

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

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

Assuming the encoder gives us  $2^{14}$  bit values,

we can convert from encoder step to mechanical angle:

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

$\alpha_{\text{mech}}^\circ$  goes from  $0 \rightarrow 359^\circ$  before looping, by this point we would have gone through 7 electrical cycles.

So a full electrical cycle of  $0 \rightarrow 359^\circ$  elctric every

$$360^\circ / 7 = 51.428^\circ_{\text{electrical}}$$

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

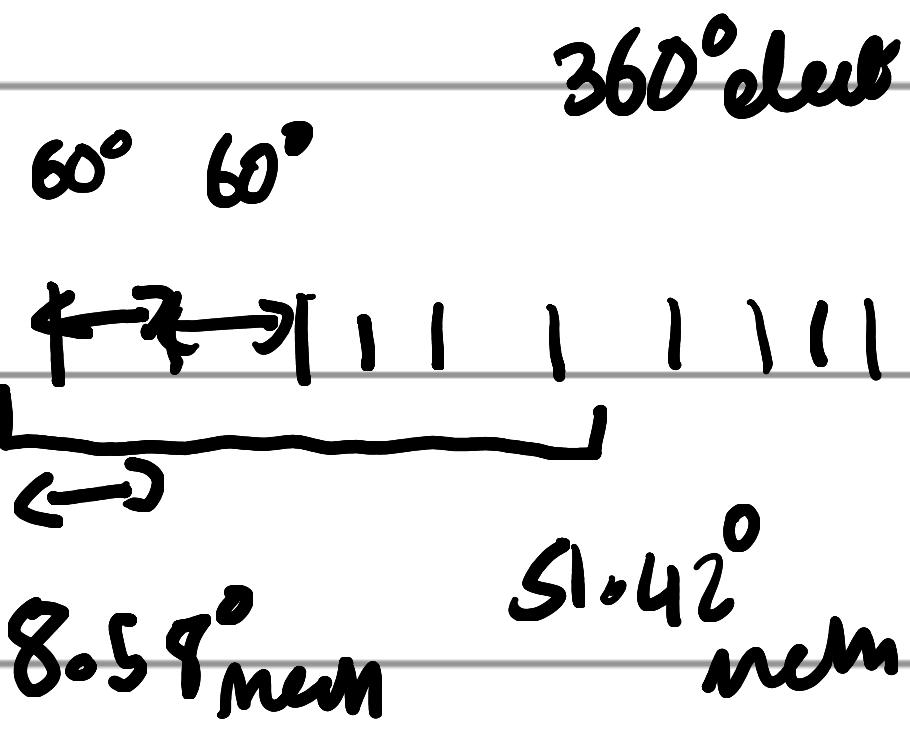
$$\frac{6 \times \text{pdes}}{2} = 3 \times \text{pdes} \quad 42 \text{ zero crossings for } 14$$

poles. So for  $360^\circ$  mechanical we could have

$$\frac{360^\circ \text{ mech}}{\text{zero crossings}} = \frac{360^\circ \text{ mech}}{3 \times \text{pdes}} \text{ in reverse} \quad \frac{360^\circ \text{ mech}}{42}$$

$= 8.78^\circ$  mechanical deg per zero crossing.

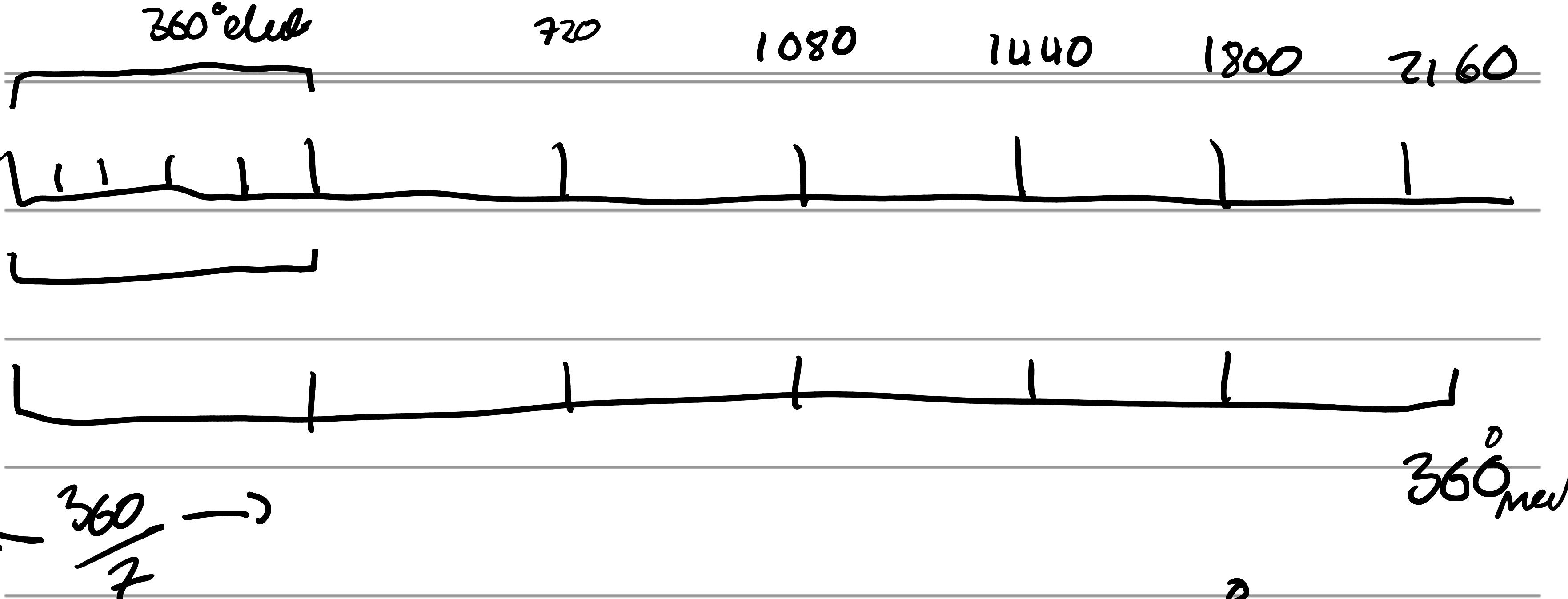
$$\frac{360^\circ}{6 \text{ electrical revs}} = 60^\circ$$



It would be good to convert  $^\circ$  mech to  $^\circ$  electrical

$$\text{elect} = \frac{(\text{mech}^\circ \times \text{pdes})}{(\text{phases} \times 2)} = \frac{\text{mech}^\circ \times \text{pdes}}{\text{phases} \times 2} = \frac{1^\circ \times 14}{3 \times 2} = \frac{7}{3}^\circ \text{ mech}$$

check  $8.58^\circ$  mech  $\rightarrow$  39.999 electrical? should be 60



$\leftarrow \frac{360}{7} \rightarrow$

51.42

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

$$360^\circ \text{ electrical} = \frac{360^\circ \text{ mech}}{(p_{\text{des}}/2)}$$

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

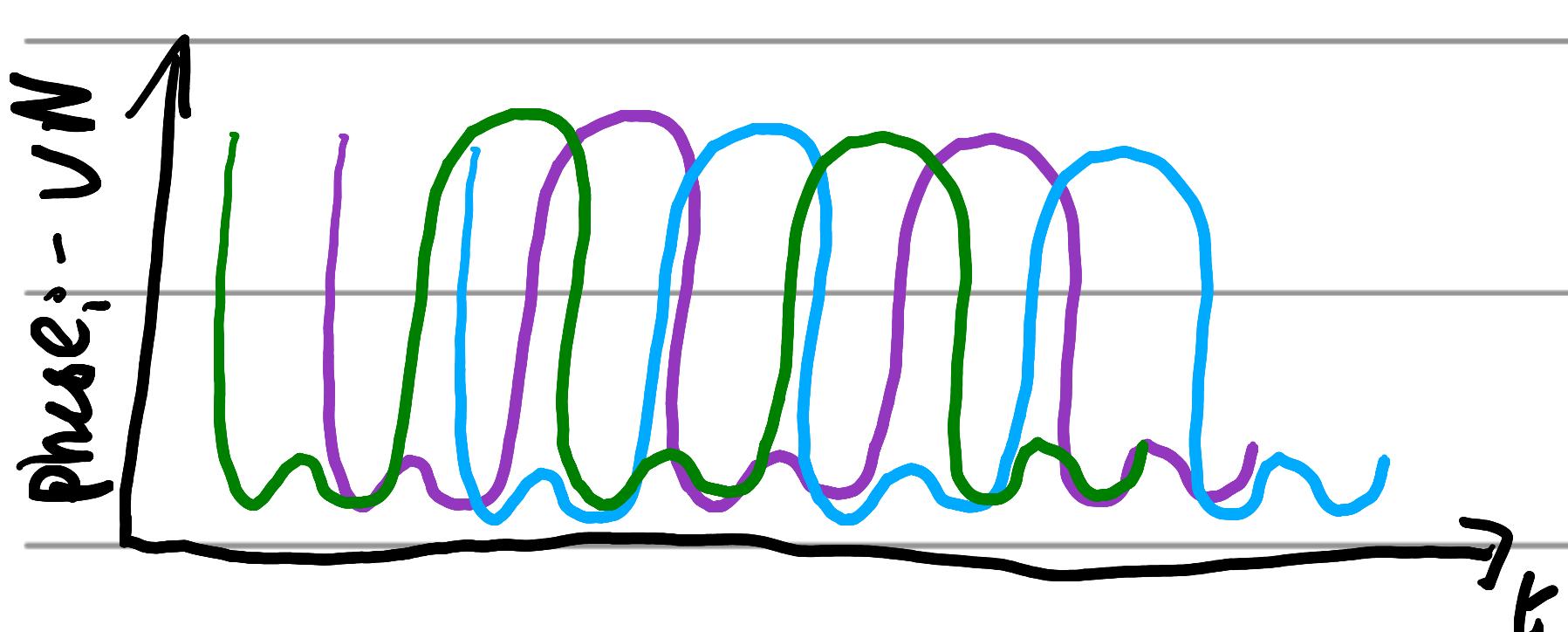
think 360° electrical should be 51.42°<sub>mech</sub> ✓

or

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

clerk transformation?

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



Note this is somewhat problematic  
(virtual natural)

the ADC phase; -  $V_n$  signals  
are not simple sine waves as the ADC only measures the voltages  
thus half of the waveform for phase; is cut off... it leads  
to a periodic pattern, but not a simple sinusoid.

When we start calibration the encoder angle is random vs  
the motors  $0^\circ$  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  $120^\circ$ ).

To start with we need a model, now we know the function is periodic our model will generate the phase currents ( $A-U_n, B-U_n, C-U_n$ ) from encoder position, angular offset (displacement of encoder from ideal shaft position) & phase current displacement (ideally  $120^\circ$ ).

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

$$\text{current-}a = \sin(\text{angular-position} + \text{angular-displacement})$$

$$-b = \sin(\text{angular-position} + \text{angular-displacement} + \text{phase-current-displacement})$$

$$-c = \sin(\text{angular-pos} + \text{angular-disp} + 2 \times \text{phase-current-disp})$$

but the amplitude will not be 1 & the function is periodic but not a simple sin wave. One could do a series expansion (Fourier)

I model the system as a sum of sine waves with some number of coefficients which we could call the Fourier coefficients thus we could obtain for some number coefficients & thus our

$f(\text{angular-pos}, \text{phase-disp}, \text{coefficients}) \xrightarrow{\text{Fourier}} \text{currents}$

for i, current in coefficients:

$$\text{plus-d } t = (\text{coefficient} \times \sin((i+1) \times \text{angular-pos}))$$

$$-b + = (\text{coefficient} \times \sin((i+1) \times (\text{angular-pos} + \text{phase-disp})))$$

$$-c - = (\text{coefficient} \times \sin((i+1) \times (\text{angular-pos} + 2 \times \text{phase-disp})))$$

Then using some curve-fitting function it could shift the quantified phase currents by the angular-position & optimising the phase displacement like 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 & optimise one final time this time keeping the coefficients constant & optimising only for the angular-disp & phase-disp.

Furier

f (cyclic pos, cyclic-dip, phase-dip, coefficients)  $\rightarrow$  current data  
FIXED!

for coefficient in furier-coefficients

$$\text{phase-a-current} += \text{coefficient} \times \sin((i+1) \times (\text{cyclic-pos} + \text{cyclic-dip}))$$

$$-b- += \text{coefficient} \times \sin(i+1) \times \text{cyclic-pos} + \text{cyclic-dip} + \text{phase-dip}^3)$$

$$-c- += (\text{coefficient} \times \sin(i+1) \times \text{cyclic-pos} + \text{cyclic-dip} + 2 \times \text{phase-dip})$$

The optimised cyclic-dip will then represent the offset & the phase-dip represents the ideal phase current dip (this accounts better for errors & manufacturing defects of the motor, away from 180° separation)

If the optimisation fitting function generates a covariance matrix then the diagonal elements can be summed with a trace,

over the variances of the estimated errors.  $\sqrt{\text{trace}(\text{cov})}$  would be the errors for the optimised parameters.

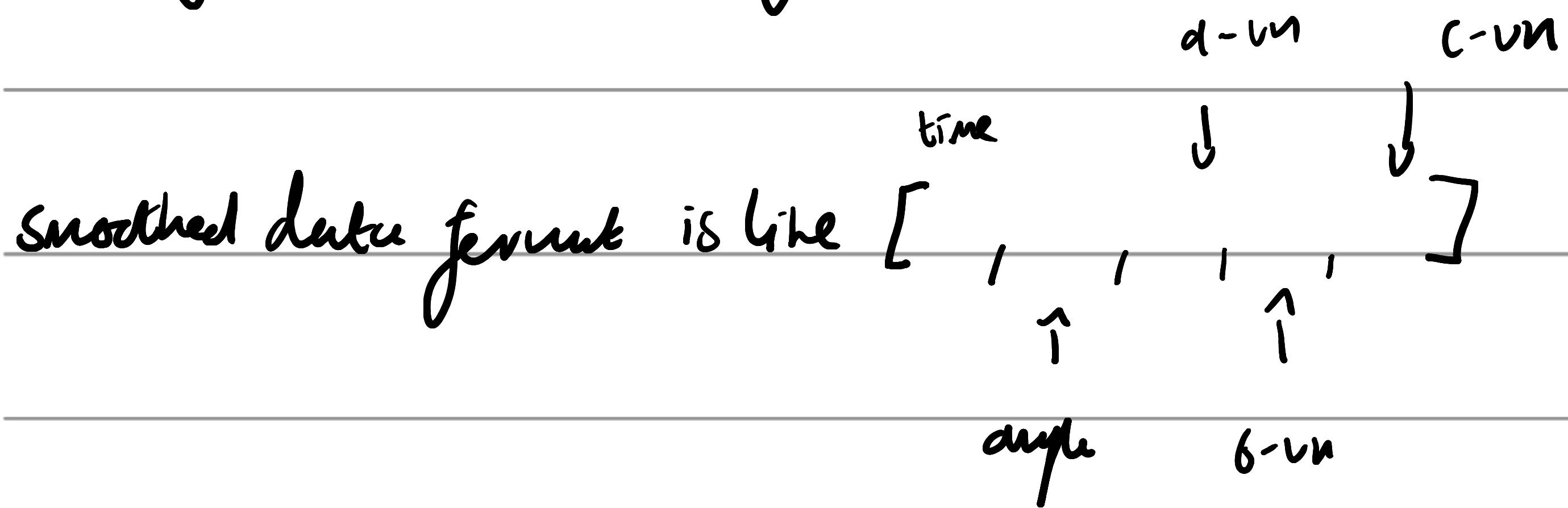
To make sure the model & fit are sensible, it would be a good idea to display a graph of the model & the experimental data points & make sure the fit looks sensible.

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 can continue project  
laptop powersupply failed.

Looking at alternative fitting method.



13/11/23

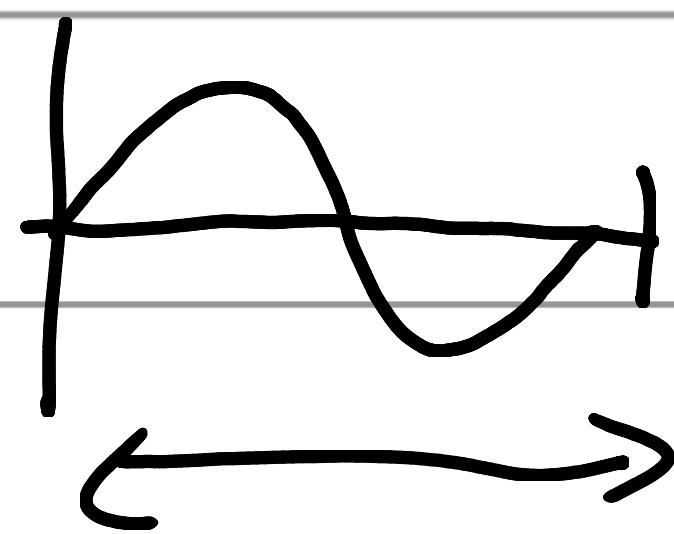
So wrote analysis-test 3, initial results frequency of sin waves over angular period is wrong, model does not account for # poles.

Think about it 3 phases,  $\frac{\text{Poles}}{2} \times 6$

$$\frac{14}{2} \times 6 = 7 \times 6 = 42$$

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

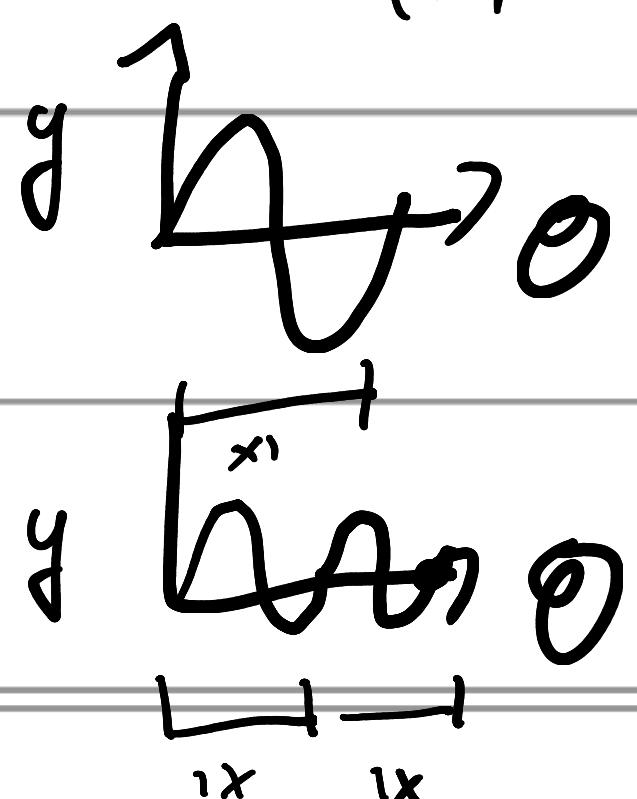
or  $\frac{\text{Poles}}{2} \times 6 \dots \frac{\text{poles} \times 3}{\text{phases}} \text{ zero's per phase}$



1 period in  $360^\circ$

How many electrical cycles per angular cycle? 6 steps  $7 \left( \frac{\text{Poles}}{2} \right)$  cycles

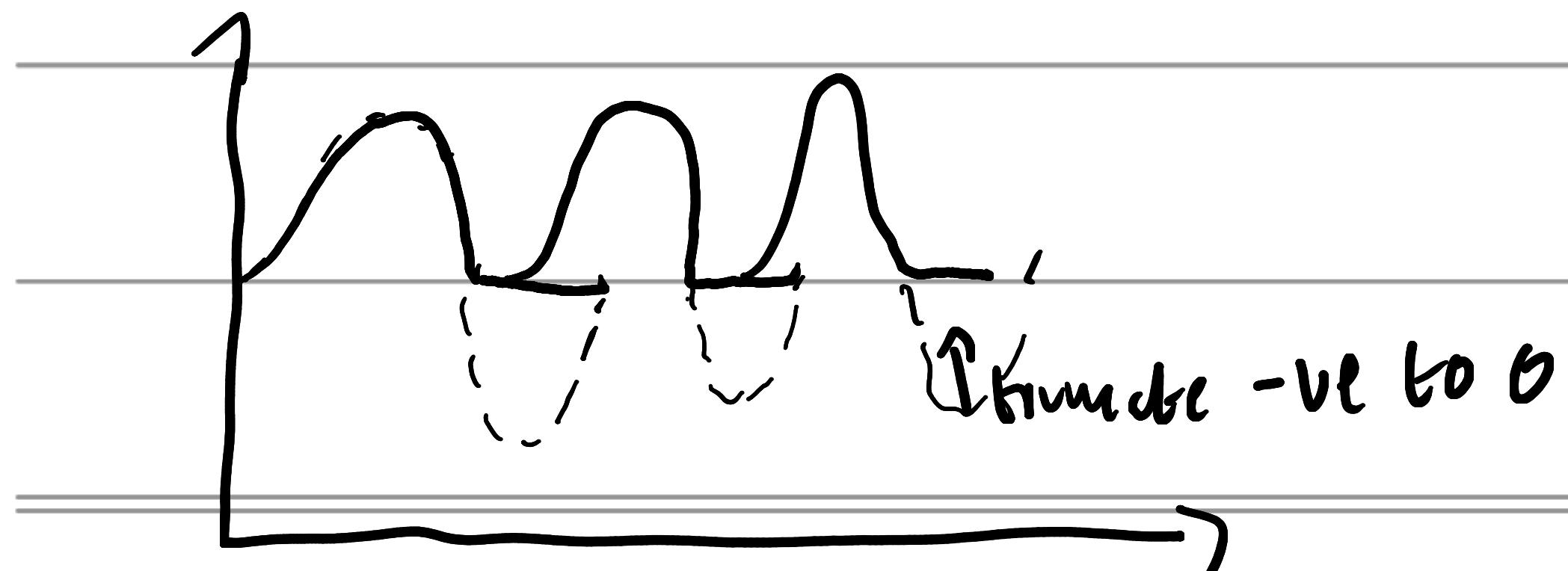
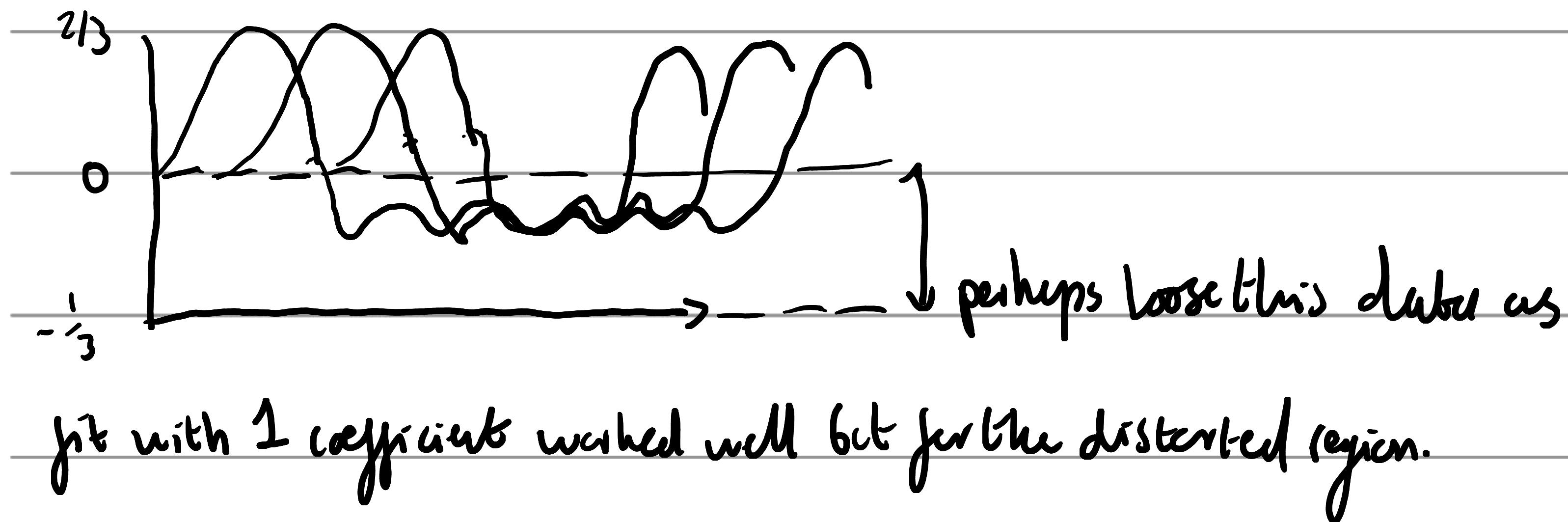
$$y = A \sin \frac{2\pi}{\kappa} \theta$$



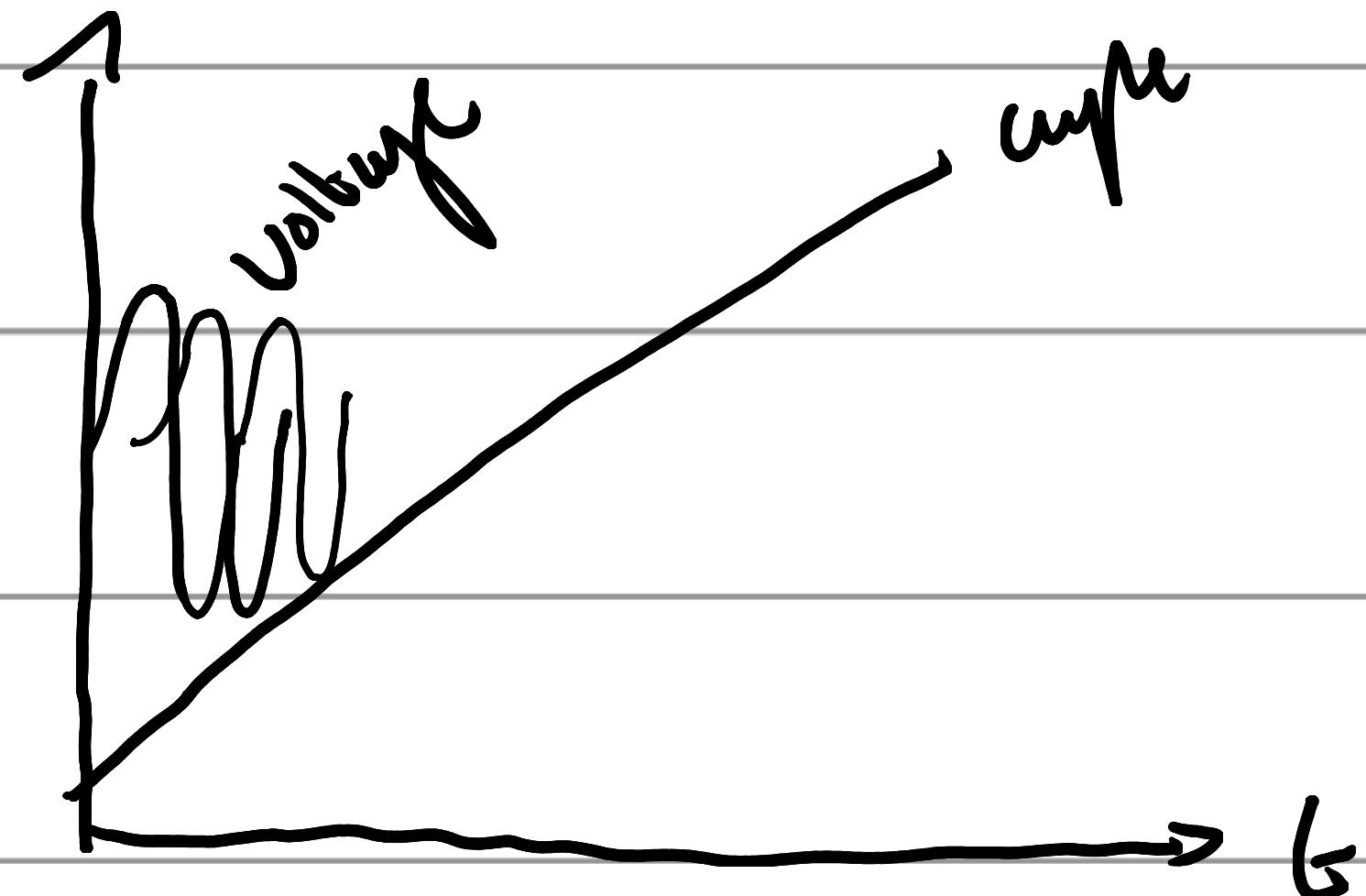
so each needs 7 periods  
within each  $360^\circ$  mech  
 $2\pi^\circ$  mech

## Model fitting results:

So the Fourier idea didn't pan out so well, fit is best with a single coefficient. Perhaps adding a lot of coefficients would eventually work but the single coefficient 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 filter for the fit data to exclude -ve results:

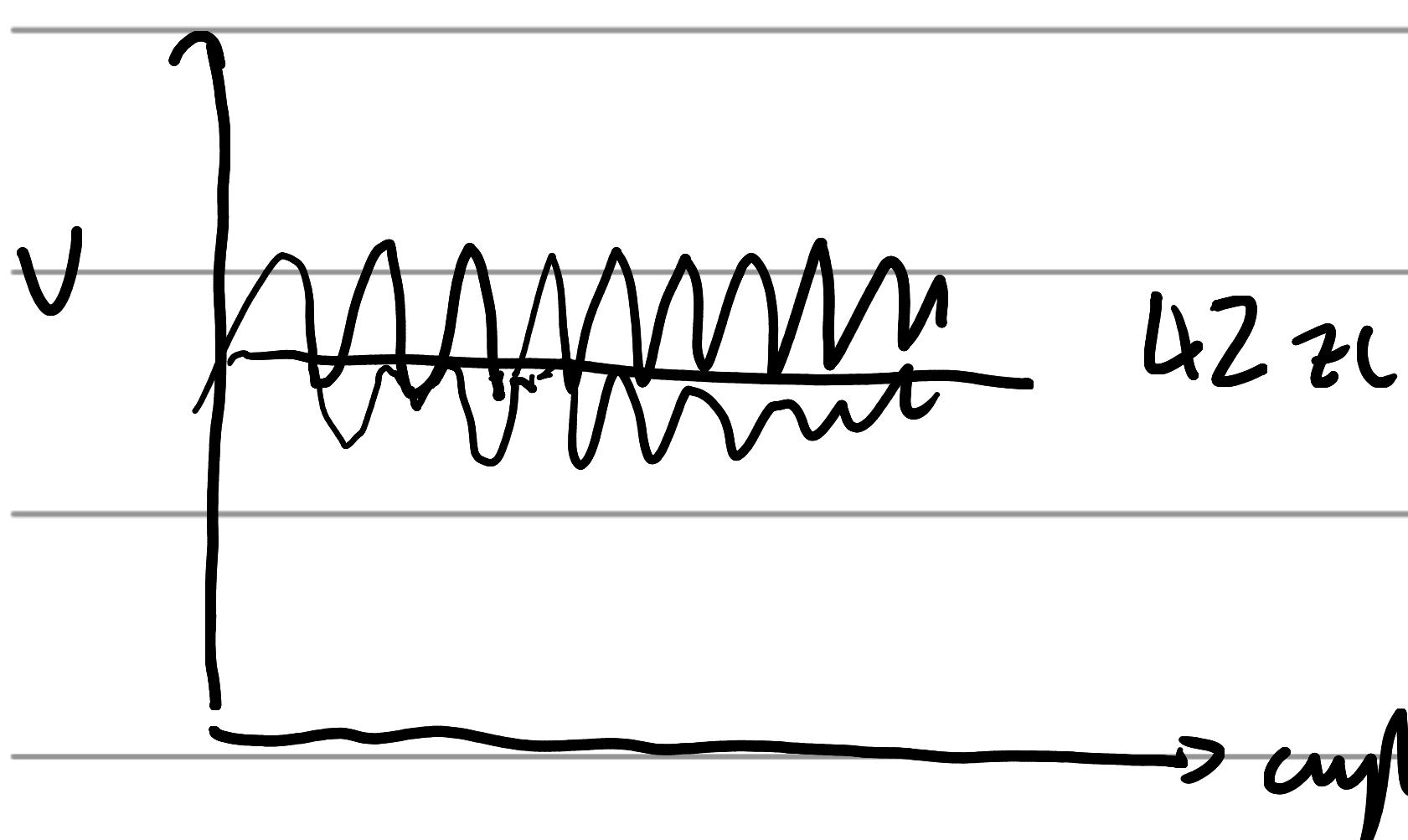


Another question is if switching points are really a function of speed.



I agree that as a function

of time, the frequency would  
change



But if we have it in

terms of cycles depth then

cycles would be swept through would inverse the voltage frequency  
per cycle sweep might not.

Next to try would be remove the -ve data, or where all the phase ABCC  
are near zero. Filter this data & attempt to fit the curves.

16/11/23

(chopped off all -ve (phase  $\chi_i - V_n$ ) values. Fit yields

$239.96 \pm 1.0$  phase current displacement  $\ell - 27.17 \pm 1.03^\circ$

angular-shape displacement. Data was from a counter clockwise  
run sept-x-ccw, phase current of  $\sim 240$  mches sense as

fit was against CW model,  $\ell$  a phase current displacement

of 240 swaps the order from black, yellow, red (cw)  
 $a \quad b \quad c$

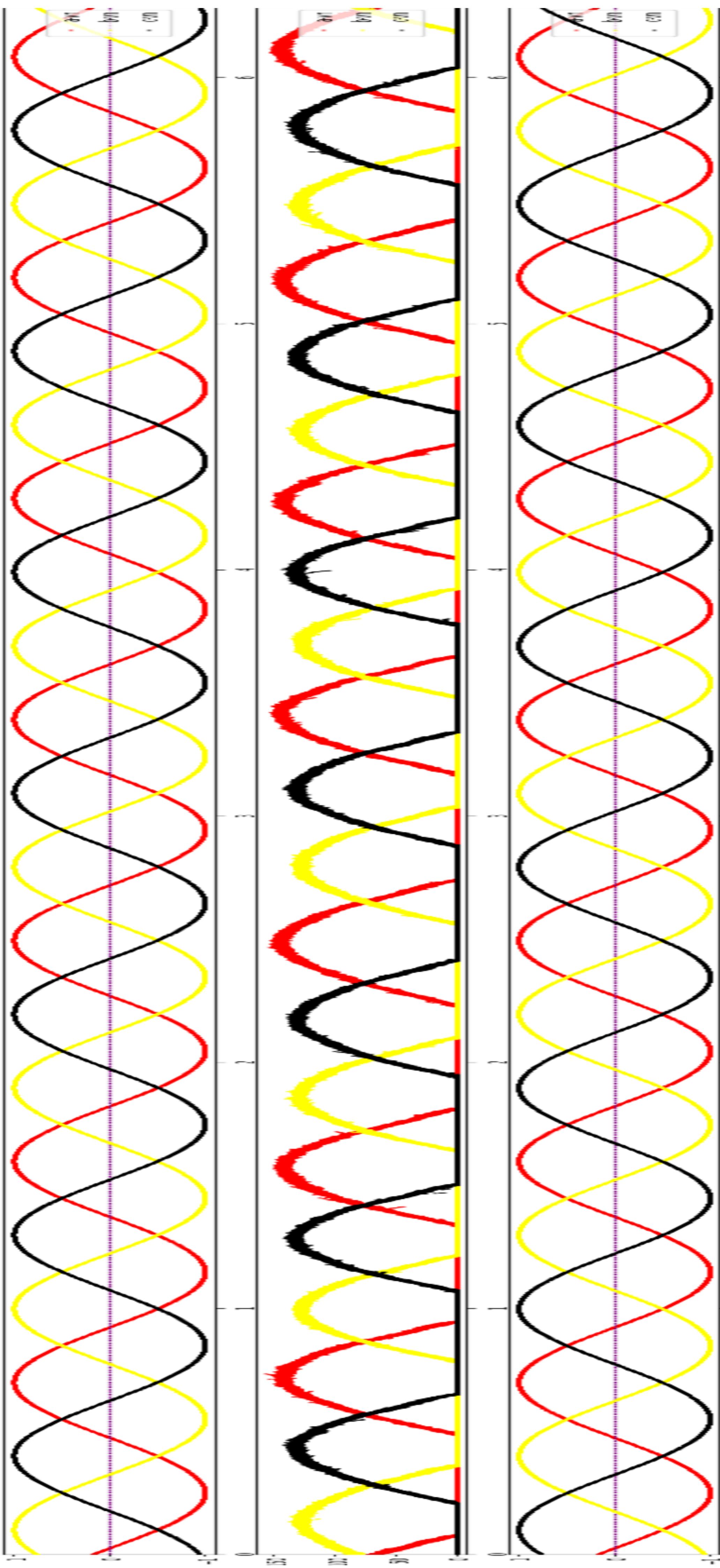
to red, yellow, black (ccw) while still maintaining a  $120^\circ$

separation

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

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

Parameters:  
alpha [type=1111111111111111]  
beta [type=1111111111111111]



Tried sept-2-cw & sept-4-cw

$$d-d = -2.4 \pm 1$$

$$\downarrow$$
$$d-d = -2.4 \pm 1$$

$$p-c-d = 137.1 \pm 0.9$$

$$p-c-d = 137.1 \pm 0.8$$

hm ... this makes much less sense, each cw run is consistent

& quite far from  $120^\circ$ , angular displacement disagree for cw & ccw!

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

cw pattern

→ direction  
in graph

A B C

R, Y, B

ccw pattern

← direction  
in graph

A C B

R B Y

← direction  
in graph

A C B

R, B, Y

→ direction  
in graph

A B C

R Y B

See if cycle repeats

clearly in point

and cheap start! stick 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 dd disk. The combined reports attempted had an average between them. Could try & extract state maps generated from the h5mt files

Name disk is a M.2 NVMe 2280 (SIE 3.0) <sup>(ben)</sup> x4

File under S10. Ordered an adaptor.

M key S

28/11/23

Recovered data successfully.

Things to do 1 → Implement FSC assuming analysis 1 is correct, 2 → export binary spike train data.

lets do 2) :

- Reinstall project dir on new laptop.
- update project packages.
- re-run analysis - all works
- Need to adapt code to export a JSON file of the ZC data (binary sprite brain)
- New mem / new scd

process is take new-mean... get ordered 1kv list,  
for each channel filters 1kv list of rising/falling edges & store  
in 3 lists 2 for each channel, convert 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  
 $\begin{pmatrix} \text{channel-name, value} \\ +1.0/-1.0\% \end{pmatrix}, \text{cycle value}$ ) do this  
for the first, midpoint & second. (emit these for all

channels & gotten & order. At this point we have not just ZC values but max & min values. Finally we can process the filtered list & collect a npy of data  $\{a_{\text{yles}}:[], a:[], f:[], c:[]\}$  & 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 Z methods.

26/11/23

Before I try & fit the covariance Kalman condition with a model I need to revisit work yesterday.

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

Works ok by reversing cw duba to form ccw duba. Fit with  $240^\circ$  initial is very close  $239.99$  for both if  $120^\circ$  initial we get  $137.13^\circ$  for both.

Now reverse analysis for all these runs & fit use the raw kulman voltage duba merge & flip!

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

27/11/23

Main files analysis - raw - recursive.py

Other main analysis analysis - zc - recursive

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

Reports should be able to generate a file with  
a chosen name (single) (title, file-name, file-pattern  
, run-ids)

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

Tested to work with 2 args, working with 3 args!

Tested & download zc & run fitting.

Next need to check stube map as in ccw rising

should be seen as falling edges vice versa

Counter-clockwise rotation (-ve)						clockwise rotation (+ve)							
	5	4	3	2	1	0		0	1	2	3	4	5
ESC	C	C	B	B	A	A	IN	A	A	B	B	C	C
IN	B	A	A	C	C	B	SD	B	C	C	A	A	B
SD	X	X	C	X	B	X	RISING	C	X	A	X	B	X
RISING	X	B	X	A	X	C	PULLING	X	B	X	C	X	A
PULLING													

This is expected to trigger this state

Rising / Falling indicates the NEXT expected edge.

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

For each direction there are 6 total steps

Diagram illustrating two sequences of steps:

- CCW Sequence:** 5 steps. Phase detected at the start of each step. The sequence is: A → B → C → A → B → C. The steps are labeled as "fall" (downward arrows) or "rise" (upward arrows). The sequence is labeled "ccw seq".
- CW Sequence:** 6 steps. Phase detected at the start of each step. The sequence is: C → B → A → C → B → A. The steps are labeled as "rise" (upward arrows) or "fall" (downward arrows). The sequence is labeled "cw seq".

CW checks out.

100

$$A \uparrow \Rightarrow 4$$

But we shall

cl  $\rightarrow$  s

flip rising falling  
decelerates.

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  
detector always goes in  
CW direction & thus sees  
 $A \uparrow$  as  $A \downarrow$ .

### 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

- $A \uparrow \rightarrow 1$  2
- $A \downarrow \rightarrow 4$  5
- $B \uparrow \rightarrow 3$  4
- $B \downarrow \rightarrow 0$  1
- $C \uparrow \rightarrow 5$  6
- $C \downarrow \rightarrow 2$  3

Re check ccw post analysis  
as on cubics Translated (flipped) (strike up) we get should have

$A \uparrow \dots A \downarrow$

0

1

$A \downarrow \dots A \uparrow$

3

2

we need

	ccw
$A \uparrow$	3
$A \downarrow$	0
$B \uparrow$	5
$B \downarrow$	2
$C \uparrow$	1
$C \downarrow$	4

values in map logic

2

5

4

1

0

3

ult veget

$A \uparrow$

$A \downarrow$

$B \uparrow$

$B \downarrow$

$C \uparrow$

$C \downarrow$

$F$

$F$

$S$

$S$ </p

Counter-clockwise rotation (-ve)

	5	4	3	2	1	0
ESC						
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

S U 3 2 1 0

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

check above

A ↑      looking for A↑ in const up  
 A ↓      A↑ in cw state  
 B ↑      B↓ in const state  
 B ↓      B↑ in ccw state  
 C ↑      C↓ in const state  
 C ↓      C↑ in ccw state

4	✓
1	✓
0	✓
3	✓
2	✓
5	✓

checks out... Actually doesn't... everything to older code.

Added dark theme.

29/11/22

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

CSN      MOSI      MISO      CLK

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

[327] are being used as inputs (optos), (GND & 3.3v also for encoder). We need another encoder (in picture) & 2x set

of Pwm pins, from two Pwm modules (half cycle offset).

MISO2 S0S12 SPI pins need to be soldered not deny that!

try random other pins & benchmark.

Flex 1.0 1

1 0 1 3 0  
1 0 3 7

A B C

Pwm modules 1 IN (2, 9, 8)

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

analog write frequency

sm0 clock INIT-SCL(2) AUX\_CLK (only for sm's)

FlexPwm EXT-CLK

Pwm-EXT-SYNC

SYNC

# Rewiring PWM modules

2.0 4, 33

2.1 5 4, 5, 6 looks good

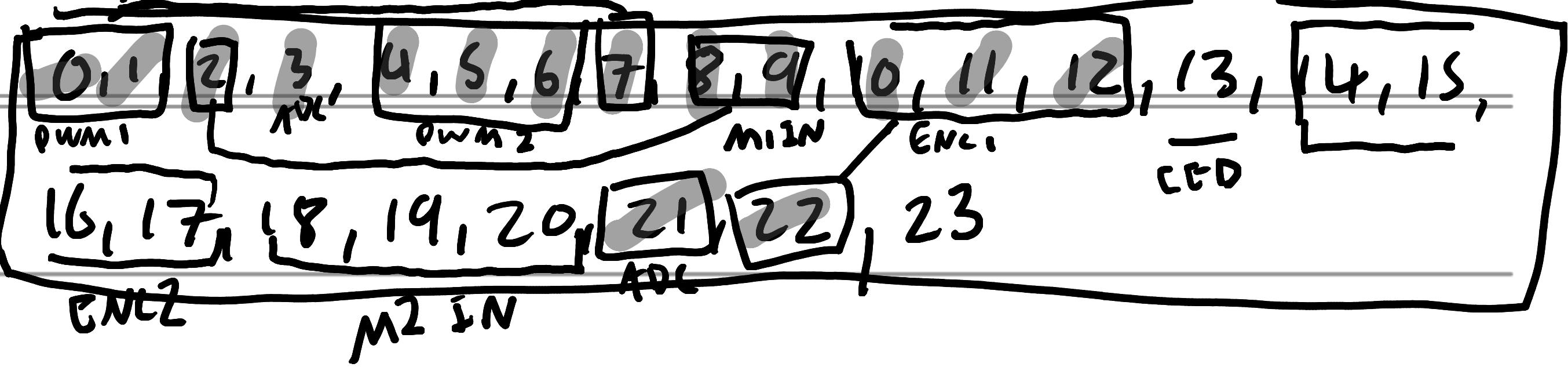
2.2 6, 9

3.0 1 28, 29

4.0 6 22

4.0 1 23

4.0 2 2, 3



Final pinout

MOTOR1	Flex SD PSN	MOTOR2	Flex SD PIN
Flex PWM	1.0 1	Flex PWM	2.0 0 1 4
	1.0 1 2 0		2.0 1 2 5
	1.0 3 3 7		2.0 2 3 6

MOTOR1 IN PIN

MOTOR2 IN PIN

1 2

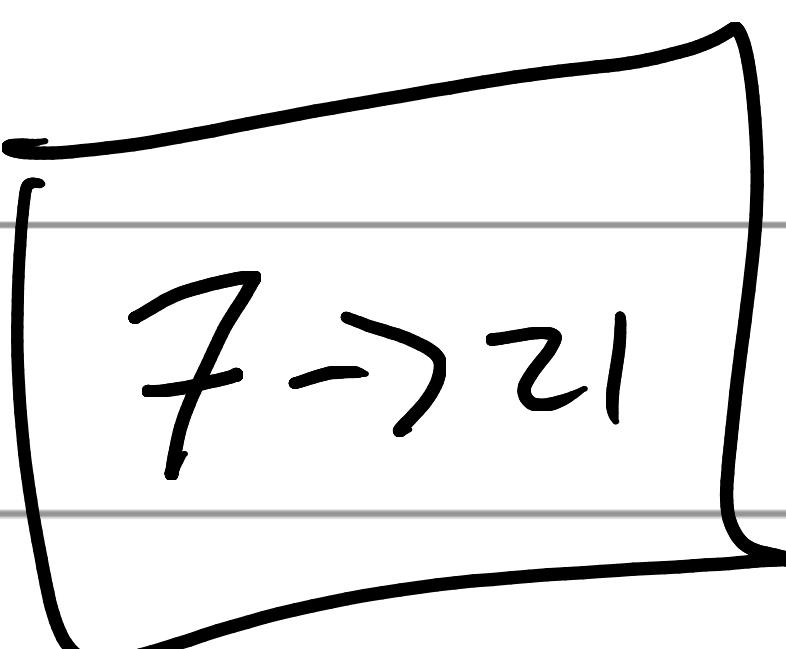
1 18

2 9

2 19

3 8

3 20



Encoder 1

SN 10

MOTOR1

Flex

SD

PSN

Encoder 2 SN

out

14

Flex PWM

MOSI

II

MOSI

15

MISO

I

MISO

16

1.0

I

MISO

I

1.0

2

1.0

0

1.0

22

1.0

3

1.0

7

ADC sym 3 l \* 21

change this  
in driver!

commutative 6 step

# Basic ESC (BLDC)

Vars

commutation

cw

ccw

State map  $\begin{bmatrix} [0, 1, 2, 3] \\ [5, 4, 3, 2] \end{bmatrix}$

State config  $[0: [IN, SD], 1: [SN, SD] \dots]$

IN/SD pins

thrust setting

Encoder pins

Direction

last encoder Value

Routines

Setup  $\rightarrow$  setup PWM sync/grey Set-commutation table  
/encoder  
started = false

readusb()  $\rightarrow$  profile

loop  $\rightarrow$  if !started  
should  
be first  
    → wait for first signal  
    started = true  
    if started  
        → read encoder  
        → has state changed  
        → yes set state  
        → read hose profile  $\rightarrow$  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 (mrsignt).

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

Syn between module 0 & 1 is done by msk. 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 synchronized per period.

(connect via XBAR/crossbar)

61.3.2. "Functional Mode"

11/2/23

Programming esc, IMU created, folders created  
for calibration state map.

logic started = false?  
read profile

Started = true

state != old state

Engage combination

read profile

MOTOR 1	Flex	SD PSN	MOTOR 1	IN PIN
Flex PWM	1.0	1	1	2
	1.1	2	0	2
	1.3	3	7	3
				8

Did first draft of the ESC.

would be nice  
to have line number  
comments

→ check

- Need to reformat 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 some com scale input

2/2/23

Refactored PS42 into a nice library so you can easily write PS4 to serial port code will allow easy network extensions for additional outputs for charting etc. Need to rename it something good. Dronium.js? HID? BIN DIR! Index files!

Chyed engine communication state function to turn everything off first.

Need to put in some sort of protection so that  
if high speed cannot reverse direction... perhaps  
if direction is 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.

(PP Kalman lib (multi-core for analysis))

Analyses cleanup

Separate into submodels Beersy 10 - block - calibration  
- block - driver - 6step - encoder  
- cumulative sine

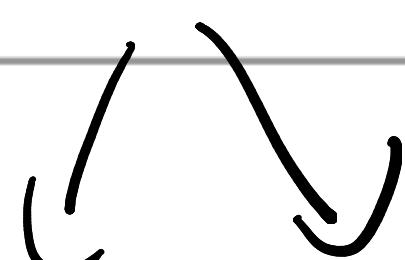
TODO

encoder

full range

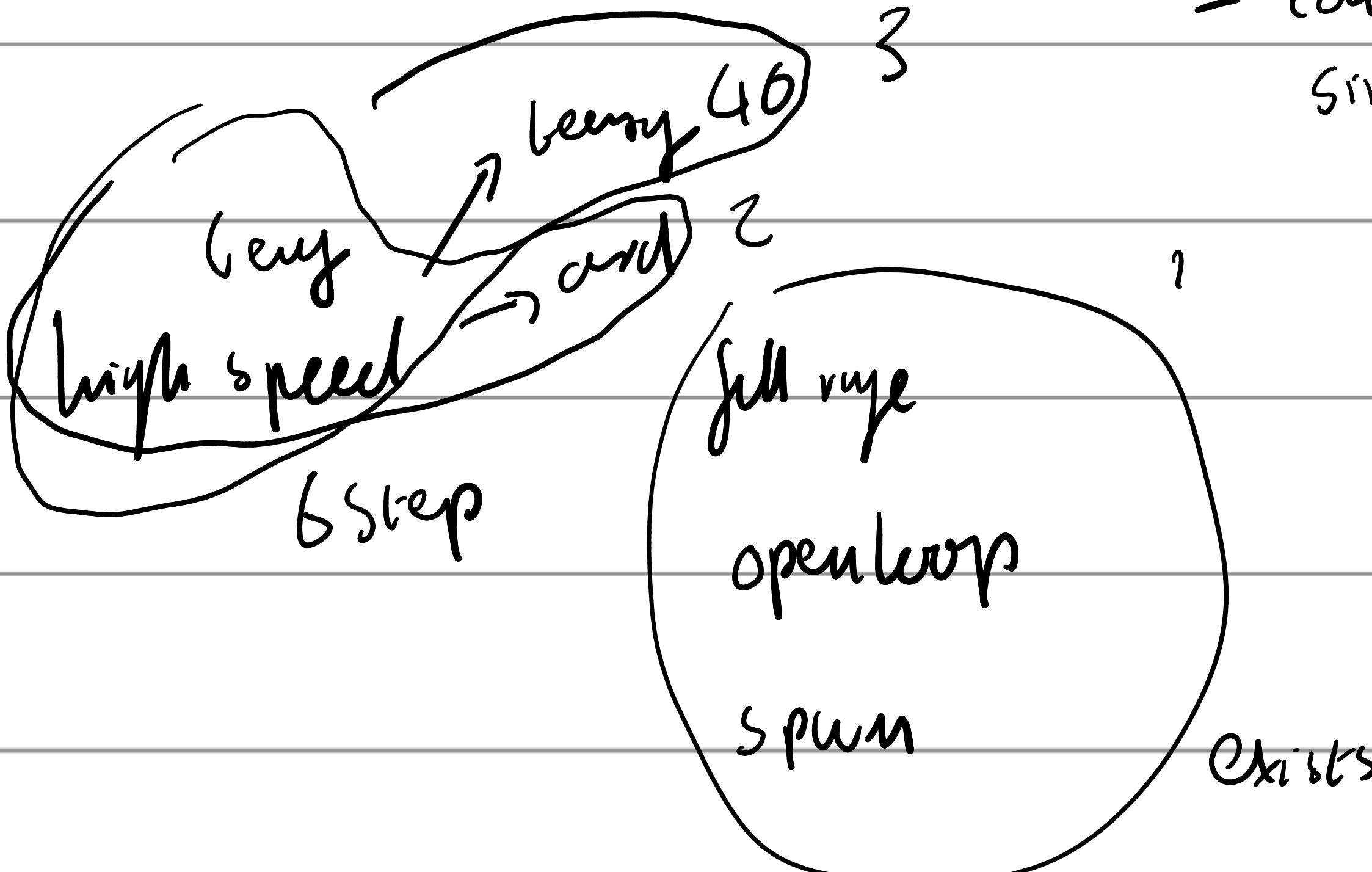
closed

loop



Sine wave 6step

SPWM



Geny - high speed (Terry cont) 6step

encoder - full range (Terry) 6step (SPWM)

Geny - high speed - underactuated (Terry cont) → current  
encoder - full range - underactuated. → closed loop <sup>SM1 QD</sup>

312123

Adding prints to prevent the state valve 1  
angle in debug mode. Angle changes when the  
direction changes! This is bad. Need to try this  
with a real encoder. Flush hardware check.

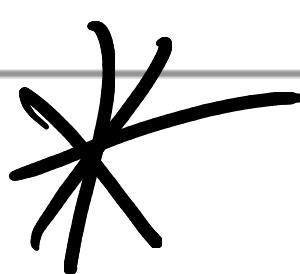
changing ADC sync pin to 21 from 7, updated  
readme & logs.

Encoder setup was missing

lesson learnt... Need to check 600MHz does  
not work over under clocked.

Picked 'Furtek' capile type

Some times need to touch encoder wires to



get them stable ! Please check in DEBUG mode  
before destroying the hardware !

Fault mode → States must be consecutive

(less than -) allowed in cw direction in case you  
cross a boundary of states abra 0 1 2 1 2 3 is  
allowed.

Enforce Net validation (apart from # style events)

... how? last/last stable compare? uniqueness?

Timing GPT 1sec timer should print stable, thrust,  
direction. In number + = 1 for each motor loop, then  
when GPT timer fires print opslsec. Keep this as delivery  
mode... limited mode only opslseconds.

(reduces safe direction reverses.)

( $a, b$ )

watch for the loop

0, 1, 2, 1, 2, 3, 4, 3, 4, 5, 0, 5, 0

+1   +1   -1   +1   +1   +1   -1   +1   +1   (-s)  $\xrightarrow{+s}$  -s

$$l \bmod 6 = 1 \quad s \bmod 6 = 5$$

$$2 \bmod 6 = 2$$

$$-5 \bmod 6 = 1$$

$(a, b)$  pairs.

$$-1 \bmod 6 = 5$$

$$0 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 4 \quad 3 \quad 4 \quad 5 \quad 0 \quad 5 \quad 0$$

$$+1, +1, -1, +1, +1, +1, -1, +1, +1, -5, +5, -5$$

FWD      BKWD

$I_S(s_{\text{tube}} = 0)$  allows  $(+1, +5)$       Allow wrong

If ( state = 5 ) allow ( -5, -1 ) distribution n times.

else if  $\text{abs}(6-a) == 1$  [  $y = (x \gg 31)$   
 $\text{return } (\pi^1 y) - y;$  or  $x < 1$   $x = ~x + 1$

If Fault resets motor & block outer loop with early return.

From experimental data

6  
cw

cw [0, 1, 2, 3, 4, 5] ccw

ccw [5, 4, 3, 2, 1, 0]

(in validate lib) but can oscillate 0, 1, 0, 1, 2

This is OK. Maybe allow -1 / +1 (& mod +5, -5 etc)

but stored the # of times we went the wrong way

cstable

→ loop

Next state = end cycle // % 6

If change

→ If many in expected direction

cabine

→ move wrong way (incorrect direction) cabine  
... done this multiple times? → fail

Fault should light status LED Fixed on.

close this (-o  
'didn't' make progress  
rather than went  
backwards)

where to put cli/sei interruption blockers? Should  
prioritize updating user state over user input / so read cycle  
& update state logic should be wrapped in real user  
profile outside.

### STALL

Timer for lack of progress (100Hz minimum action) Need to  
work this out as a min speed.

check current direction

The check wrong direction the wrong

STATE\_IDX STATE\_cw STATE\_ccw NEXT\_cw NEXT\_ccw

0	0	5	1	4
---	---	---	---	---

1	1	4	2	3
---	---	---	---	---

2	2	3	3	2
---	---	---	---	---

3	3	2	4	1
---	---	---	---	---

4	4	1	5	0
---	---	---	---	---

5	5	0	0	5
---	---	---	---	---

Fault states "WRONG-DIRECTION"

"SKIPPED-STEPS"

"STALL"

NEXT-CW      NEXT-CCW

STATE-VALIDATOR-STRUCT [ 1 , 5 ]

[ 2 , 0 ]

[ 3 , 1 ]

[ 4 , 2 ]

[ 5 , 3 ]

[ 0 , 4 ]

Next-state = StateNext(direction)(angle)

State-validation = STATE-VALIDATOR.STRUCT[Current-state]

Expected = State-validation[direction]

If STATE-CHANGED

→ If Expected = Next-state  
        contine wrong way ctr=0

Else if Next=Expected [other direction]

in anyway way ctr, if ctr > max FAULT WRONG DIRECTION

Else → FAULT-SKIPPED-STEPS

6/2/23

Prepare for Open data conference.

→ Timing

→ load data

→ website

→ Analytics (graph layout)

1) Analytics plot layout. Try reduce from 58,000 -> 40,000.

2) Timing plan ... only print every so often e.g. 1s

Interval Timer

Get some experimental data for loop

frequency / sample rate...

$$\frac{60 \cdot S_R}{z^{12}} = \frac{60 \cdot 1,300,000}{z^{12}}$$

$$= 19,042 \text{ RPM!}$$
