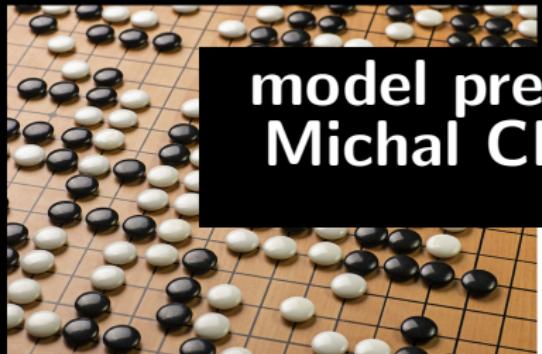


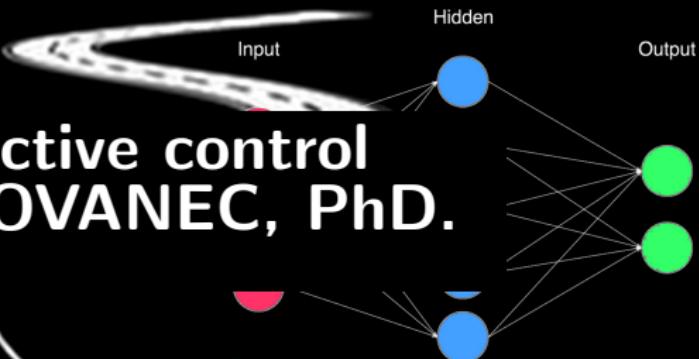
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

(The New Action Value = The Old Value) + The Learning Rate \times (The New Information — the Old Information)

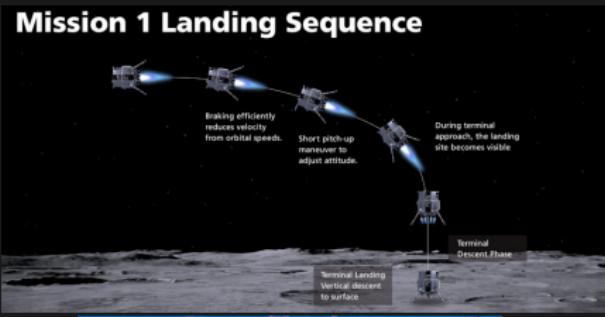


model predictive control

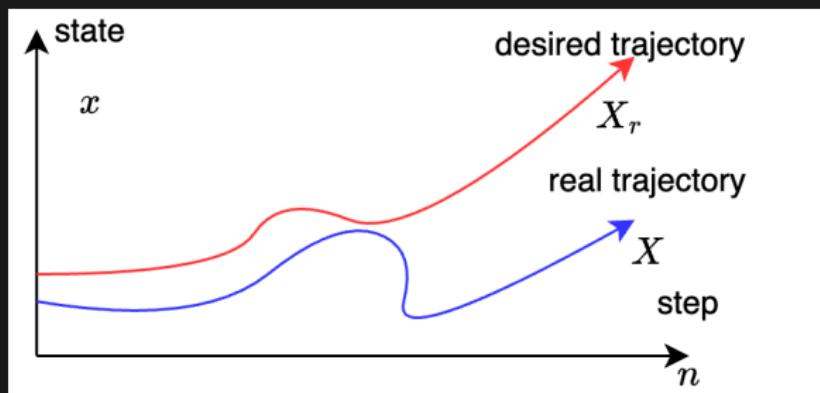
Michal CHOVANEC, PhD.



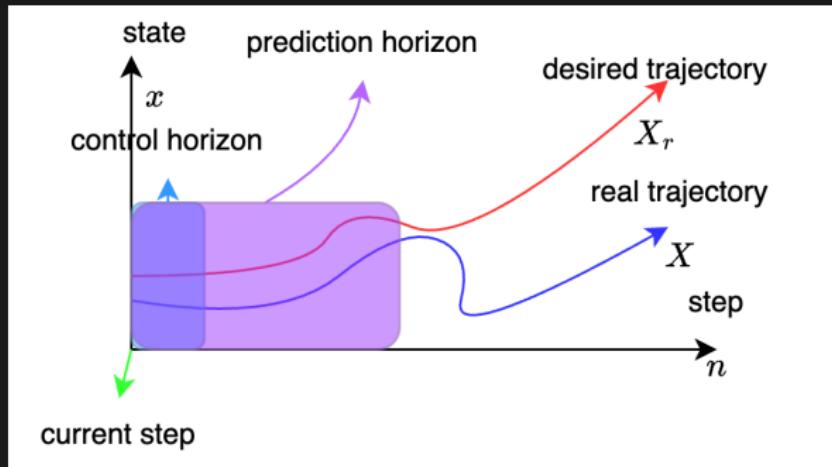
MPC - applications



MPC - overview



MPC - overview



state vector,

e.g. (position, velocity, angle,
angular rate)

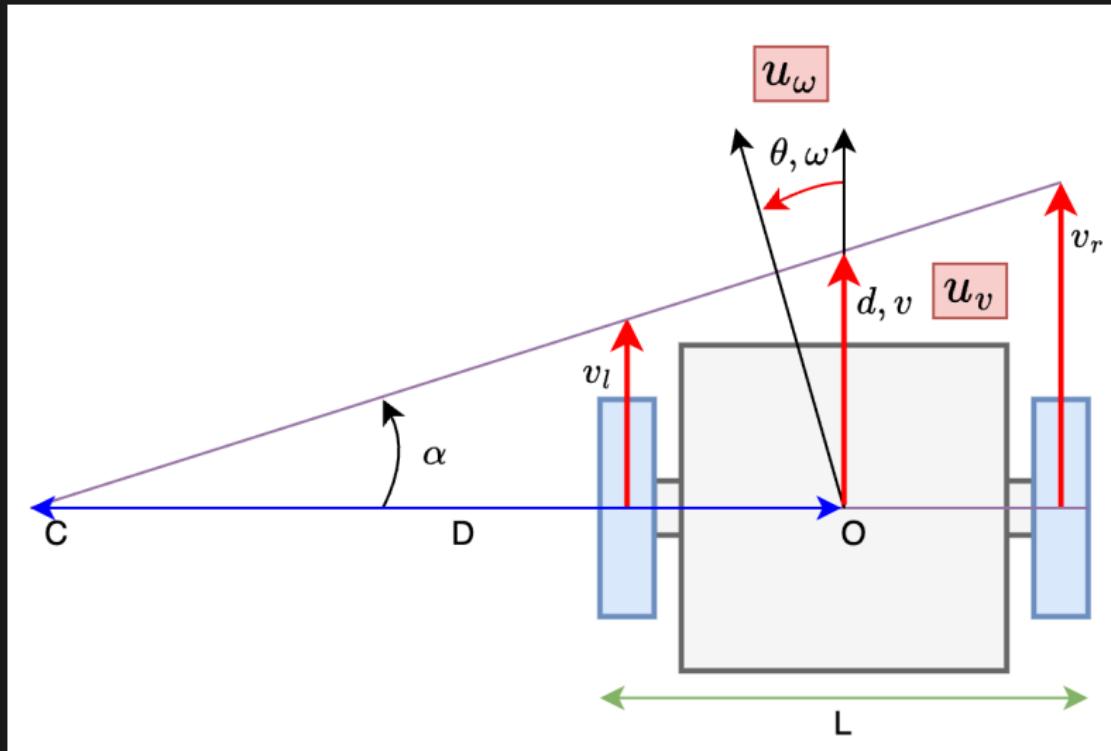
control variable,

e.g. (left speed, right speed)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ .. \\ x_{n-1} \end{bmatrix}$$

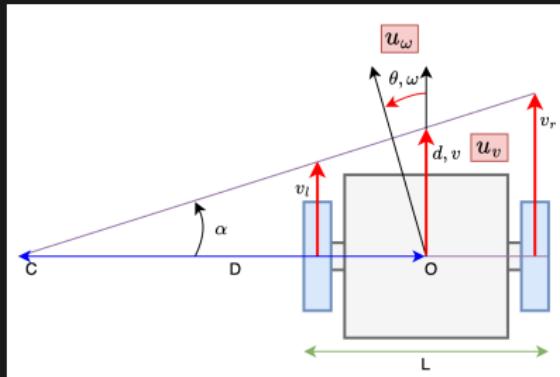
$$u = \begin{bmatrix} u_0 \\ u_1 \\ .. \\ u_{m-1} \end{bmatrix}$$

robot trajectory tracking example



robot trajectory tracking example

- state



$$x = \begin{bmatrix} v \\ d \\ \omega \\ \theta \end{bmatrix}$$

- control variable

$$\Delta u = \begin{bmatrix} \Delta u_v \\ \Delta u_\omega \end{bmatrix}$$

$$u(n) = u(n-1) + \Delta u(n)$$

state space model (continuos)

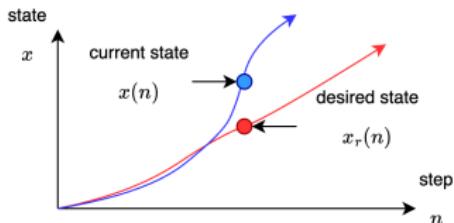
$$dx = Ax + Bu$$

$$A = \begin{pmatrix} -\frac{1}{\tau_{forward}} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau_{turn}} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} -\frac{k_{forward}}{\tau_{forward}} & 0 \\ 0 & 0 \\ 0 & -\frac{k_{turn}}{\tau_{turn}} \\ 0 & 0 \end{pmatrix}$$

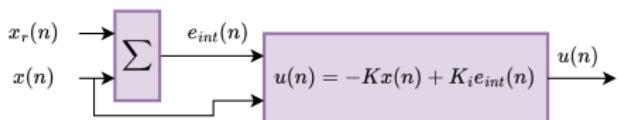
- 4 poles : double in 0, $-\frac{1}{\tau_{forward}}$ and $-\frac{1}{\tau_{turn}}$
- 2 control inputs : forward velocity and turning velocity
- constrains for kick avoid (jerk) : $\Delta u_{min} < \Delta u < \Delta u_{max}$
- constrains for motor limits : $u_{min} < u < u_{max}$

MPC - overview

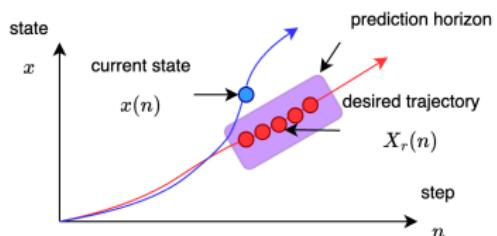
LQR regulating



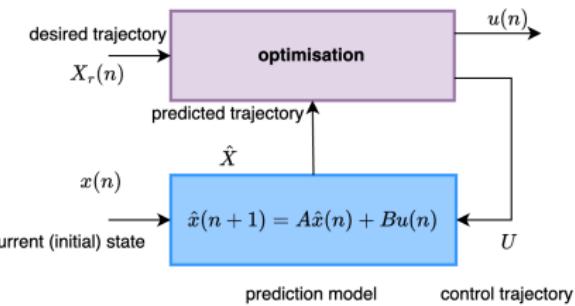
LQR controller



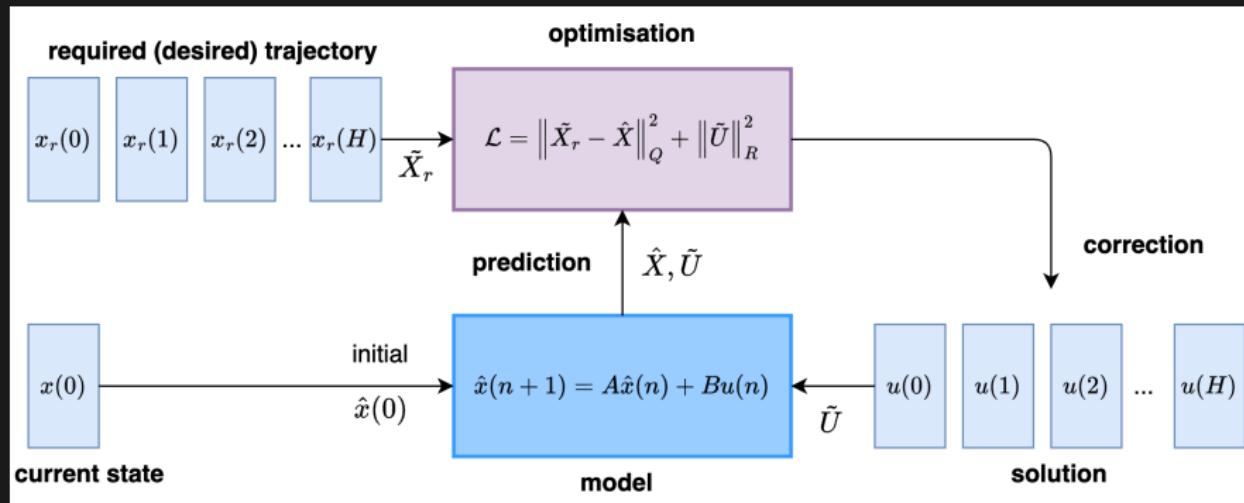
MPC tracking



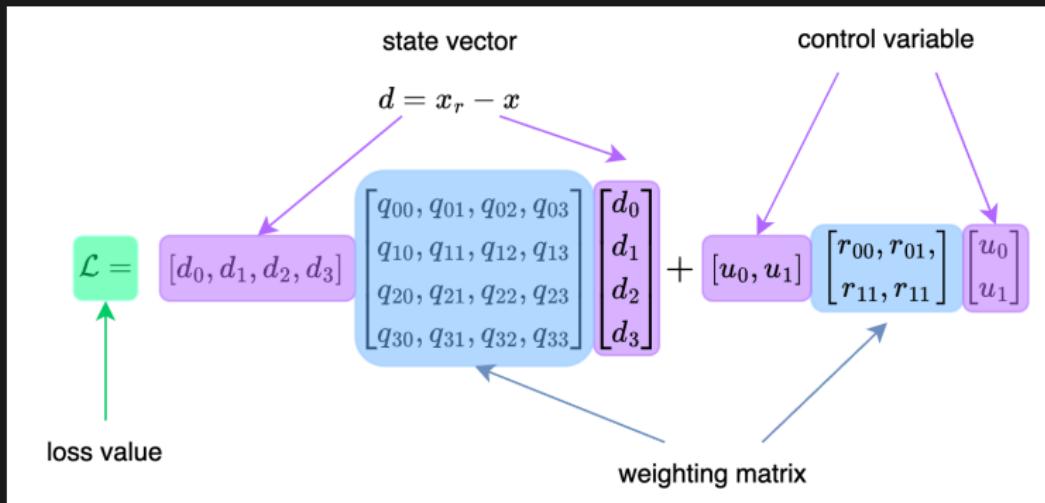
MPC controller



MPC - overview



quadratic problem formulation



quadratic problem formulation

loss (cost) function is quadratic, with weighting terms Q and R ,

$$\mathcal{L} = \sum_{h=0}^{H-1} ((x_r(h) - x(h))^T Q (x_r(h) - x(h))) +_{\Delta} u^T(h) R_{\Delta} u(h),$$

$$u(n) = u(n-1) +_{\Delta} u(n), \\ \text{s.t. } x(n+1) = Ax(n) + Bu(n).$$

where :

- H is prediction horizon steps
- A is matrix, $n \times n$
- B is matrix, $n \times m$
- Q is matrix, $n \times n$
- R is matrix, $m \times m$
- Δu is controller output
- where n is system orders, and m system inputs count

unrolling sequence

rewrite into form where initial conditions **depends only** on $x(n)$ and $u(n - 1)$

$$x(n + 1) = Ax(n) + Bu(n)$$

$$= Ax(n) + B(u(n - 1) +_{\Delta} u(n))$$

$$x(n + 2) = Ax(n + 1) + B(u(n) +_{\Delta} u(n + 1))$$

$$= A^2x(n) + (AB + B)u(n - 1) + (AB + B)_{\Delta}u(n) + B_{\Delta}u(u)$$

$$x(n + 3) = Ax(n + 2) + B(u(n + 1) +_{\Delta} u(n + 2))$$

$$= A^3x(n) + (A^2 + AB + B)u(n - 1)$$

$$+ (A^2B + AB + B)_{\Delta}u(n)$$

$$+ (AB + B)_{\Delta}u(n + 1) + B_{\Delta}u(n + 2)$$

...

matrix formulation

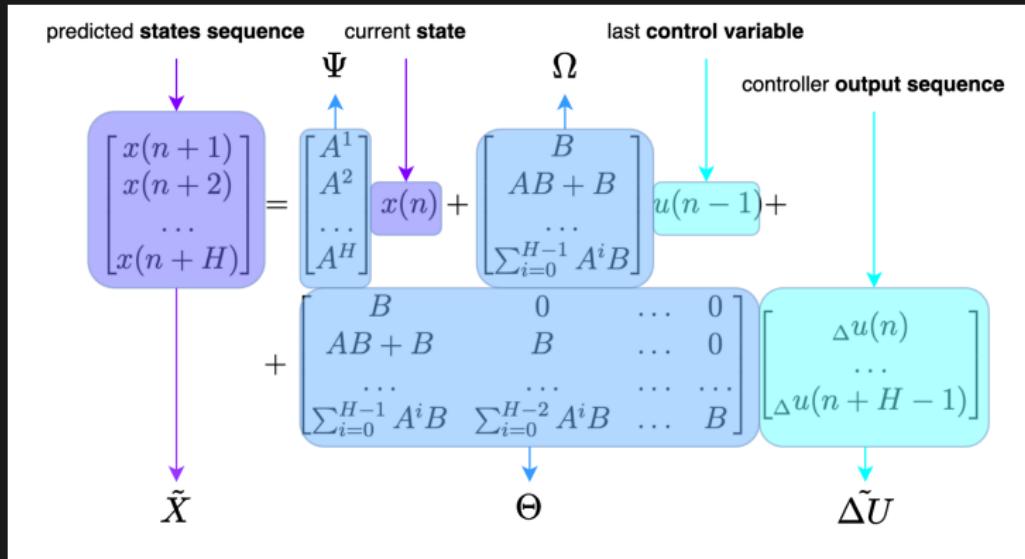
rewrite into matrix form

$$\begin{bmatrix} x(n+1) \\ x(n+2) \\ \dots \\ x(n+H) \end{bmatrix} = \begin{bmatrix} A^1 \\ A^2 \\ \dots \\ A^H \end{bmatrix} x(n) + \begin{bmatrix} B \\ AB \\ \dots \\ \sum_{i=0}^{H-1} A^i B \end{bmatrix} u(n-1) + \\ + \begin{bmatrix} B & 0 & \dots & 0 \\ AB+B & B & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \sum_{i=0}^{H-1} A^i B & \sum_{i=0}^{H-2} A^i B & \dots & B \end{bmatrix} \begin{bmatrix} \Delta u(n) \\ \dots \\ \Delta u(n+H-1) \end{bmatrix}$$

in compact form

$$\tilde{X} = \Psi x(n) + \Omega u(n-1) + \Theta \tilde{U}$$

matrix formulation



$$\tilde{X} = \Psi x(n) + \Omega u(n-1) + \Theta \Delta U$$

n-th step input to output projection

$$\begin{bmatrix} x(n+1) \\ x(n+2) \\ \vdots \\ x(n+H) \end{bmatrix} = \begin{bmatrix} A^1 \\ A^2 \\ \vdots \\ A^H \end{bmatrix} x(n) + \begin{bmatrix} B \\ AB+B \\ \vdots \\ \sum_{i=0}^{H-1} A^i B \end{bmatrix} u(n-1) +$$
$$+ \begin{bmatrix} B & 0 & \dots & 0 \\ AB+B & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{H-1} A^i B & \sum_{i=0}^{H-2} A^i B & \dots & B \end{bmatrix} \begin{bmatrix} \Delta u(n) \\ \vdots \\ \Delta u(n+H-1) \end{bmatrix}$$

projection how **current step** effects 1st output step

projection how **current step** effects 2nd output step

projection of control $\Delta u(n)$ to output sequence

$$\begin{bmatrix} x(n+1) \\ x(n+2) \\ \vdots \\ x(n+H) \end{bmatrix} = \begin{bmatrix} A^1 \\ A^2 \\ \vdots \\ A^H \end{bmatrix} x(n) + \begin{bmatrix} B \\ AB+B \\ \vdots \\ \sum_{i=0}^{H-1} A^i B \end{bmatrix} u(n-1) +$$

causality : 2nd input step can't effect 1st output step

$$+ \begin{bmatrix} B \\ AB+B \\ \vdots \\ \sum_{i=0}^{H-1} A^i B \end{bmatrix} \begin{bmatrix} 0 \\ B \\ \vdots \\ \sum_{i=0}^{H-2} A^i B \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ B \end{bmatrix} \begin{bmatrix} \Delta u(n) \\ \vdots \\ \Delta u(n+H-1) \end{bmatrix}$$

projection how 1st input step effects whole output projection how 2nd input step effects whole output last input step effects only last output step

Q and R augmentation

prediction horizon steps H

prediction horizon steps H

$$\tilde{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Q \end{bmatrix}$$

control horizon steps Hc

$$\tilde{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

put into quadratic loss (cost) function

$$\mathcal{L} = {}_{\Delta}\tilde{U}^T \tilde{R}_{\Delta} \tilde{U} + (\tilde{X}_r - \tilde{X})^T \tilde{Q} (\tilde{X}_r - \tilde{X})$$

after substitution

$$S = \tilde{X}_r - \Psi \tilde{X} - \Omega u(n-1)$$

we obtain

$$\begin{aligned}\mathcal{L} &= {}_{\Delta}\tilde{U}^T \tilde{R}_{\Delta} \tilde{U} + (S - \Theta_{\Delta} \tilde{U})^T \tilde{Q} (S - \Theta_{\Delta} \tilde{U}) \\ &= {}_{\Delta}\tilde{U}^T \tilde{R}_{\Delta} \tilde{U} + S^T \tilde{Q} S - S^T \tilde{Q} \Theta_{\Delta} \tilde{U} - {}_{\Delta}\tilde{U}^T \Theta^T \tilde{Q} S + {}_{\Delta}\tilde{U}^T \Theta^T \tilde{Q} \Theta_{\Delta} \tilde{U}\end{aligned}$$

finding minima

find derivative with respect to $\Delta \tilde{U}$

$$\frac{\partial \mathcal{L}}{\partial \Delta \tilde{U}} :$$

$$\frac{\partial \Delta \tilde{U}^T \tilde{R}_\Delta \tilde{U}}{\partial \Delta \tilde{U}} = 2 \tilde{R}_\Delta \tilde{U}$$

$$\frac{\partial S^T \tilde{Q} S}{\partial \Delta \tilde{U}} = 0$$

$$\frac{\partial -S^T \tilde{Q} \Theta_\Delta \tilde{U}}{\partial \Delta \tilde{U}} = -\Theta^T \tilde{Q} S$$

$$\frac{\partial -\Delta \tilde{U}^T \Theta^T \tilde{Q} S}{\partial \Delta \tilde{U}} = -\Theta^T \tilde{Q} S$$

$$\frac{\partial \Delta \tilde{U}^T \Theta^T \tilde{Q} \Theta_\Delta \tilde{U}}{\partial \Delta \tilde{U}} = 2 \Theta^T \tilde{Q} \Theta_\Delta \tilde{U}$$

finding minima

put derivate equal to zero, and solve

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tilde{U}} &= 2\tilde{R}_{\Delta}\tilde{U} - 2\Theta^T\tilde{Q}S + 2\Theta^T\tilde{Q}\Theta_{\Delta}\tilde{U} \\ 0 &= 2\tilde{R}_{\Delta}\tilde{U} - 2\Theta^T\tilde{Q}S + 2\Theta^T\tilde{Q}\Theta_{\Delta}\tilde{U} \\ (\tilde{R} + \Theta^T\tilde{Q}\Theta)_{\Delta}\tilde{U} &= \Theta^T\tilde{Q}S\end{aligned}$$

and obtain **analytical solution** for model predictive control

$$\Delta\tilde{U} = (\tilde{R} + \Theta^T\tilde{Q}\Theta)^{-1}\Theta^T\tilde{Q}S$$

full algorithm

$$\tilde{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

$$\Theta = \begin{bmatrix} B & 0 & \dots & 0 \\ AB + B & B & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \sum_{i=0}^{H-1} A^i B & \sum_{i=0}^{H-2} A^i B & \dots & B \end{bmatrix}$$

$$\tilde{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Q \end{bmatrix}$$

precompute

$$\Xi = (\tilde{R} + \Theta^T \tilde{Q} \Theta)^{-1} \Theta^T \tilde{Q}$$

$$\Psi = \begin{bmatrix} A^1 \\ A^2 \\ \dots \\ A^H \end{bmatrix}$$

$$\Omega = \begin{bmatrix} B \\ AB + B \\ \dots \\ \sum_{i=0}^{H-1} A^i B \end{bmatrix}$$

error estimation

$$E(n) = X_r(n) - \Psi x(n) - \Omega u(n-1)$$

run in loop

future prediction using model

correction

$$\Delta u(n) = \Xi E(n)$$

output

$$u(n) = u(n-1) + \Delta u(n)$$

full algorithm

given matrices :

$$\tilde{Q}, \tilde{R}, \Psi, \Omega, \Theta$$

initialization (precompute) :

$$\Xi = (\tilde{R} + \Theta^T \tilde{Q} \Theta)^{-1} \Theta^T \tilde{Q}$$

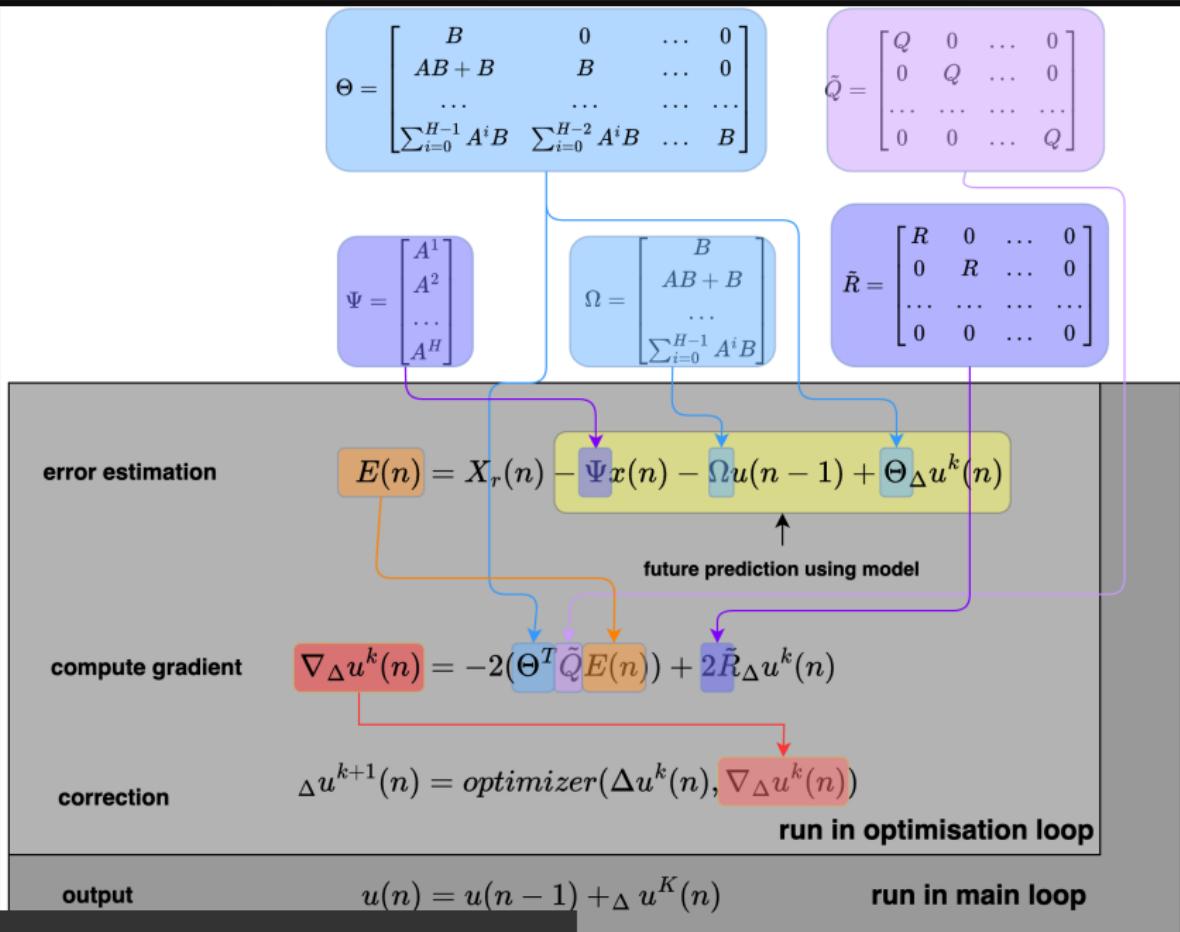
in loop :

$$E(n) = X_r(n) - \Psi x(n) - \Omega u(n-1)$$

$$\Delta u(n) = \Xi E(n)$$

$$u(n) = u(n-1) + \Delta u(n)$$

full algorithm - optimisation



handling constraints

most common clipping ranges:

$$\begin{aligned}\Delta \tilde{u}(n) &= \text{clip}(\Delta u(n), \Delta u_{min}, \Delta u_{max}) \\ u(n) &= \text{clip}(u(n-1) + \Delta \tilde{u}(n), u_{min}, u_{max})\end{aligned}$$

keep u_ω maximum possible and find constraints for u_v

$$v_{max} = 2\pi R_{wheel} \frac{rpm_{max}}{60}$$

$$v_{upper} = \min(v_{max}, -0.5Lu_{omega} + v_{max})$$

$$v_{lower} = \max(-v_{max}, -0.5Lu_{omega} - v_{max})$$

$$\tilde{u}_v = \text{clip}(u_v, v_{lower}, v_{upper})$$

recommended reading

