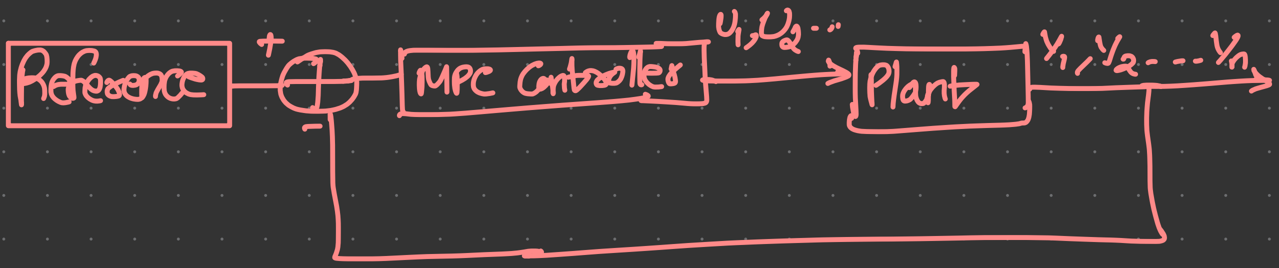
Model Predictive Control

→ Feedback control algorithm uses model to make future outputs of a process

Example: Keeping car in lane

→ Can handle multi-input multi-output systems
- ↳ Change in second output can affect change in first output etc
- ↳ Designing larger controller will increase complexity

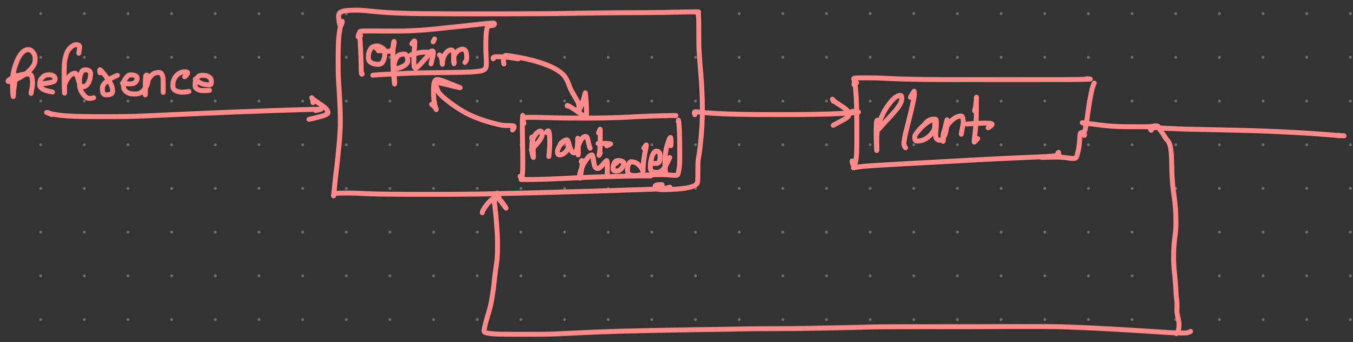


→ Can handle constraints due to rules, physical limitations etc

→ Has preview capablity [know in advance similar to feed-forward]

→ Solves an optimisation problem to select optimal control

→ Objective is to predict future



→ P = Prediction Horizon [How far ahead we look into the future]

→ Cost function is used to predict the simulation movements and identify best-case scenario.

↳ Constraints also contribute

Predicted with smallest J [Cost Function] is most stable

↳ After predict MPC applies only first step of predict at current time step

MPC is also called [Receding Horizon Control]

Variables sent to plant from MPC are called manipulated variables

Variables given by plant as output is called output variables

State estimator estimates states based on output variables that are sent to MPC controller

MPC Design Parameters

→ Sample Time (T_s) - Rate of execution of control algorithm

If too big, controller won't recognise (react)

If too small, react too much increasing computational effort

T_r - Time taken to go from 10% - 90% of steady state response

Ideally, $\frac{T_r}{20} \leq T_s \leq \frac{T_r}{10}$

→ Control Horizon(m): Smaller the control horizon, fewer the computations

Ideally, $0.1p \leqslant m \leqslant 0.2p$, p: Prediction horizon

Types of constraints:

→ Hard: Can't be violated

→ Soft: Can be violated

Ideally keep soft constraints for output. Also don't keep hard constraints on both input, input rate simultaneously

Has multiple goals:

1) Setpoint - tracking

2) Smooth control moves

} View input weights and outputs by comparing them and prioritise

If Linear system + Linear constraints + Quadratic cost function

⇓

Linear Time-invariant MPC

[Convex Optimisation problem - find global optimum]

If Non-linear system ⇒ can still use linear MPC [Adaptive MPC & Gain-scheduled MPC]

← Based on linearisation

In adaptive MPC, a linear model is computed on the fly as operating conditions change. After each timestep we update internal plant model used by MPC controller with linear model.

In adaptive MPC, the structure of optimisation problem remains same across different operating points

So, no of states & operating constraints remains the same across various operating points

⇓

If they change, use gain-scheduled MPC

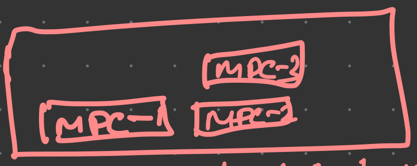
↵ linearise at operating points of interest

↳ Design linear MPC at each op. point. [All controllers are independent to each other]

Need switching ← algorithm

↓

Uses more memory than adaptive MPC



Gain-scheduled MPC

Finally

Use adaptive MPC when: structure of optimisation problem remains fixed across different operating conditions

Use gain-scheduled MPC when: structure changes across different operating points

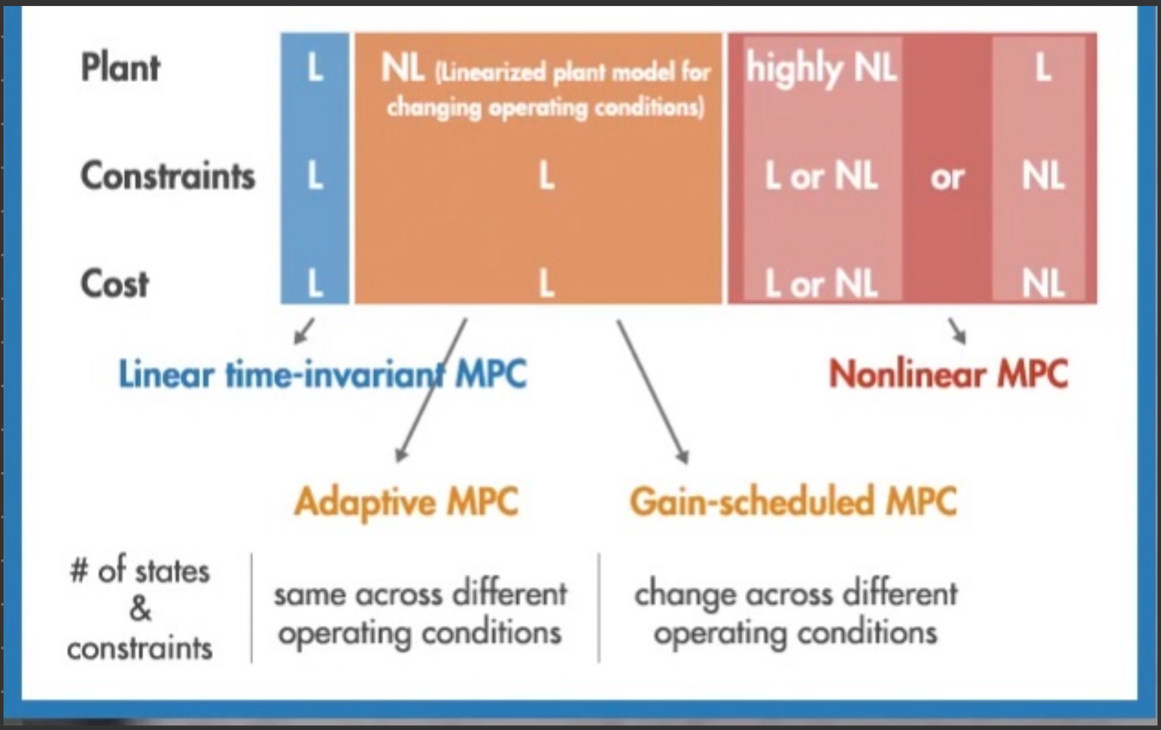
If everything is non-linear, then use non-linear MPC.

It is most powerful as it uses most accurate representation of plant

↳ Predictions made by them are more accurate, better control actions.

↳ More complex to solve in real-time as we get non-convex optimization problem

Conclusions:



Plant	L	NL (Linearized plant model for changing operating conditions)	highly NL		L
Constraints	L	L	L or NL	or	NL
Cost	L	L	L or NL		NL

Linear time-invariant MPC

Adaptive MPC

Gain-scheduled MPC

Nonlinear MPC

# of states & constraints	same across different operating conditions	change across different operating conditions