Rethinking the analysis. 18/12/22

So zero crossings points move as a function of speed.

Lets assume we have 14 poles therefore $\frac{14}{2} \cdot 6 = 36$

Zero crossing. 360° in 1 electrical cycle, 7 electrical cycles per mechanical cycle. Thus if the encoder & the motor shaft are calibrated such that 0° from the encoder is 0° in terms of electrical cycles then we could easily know the zero crossing points (ideal) but the real zero-crossing point are a function of speed (voltage jumping changes)

To know the real points we can just do this from geometry, we need to convert from encoder values to electrical cycle.

Assuming the encoder gives us 2^{14} bit values,
we can convert from encoder step to mechanical angle:

$$\frac{\text{encoder-step}}{2^{14}} \cdot 360^\circ_{mech} = x^\circ_{mech}$$

x°_{mech} goes from $0 \to 359^\circ$ before looping, by this point we would have gone through 7 electrical cycles.
So a full electrical cycle of $0 \to 359^\circ$ elec is every
$360^\circ / 7 = 51.428^\circ$ mechanical.

A 360° elec is a transition from "north", "south", "north"
a 6 pole motor has 3 south & 3 north poles so for a full
electrical cycle the shaft has rotated $\frac{360}{(6/2)} = \frac{360}{3} = 120^\circ$
mechanical degrees, for a 14 pole motor $\frac{360}{7} = 51.428^\circ$ mech

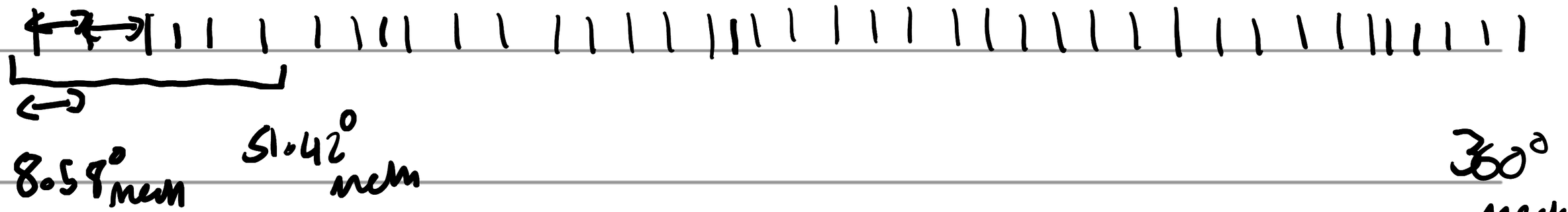
$\frac{6 \times \text{poles}}{2} = 3 \times \text{poles}$ 42 Zero crossings for 14 poles. So for 360° mechanical we could have

$\frac{360°\text{mech}}{\text{Zero crossings}} = \frac{360°\text{mech}}{3 \times \text{poles}}$ inverse $\frac{360°\text{mech}}{42}$

$= 8.78$ mechanical deg per zero crossing.

$\frac{360°}{6 \text{ electrical poles}} = 60°$

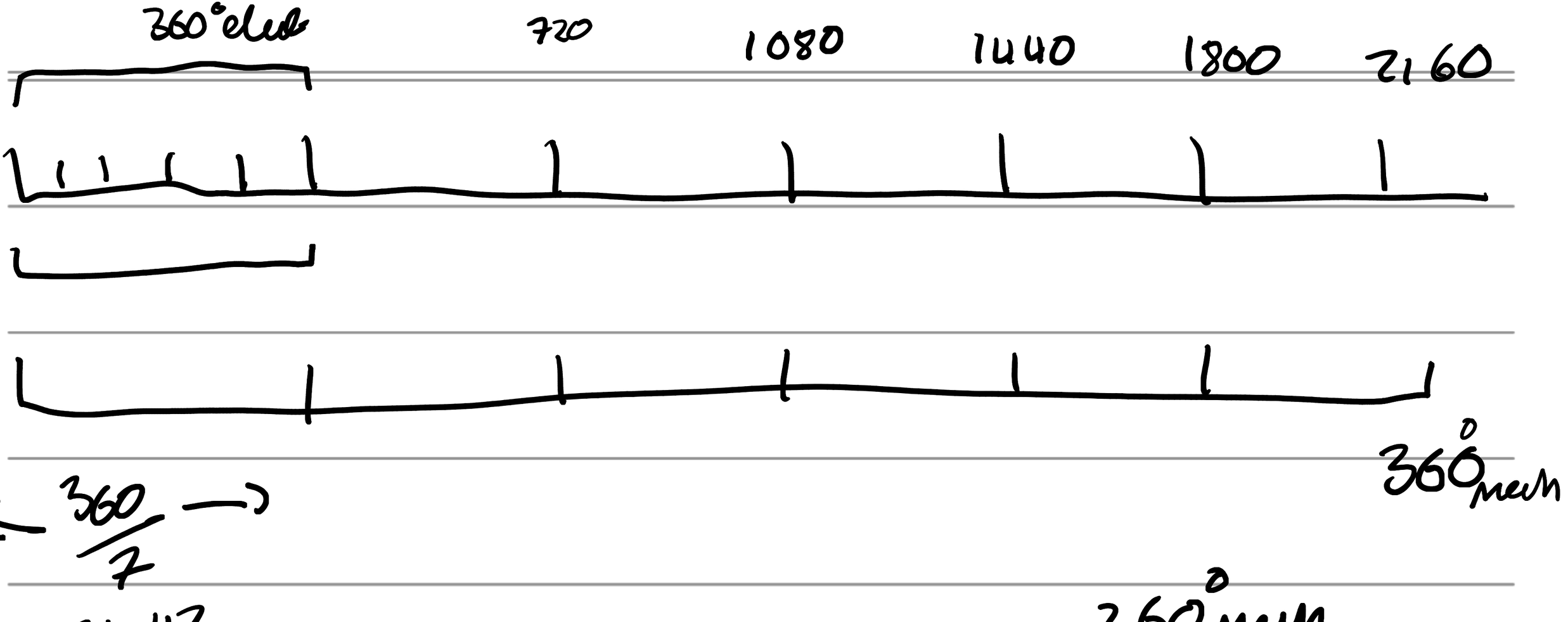
60° 60° 360° elec



It would be good to convert °mech to °electrical

$°\text{elec} = \frac{(°\text{mech} \times \text{poles})}{(\text{phases} \times 2)} = \frac{°\text{mech} \times \text{poles}}{\text{phases} \times 2} = \frac{1° \times 14}{3 \times 2} = \frac{7}{3}°\text{mech}$

check $8.58°\text{mech} \rightarrow 39.994$ elec ? should be 60



S1.42

$$1 \text{ electrical cycle} = \frac{360^\circ \text{ mech}}{(\text{poles}/2)}$$

$$360^\circ_{\text{electrical}} = \frac{360^\circ_{\text{mech}}}{(\text{poles}/2)}$$

$$\therefore 1^\circ \text{ electrical} = 1^\circ \text{ mechanical} \times \frac{2}{\text{poles}}.$$

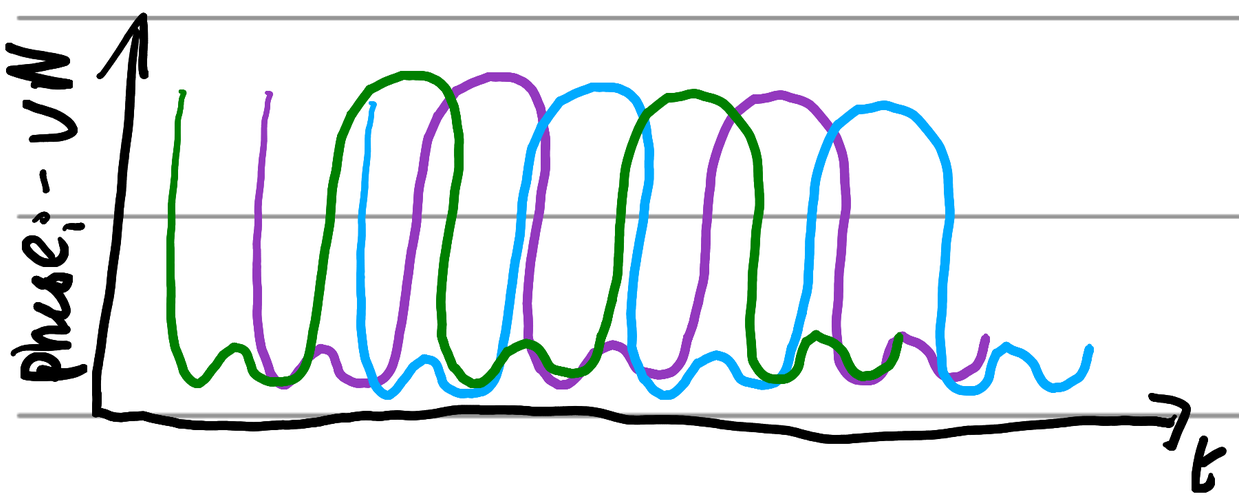
think 360° electrical should be S1.42° mech ✓

or

$$1^\circ \text{ electrical} = 1^\circ \text{ mech} \times \frac{\text{Poles}}{2 \cdot \text{phases}}$$

clearer transformation?

To calibrate the motor/encoder system one must calibrate the encoder such that step 0 lines up with 0° electrical, that way 0° mech 0° elect. To do this one could take the voltage/encoder values, we already have:



Note this is somewhat problematic (virtual neutral)

the ADC phase - VN signals are not simple sine waves as the ADC only measures +ve voltages & thus half of the waveform for phase is cutoff... it leads to a periodic pattern, but not a simple sinusoid.

When we start calibration the encoder angle is random vs the motors 0° shaft angle, we will need to find the offset.

We will need to fit some model to the recorded data & the use such a model & to calculate optimal values for the parameters for angular displacement offset & the current displacement between the

3 phases (in an ideal world should be 120°).

To start with we need a model, now we know the future is periodic our model will generate the phase currents (A-Un, B-Un, C-Un) from encoder position, angular offset (displacement of encoder from ideal startup position) & phase current displacement (ideally 120°).

The closest model we know for the phase currents is something like

current-a = sin (angular-position + angular-displacement)

-b = sin (angular-position + angular-displacement + phase_current-displacement)

-c = sin (angular_pos + angular_disp + 2 x phase-current_disp)

but the amplitude will not be 1 & the function is periodic but not a simple sin wave. One could do a series expansion (Fourier) & model the system as a sum of sine waves with some number of coefficients which we could call the fourier coefficients thus we could approach for some number coefficients & like our

f(angular-pos, phase-disp, Fourier coefficients) → current curves

for i, coefficient in coefficients:

phase-a t= coefficient x sin((1+i) x angular-pos)

-b t= coefficient x sin((i+1) x(angular-pos + phase_disp)

-c t= coefficient x sin((i+1) x (angular-pos + 2x phase_disp)

Then using some curve-fitting function it could shift the guessed phase currents by the angular-position & optimizing the phase displacement the angular-disp & fourier coefficients until the model fits the data...

at this point we will have some good approximation of the model coefficients.

Now with coefficients we can create a new model & optimize one final time this time keeping the coefficients constant & optimizing only for the angular_disp & phase_disp.

f (angular pos, angular_disp, phase_disp, Fourier coefficients) → current data

FIXED!

for i, coefficient in fourier_coefficients

phase_a_current += coefficient x sin ((i+1) x (angular_pos + angular_disp))

_b_ += coefficient x sin((i+1) x angular_pos + angular_disp + phase_disp)

_c_ += coefficient x sin((i+1) x angular_pos + angular_disp + 2x phase_disp)

The optimised angular_disp will then represent the offset & the phase_disp represents the ideal phase current disp (this enables better for errors & manufacturing defects of the motor, away from 120° separation)

If the optimisation fitting function generates a covariance matrix then the diagonal elements can be summed with a trace, are the variances of the estimation errors. $\sqrt{\text{trace}(cov)}$ would be the errors for the optimised parameters.

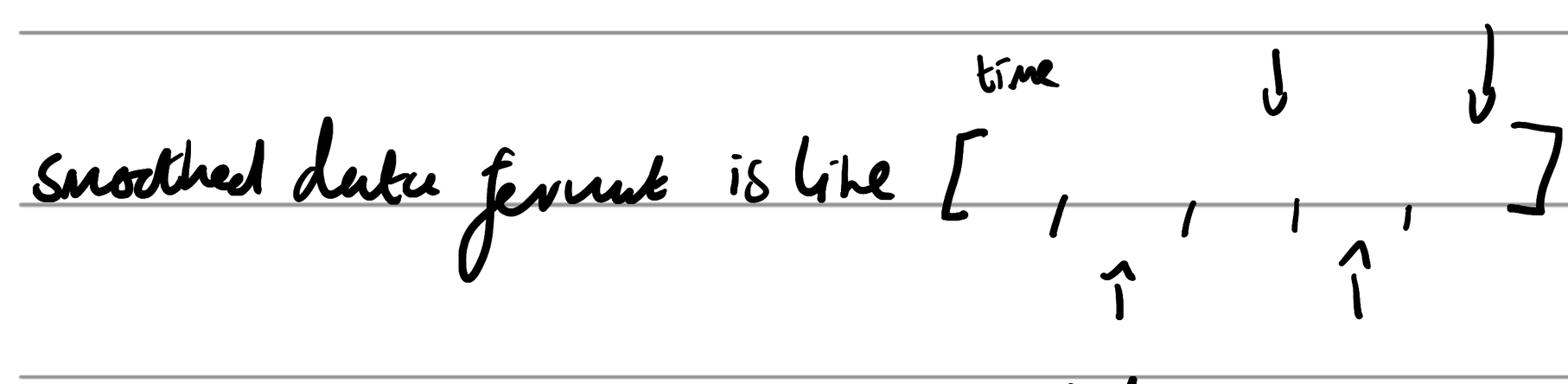
To make sure the model fit curve sensible, it would be a good idea to display a graph of the model & the experimental data points & make sure the fit looks sensible.

Artifacts / bad runs from the system would have a-periodic issues that could ruin the fitting system. (would be reflected in the errors)

5/1/23

Finally have a new working laptop so I can continue project dad laptop power supply fanled.

looking at alternative fitting method.



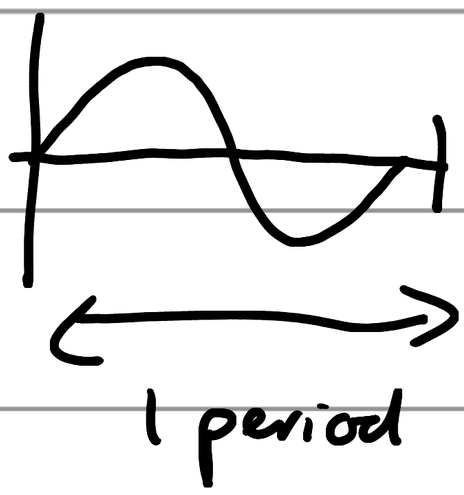
13/11/23

So wrote analysis - test 3, initial results frequency of sin waves over angular period is wrong, model does not account for # poles.

Think about it 3 phases, $\frac{\text{Poles}}{2} \times 6$ $\quad \frac{14}{2} \times 6 = 7 \times 6 = 42$

42 crosses for 3 phases $\quad \frac{42}{3} = 14$ ZC's per angular period

or $\frac{\text{poles}}{2} \times 6 \ldots \frac{\text{poles} \times 3}{\text{phases}}$ ZC's per cycle period per phase

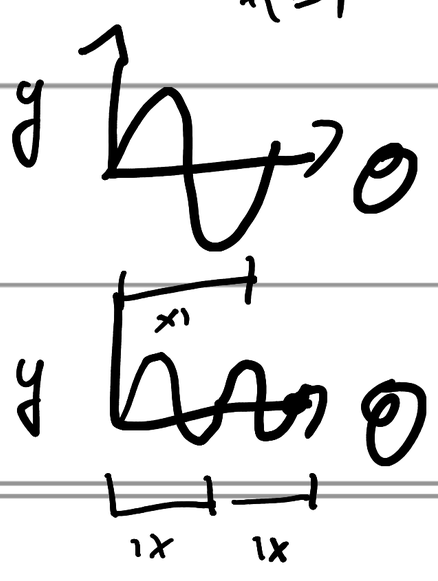


how my electrical cycles per angular cycle? 6 steps $7\left(\frac{\text{poles}}{2}\right)$ cycles

so each needs 7 periods within each 360° mech

2π° mech

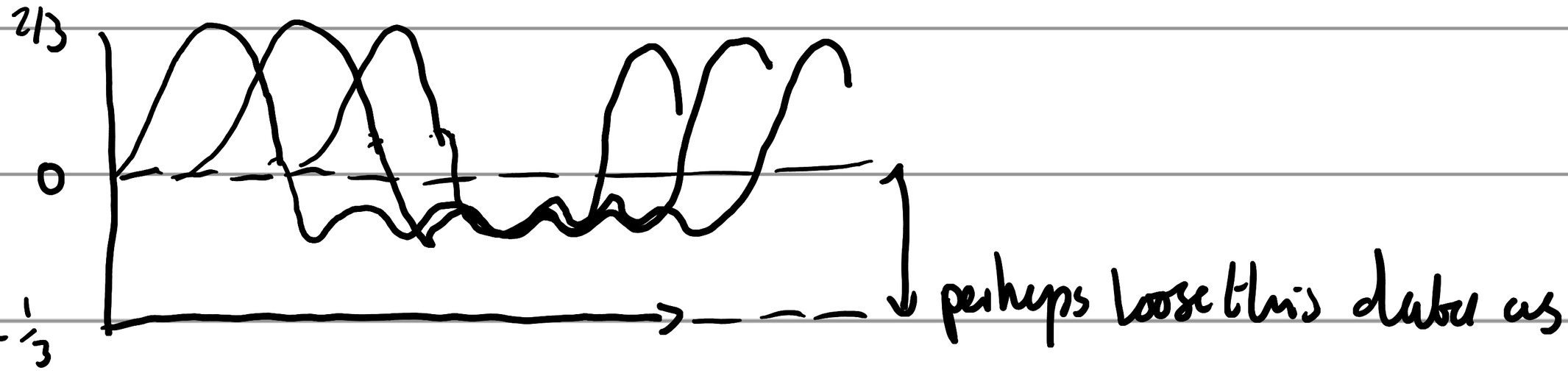
$y = A \sin \frac{2\pi}{k} \theta$



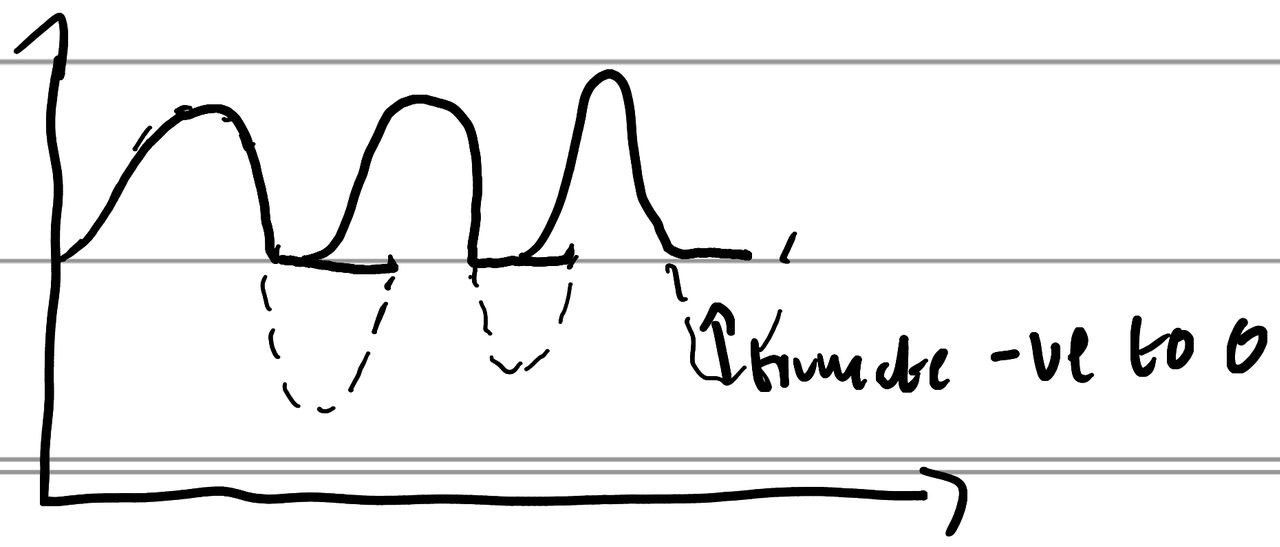
14/1/23

Model fitting results:

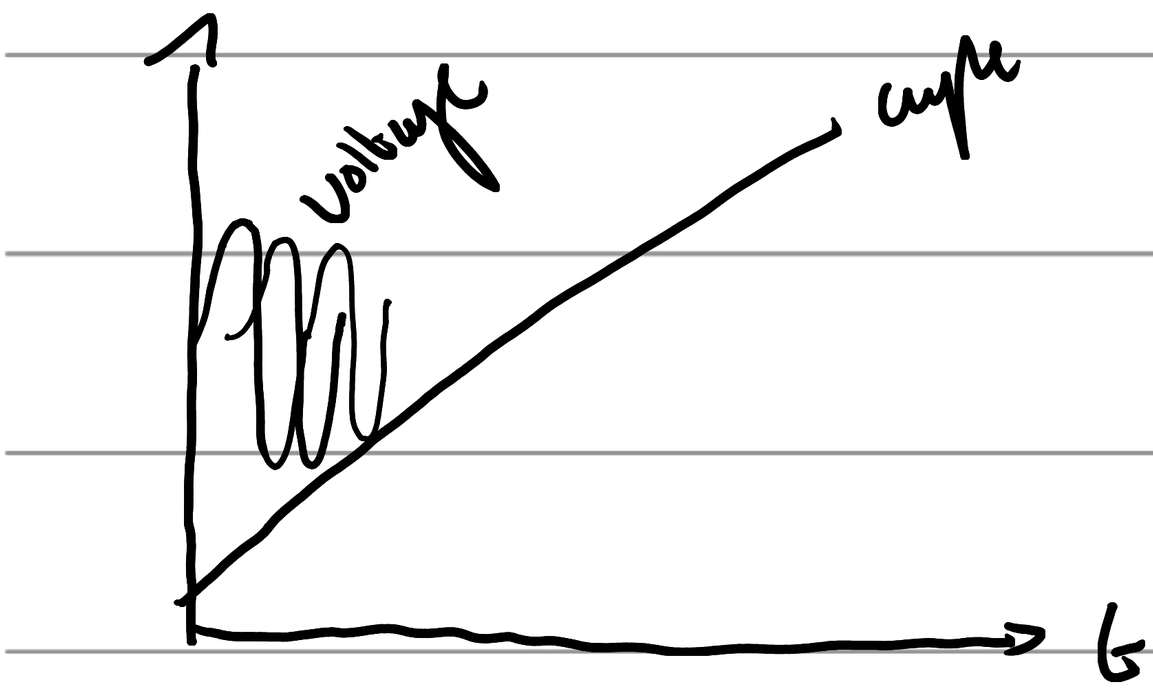
So the fourier idea didn't pan out so well, it is best with a single coefficient. Perhaps adding a lot of coefficients would eventually work but like single truncated fit is very slow. If I wish to continue the perhaps looking at a GPU fitting library. The other alternative is to create a bespoke fitter for the fit data to exclude -ve resnts:



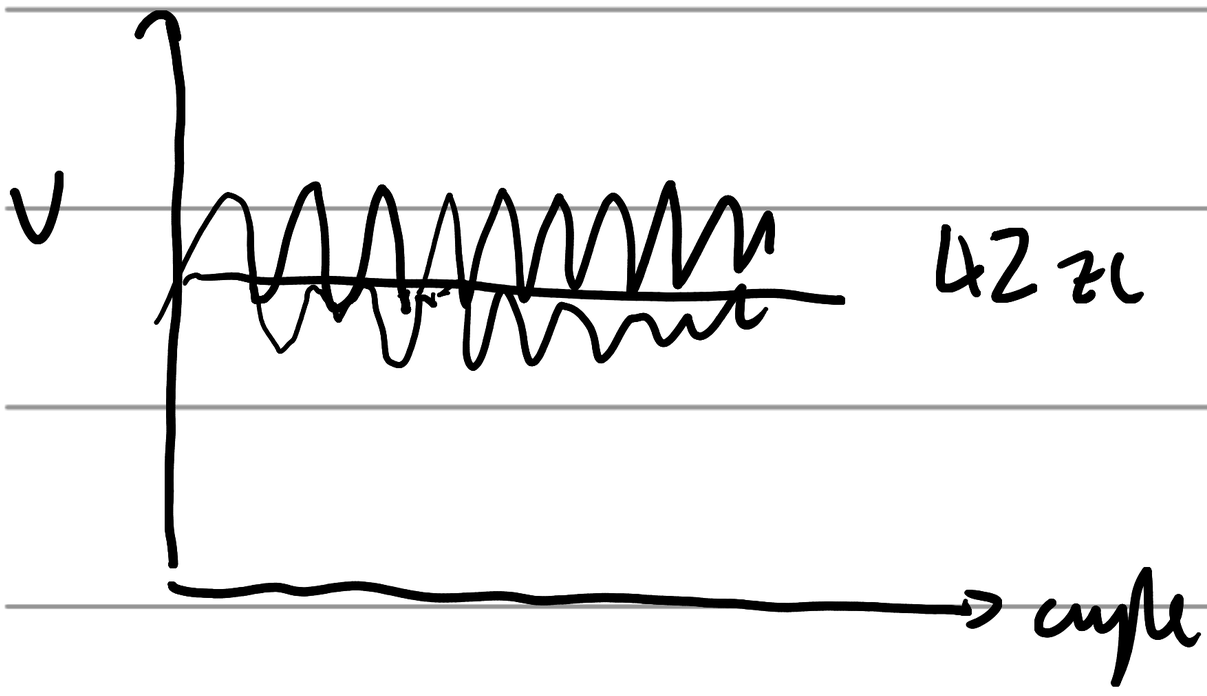
fit with 1 coefficient worked well but for the distorted region.



Another question is if switching points are really a function of speed.



I agree that as a function of time, the frequency would change



But if we have it in terms of cycles then although the ratio at which cycles would be swept through would increase the voltage frequency per cycle sweep might not.

Next to try would be remove the -ve data, or where otherwise plus ABC are near zero. Filter this data & attempt to fit the curves.

Chopped off all -ve (phase $X_i - V_n$) values. Fit yields

239.96 ± 1.0 phase current displacement & $-27.17 \pm 1.3^\circ$

angular-shaft displacement. Data was from a counter clockwise

run sept-x-ccw, phase current of ~240 makes sense as

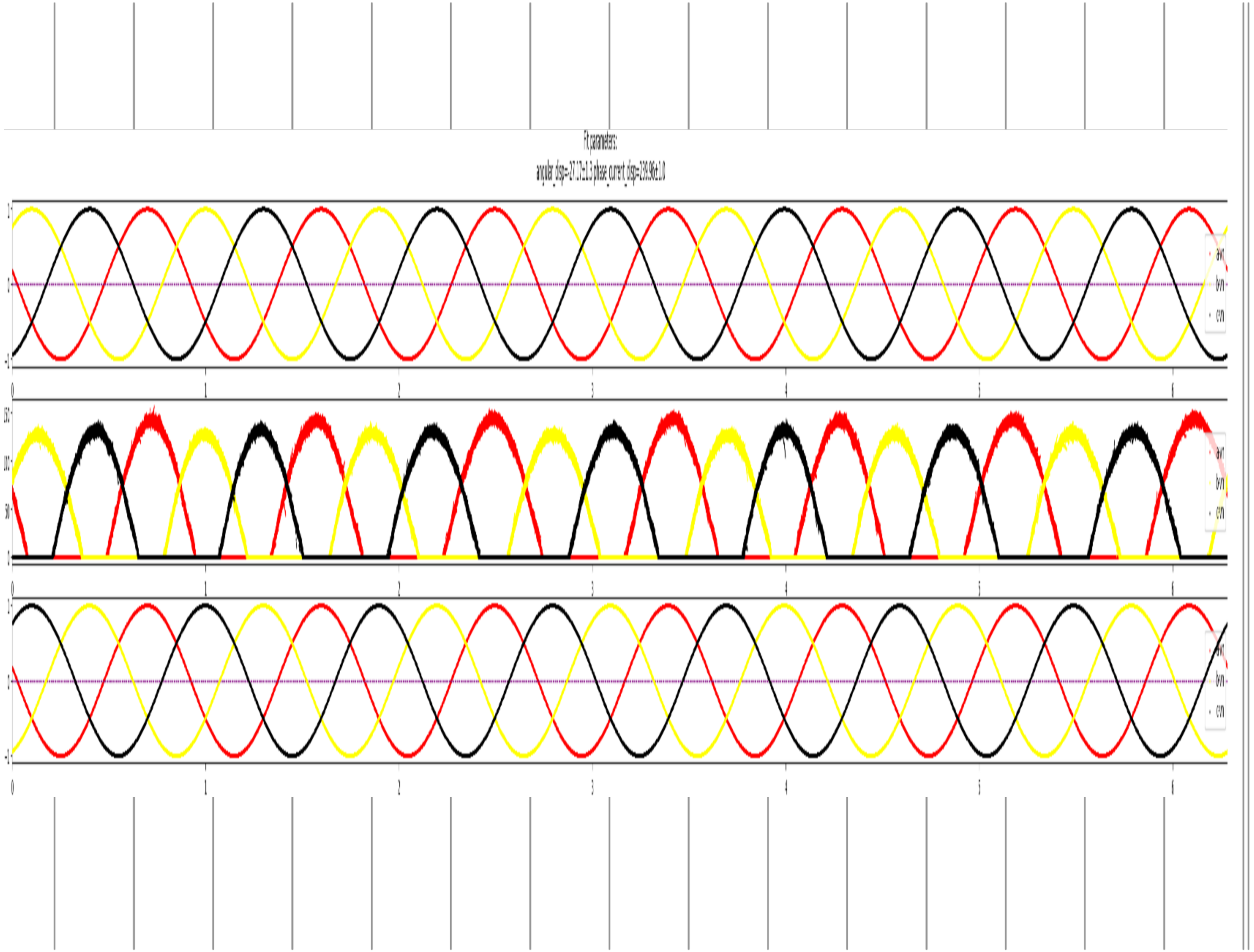
fit was against CW model, & a phase current displacement

of 240 swaps the order from blue, yellow, red (CW)
a b c c b a

to red, yellow, blue (ccw) while still maintaining a 120°

seperation $0^\circ + 240^\circ$

$240^\circ + 240^\circ = 480^\circ \% 360 = 120^\circ$


Fit parameters:
angular_disp=-27.17±1.3 phase_current_disp=239.96±1.0
0
1
2
3
4
5
6
ave
bvm
cvm

Tried sepe-2-cw & sepe-4-cw

d-d = -2.4 ± 1

p-c-d = 137.1 ± 0.8

d-d = -2.4 ± 1

p-c-d = 137.1 ± 0.9

hmm... this makes much less sense, each cw run is consistent & quite far from 120°, angular displacement discrepancy for cw & ccw!

Need to rework the model maybe? Not sure what to conclude.

---

cw pattern

A B C

→ direction in graph: R, Y, B

← direction in graph: A C B; R, B, Y

ccw pattern

A C B

← direction in graph: R B Y

→ direction in graph: A B C; R Y B

See if cycle repeats clearly in paint and chop start & stitch on to end of plot... is it continuous?

17/11/23

Thinking about old analysis, could use zc data & try to fit a model to the limited data. Need to do data recovery on the old disk. The combined reports attempted head angle between cw & ccw. Could try & extract stable maps generated from the hbmf files

Nvme disk is a M.2 NVMe 2280 PCIe 3.0 (4 Gen) x4

File cuddy SIO. ordered an adaptor.

M key S

25/11/23

Recovered data successfully.

Things to do 1 -> Implement GSC assuming analysis 2 is correct, 2 -> export binary spike train data.

Labs do 21:

→ Reinstall project dir on new laptop.

→ update project prefixes.

→ re run analysis - all works

→ Need to adjust code to export a JSON file of the

ZC data (binary spike train)

→ New mem / new std

Process is take new-mem ... get ordered lkv list,

for each channel filter lkv list of rising/falling clusters & store

in 3 lists 2 for each channel, arrange into kv map. Next

of each list of rising/falling get pairs of item i & i+1,

for each pair find midpoint & emit 3 entries

(channel-name, value $^{+1.0/-1.0\%}$, cycle value) do this

for the first, midpoint & second. Collect these for all

channels & gotten & order. At this point we have not just ZC values but max & min values. Finally we can process the gathered list & collect a map of data {apples: [], a: [], f: [], c: []} & errors {a: [], b: [], c: []} errors are taken as 1.0 channel where we have a ZC or a midpoint, if we had to pad a zero for the remaining channels then the error is taken as Infinity

Next steps... could take the combined id or just the folder run names, combine all data for all runs, chop off -ve data... fit curve... compare 2 methods.

26/11/23

Before I try & fit the curves Rahim consolation with a model need to revisit work yesterday.

The zc analyser only detects zc events in the +ve angular (cw) direction. So to fit for ccw (-ve) angular direction we need to flip rising to falling & refit.

Works ok by reversing cw data to form ccw data. Fit with 240° initial is very close 239.99 for both if 120° initial we get 137.13° for both.

Now rewrite analysis for all three runs & use the raw Kalman voltage data merge & flip!

Done, with disp(plus) 240 get somewhat similar fit params, visually very similar & consistent.

27/11/23

Main files analysis-raw-recursion.py

Other main analysis analysis-ZC-Recursion

For combined runs would be nice to just stick to the id file conventions, change the code in repeat to always generate an id file.

Repeats should be able to generate a file with a chosen name (simple) (title, file_name, file_pattern, run_ids)

Combined repeats should be given a file name pattern & the code should produce an id

Tested to work with 2 exps, working with 3 exps!
Tested & documented ZC raw fitting.

Next need to check state map as in CCW rising should be seen as falling edges & vice versa

Counter-clockwise rotation (-ve)

ESC	5	4	3	2	1	0
IN	C	C	B	B	A	A
SD	B	A	A	C	C	B
RISING	A	X	C	X	B	X
FALLING	X	B	X	A	X	C

This is expected to trigger this state

5 4 3 2 1 0

clockwise rotation (+ve)

ESC	0	1	2	3	4	5
IN	A	A	B	B	C	C
SD	B	C	C	A	A	B
RISING	C	X	A	X	B	X
FALLING	X	B	X	C	X	A

This is expected to trigger this state

0 1 2 3 4 5

A Red
B Yellow
C Black

Rising / Falling indicates the NEXT expected edge.

For each direction there are 6 total steps

	0	1	2	3	4	5
	A	A	B	B	C	C
	B	C	C	A	A	B

CCW seq	Step	5	4	3	2	1	0
	rise/fall	↑	↓	↑	↓	↑	↓
	Phase Detect	A	B	C	A	B	C

CW seq	Step	0	1	2	3	4	5
	rise/fall	↑	↓	↑	↓	↑	↓
	Phase detect	C	B	A	C	B	A

CW checks out.

CCW... A↑ → 4

C↓ → 5

But we should flip rising/falling detections.

create_bi_directional_state_cycle_map is the

function. which derives its state from function

channel_name_and_direction

Counter-clockwise rotation (-ve)

ESC	5	4	3	2	1	0
IN	C	C	B	B	A	A
SD	B	A	A	C	C	B
RISING	A	X	C	X	B	X
FALLING	X	B	X	A	X	C

This is expected to trigger this state

5 4 3 2 1 0

so this needs correcting a ZC detection always goes in CW direction & this sees A↑ as A↓.

CORRECTED

Counter-clockwise rotation (-ve)

ESC	5	4	3	2	1	0
IN	C	C	B	B	A	A
SD	B	A	A	C	C	B
RISING	X	B	X	A	X	C
FALLING	A	X	C	X	B	X

This is expected to trigger this state

5 4 3 2 1 0

A↑ → 1 → 2

A↓ → 4 → 5

B↑ → 3 → 4

B↓ → 0 → 1

C↑ → 5 → 6

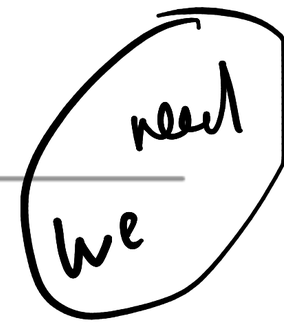
C↓ → 2 → 3

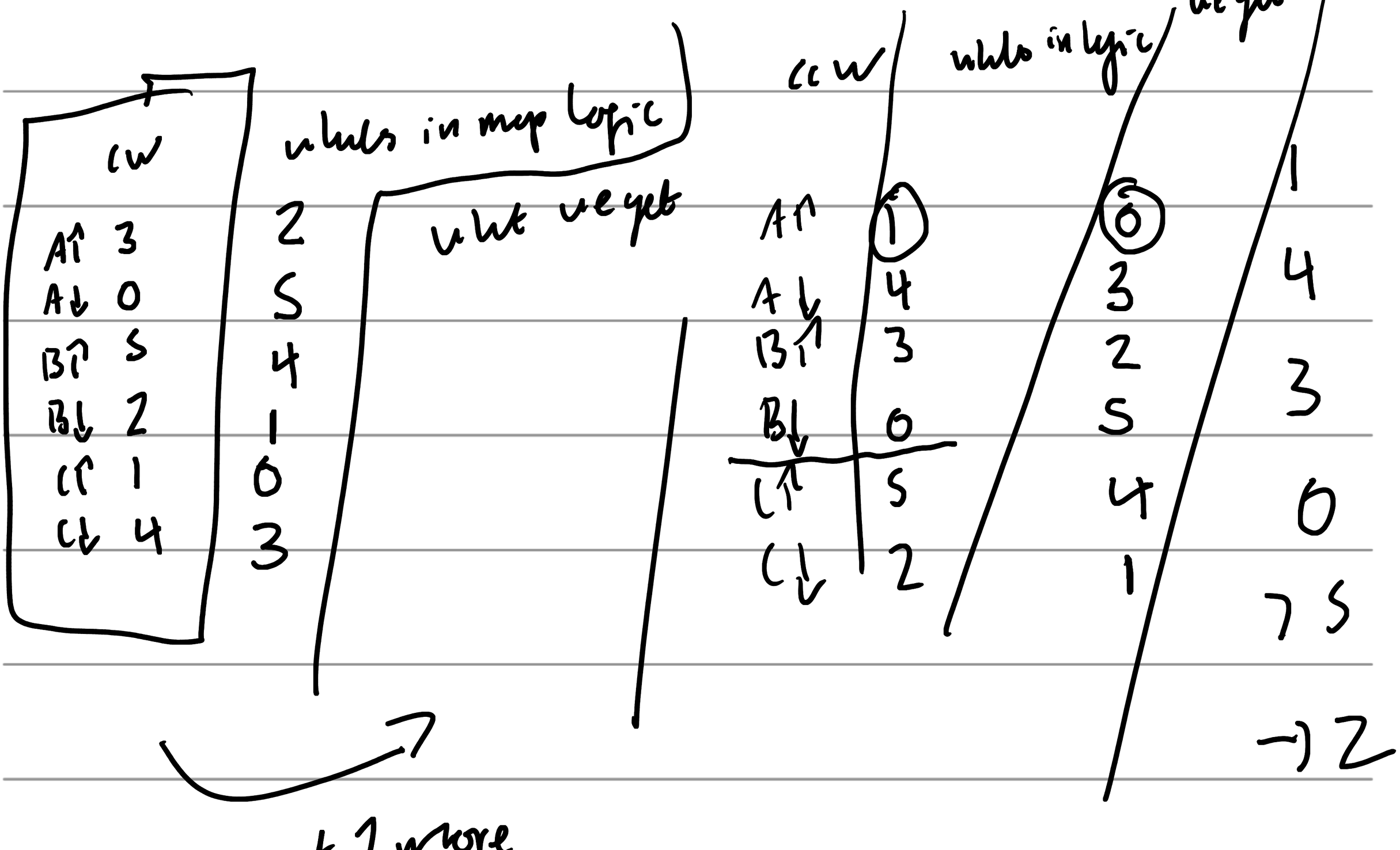
Re check ccw post analysis (state map) should have

as encoded Translated (gripped) we get

A↑ ... A↓ 0 1

A↓ ... A↑ 3 2





changes applied ... retesting

Counter-clockwise rotation (-ve)

ESC	5	4	3	2	1	0
IN	C	C	B	B	A	A
SD	B	A	A	C	C	B
RISING	A	X	C	X	B	X
FALLING	X	B	X	A	X	C

This is expected to trigger clock state

Remember when reading of the chart to inverse interpretation of rising / falling when going

5 4 3 2 1 0 CW

check above

	looking for	
A ↑	A↓ in counterclockwise	4 ✓
A ↓	A↑ in cw state	1 ✓
B ↑	B↓ in ccw state	0 ✓
B ↓	B↑ in ccw state	3 ✓
C ↑	C↓ in ccw state	2 ✓
C ↓	C↑ in ccw state	5 ✓

Checks out... Actually doesn't... reverting to older code.

Added dark theme.

29/11/22

Next step ... time to program the CSG. Think about pins...

CSN MOSI MISO CLK

we are using pin [10, 11, 12, 22] for the encoder,

[3&7] are being used as inputs (optos), (GNDe 3.3v also for encoder). We need another encoder (in future) & 2x sets of pwm pins, from two pwm modules (duty cycle apart).

Miso2 sosi2 spi pins need to be soldered not daisy chain!

Try random other pins & benchmark.

Flex 1.0 1

10 1 0
1. 3 7

pwm	A	B	C
SD	(1,	0,	7)
	1-0	1-1	1-3

PWM modules 1 IN (2, 9, 8)

A B C)

analogue write Frequency

sm0 clock INIT-SCL(2) AUX_CLK (only for sm's)

Flexpwm_EXT-CLK

SYNC

PWM-EXT-SYNC

Remiing PWM modules

2.0 4, 33

2.1 4, 5, 6 looks good

2.2 6, 9

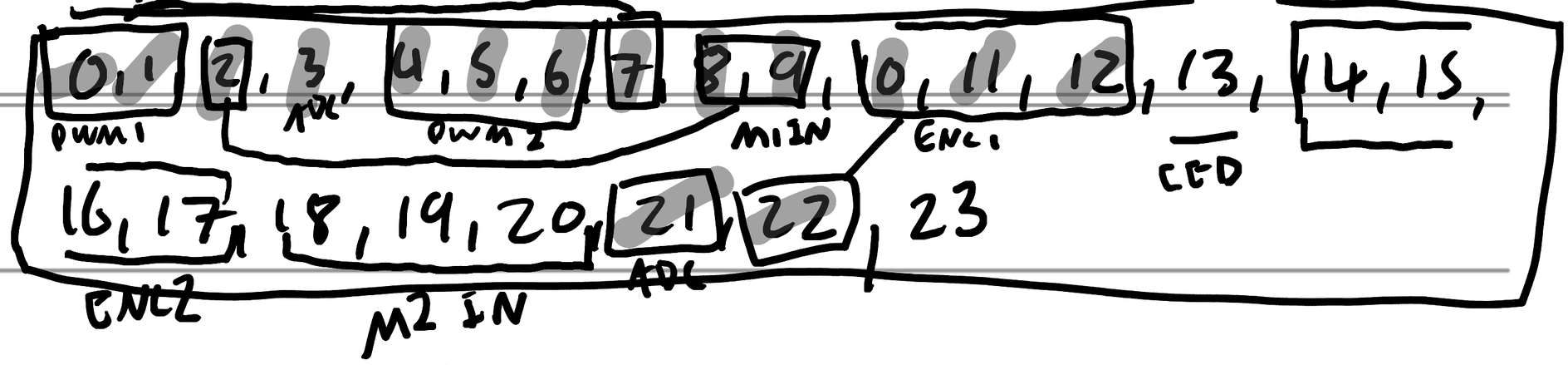
3.1 28, 29

4.0 22

4.1 23

4.2 2, 3

Final pinout



MOTOR 1	Flex	SD	PIN	MOTOR 2	Flex	SD	PIN
FlexPWM	1.0	1	1	FlexPWM	2.0	1	4
	1.1	2	0		2.1	2	5
	1.3	3	7		2.3	2	6

MOTOR 1	IN	PIN	MOTOR 2	IN	PIN
	1	2		1	18
	2	9		2	19
	3	8		3	20

7 -> 21

Encoder 1		SN	10	Encoder 2	SN out	14
MOTOR 1	Flex	SD	PIN		MOSI out	15
	MOSI	11				
FlexPWM	1.0	1	1		MISO in	16
	MISO	12				
	1.1	2	0		CLK out	17
	CLK	22				
	1.3	3	~~7~~			
ADC sync	3	~~7~~	21			

change this in driver!

commutative 6 step

Basic ESC (BLDC)

Vars

commutation cw ccw

State map [[0, 1, 2, 3] , [5, 4, 3, 2]]

State config [0: [IN, SD], 2: [SN, SD] ...]

IN/SD Pins thrust setting

Encoder pins

Direction last Encoder Value

Rutines

Setup → setup pwm sync/freq set-commutation state /enable

started = false

read usb() → profile

loop → if ! started

→ wait for first signal

started = true

should before

if started

→ read encoder

→ has state changed

→ yes · set state

→ read host profile → If 0 then stop

how to sync submodules EXT_SYNC EXT_FORCE

FlexPWM_0 master reload → EXT_SYNC FlexPWM_1

FlexPWM_0 configured as master, generate master reload signal (MRS) & center sync signal (mcenter sync).

The MRS signal is generated at every occurrence of SM0 VAL1 compare (half cycle reload), VAL defines the center value, INIT the start.

Sync between module 0 & 1 is done by MSR. This signal is connected internally to EXT_SYNC input module 1. Module 1 is set to generate the MRS in the middle of the PWM period.

PWM signals are synced by cic period. connect via XBAR/ crossbar 61.3.2. "Functional Mode"

1/2/23

Programming esc, ino created, folder created for calibration state mgp.

Logic started = false ?
  read profile

  started = true

    state != old state
      Enter combination
      read profile

MOTOR 1 Flex SD PSN MOTOR 1 IN PIN

Flex pwm	1.0		1	2
	1.0	2	2	9
	1.3	3	3	8

Did first draft of the ESC.

would be nice to have line number comms

↗ check ☑

→ Need to refactor statefile multiline issues.

RM || check ☑

→ Need to cleanup startup mode (delay means we don't get junk!

→ Need to cleanup prints!

(we are going for high speed) ☑ DEBUG mode

→ parameter so we can scale input ✓

2/2/23

Refactored PS4Z into a nice library so you can easily write PS4 to serialport code will allow easy network extensions for additional outputs for charting etc. Need to rename it something good. DRONIUM.IO? HID. BIN DIR! Index files!

Changed engine communication state functions. to turn everything off first.

Need to put in some sort of protection so that if high speed cannot reverse direction... perhaps if direction if flipped then emit 0% thrust (overriding preference... wait for user to bring throttle below 5% before ramping up again.

Maybe need to reduce its power consumption as its like 40% of a whole core.

Cpp Kalman lib (multi-core for analysis)

Analysis cleanup

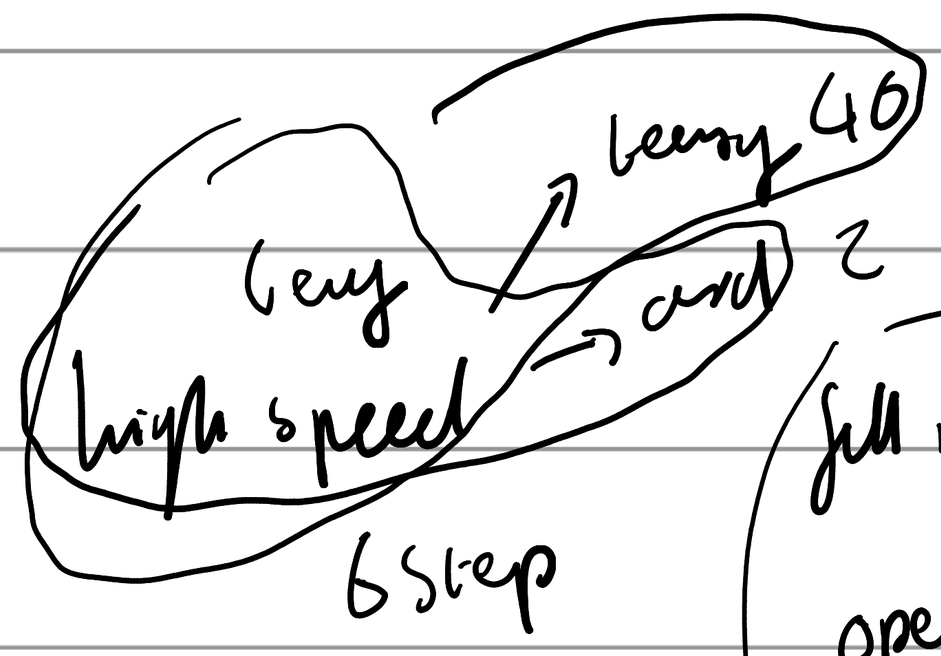
Seperate into submodules teensy40 - blde-calibration

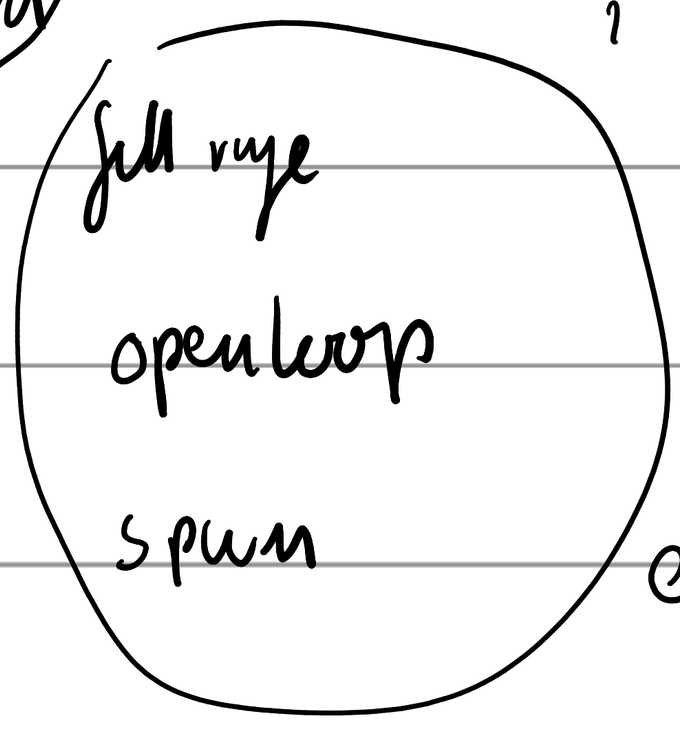
- blde-driver - 6step - encoder

- commutative

- commutative sine

TODO

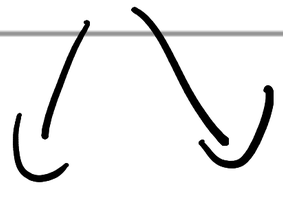




encoder

full range

closed loop



sinewave 6step

spwm

exists

teensy-highspeed (teensy card) 6step

encoder-fullrange (teensy) 6step|spwm

teensy-highspeed-undercurrented (teensy card)

encoder-fullrange-undercurrented.

speed control SMI OLD

closedloop

3/21/23

Adding prints to previous the state value & cycle in debug mode. Angle changes when the direction changes! This is bad. Need to try this with a real encoder. Flash hardened check.