$Q_{(k,k+1)} = Q(dt)$ $[4\times4]$ Variance matrix at step $k \to k+1$

$F_{(k,k+1)} = F(dt)$ $[4\times4]$ state transition matrix

take last known state $X_k = \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \\ \dots \\ \vec{x} \end{pmatrix}$ & project it ahead in

time using the transition matrix.

Ignore as this comes from the environment

$$X_{(k+1)} = F_{(k,k+1)} \cdot X_{(k)} + \overbrace{B \cdot W_{(k)}}$$

$[4\times1]$ $[4\times4]$ $[4\times1]$ $[4\times4]$ $[4\times1]$

Project the error covariance ahead note $P_{(k,k+1)}$ is the covariance

matrix at step $k \to k+1$, $P_{(k-1,k)}$ same for $k-1 \to k$.

$$P_{(k,k+1)} = F_{(k,k+1)} \cdot P_{(-k,k)} \cdot F_{(k,k+1)}^{T} + Q_{(k,k+1)}$$

$[4\times4]$ $[4\times4]$ $[4\times4]$ $[4\times4]$ $[4\times4]$

? not $(k-1,k)$ seems wrong

Calculate uncertainty in measurement + uncertainty in estimate:

where H is the measurement matrix & R is the measurement uncertainty

$$S = \underset{[1\times4]}{H} \cdot \underset{(4\times4)}{P_{(k,k+1)}} \cdot \underset{[4\times1]}{H^T} + \underset{[1\times1]}{R}$$

$[1\times4]$ $[1\times4]$

$$[1\times1] + [1\times1] = [1\times1]$$

Calculate uncertainty in estimate

$$C = P_{(k,k+1)} \cdot H^T$$

$[4\times1]$ $[4\times4]$ $[4\times1]$

Calculate the gain

$$K = C \cdot S^{-1}$$

$[4\times1]$ $[4\times1]$ $[1\times1]$

Update measurement Z

$Z = x_c.\text{reshape}(H.\text{shape}[0], 1)$ $[1\times1]$ in 1d

Calculate innovation

$$Y = Z - (H \cdot X_{(n+1)})$$

$$[1\times1] - ([1\times4][4\times1]) = [1,1]$$

Calculate the filtered state

$$\underset{[4\times1]}{X_{(n+1)}} = \underset{[4\times1]}{X_{(n+1)}} + \underset{[1\times1]}{K \cdot Y}$$

kalm

Assign old to new state

$$\underset{[4\times1]}{X_{(n)}} = \underset{[4\times1]}{X_{(n+1)}} + \underset{[4\times1]}{(K \cdot Y)}$$

$$[4\times1] \quad [4\times1]$$

Update error covariance

$$\underset{[4\times4]}{P_{(k,k)}} = \underset{[4\times4]}{(1 - K \cdot H)} \cdot \underset{[4\times4]}{P_{(n,n-1)}}$$

$$[4\times4] \quad [4\times1][1\times4]$$

$$[4\times4] \; [4\times4] = [4\times4]$$

K+1 then loop for next stabe.

So need a simple 1D non generic version which can be coded specially.

$$Q(dt) = \begin{bmatrix} \frac{dt^7}{286} & \frac{dt^6}{72} & \frac{dt^5}{30} & \frac{dt^4}{24} \\ \frac{dt^6}{72} & \frac{dt^5}{20} & \frac{dt^4}{8} & \frac{dt^3}{6} \\ \frac{dt^5}{30} & \frac{dt^4}{8} & \frac{dt^3}{3} & \frac{dt^2}{2} \\ \frac{dt^4}{24} & \frac{dt^3}{6} & \frac{dt^2}{2} & dt \end{bmatrix}$$

$\sigma^2 = V$

$$P(dt) = \begin{bmatrix} \sigma_\Theta^2 & \sigma_\Theta^2/dt & \sigma_\Theta^2/dt^2 & 0 \\ \sigma_\Theta^2/dt & 2\sigma_\Theta^2/dt^2 & 3\sigma_\Theta^2/dt^3 & 5\sigma_J^2 dt^2/6 \\ \sigma_\Theta^2/dt^2 & 3\sigma_\Theta^2/dt^3 & 6\sigma_\Theta^2/dt^4 & \sigma_J^2 dt \\ 0 & 5\sigma_J^2 dt^2/6 & \sigma_J^2 dt & \sigma_J^2 \end{bmatrix}$$

$$F(dt) = \begin{bmatrix} 1 & dt & dt^2/2 & dt^3/6 \\ 0 & 1 & dt & dt^2/2 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To simplify this we could just use the components of P, F & Q as e.g. $P_{ij}(dt)$ & sub them in later.

Step by step: Log k = k

$$[F_{ij}]_\pi = F(dt_\pi) = \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \\ F_{41} & F_{42} & F_{43} & F_{44} \end{bmatrix}$$

$$[Q_{ij}]_\pi = Q_\pi = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{bmatrix}$$

same with $[P_{ij}]_\pi$

$[4\times1]$

We start with state $X_{(\pi)} = [X_{11}^{\pi} \; X_{12}^{\pi} \; X_{13}^{\pi} \; X_{14}^{\pi}]$

$[4\times1]$ $\quad$ $[4\times4]$ $[4\times1]$

1) Propuls $X_{(\pi+1)} = F X_{(\pi)}$

$$\Rightarrow [X_{11}^{\pi+1}, X_{12}^{\pi+1}, X_{13}^{\pi+1}, X_{14}^{\pi+1}] \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \\ F_{41} & F_{42} & F_{43} & F_{44} \end{bmatrix}^T$$

$$\Rightarrow \begin{bmatrix} X_{11}^{\pi+1} F_{11}^{dt} + X_{12}^{\pi+1} F_{21}^{dt} + X_{13}^{\pi+1} F_{31}^{dt} + X_{14}^{\pi+1} F_{41} \\ X_{11}^{\pi+1} F_{12}^{dt} + X_{12}^{\pi+1} F_{22}^{dt} + X_{13}^{\pi+1} F_{32}^{dt} + X_{14}^{\pi+1} F_{42} \\ X_{11}^{\pi+1} F_{13}^{dt} + X_{12}^{\pi+1} F_{23}^{dt} + X_{13}^{\pi+1} F_{33}^{dt} + X_{14}^{\pi+1} F_{43} \\ X_{11}^{\pi+1} F_{14}^{dt} + X_{12}^{\pi+1} F_{24}^{dt} + X_{13}^{\pi+1} F_{34}^{dt} + X_{14}^{\pi+1} F_{44} \end{bmatrix}_{[4\times1]}$$

$$F(dt_n)^T = \begin{bmatrix} F_{11} & F_{21} & F_{31} & F_{41} \\ F_{12} & F_{22} & F_{32} & F_{42} \\ F_{31} & F_{23} & F_{33} & F_{43} \\ F_{14} & F_{24} & F_{34} & F_{44} \end{bmatrix}$$

2) pryct error cover ahead.

$$P_{(k+1)} = F_{(n)} P_{(k-1)} F_{(n)}^T + Q_{(k)}$$

$$\begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \\ F_{41} & F_{42} & F_{43} & F_{44} \end{bmatrix} \begin{bmatrix} P_{11}^{k-1} & P_{12}^{k-1} & P_{13}^{k-1} & P_{14}^{k-1} \\ P_{21}^{k-1} & P_{22}^{k-1} & P_{23}^{k-1} & P_{24}^{k-1} \\ P_{31}^{k-1} & P_{32}^{k-1} & P_{33}^{k-1} & P_{34}^{k-1} \\ P_{41}^{k-1} & P_{42}^{k-1} & P_{43}^{k-1} & P_{44}^{k-1} \end{bmatrix} \begin{bmatrix} F_{11} & F_{21} & F_{31} & F_{41} \\ F_{12} & F_{22} & F_{32} & F_{42} \\ F_{31} & F_{23} & F_{33} & F_{43} \\ F_{14} & F_{24} & F_{34} & F_{44} \end{bmatrix} + \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{bmatrix}$$

Need a symbolic solver for this gonna be unwieldy!

2/4/23

Symp solver for kalman process.

3/14/23

Need dependency graph.

$P_{k+1} = (I - K_{k+1}) P_{k+1}$ * need for next loop

$X^{KALMAN}_{k+1} = X_{k+1} + (K Y)$ * final state

↳ also taken as final state.

Dependencies...

NO

$Y = Z - (H X_{k+1})$ ← last state (Kalmen?)

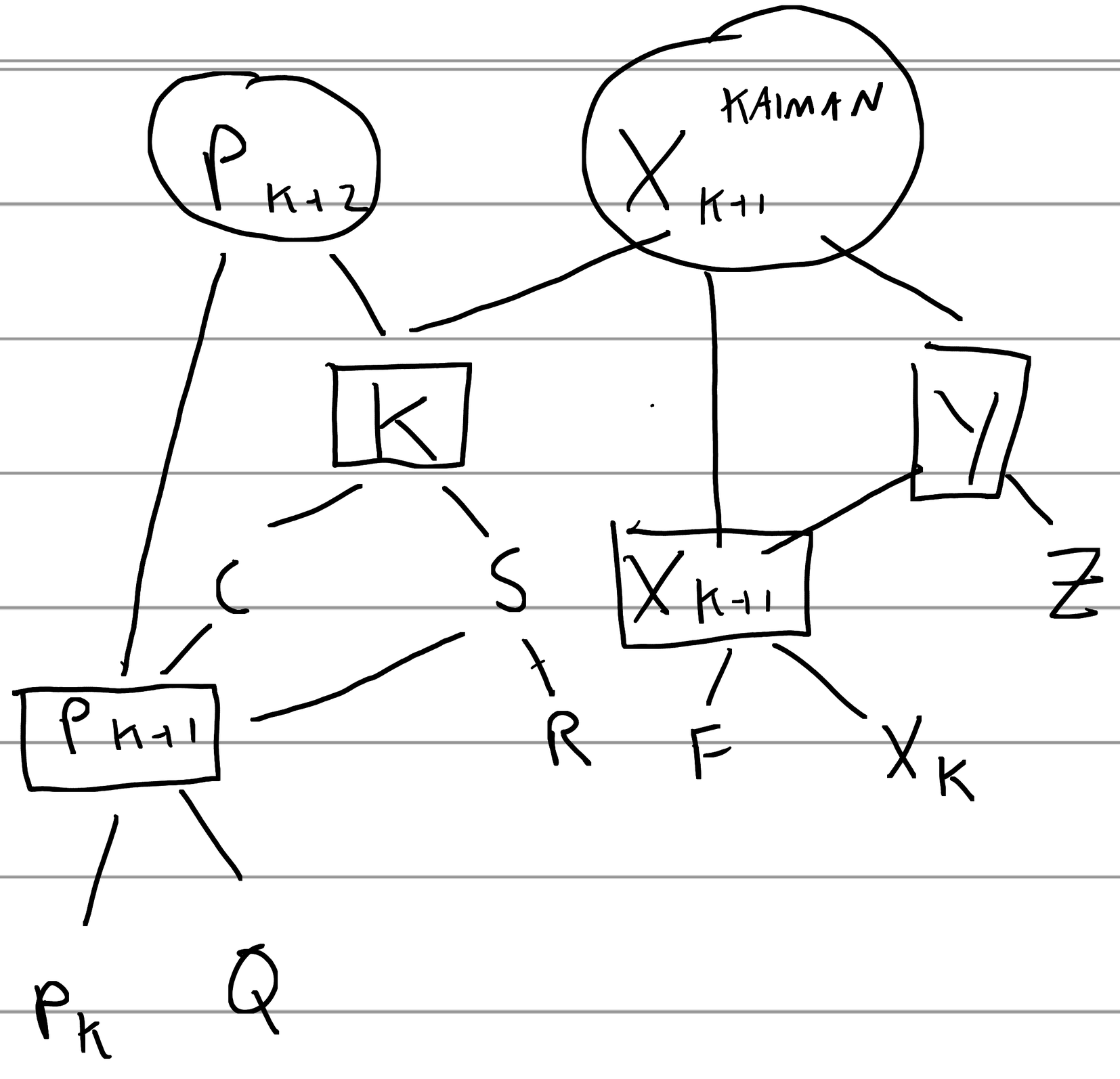
Z comes from new input ds
reshaped via H

$K = C S^{-1}$

$C = P_{k+1} H^T$ $S = H P_{k+1} H^T + R$

$P_{k+1} = F P F^T + Q$ $X_{k+1} = F X_k$

H omitted



Each cycle we need P_{k+2} & X_{k+1} > what common computation can we leverage?

Need P_{k+1} > K > X_{k+1} & Y then compute the "rest" in terms of them

KALMAN

$\hookrightarrow P_{(k+2)}$ > X^{KALMAN}_{k+1} > P_{k+1} > X_{k+1}

(P_{k+2}) (X^{KALMAN}_{k+1})

↓

Can generate error from this.

4/4/23

missing error Lucy stuff

10/4/23

first version of cpp... need option for absolute/relative time.

need python bindings. ✓

11/9/23

cleanup unnecessary files Kalman1D.cpp

params include time absolute/relative

clean up 0.0 unnecessary computations.

Cat test

rm stdout

check fpow / fmod operations

auto model optimizer. inputs computed outputs (symbols) easier shapes type defs

17/4/23

Cleaned up & harmonised pg & cython Kalman class.

added args for selective time

testing against old double pendulum code

cpp code works but Pykalman & Cykalman behave differently

based on input params.

Can get same smoothing but with different args. Cython version much

smaller alpha needed.

Might not matter but would be nice to have almost exactly the same

behaviour between Pykal & Cykal

Model for white noise acceleration might be computationally simpler.

Next rotary sensor

24/4/23

Todo optimise kalman loop.

Vm stat out prints

create ino without sensor find freq at which we are saplin.

$2^{12} = 4096$

rpm / 60 = Hz

Sample rate x 60 = max rpm

encoder res

High speed

260Hz

78rpm

1.3 Hz

33.3 rpm

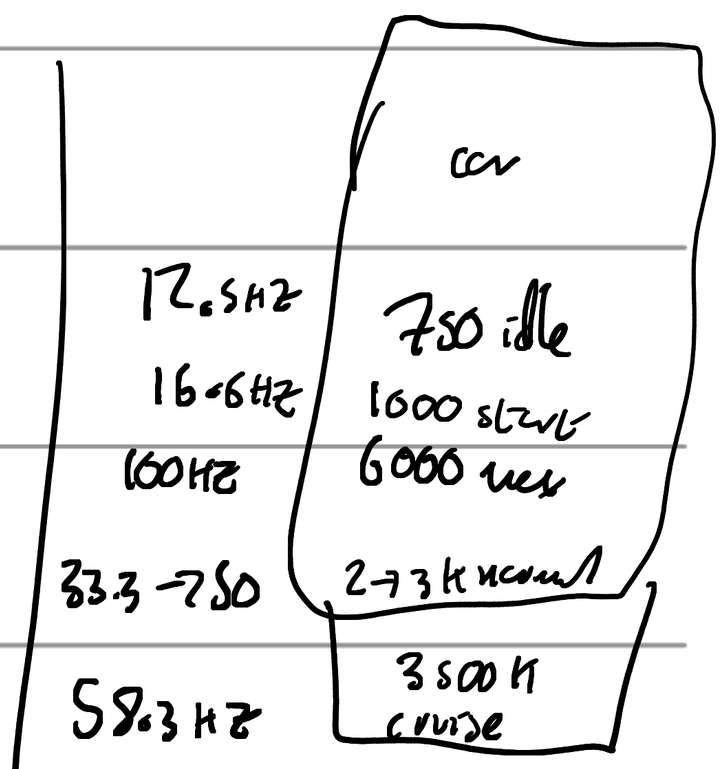
0.555 Hz

↓

1.1x

1.43

could it 1.5Hz



Sample rate needed = max rpm needed x enc res = max Hz needed x enc res

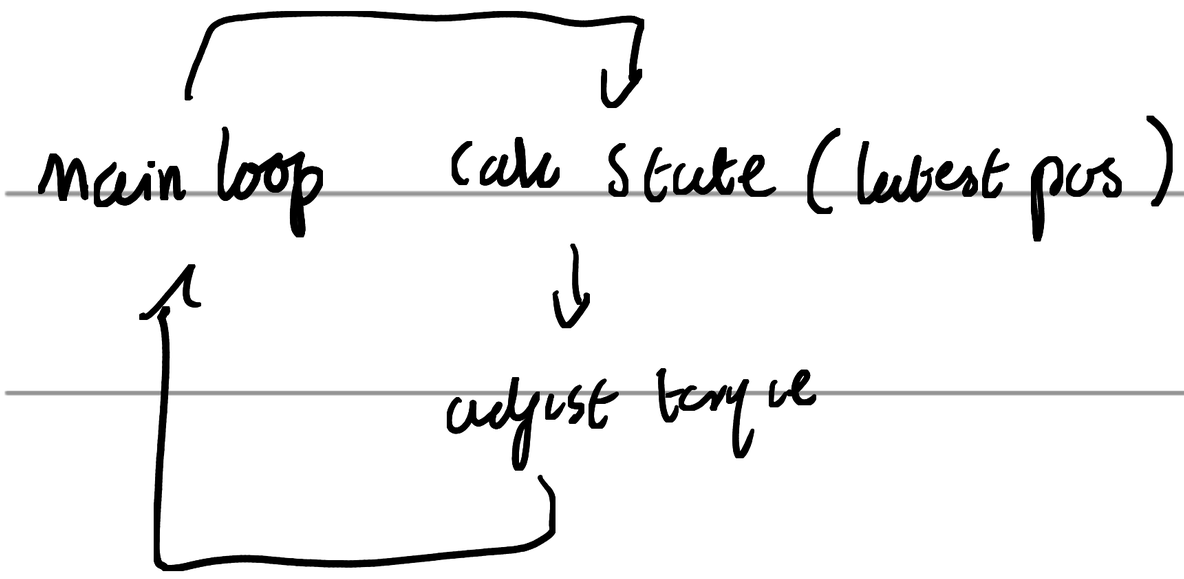
60

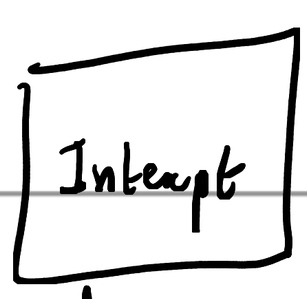
samp b (max Hz = 1.5) = 6144 Hz

Probably teensy 40 will be able to do more than a few 1000's kalman points per second.

In order to make sure we move the wave forward enough to not

disrupt motion need to seperate out the process of speed calc & enc saving





GPT?

every X mechrespeed,

Slight synchronize, do error/fault logic. [no stall protection]

what is X? well preferably its enough to ensure we update the wave

for each possible encoder position. what's that value?

Sample rate = rotations per second x enc res = $\frac{rpm}{}$ x enc res

66

lets say we define mid speed as 6000 rpm, 100 Hz so we need 40q 600 Hz

which leaves quite a bit from the 1of MHz o left loop time for doing measurements

$\frac{400,000}{1,100,000}$ = 0.37% so we have 62% of the time for speed calc.

optimising code abs leg results install hyperfine

hyperfine --warmup 3 "(cmd)"

best of 5 1.175 x̄ 1.174

1.155

1.21

1.23

1.11

POSE optimisation in (0.0)

1.15 x̄ 1.16

1.18 1% increase.

1.19

1.18

1.1

change to Kuepeh

remove get jacobians Kalman ~~state~~ vec X

euler ~~state~~ vec

covariance matrix P ✓.