algorithms for robotics

Michal CHOVANEC, PhD.

overview

- noise reduction : digital filters, Kalman filter
- control : PID, LQR, MPC
- motor controll : park, clark transformations, real time PID
- planning, decision making : image segmentation, reinforcement learning

digital filters

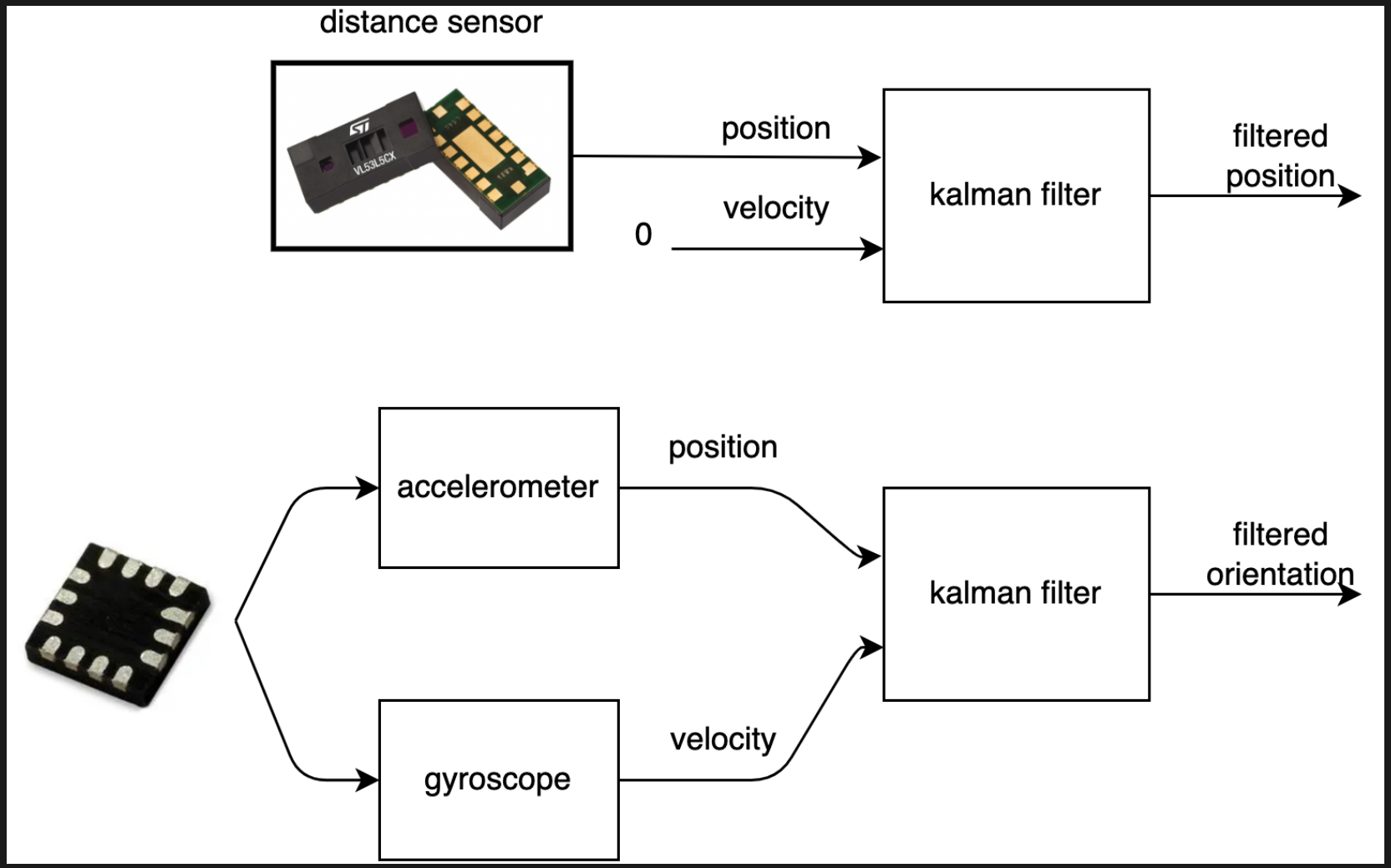
- low pass filter (1st order)

$$y_n = (1 - a)y_{n-1} + ax_n$$
$$a = \frac{\Delta t}{RC + \Delta t}$$

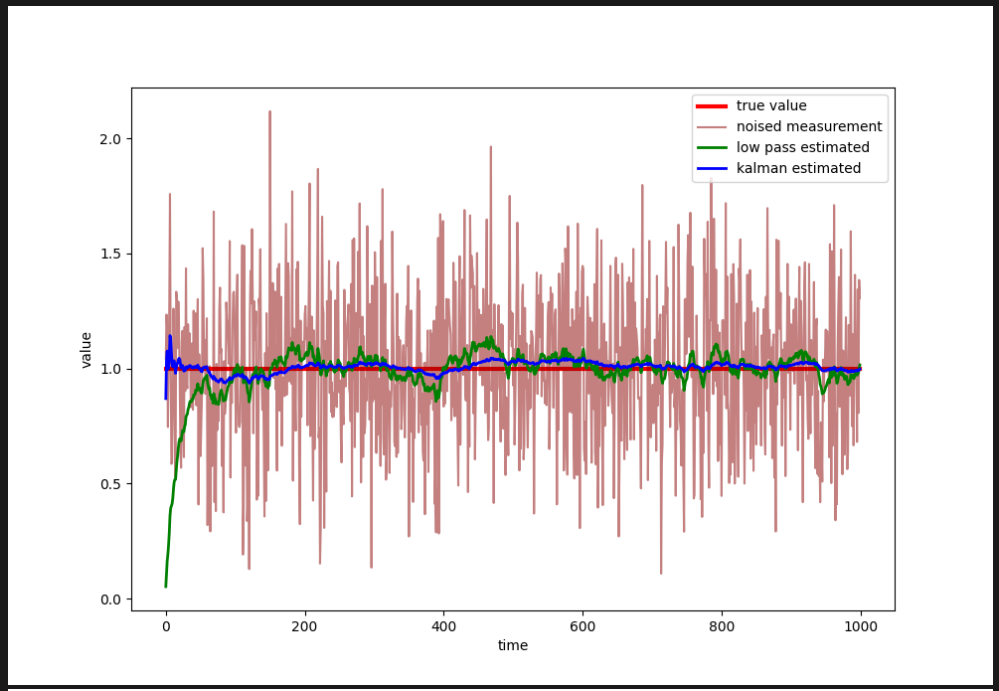
- resonant filter (2nd order)

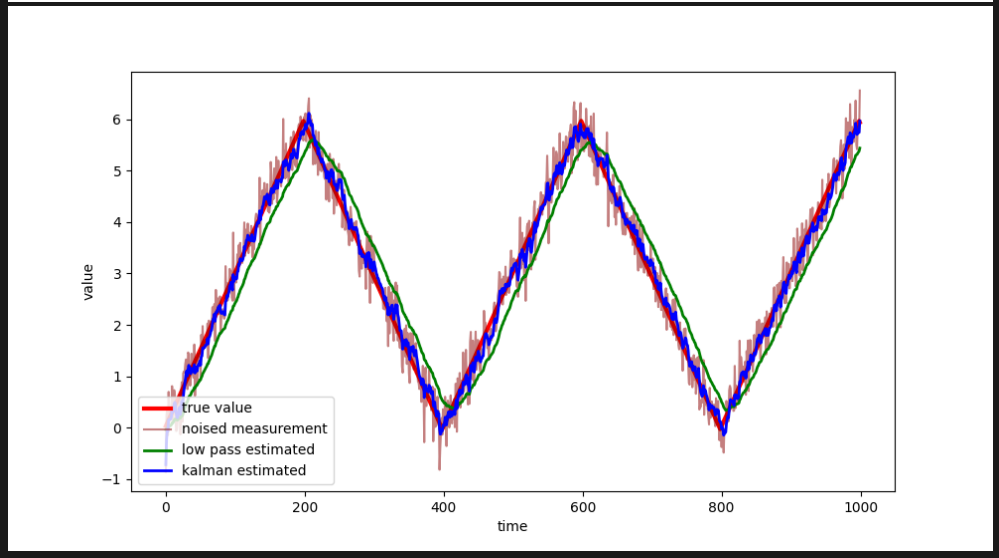
$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2}$$

digital filters - usage



digital filters - low pass vs Kalman





1D kalman is simple

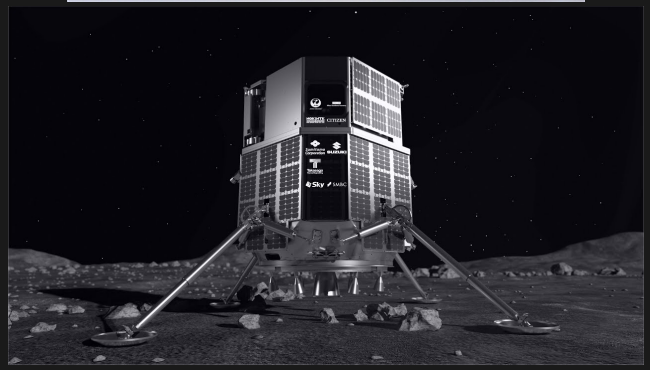
```
class KalmanFilter:

    def __init__(self, dims = 1):

        #initial guess
        self.x_hat = torch.zeros(dims)
        self.p     = 0.5*torch.ones(dims)

    '''
        z   - position measurement
        dz  - velocity measurement
        pz  - position measurement uncertainty
        pdz - velocity measurement uncertainty
    '''
    def step(self, z, dz, pz, pdz, dt = 1):
        #1, prediction
        #predict the state and uncertainty
        self.x_hat  = self.x_hat + dz*dt
        self.p      = self.p + pdz*(dt**2)

        #2, kalman gain
        k      = self.p/(self.p + pz)

        #3, update
        self.x_hat  = self.x_hat + k*(z - self.x_hat)
        self.p      = (1.0 - k)*self.p

        return self.x_hat
```

- 1, calibrate sensor variance (pz, pdz)
- 2, measure position z and velocity v
- 3, dynamics model

$$z_{n-1} = z_n + v_n dt$$

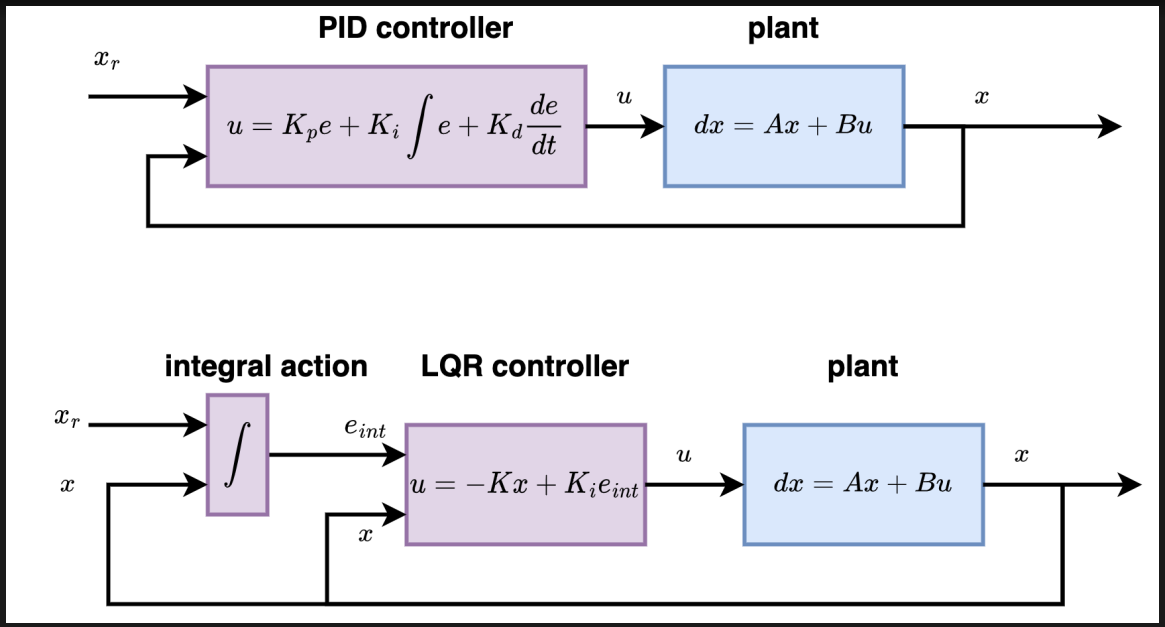
- 4, compute filter step

usage



- sensors de-noising
- sensors fusion (gyro + accelerometer), GPS
- navigation, error checking
- Apollo guidance computer (Apollo 8, 1968)
- Hakuto lander crash !!! (2023, April 2023)

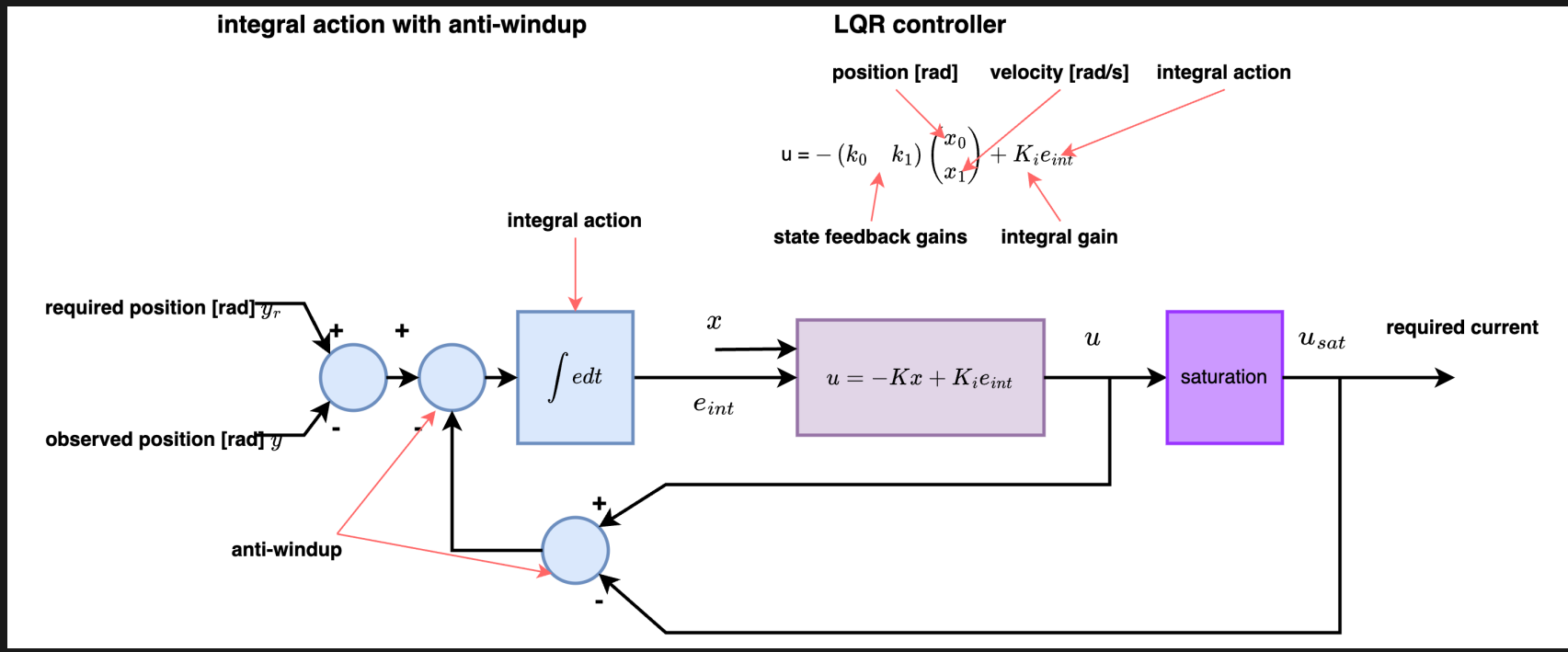
controll : PID vs LQR



- PID
- single input, single output
- simple systems (motor control, temperature ...)
- empirical "tunning"

- LQR
- multiple inputs, multiple outputs
- complex dynamics (drones, F16, rockets, walking robot, segway)
- exact solution
- robust

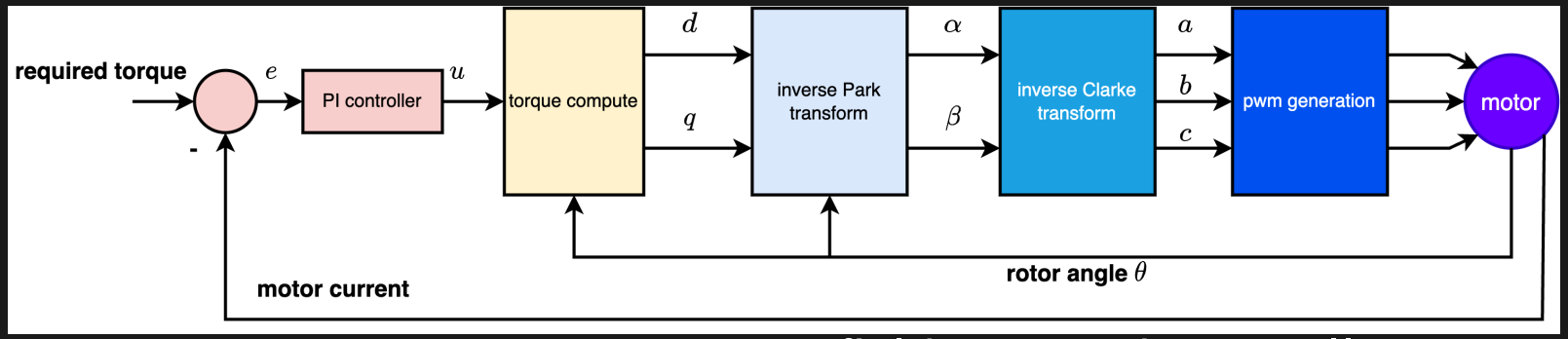
LQR for servo



three phase motor controll

- MCU : stm32f051
  - ARM cortex m0, 48MHz
- encoder : AS5600, 12bit
- driver : MP6540H, 5A, 50V

field oriented controll



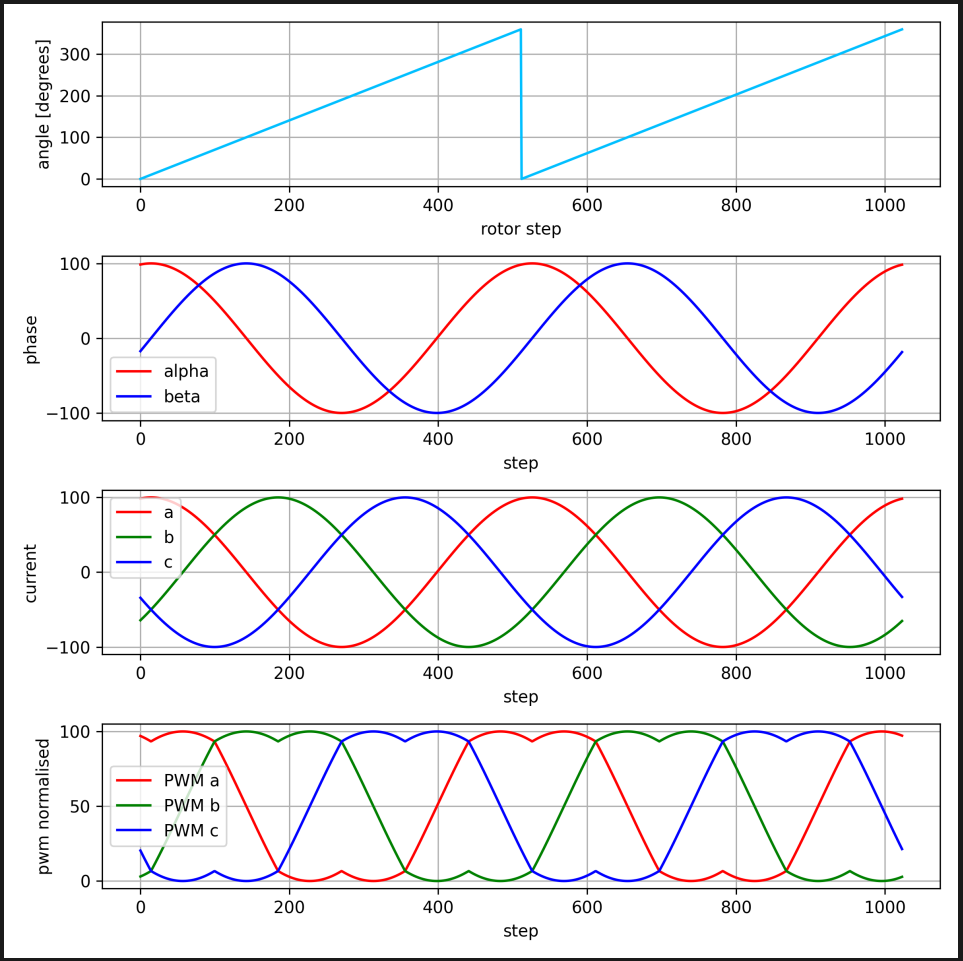
trapezoidal control

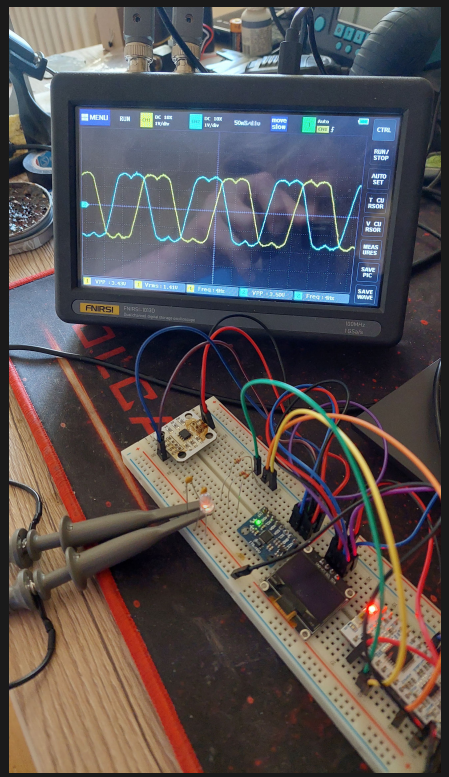
- highest maximum speed
- lowest switching losses
- easiest implementation

field oriented controll

- highest power output
- lowest noise
- best torque ripple
- maximum motor efficiency
- coding experience needed

field oriented controll





recommended sources

- book : Maxim Lapan, 2020, Deep Reinforcement Learning Hands-On second edition
- book : Enes Bilgin, 2020, Mastering Reinforcement Learning with Python
- youtuber : Yannic Kilcher, link
- youtuber : Two Minute Papers, link
- web : Paper With Code, link
- web : Intellabs, link

Q&A

- https://github.com/michalnand/
- michal.nand@gmail.com