MPC control Manual

Akshay
Arjun

Integration:

One of the most basic things when it comes to discrete / digital control, are the mathematical operations that let them happen.

$$\int_{i=0}^{t} P_i \, dt = \sum_{i=0}^{t} \left[\frac{P_i + P_{i+1}}{2}\right] dT$$

here dT is the small sampling time

This is called the trapezoidal rule for integration.

but the issue with this is that its impractical for large integrations:

=> because we will need to know values from $i = 0 \rightarrow t$ thats t different variables.

hence we go with velocity control mode

$$\gamma(+) = \gamma_0 + \sum \left(\frac{P_i + P_{i+1}}{2}\right) \Delta T$$

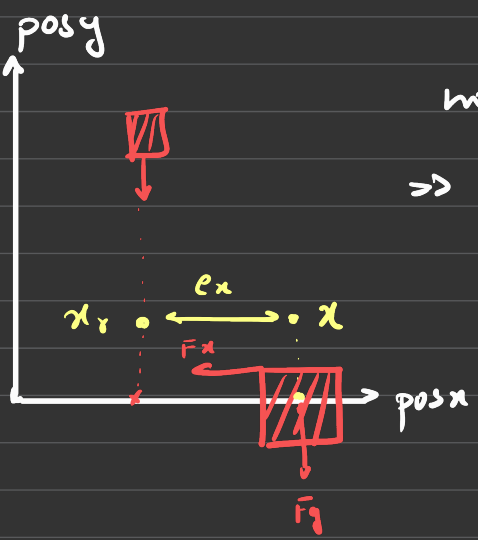
$$\gamma(-1) = \gamma_0 + \sum \frac{P_{i-1} + P_i}{2} \Delta T$$

$$\Rightarrow \gamma(t) - \gamma(t-1) = \left(P_i + \frac{P_{i+1}}{2}\right)\Delta T$$

$$\Rightarrow \boxed{\gamma(t) = \gamma_{t-1} + \left(P_i + \frac{P_{i+1}}{2}\right)\Delta T}$$

next state = prev state + control output

[problem statement :] Control of Magbu to catch randomly falling objects.

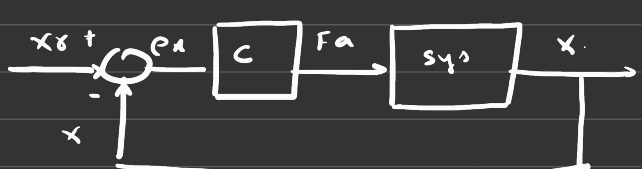


now $posy = \frac{1}{2}gt^2$

$\Rightarrow t = \sqrt{\frac{2y}{g}}$

System modelling :

Control diagram :

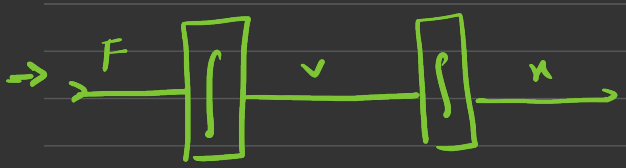


$$F = m\frac{dv}{dt} \Rightarrow \quad v = \int \frac{1}{m} Fa$$

$$\Rightarrow \quad v(i) = v(i-1) + \frac{1}{m} \frac{Fa(i-1) + Fa(i)}{2} \quad -(I)$$

$$\Rightarrow \quad \frac{dx(i)}{dt} = \frac{dx(i-1)}{dt} + \frac{1}{m} \frac{Fa(i-1) + Fa(i)}{2}$$

$$\Rightarrow x(i) = x(i-1) + \frac{1}{m}\left[v(i) + \frac{v(i-1)}{2} \right] \Delta T \quad -(II)$$



- This is system modelling. Take the control input F and return the present state x as described by the two systems above (I) & (II)
- In our case we are using a regular PD controller.

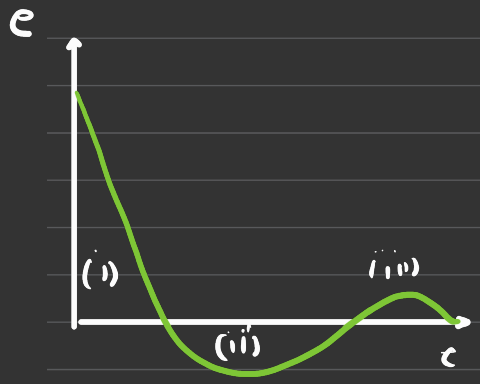
Why not PI? → P cannot handle more than 2 integrations.

=> for a PD controller:

$$F_a = k_p e + k_d \dot{e} \qquad e = x_d - x$$

($k_p e$ → prop, $k_d \dot{e}$ → derivative)

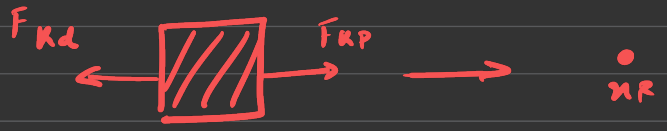
lets do some error modelling:



this is an approx model of how the error is changing wrt time

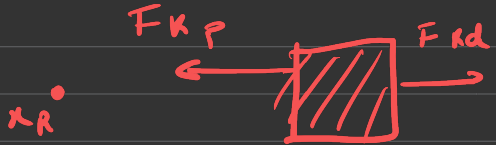
→ here we can see:

(i) the error is +ve and $\dot{e}$ is -ve



(ii) on over shoot: e is -ve and also $\dot{e}$ till the minima

ii b): at mima point + : e still -ve but $\dot{e}$ is +ve



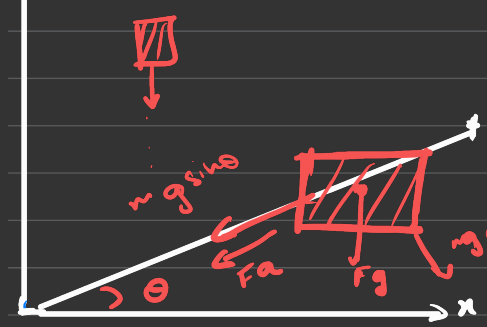
iii): on and over shoot the same thing happens but in inverse

=> hence in some ways Kd is somewhat dampening the motion caused by Kp ;

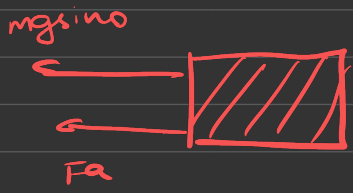
[problem statement 2:] sloped magev

[

posy



now:



now the issue with simple PD controller here is that

@ $e = 0$

$F = mg\sin\theta$

and as the error increases to fix this

imbalance ; eventually body reaches a steady state

where [Fa - Fmgsinθ = 0]

But !! => @ this Fa e ≠ 0

→ this is called steady state error.

- This is one of the demerits of PD controllers.
- Hence, introducing PID !!

now: we have a new term

$Fa = kpe + kd\dot{e} + ki\int e\,dt$

↓

@ steady state the accumulation of error over time will fix the Magnitude.

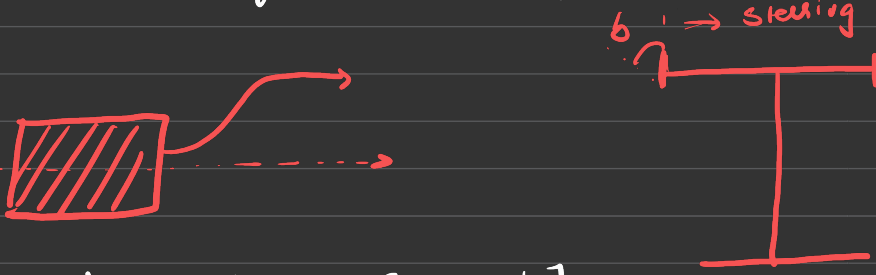
=> It overcomes the steady state error and pushes until "e" = 0 or NOT ET

Car Steering control with MPL

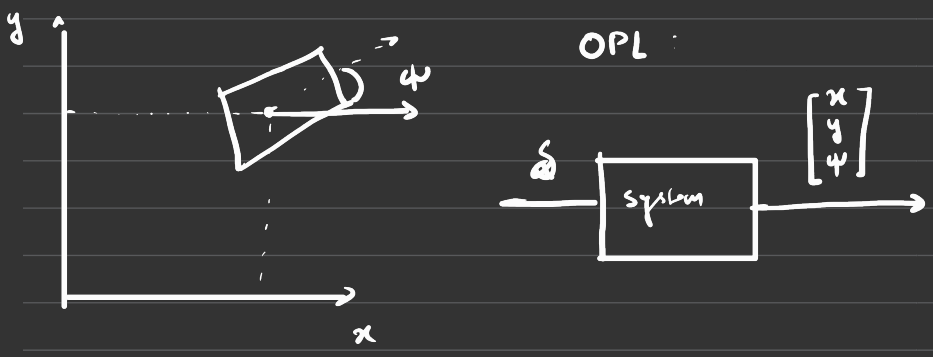
eg: we take a car [lane changing maneuver]

=> constant forward velocity # 1mp point.

δ → steering angle.

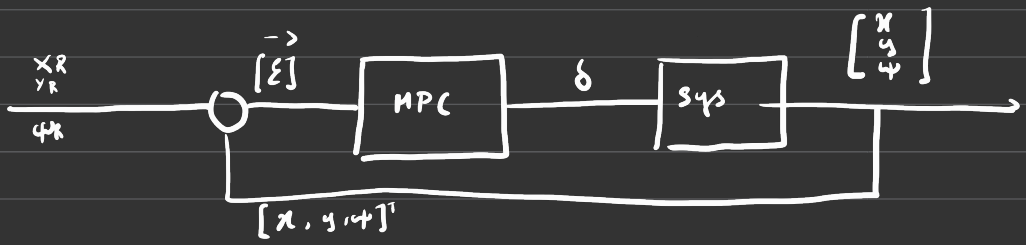


3 DOF systems: $[x, y, \Phi]$



OPL:

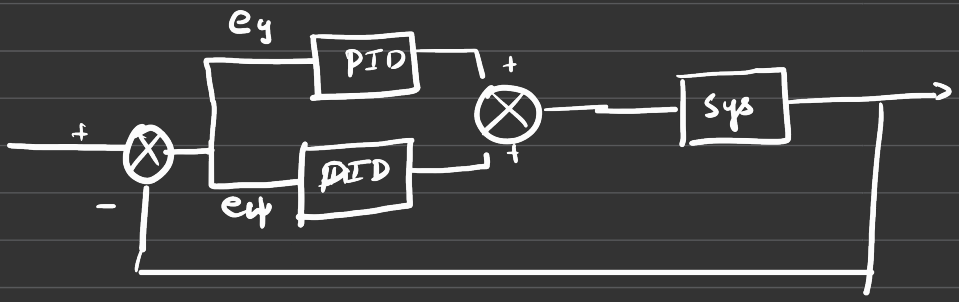
Controller:



now, initially: we made an assumption; horizontal velocity is constant.

⇒ we only need to control y_R and ψ_R.

PID drawbacks: can only handle SISO so we need 2 pids; 2 delays; & we need to combine them.



here in this PID loop we have 6 variables to tune; which is alot.

and if we also combine the linear combination weights, its a total of 8 variables.

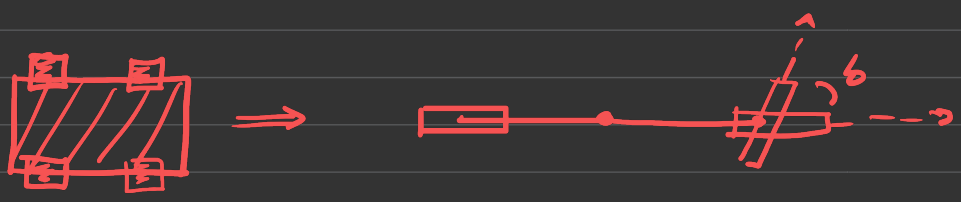
MPC fixes that. MPC is also scalable. MPC is overall better for MIMO systems.

First we work on System block:

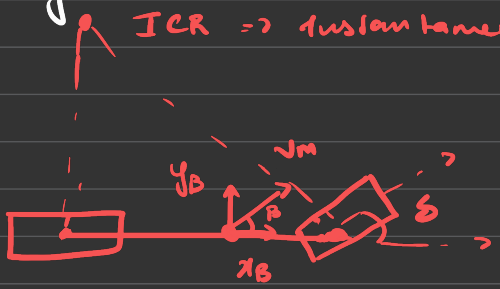
Kinematics & Dynamics of the System

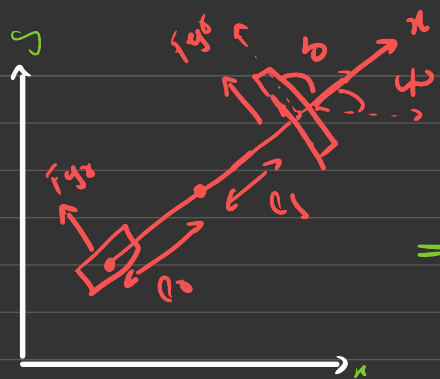
for our case there are different steering model we can use. eg ackermann

however for ease of calculation, we are gonna use bicycle model:



Bicycle model





for lateral direction → in body frame

$$\Sigma F = ma$$

$$\Rightarrow F_{yr} + F_{yf} = ma_y$$

$$\Sigma M = I\alpha$$

$$\Rightarrow [F_{y_f} l_f - F_{y_r} l_r] = I\ddot{\psi}$$

$$\Rightarrow a_y = \ddot{y} + \dot{x}\dot{\psi} - \frac{mv^2}{R} = \dot{x}\left(\frac{\dot{x}}{R}\right) = \dot{x}\dot{\psi}$$

centripetal / rotational correction

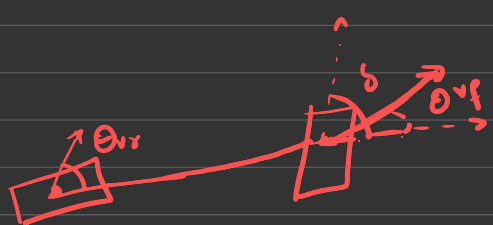
additional: slip angle due to deformation of tires:

we define new angle: θ_{vr}, θ_{vf}

$F_{y_f} \propto \alpha_f$ $\quad$ $F_{y_r} \propto \alpha_r$

$$\alpha_r = \delta_r - \theta_{vr}$$

$$\alpha_f = \delta_f - \theta_{vf}$$



$$\Rightarrow F_{yr} = 2C_{vr}(\delta_r - \theta_{vr})$$
$$F_{yf} = 2C_{vf}(\delta_f - \theta_{vf})$$

$$\Rightarrow F_{yr} = 2C_{vr}(-\theta_{vr})$$
$$F_{yf} = 2C_{vf}(\delta_f - \theta_{vf})$$

$$\theta_{vf} = \text{atan}\left([\dot{y} + \dot{\psi} l_f] / \dot{x} \right)$$

$$\theta_{vr} = \text{atan}\left([\dot{y} - \dot{\psi} l_r] / \dot{x} \right)$$

now ∵ we assume a small steering angle $\tan\theta \approx \theta$

$$= \theta_{vf} = (\dot{y} + \psi l_f) / \dot{x} \qquad \theta_{vr} = \dot{y} - \dot{\psi} l_r / \dot{x}$$

{Now all derived eqns:}

$$F_{yf} + F_{yr} = m[\ddot{y} + \dot{x}\dot{\psi}]$$
$$F_{yf} l_f - F_{yr} l_r = I \ddot{\psi}$$

} Force moment eqns

$$F_{yf} = 2C_{sf}(\delta - \theta_{vf})$$
$$F_{yr} = 2C_{sr}(-\theta_{vr})$$

} deformation derived

$$\theta_f = \frac{\dot{y} + l_f \dot{\psi}}{\dot{x}}$$

$$\theta_r = \frac{\dot{y} + l_r \dot{\psi}}{\dot{x}}$$

} post deformation steering angle

State space:

we need to write the before mentioned eqns in this form:

$$\begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix} = A \begin{bmatrix} y \\ \psi \end{bmatrix} + B\,\delta$$

$$\Rightarrow \dot{\vec{x}} = A\vec{x} + B\vec{u} \rightarrow$$ This is called state space form

⇒ On derivation we get:

$$A = \begin{bmatrix} \underbrace{-2[C_f + C_r]}_{m\dot{x}} \ (a_1) & \underbrace{-2\left[C_f l_f - C_r l_r - \frac{\dot{x}}{m}\right]}_{m\dot{x}} \ (b_2) \\ \underbrace{-2[l_f C_f - l_r C_r]}_{I\dot{x}} \ (a_2) & \underbrace{-2[l_f^2 C_r + l_f^2 C_f]}_{I\dot{x}} \ (b_2) \end{bmatrix}$$

$$B = \begin{bmatrix} 2C_f \ (\delta_1) \\ \frac{2l_f C_f}{I} \ (\delta_2) \end{bmatrix}$$

however the input states aren't $\dot{y}, \dot{\psi}$ rather Y and ψ

hence we add new states and rewrap a new state matrix

$$\vec{X} = [\, \dot{y} \quad \psi \quad \dot{\psi} \quad Y \,]$$

($\dot{y}$ -> body frame $\dot{y}$; Y -> global Y)

$$\dot{x}_1 = \ddot{y} \;;\; x_2 = \dot{\psi} \quad \dot{x}_3 = \ddot{\psi} \quad \dot{x}_4 = \dot{Y}$$

$$\Rightarrow \dot{x}_1 = \ddot{y} = a_1 x_1 + b_1 x_3 + g_1 u$$
$$\dot{x}_2 = x_3$$
$$\dot{x}_3 = a_2 x_1 + b_2 x_3 + g_2 u$$

$$\Rightarrow \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \ddot{y} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & b_1 & 0 \\ 0 & 0 & 1 & 0 \\ a_2 & 0 & a_3 & 0 \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} \dot{y} \\ \psi \\ \dot{\psi} \\ Y \end{bmatrix}$$

Now: $\dot{Y} = \dot{y}\cos\psi + \dot{x}\sin\psi$

@ small ψ = $\dot{Y} = \dot{y} + \dot{x}\psi$

= last column = $[\, 1 \quad \dot{x} \quad 0 \quad 0 \,]$

So we have

$$\begin{bmatrix} \ddot{y} \\ \dot{\psi} \\ \ddot{\psi} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & b_1 & 0 \\ 0 & 0 & 1 & 0 \\ a_2 & 0 & b_2 & 0 \\ 1 & \dot{x} & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \psi \\ \dot{\psi} \\ \gamma \end{bmatrix} + \begin{bmatrix} g_1 \\ 0 \\ g_2 \\ 0 \end{bmatrix} u$$

and the output [which we need to compare with reference states are

$$\begin{bmatrix} \psi \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y} \\ \psi \\ \dot{\psi} \\ \gamma \end{bmatrix} + \underbrace{\begin{bmatrix} 0_{4\times 1} \\ \end{bmatrix}}_{} u$$

hence we have a standard eqns of form

no need for D here

$$\left.\begin{aligned} \dot{X} &= AX + Bu \\ Y &= CX + Du \end{aligned}\right\} \text{ state space fundamental}$$

This is the system modelled

assumptions : i) $\theta_{ref} \to 0$ ii) $\psi \to 0$

iii) $\dot{x} \to$ constant

Integration tips:

since we have $\dot{x}$ we calculate x by

$$\vec{x}_{t+1} = \vec{x}_t + \dot{x}\,\Delta T$$ } euler method → very imprecise.

→ so what we do is loop it over even smaller sample for the same control input u [in this case 5] and calc.

=> eg: $\vec{x}_{\frac{t+1}{n}} = \vec{x}_t + \dot{x}\,\frac{\Delta}{n}$ | sample time = 0.1

eg: $\dot{y}_{0.02} = \dot{y}_0 + \ddot{y}\,\frac{T_s}{5}$

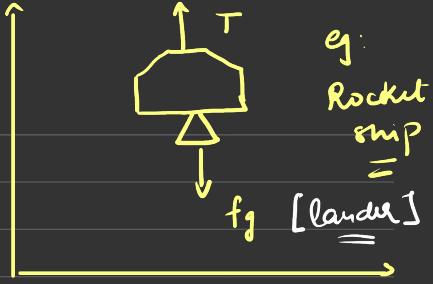
$\dot{y}_{0.04} = \dot{y}_{0.02} + \ddot{y}\,\frac{T_s}{5}$

$\vdots$

$\dot{y}_{0.1} = \dot{y}_{0.08} + \ddot{y}\,\frac{T_s}{5}$

In all these cases control input is kept constant; but states change

The MPC controller



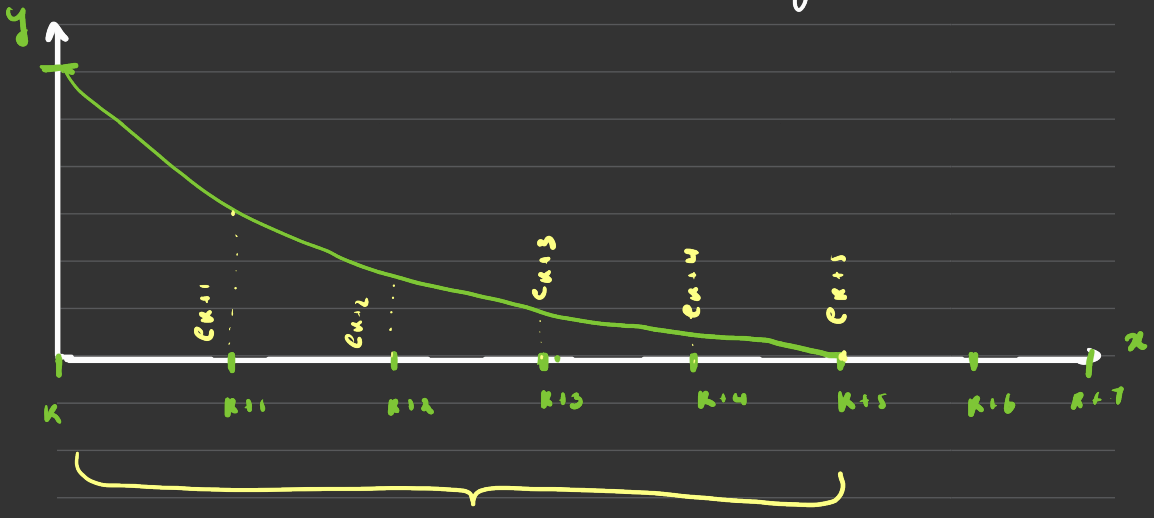
Cost fnc

$$J(i) = w_1 e^2_{k+1} + w_2 e^2_{k+2} + w_3 e^2_{k+3} + w_4 e^2_{k+4} + w_5 e^2_{k+5}$$

↳ cost fnc pt. 1

horizon period

The number of samples ahead of time that the controller takes, ahead of time



horizon period

we find a unique combo of errors, such that it gives us J_{min}.

now each e_{k+1} is dependent on thrust applied on the prev step

=> so we use every e_{k+1} is dependent on the thrust applied at the previous time step.

- while it will give us optimal values, cf might give us extremely impractical thrust values

- hence we add a new part to the cost func

=> we add more dimension to the cost func we are gonna add thrust at each sample time to optimise

$$J_{(2)} = w_6 T_k^2 + w_7 T_{k+1}^2 + w_8 T_{k+2}^2 + w_9 T_{k+3}^2 + w_{10} T_{k+4}^2$$

$$\Rightarrow J_T = w_1 e_{k+1}^2 + w_2 e_{k+2}^2 + w_3 e_{k+3}^2 + w_4 e_{k+4}^2 + w_5 e_{k+5}^2 + w_6 T_k^2 + w_7 e_{k+1}^2 + w_8 e_{k+2}^2 + w_9 e_{k+3}^2 + w_{10} e_{k+4}^2$$

=> optimising (i) gives us a time optimal soln
=> optimising (ii) gives us an energy optimal soln

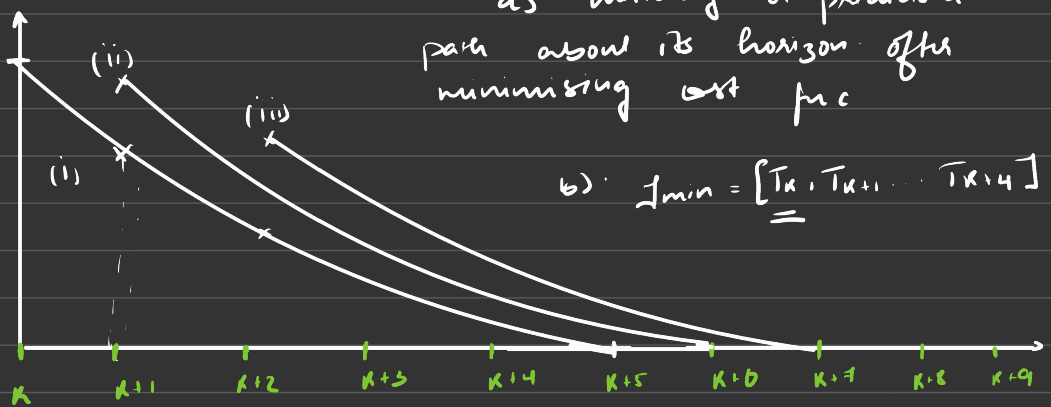
(i) -> because cf minimises errors "quickly"
(ii) -> because it minimises thrust "optimally"

=> assigning weights assign which variable has to be minimised, eg if $W_5 \to 1000$ then e_{k+5} will be a very very small value.

MPC iterative loop :

a) Initially el predicts a path about its horizon after minimising cost fnc

b) $J_{min} = [T_k, T_{k+1} \cdots T_{k+4}]$



=> after taking T_k lets say due to imperfection of model sensor o/p is at (ii) -> repeat process from step (a) but now from point (ii).

=> repeat steps until reference is reached.

Kalman Filtering Application :

=> here we have reached the classic case of where we have an "unreliable model" and imperfect sensor.

⇒ we apply kalman filtering to get a good approximate position after thrust input was taken.

MPI on our car problem:

1) Our state space is on continuous time.
   ⇒ MPC works on discrete "time steps"

⇒ conversion from continuous time to discrete time model

now $\dot{X} = AX + BU$ ⇒ Continuous model.

now: $\dot{x} = \left[\frac{X_{k+1} - X_k}{T}\right]$

⇒ $X_{k+1} = \underbrace{[AT + I]}_{Ad} X_k + \underbrace{BT}_{Bd} u$ — (1)

↳ control.

$Y = \underbrace{CT}_{Cd} X + \underbrace{DT}_{Dd} u$

↳ Observation

Now MPC "predicts" the system across a horizon period. Let the horizon period be 5
@ initial $K = 0 \rightarrow K = 5$

$$
\begin{aligned}
x_1 &= A_d x_0 + B_d \delta_0 \\
x_2 &= A_d x_1 + B_d \delta_1 \\
x_3 &= A_d x_2 + B_d \delta_2 \\
x_4 &= A_d x_3 + B_d \delta_3 \\
x_5 &= A_d x_4 + B_d \delta_4
\end{aligned}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}
=
\begin{bmatrix}
B_d & 0 & 0 & 0 \\
A_d B_d & B_d & 0 & 0 \\
A_d^2 B_d & A_d B_d & B_d & 0 \\
A_d^3 B_d & A_d^2 B_d & A_d B_d & B_d \\
A_d^4 B_d & A_d^3 B_d & A_d^2 B_d & A_d B_d
\end{bmatrix}
\begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix}
+
\begin{bmatrix} A_d \\ A_d^2 \\ A_d^3 \\ A_d^4 \\ A_d^5 \end{bmatrix} x_0
$$

New updated cost fnc: [for matrices J]

$$
J = \frac{1}{2} e_{k+N}^T S e_{k+N} + \sum_{i=0}^{N-1} e_{k+i}^T Q e_{k+i} + \delta_{k+i}^T R_i \delta_{k+i}
$$

S, Q, R are all diagonal matrices;

=> If we look closely it's still quadratic:

eg: $e_{k+N} = \begin{bmatrix} e_{\psi_{k+N}} \\ e_{y_{k+N}} \end{bmatrix} \quad S = \begin{bmatrix} S_\psi & 0 \\ 0 & S_y \end{bmatrix}$

$$
\Rightarrow e^T S e = \begin{bmatrix} e_\psi & e_y \end{bmatrix} \begin{bmatrix} S_\psi e_\psi \\ S_y e_y \end{bmatrix} = \underline{\underline{\begin{bmatrix} S_\psi e_\psi^2 + S_y e_y^2 \end{bmatrix}}}
$$

=> The reason we assign a separate weight S is so that the last term can be reduced as much as needed.

State Change is from $x \rightarrow \tilde{x}$ ie: $\delta \rightarrow \Delta\delta$

$$\Delta\delta_k = \delta_k - \delta_{k-1}$$
$$\delta_k = \Delta\delta_k + \delta_{k-1}$$

$$x_{k+1} = A_d x_k + B\delta_k = A_d x_k + B_d \Delta\delta + B_d \delta_{k-1}$$
$$\delta_k = \Delta\delta_k + \delta_{k-1}$$

$$\begin{bmatrix} x_{k+1} \\ \delta_k \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ \delta_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta\delta_k$$

$$\tilde{x}_{k+1} \quad \tilde{A} \quad \tilde{x}_k \quad \tilde{B} \quad u$$

$$y_k = \begin{bmatrix} C_d & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \delta_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \Delta\delta$$

$$y_k = \tilde{C} \quad \tilde{x}_k \quad \tilde{D} \quad u$$

$$\begin{bmatrix} \tilde{x}_{k+1} \\ \tilde{x}_{k+2} \\ \tilde{x}_{k+3} \\ \tilde{x}_{k+4} \\ \tilde{x}_{k+5} \end{bmatrix} = \begin{bmatrix} \tilde{B} & & & & \\ \tilde{A}\tilde{B} & \tilde{B} & & & \\ \tilde{A}^2\tilde{B} & \tilde{A}\tilde{B} & \tilde{B} & & \\ \tilde{A}^3\tilde{B} & \tilde{A}^2\tilde{B} & \tilde{A}\tilde{B} & \tilde{B} & \\ \tilde{A}^4\tilde{B} & \tilde{A}^3\tilde{B} & \tilde{A}^2\tilde{B} & \tilde{A}\tilde{B} & \tilde{B} \end{bmatrix} \begin{bmatrix} \Delta\delta_1 \\ \Delta\delta_2 \\ \Delta\delta_3 \\ \Delta\delta_4 \\ \Delta\delta_5 \end{bmatrix} + \begin{bmatrix} \tilde{A} \\ \tilde{A}^2 \\ \tilde{A}^3 \\ \tilde{A}^4 \\ \tilde{A}^5 \end{bmatrix} \tilde{x}_k$$

$$X_m \quad \bar{\bar{C}} \quad \Delta\delta_m \quad \hat{\tilde{A}}$$

final recursive eqn : $X_{k+1} = \bar{\bar{C}}\,\Delta\delta_{k+1} + \hat{\bar{A}}\,X_k$

Cost fnc simplification for MPC

=> check the PDF on MPC cost fnc for derivation;

Final cost fnc : $\bar{\bar{H}}$

$$J = \frac{1}{2}\Delta u_k^T\left(\bar{\bar{C}}^T\bar{\bar{Q}}\,\bar{\bar{C}} + \bar{\bar{R}}\right)\Delta\delta_{k+1} + \left[\hat{x}_k^T\ \gamma_k^T\right] J \begin{bmatrix} \hat{\bar{A}}^T\bar{\bar{Q}}\,\bar{\bar{C}} \\ -\bar{\bar{T}}\,\bar{\bar{C}} \end{bmatrix}\Delta\delta_{k+1}$$

$\bar{\bar{F}}^T$

$$\Rightarrow J = \frac{1}{2}\Delta\vec{U}_k^T\,\bar{\bar{H}}\,\Delta\delta_{k+1} + \left[\hat{x}_k^T\ \gamma_k^T\right]\bar{\bar{F}}^T\Delta\delta_{k+1}$$

now $\nabla J = \bar{\bar{H}}\,\Delta\vec{U}_k + \bar{\bar{F}}\begin{bmatrix}\vec{x}_k \\ \vec{r}_k\end{bmatrix} = 0$

$$\Rightarrow \Delta\delta_{k+1} = -\bar{\bar{H}}^{-1}\bar{\bar{F}}\begin{bmatrix}\vec{x}_k \\ \vec{r}_k\end{bmatrix}$$

The Control Architecture