RESET
LQR & MPC
Design and Implementation

Maksim Katerishich

Sausar Karaf

Aleksandr Kashirin

Zhanibek Darush

Goal

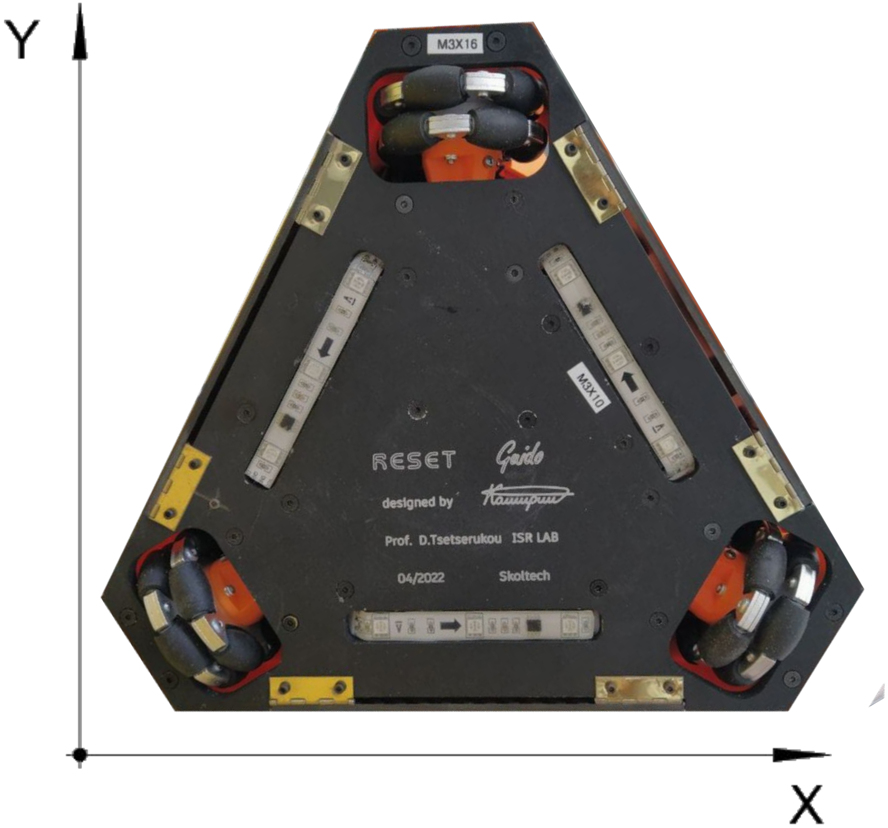
Design, comparison of LQR and MPC, implementation on the platform

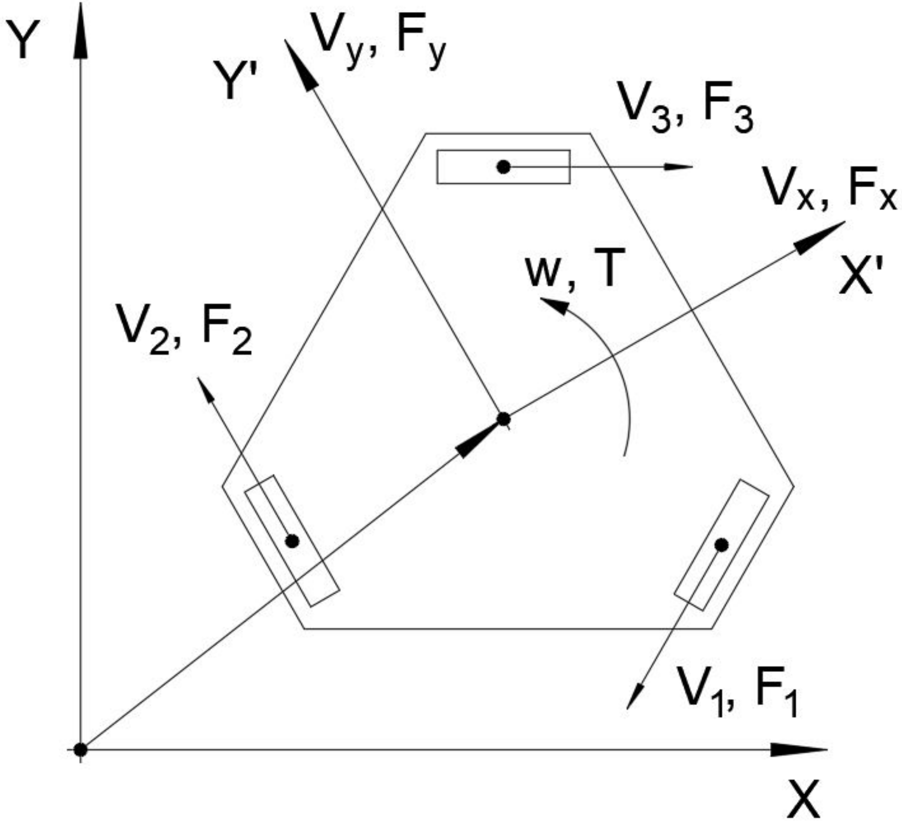
Prerequisites

1. Robot Guido by RESET
2. Pipeline: Matlab + Simulink, Python, C, ROS, SOLIDWORKS, do-mpc.
3. Localization and path-planning are completed task for the project

Plan

1. Mathematical model synthesis
2. Model verification
3. Controller design: LQR & MPC
4. Implementation
5. Results evaluation


Y
X
M3X16
M3X10
RESET
designed by
Prof. D.Tsetserukou ISR LAB
04/2022
Skoltech


Y
Y'
V_y, F_y
V_3, F_3
V_x, F_x
X'
w, T
V_2, F_2
V_1, F_1
X

System Equations

$$M\frac{dv_x(t)}{dt} = \sum F_{v_x}(t) - B_{v_x} v_x(t) - C_{v_x}\operatorname{sign}(v_x(t))$$

$$M\frac{dv_y(t)}{dt} = \sum F_{v_y}(t) - B_{v_y} v_y(t) - C_{v_y}\operatorname{sign}(v_y(t))$$

$$J\frac{d\omega(t)}{dt} = \sum T(t) - B_{\omega}\omega(t) - C_{\omega}\operatorname{sign}(\omega(t))$$

Reference: [Dynamical Models for Omni-directional Robots]

Force along X

System Equations

Coulomb friction coefficient

Mass

$$M\frac{dv_x(t)}{dt} = \sum F_{v_x}(t) - B_{v_x}v_x(t) - C_{v_x}\mathrm{sign}(v_x(t))$$

Linear speed along Vy

$$M\frac{dv_y(t)}{dt} = \sum F_{v_y}(t) - B_{v_y}v_y(t) - C_{v_y}\mathrm{sign}(v_y(t))$$

Inertia Moment

Viscous friction coefficient

$$J\frac{d\omega(t)}{dt} = \sum T(t) - B_{\omega}\omega(t) - C_{\omega}\mathrm{sign}(\omega(t))$$

Torque

Reference: [Dynamical Models for Omni-directional Robots]

Dynamic Continuous State-Space Model of the System

$$X = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix} \quad \longrightarrow \dot{X} = \begin{bmatrix} v_x \\ v_y \\ \omega \\ a_x \\ a_y \\ \varepsilon \end{bmatrix} \qquad U = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix}$$

X:
- x, y — positions, [m]
- θ — angle, [rad]
- v_x, v_y — velocities, [m/s]
- ω — angular velocity, [rad/s]

$\dot{X}$:
- v_x, v_y — velocities, [m/s]
- ω — angular velocity, [rad/s]
- a_x, a_y — accelerations, [m/s^2]
- ε — angular acceleration, [rad/s^2]

U: motor current control, [A]

Dynamic Continuous State-Space Model of the System

$$\begin{bmatrix} \dot{X} \\ Y \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ Cx + Du \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-B_{v_x}}{M} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-B_{v_y}}{M} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-B_w}{J} \end{bmatrix} ; \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{-cos(30^\circ) \cdot K_m}{M} & 0 & \frac{cos(30^\circ) \cdot K_m}{M} \\ \frac{-cos(60^\circ) \cdot K_m}{M} & \frac{K_m}{M} & \frac{-cos(60^\circ) \cdot K_m}{M} \\ \frac{-d \cdot K_m}{J} & \frac{-d \cdot K_m}{J} & \frac{-d \cdot K_m}{J} \end{bmatrix}$$

A - System Dynamics Matrix

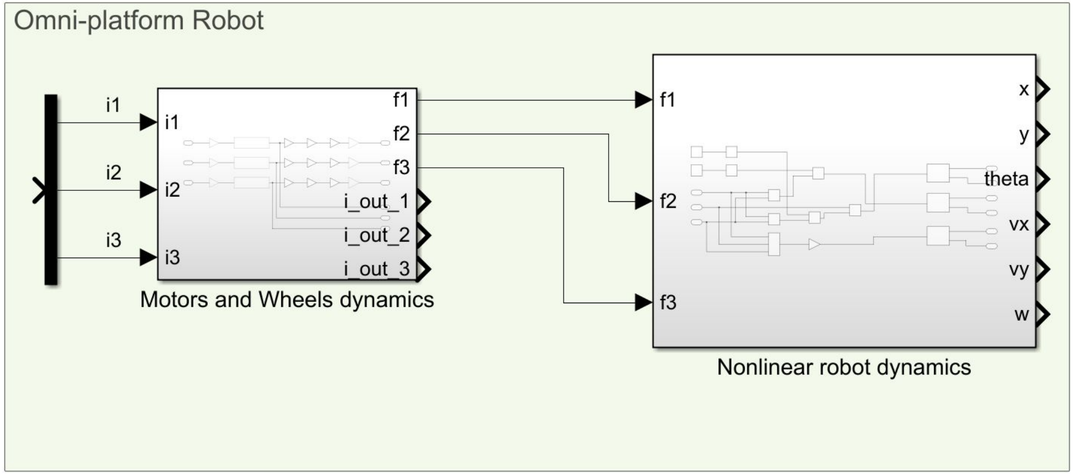
B - Input Gain Matrix

$$C = I_{6 \times 6}$$

C - Output Gain Matrix

$$D = 0$$

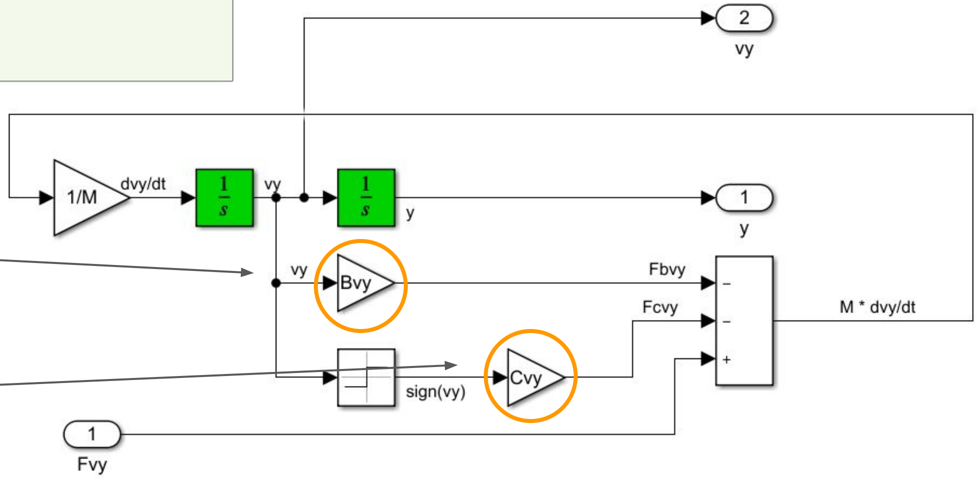
D - Feedforward Matrix



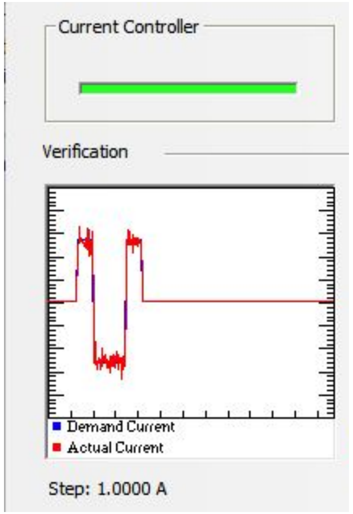
System Model in Simulink with nonlinear friction and motor dynamics

Viscous friction

Coulomb friction

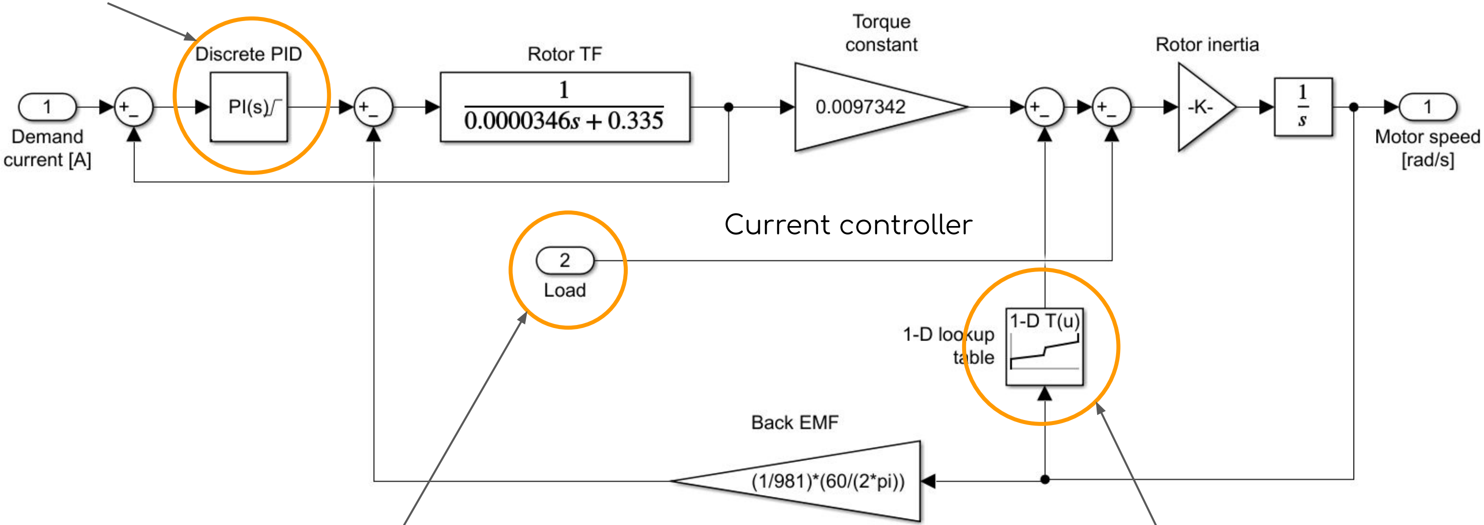


Brushed DC Motor Simulink Model



Maxon motor driver current controller tuning used for model verification

Current controller



Current controller

Robot inertia and friction load transformed to motor

Motor "freewheel" friction

Dynamic Discrete Augmented State-Space Model of the System

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} Ax_k + Bu_k \\ Cx_k + Du_k \end{bmatrix}$$

$$X_k = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \\ \sum \mathrm{err}_x \\ \sum \mathrm{err}_y \\ \sum \mathrm{err}_\theta \end{bmatrix}$$

- x, y — positions, [m]
- θ — angle, [rad]
- v_x, v_y — velocities, [m/s]
- ω — angular velocity, [rad/s]
- $\sum \mathrm{err}_x$, $\sum \mathrm{err}_y$, $\sum \mathrm{err}_\theta$ — Sums of the deviation errors, [m•s, m•s, rad•s]

LQR Cost function

$$J = \sum_{i=0}^{N-1} x^T Q x + u^T R u$$

MATLAB

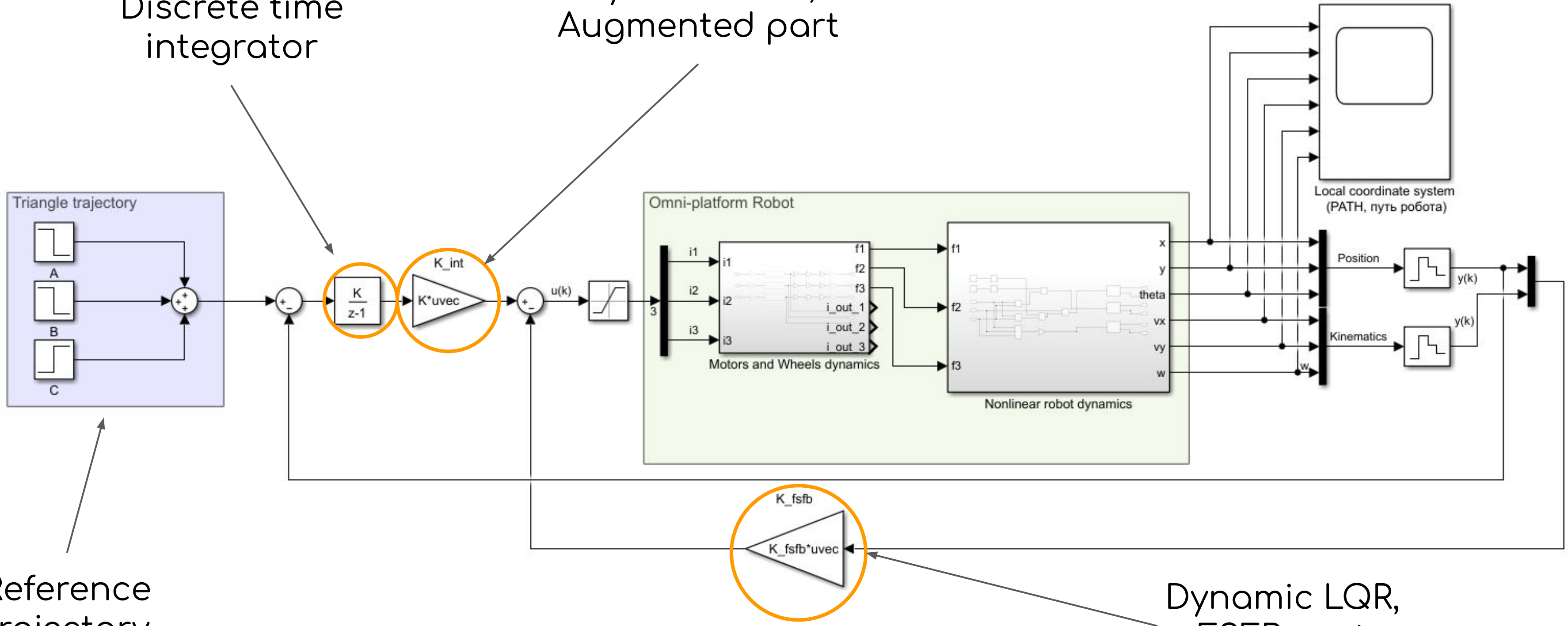
```
[K_lqr, S, e] = dlqr(A, B, Q, R)
```

Dynamic Augmented LQR Controller

$$K_{LQR} = \left[\underbrace{\begin{matrix} -1.295 & -0.748 & -0.261 & -1.142 & -0.66 & -0.101 \\ 0 & 1.495 & -0.261 & 0 & 1.32 & -0.101 \\ 1.295 & -0.748 & -0.261 & 1.142 & -0.66 & -0.101 \end{matrix}}_{\text{Full State Feedback Part}} \quad \underbrace{\begin{matrix} -0.014 & -0.008 & -0.002 \\ 0 & 0.016 & -0.002 \\ 0.014 & -0.008 & -0.002 \end{matrix}}_{\text{Integral Augmented Part}}\right]$$

Discrete time integrator

Dynamic LQR, Augmented part



Reference trajectory

Dynamic LQR, FSFB part

Project | System Model | Dynamic LQR | Kinematic LQR | MPC Controller | Embedded System | ROS

(Dynamic LQR)

Kinematic Continuous State-Space Model of the System

$$X = \begin{bmatrix} x \\ y \\ \theta \\ \upsilon_x \\ \upsilon_y \\ \omega \end{bmatrix} \longrightarrow \dot{X} = \begin{bmatrix} \upsilon_x \\ \upsilon_y \\ \omega \\ a_x \\ a_y \\ \varepsilon \end{bmatrix} \qquad U = \begin{bmatrix} v_x \\ v_y \\ w \end{bmatrix}$$

X:
- x, y — positions, [m]
- θ — angle, [rad]
- υ_x, υ_y — velocities, [m/s]
- ω — angular velocity, [rad/s]

$\dot{X}$:
- υ_x, υ_y — velocities, [m/s]
- ω — angular velocity, [rad/s]
- a_x, a_y — accelerations, [m/s^2]
- ε — angular acceleration, [rad/s^2]

U: velocity control, [m/s], [rad/s]

Kinematic Continuous State-Space Model of the System

$$\begin{bmatrix} \dot{X} \\ Y \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ Cx + Du \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-1}{T_{v_x}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{T_{v_y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{T_w} \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{T_{v_x}} & 0 & 0 \\ 0 & \frac{1}{T_{v_y}} & 0 \\ 0 & 0 & \frac{1}{T_w} \end{bmatrix}$$

A - System Dynamics Matrix

B - Input Gain Matrix

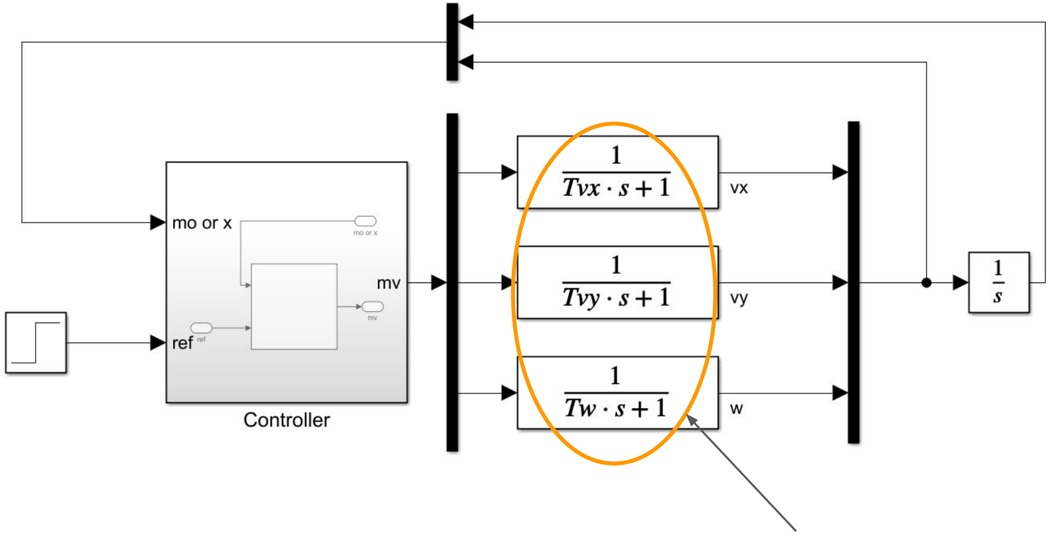
$$C = I_{6\times 6}$$

C - Output Gain Matrix

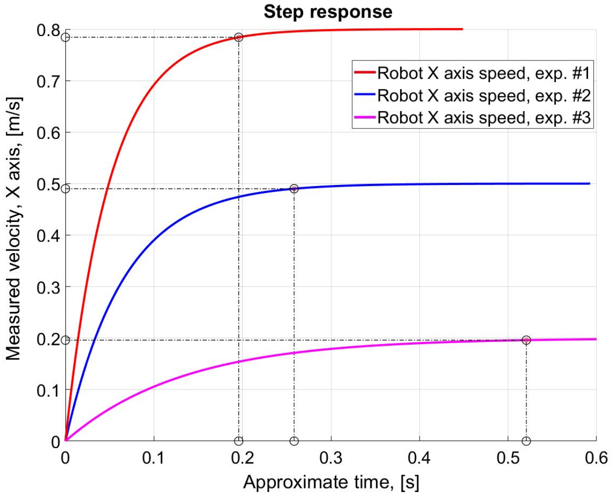
$$D = 0$$

D - Feedforward Matrix

System Model in Simulink with combined nonlinearities and dynamics



Approximated dynamics of the system - 1st order TF



Part of experimental data - Tv, Tw time constants estimation

Kinematic Discrete State-Space Model of the System

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} Ax_k + Bu_k \\ Cx_k + Du_k \end{bmatrix}$$

$$X_k = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix}$$

positions, [m] (x, y)

angle, [rad] (θ)

velocities, [m/s] (v_x, v_y)

angular velocity, [rad/s] (ω)

Kinematic LQR Controller

$$K_{LQR} = \underbrace{\begin{bmatrix} 0.8097 & 0 & 0 & 0.0666 & 0 & 0 \\ 0 & 0.8097 & 0 & 0 & 0.6616 & 0 \\ 0 & 0 & 0.8094 & 0 & 0 & 0.0377 \end{bmatrix}}_{\text{Full State Feedback Part}}$$

Kinematic Discrete State-Space Model of the System

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} Ax_k + Bu_k \\ Cx_k + Du_k \end{bmatrix}$$

$$X_k = \begin{bmatrix} x \\ y \\ \theta \\ v_x \\ v_y \\ \omega \end{bmatrix}$$

positions, [m]

angle, [rad]

velocities, [m/s]

angular velocity, [rad/s]

Kinematic LQR Controller

$$K_{LQR} = \underbrace{\begin{bmatrix} 0.8097 & 0 & 0 & 0.0666 & 0 & 0 \\ 0 & 0.8097 & 0 & 0 & 0.6616 & 0 \\ 0 & 0 & 0.8094 & 0 & 0 & 0.0377 \end{bmatrix}}_{\text{Full State Feedback Part}}$$

FSFB LQR Controller degenerates into 3 parallel PD controllers:

$$K_{\text{LQR}} \rightarrow PD(x) + PD(y) + PD(\theta)$$

do-mpc MPC cost function template

$$C = \sum_{k=0}^{n-1} \left(\underbrace{l(x_k, u_k, z_k, p)}_{\text{lagrange term}} + \underbrace{\Delta u_k^T R \Delta u_k}_{\text{r-term}} \right) + \underbrace{m(x_n)}_{\text{meyer term}}$$

Proposed cost function

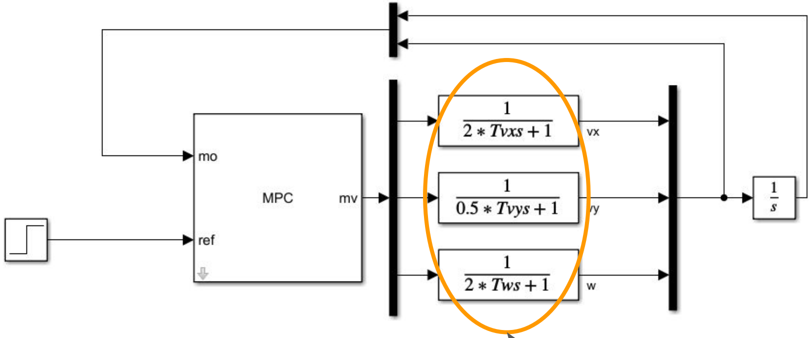
$$l = (x_{\text{ref}} - x)^2 + (y_{\text{ref}} - y)^2 + (y_{\text{ref}} - y)^2$$

$$m = l$$

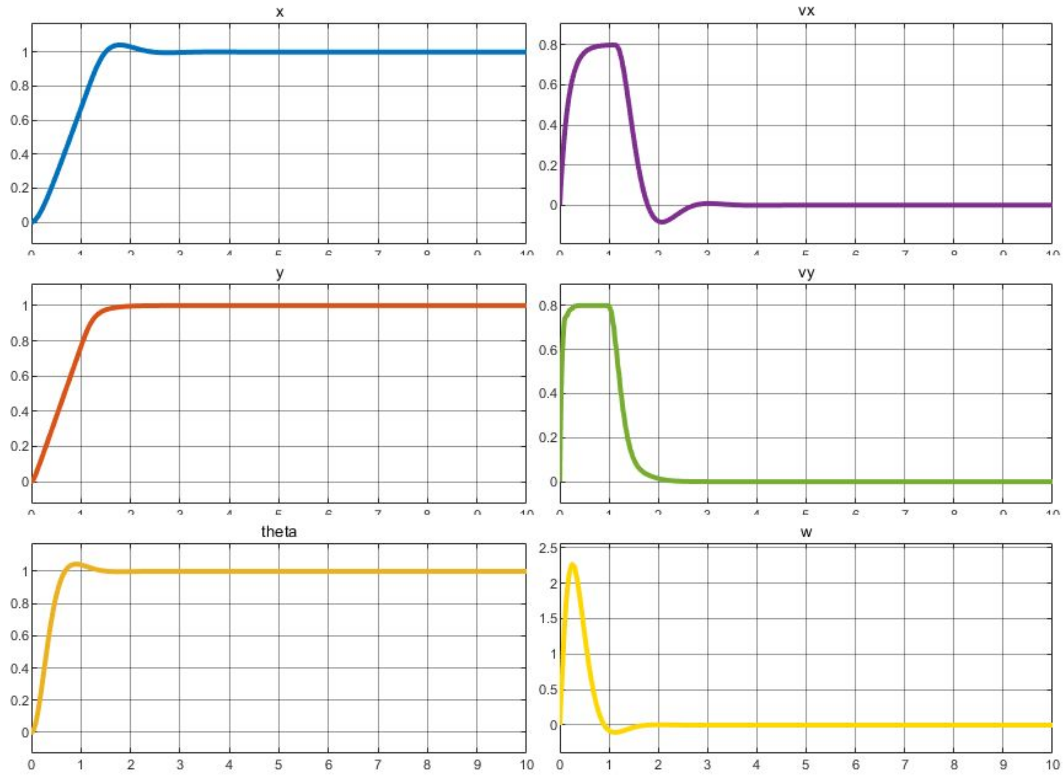
$$R = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

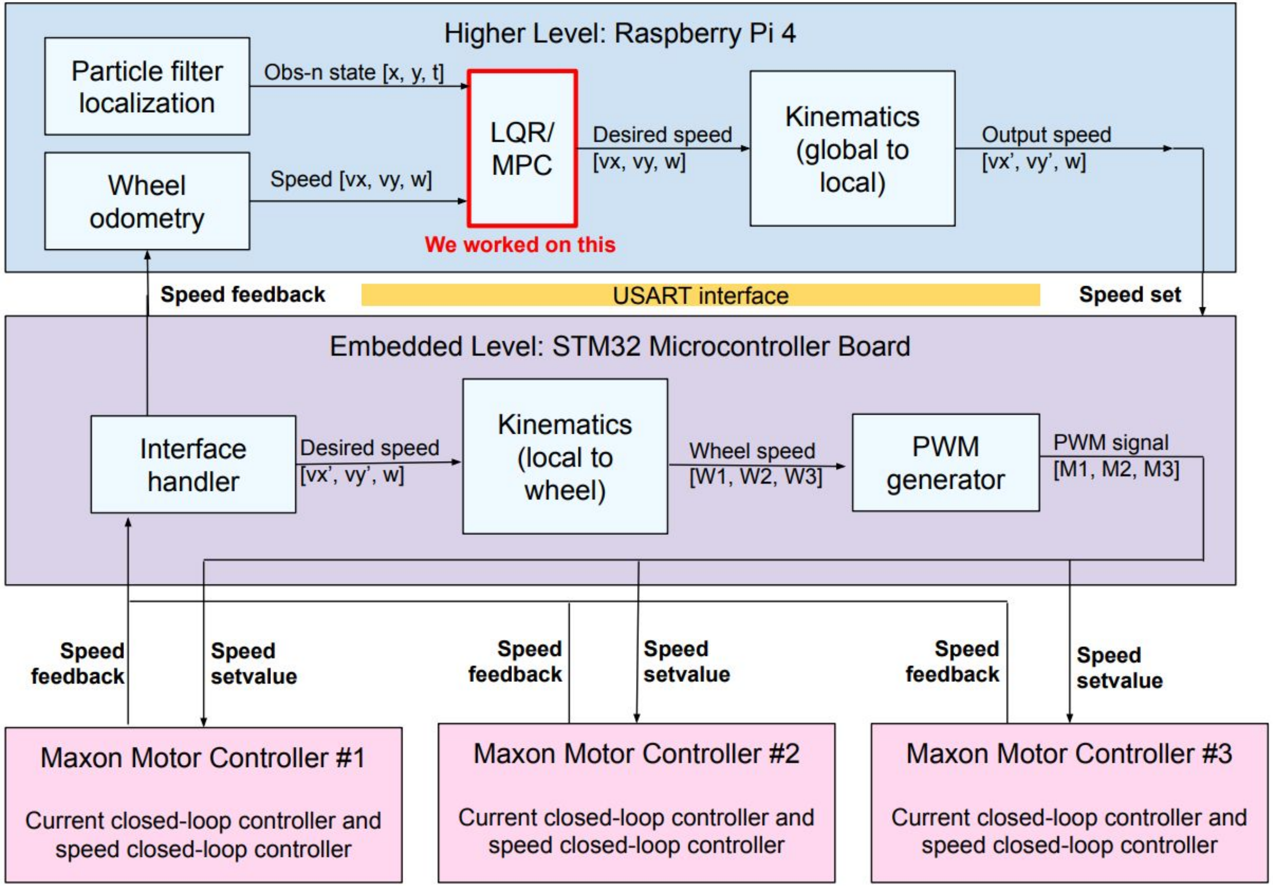
Reference: [do-mpc Python Library]

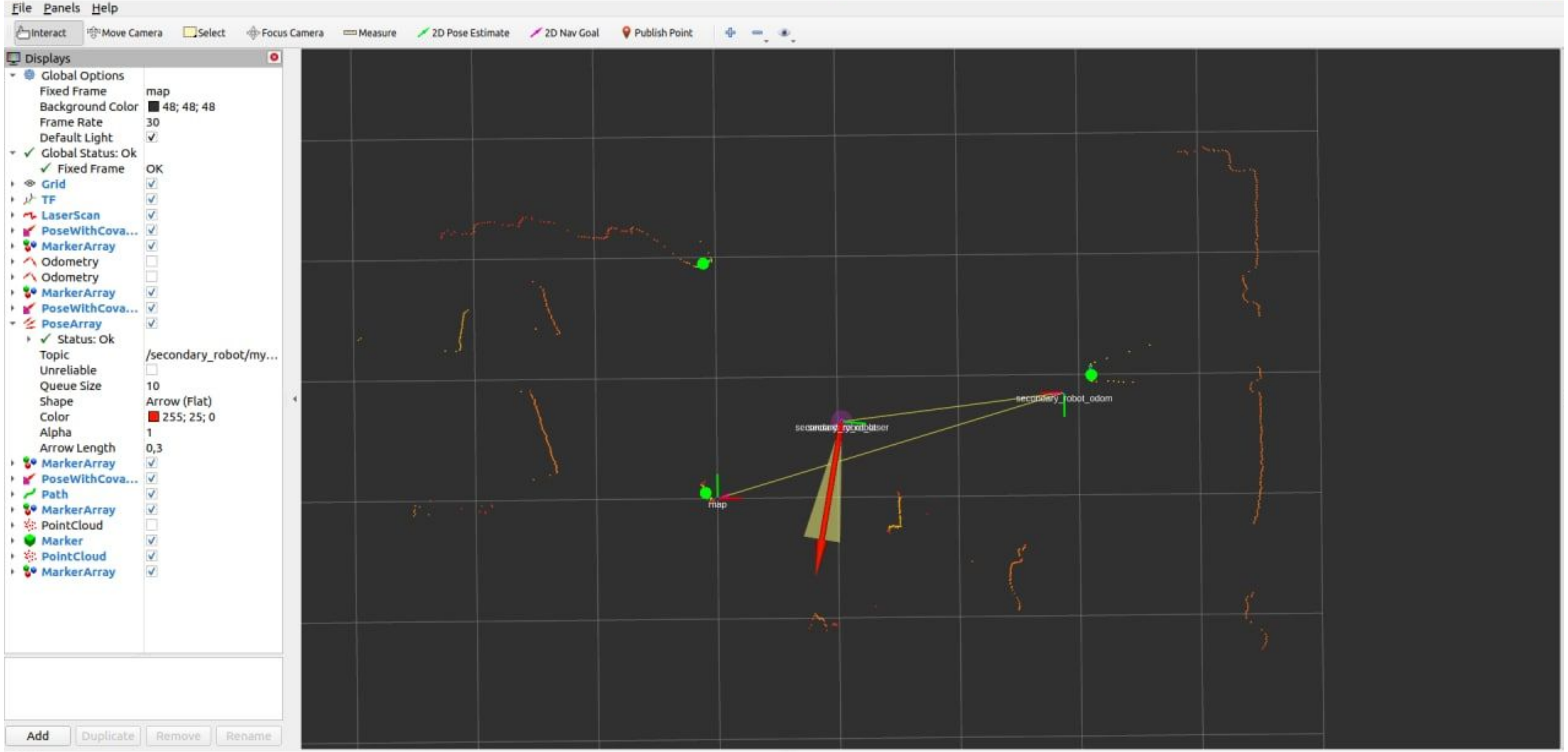
System remains the same as for Kinematic LQR



Dynamics with error assumptions




Higher Level: Raspberry Pi 4
Particle filter localization
Obs-n state [x, y, t]
Wheel odometry
Speed [vx, vy, w]
LQR/ MPC
We worked on this
Desired speed [vx, vy, w]
Kinematics (global to local)
Output speed [vx', vy', w]
Speed feedback
USART interface
Speed set
Embedded Level: STM32 Microcontroller Board
Interface handler
Desired speed [vx', vy', w]
Kinematics (local to wheel)
Wheel speed [W1, W2, W3]
PWM generator
PWM signal [M1, M2, M3]
Speed feedback
Speed setvalue
Speed feedback
Speed setvalue
Speed feedback
Speed setvalue
Maxon Motor Controller #1
Current closed-loop controller and speed closed-loop controller
Maxon Motor Controller #2
Current closed-loop controller and speed closed-loop controller
Maxon Motor Controller #3
Current closed-loop controller and speed closed-loop controller



RVIZ visualization

Evaluation: 8-shaped trajectory test

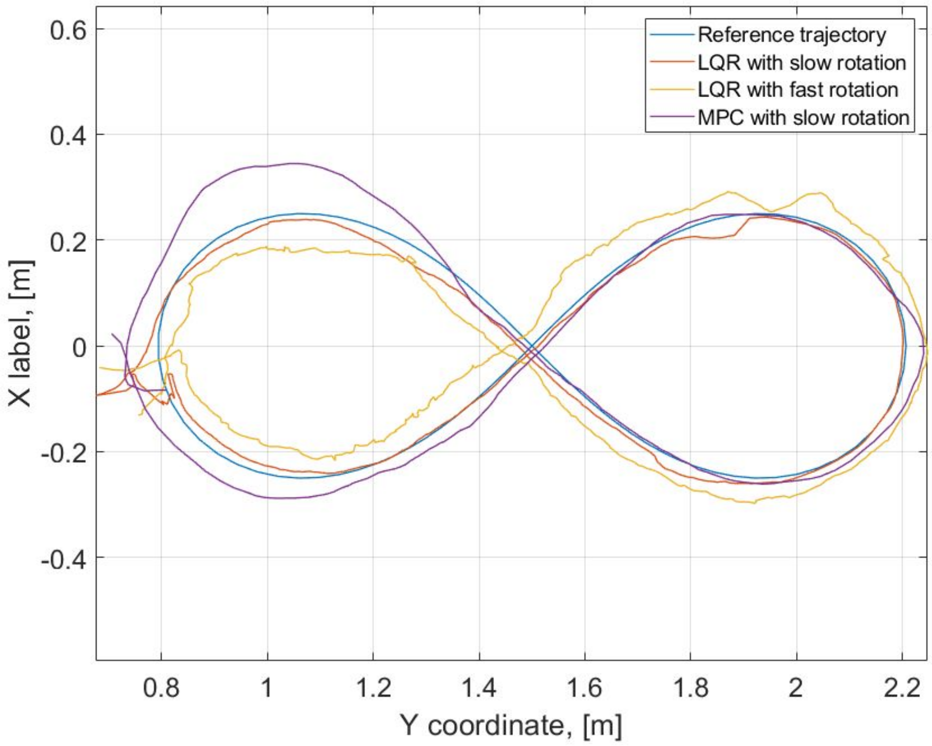
Scenario:

$$\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \dfrac{0.5\sqrt{2}\,cos(\alpha_i)}{1 + sin(\alpha_i)^2} + 1.5 \\ \dfrac{0.5\sqrt{2}\,cos(\alpha_i)\cdot sin(\alpha_i)}{1 + sin(\alpha_i)^2} \\ \alpha_i \end{bmatrix}$$

with two different angle setpoints.
α is linearly interpolated between two values.

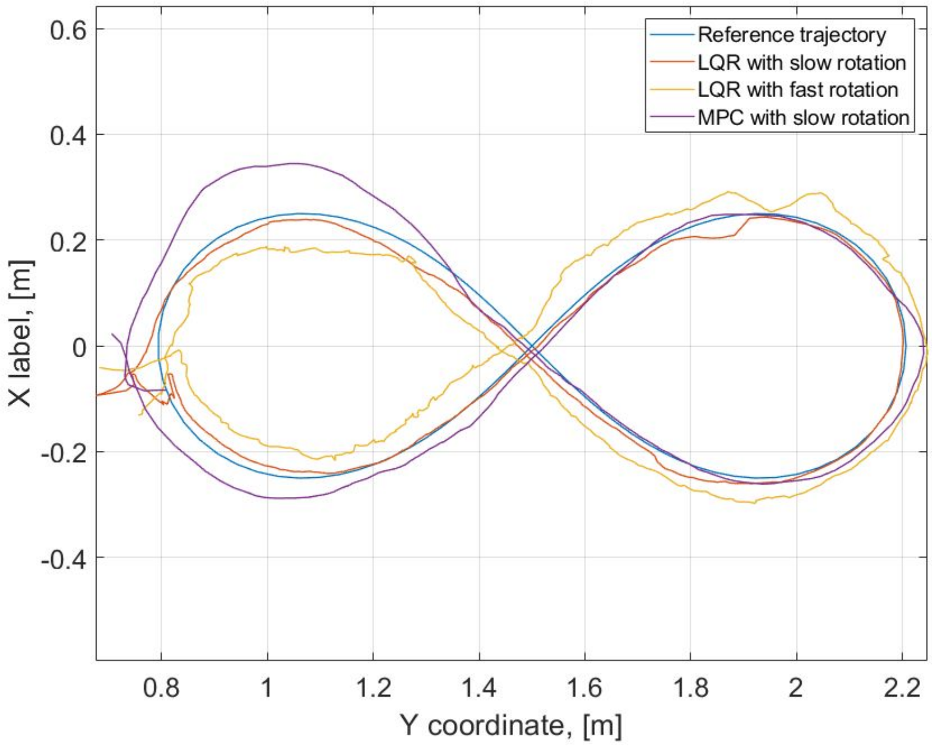
$$Case\ 1\,(slow\ rotation):\quad a_i = (0...2\pi),\quad i = 150,\quad \omega \approx 18^{\circ}/s$$

$$Case\ 2(\,fast\ rotation):\quad a_i = (0...20\pi),\quad i = 150,\quad \omega \approx 180^{\circ}/s$$

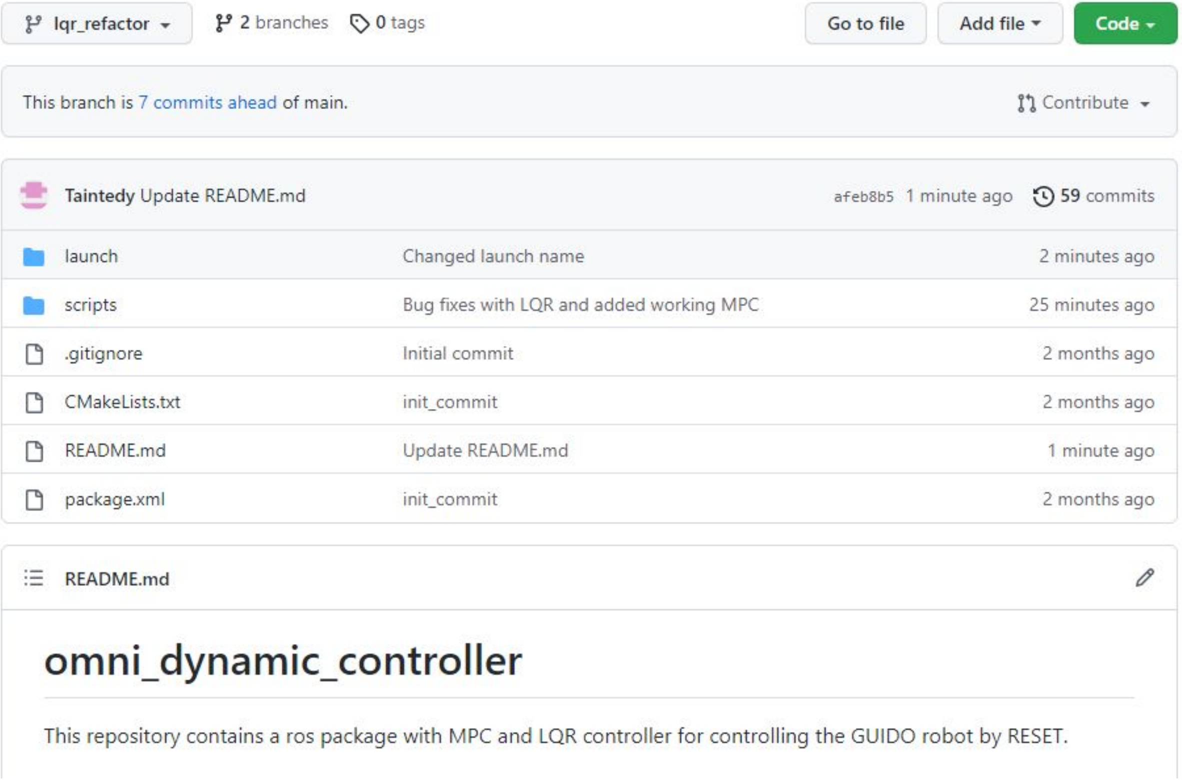



аче всмысле

*hell what


Reference trajectory
LQR with slow rotation
LQR with fast rotation
MPC with slow rotation
0.6
0.4
0.2
0
-0.2
-0.4
X label, [m]
0.8
1
1.2
1.4
1.6
1.8
2
2.2
Y coordinate, [m]

Mathematical model synthesis	✔️ The dynamic model and kinematic one
Model verification	✔️ Friction function is constructed ✔️ Tuned brushed DC motor model ✔️ System dynamics is approximated
Controller design	❌ Dynamic LQR ✔️ Kinematic LQR ✔️ MPC
Implementation	✔️ See demo video
Results evaluation	❌ Dynamic LQR - didn’t converge ✔️ Kinematic LQR - passed “slow” test ✔️ MPC - passed “slow” test ❌ MPC - “fast” test wasn’t conducted ✔️ …and much more tests



Reference: [Project Github]