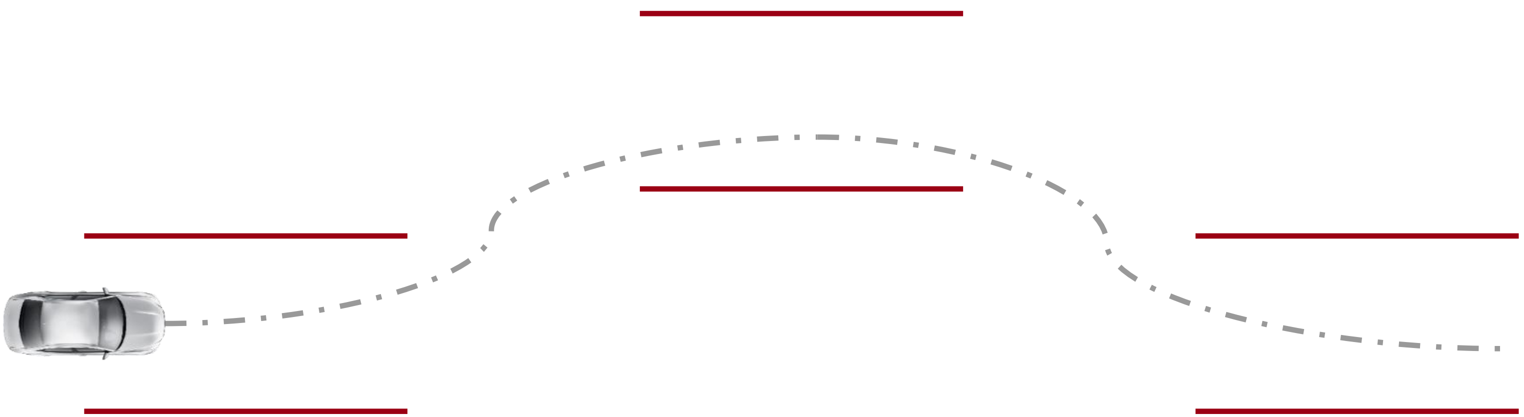
An introduction to Model Predictive Control (MPC): motivation, linear derivation, examples of application

L. Caiaffa, M. Bruschetta

Model Predictive Control: an illustrative example



- Representative scenario: obstacle avoidance maneuver at high speed
- Common approach:
  1. Find optimal trajectory (vehicle dependent)
  2. Apply a path following control technique (e.g. PID, LQR), possibly based on vehicle dynamic

Model Predictive Control: an illustrative example

- A better choice could be the following: **we both want to follow the desired speed, satisfying road constraints**

- This can be written as

This is a Model Predictive Control problem!

$$
\begin{aligned}
&\min_{x,u} \sum_{k=1}^{N} \|v_k - \bar{v}\|_Q^2 \\
&s.t.\ \ \dot{x} = f(x,u) \\
&\qquad x_{min} \leq x \leq x_{max} \\
&\qquad u_{min} \leq u \leq u_{max}
\end{aligned}
$$

Following the velocity reference

Vehicle dynamics (can be non-linear)

Road constraints and input constraint (maximum force, maximum steering)

This automatically generates a trajectory controlling the vehicle

+

Model Predictive Control: receding horizon principle

- Emulating human actions:
  1. We want to stay in the track and maintain constant velocity
  2. We choose the best action based on predicted future behavior of the vehicle (model based)
  3. We react to an unexpected system behavior (model mismatch, disturbances)
  4. We recompute the best action based on the actual state

➢ **MPC acts in the same way: receding horizon principle requires to solve an optimization problem at each time step and to apply only the first control action**

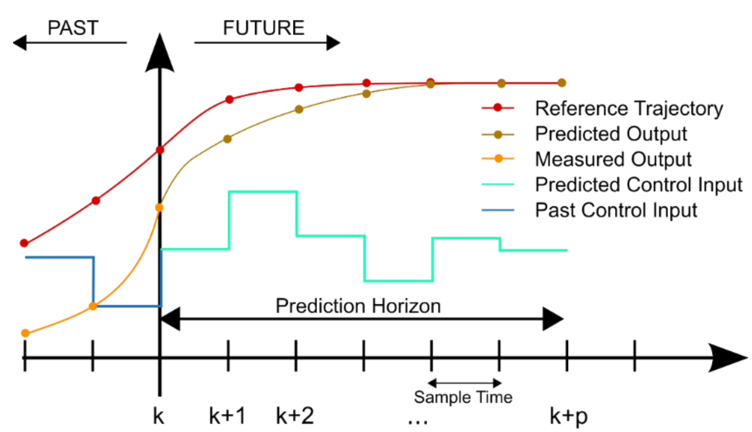
Model Predictive Control: general formulation

- A general MPC problem can be written as follows

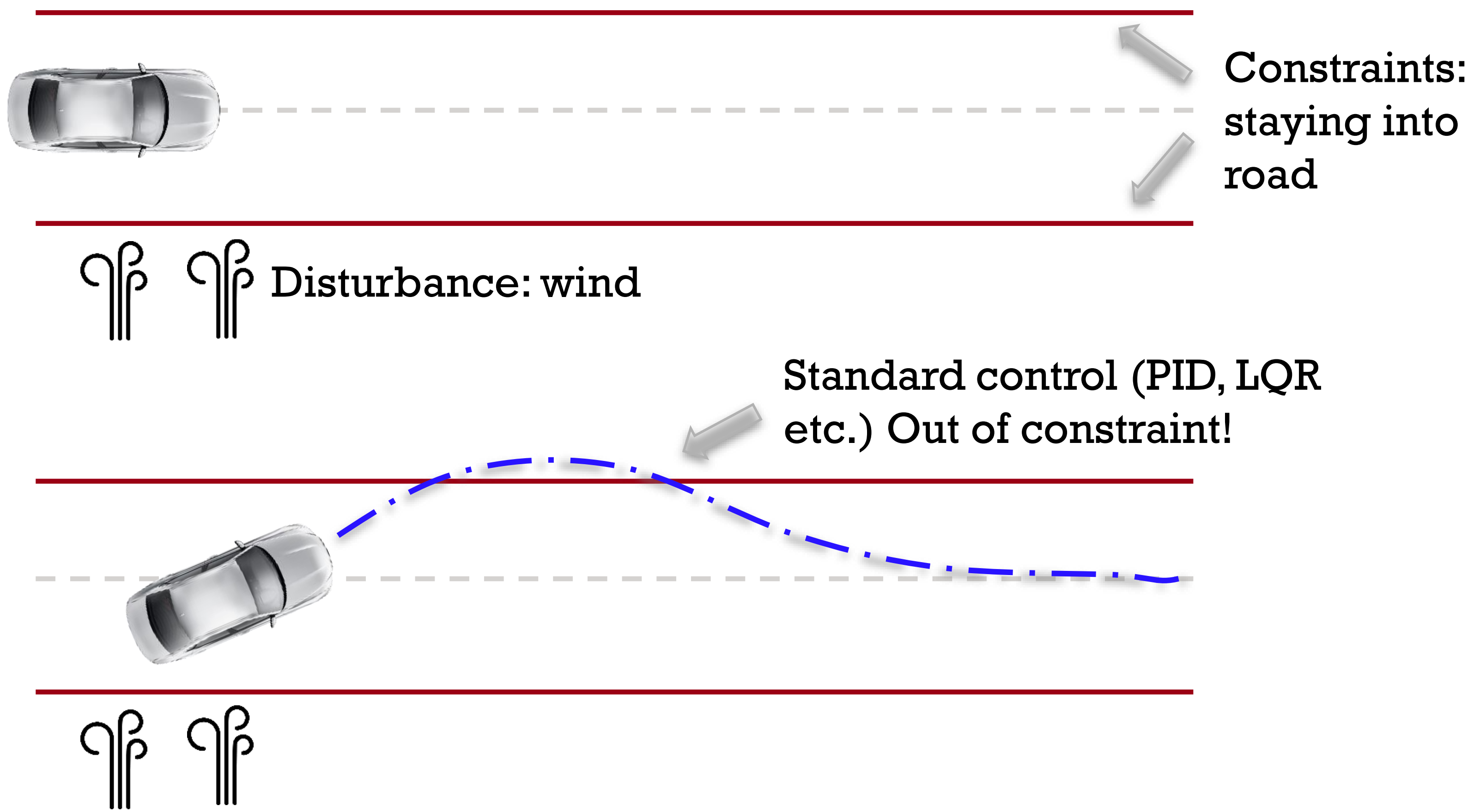
$$\min_{x,u} \sum_{k=0}^{N-1} \|x_k - \bar{x}\|_Q^2 + \|u_k - \bar{u}\|_R^2 + \|x_N - \bar{x}\|_S^2$$

$$\begin{aligned} s.t.\ \ & \dot{x} = f(x,u) \\ & x_0 = \hat{x} \\ & x_{min} \le x \le x_{max} \\ & u_{min} \le u \le u_{max} \end{aligned}$$

- What MPC do:
  1. **Minimizes** distances from references while fulfilling state/input **constraints**
  2. Uses the model for predicting future states/outputs/inputs of the system
  3. Apply the first input only
  4. Estimate x_k and restart

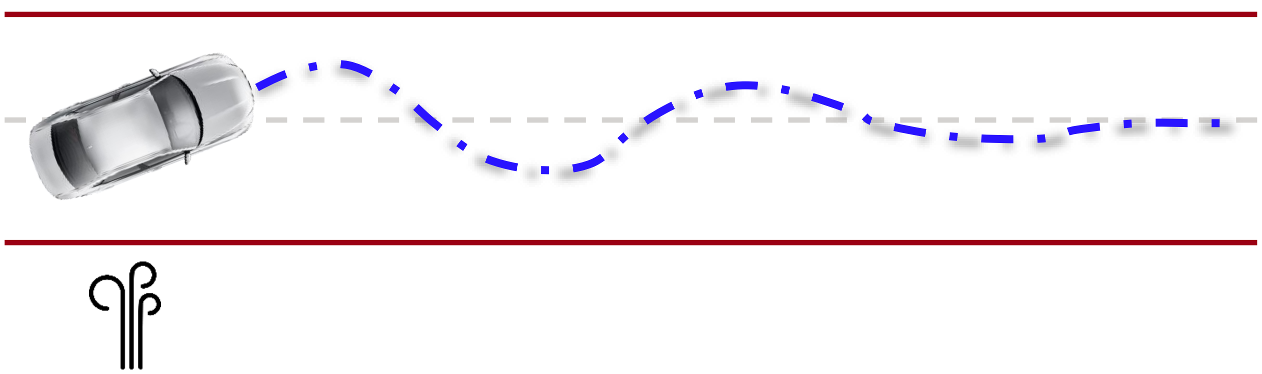


Model Predictive Control: another point of view



Model Predictive Control: an illustrative example

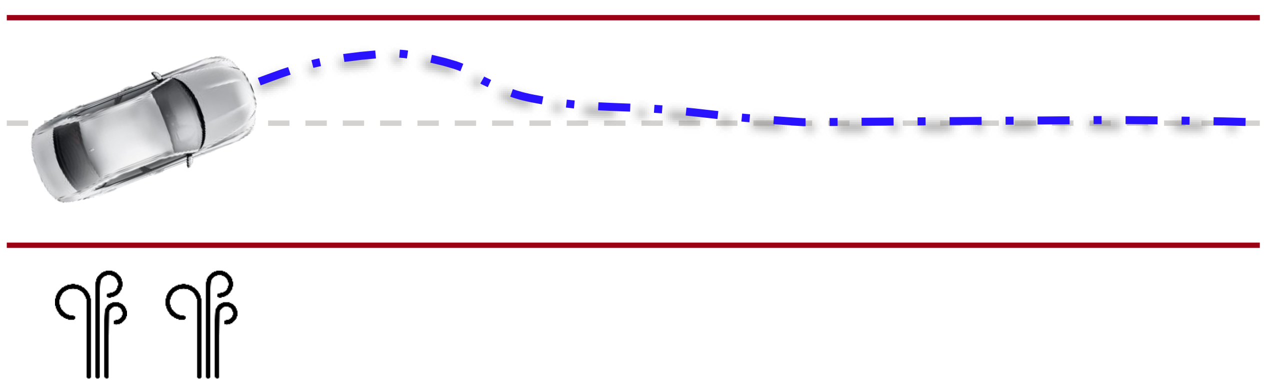
- How to avoid exiting the road? Designing the controller based on the (hypothetical) maximum wind force
  1. **Disturbance must be taken into account**
  2. **Controller has to be robust**
  3. **Can interfere with comfort**



Model Predictive Control: an illustrative example

- What we want from this controller?
  - **To be aware of the constraints (road limits)**
  - **To know what will happen to the vehicle after the applied control**

➢ **Model Predictive Control satisfies these requests**



+

Linear Model Predictive Control

$$\min_{x,u} \sum_{k=0}^{N-1} \|x_k - \bar{x}\|_Q^2 + \|u_k - \bar{u}\|_R^2 + \|x_N - \bar{x}\|_S^2$$

$$\begin{aligned} s.t.\ \ \dot{x} &= Ax + Bu \\ x_{min} &\leq x \leq x_{max} \\ u_{min} &\leq u \leq u_{max} \end{aligned}$$

Linear state space model

- We will focus on linear-time-invariant (**LTI**) systems

$$x((k+1)T) = Ax(kT) + Bu(kT)$$
$$y(kT) = Cx(kT) + Du(kT)$$

with $A \in \mathbb{R}^{nxn}$, $B \in \mathbb{R}^{nxm}$, $C \in \mathbb{R}^{pxn}$ and $D \in \mathbb{R}^{pxm}$.

- We will assume the **state to be known**, then the estimate of the state $\hat{x}(kT|kT)$ will be equal to the actual state.

- We will consider **discrete-time systems with sampling time T** as an actual discrete system or a discretized version of a continuous time one

State predictions for linear systems

- Given the initial state as $\hat{x}(kT|kT)$, the state evolution at successive time steps $k+1, k+2, \ldots$ results

$$
\begin{aligned}
\hat{x}((k{+}1)T|kT) &= A\hat{x}(kT|kT) + Bu(kT) \\
\hat{x}((k{+}2)T|kT) &= A\hat{x}((k{+}1)|kT) + Bu((k+1)T) \\
&= A^2\hat{x}(kT|kT) + ABu(kT) + Bu((k+1)T) \\
\hat{x}((k{+}3)T|kT) &= A\hat{x}((k{+}2)|kT) + Bu((k+2)T) \\
&= A^3\hat{x}(kT|kT){+}A^2Bu(kT){+}ABu((k{+}1)T){+}Bu((k{+}2)T
\end{aligned}
$$

- For a generic N we have (from induction), that the state evolution results

$$\hat{x}((k{+}N)T|kT) = A^N\hat{x}(kT|kT) + \sum_{j=1}^{N} A^{N-j}Bu((k+j-1)T)$$

State prediction: matrix version

- By omitting the sampling time T, consider the vectors of states and inputs at all times as

$$\hat{\mathcal{X}}(k) = [\hat{x}^T(k+1|k)\ \hat{x}^T(k+2|k) \dots\ \hat{x}^T(k+N|k)]^T \in \mathbb{R}^{Nn}$$
$$\mathcal{U}(k) = [u^T(k)\ u^T(k+1) \dots\ u^T(k+N-1)]^T \in \mathbb{R}^{Nm}$$

- Then, we can write all state prediction in a compact matrix version as

$$\underbrace{\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{x}(k+2|k) \\ \hat{x}(k+3|k) \\ \vdots \\ \hat{x}(k+N|k) \end{bmatrix}}_{\hat{\mathcal{X}}(k)\in\mathbb{R}^{Nn\times 1}} = \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{A}\in\mathbb{R}^{Nn\times n}} \hat{x}(k|k) + \underbrace{\begin{bmatrix} B & 0 & 0 & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \vdots & & \ddots & \ddots & \\ A^{N-1}B & A^{N-2}B & \dots & AB & B \end{bmatrix}}_{\mathcal{B}\in\mathbb{R}^{Nn\times Nm}} \underbrace{\begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \\ \vdots \\ u(k+N-1) \end{bmatrix}}_{\mathcal{U}(k)\in\mathbb{R}^{Nm\times 1}}$$

$$\boxed{\hat{\mathcal{X}}(k) = \mathcal{A}\hat{x}(k|k) + \mathcal{B}\cdot\mathcal{U}(k)}$$

Output prediction for linear systems

- The output predictions are easily obtained by state predictions as

$$\hat{y}(k+N|k) = CA^N\hat{x}(k|k) + \sum_{j=1}^{N} CA^{N-j}Bu(k+j-1)$$

- Even the sequence of output predictions can be written in compact form as

$$\underbrace{\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}}_{\hat{\mathcal{Y}}(k)} = \underbrace{\begin{bmatrix} C & & & \\ & C & & \\ & & \ddots & \\ & & & C \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} \hat{x}(k+1|k) \\ \hat{x}(k+2|k) \\ \vdots \\ \hat{x}(k+N|k) \end{bmatrix}$$

$$\mathcal{Y}\hat{(}k) = \mathcal{C}\hat{\mathcal{X}} = \mathcal{C}\mathcal{A}\cdot\hat{x}(k|k) + \mathcal{C}\mathcal{B}\cdot\mathcal{U}k)$$

MPC for linear systems: QP formulation

- The considered framework is:
  - Linear system $\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{cases}$
  - Quadratic cost function

$$\sum_{i=0}^{N-1} \left(\hat{x}^T(k+i|k) Q \hat{x}(k+i|k) + u^T(k+i) R u(k+i) \right) + \hat{x}(k+N|k)^T S \hat{x}(k+N|k)$$

  - A set of linear contraints on input and state, written in matrix form, as

$$\begin{aligned} F_{k+i} u(k+i) &\leq f_{k+i} & i &= 0, \dots N-1 \\ G_{k+i} \hat{x}(k+i|k) &\leq g_{k+i} & i &= 0, \dots N \end{aligned}$$

Cost function compact form

- Defining the weight matrices as

$$\mathcal{Q} = \begin{bmatrix} Q & & & & 0 \\ & Q & & & \\ & & \ddots & & \\ & & & Q & 0 \\ 0 & & & 0 & S \end{bmatrix} \qquad \mathcal{R} = \begin{bmatrix} R & & & 0 \\ & R & & \\ & & \ddots & \\ 0 & & & R \end{bmatrix}$$

and using the condensed version of state and input sequences, it is possible to re-write the cost function

$$J = \sum_{i=0}^{N-1} \left(\hat{x}^T(k+i|k) Q \hat{x}(k+i|k) + u^T(k+i) R u(k+i) \right) + \hat{x}(k+N|k)^T S \hat{x}(k+N|k)$$

$$\Downarrow$$

$$J = \hat{\mathcal{X}}^T(k) \mathcal{Q} \hat{\mathcal{X}}(k) + \mathcal{U}^T(k) \mathcal{R} \mathcal{U}(k)$$

+

Cost function condensing: dependence on input only

- It is possible to further simplify the problem

$$J = \hat{\mathcal{X}}^T(k)\mathcal{Q}\hat{\mathcal{X}}(k) + \mathcal{U}^T(k)\mathcal{R}\mathcal{U}(k) \quad \longleftarrow \mathcal{X}(k) = \mathcal{A}\hat{x}(k|k) + \mathcal{B}\mathcal{U}(k)$$

$$= (\mathcal{A}\hat{x}(k|k) + \mathcal{B}\mathcal{U}(k))^T\mathcal{Q}(\mathcal{A}\hat{x}(k|k) + \mathcal{B}\mathcal{U}(k)) + \mathcal{U}^T(k)\mathcal{R}U(k)$$

$$\begin{aligned} &= \hat{x}(k|k)^T\mathcal{A}^T\mathcal{Q}\mathcal{A}\hat{x}(k|k) + \\ &\quad + \hat{x}(k|k)^T\mathcal{A}^T\mathcal{Q}\mathcal{B}\mathcal{U}(k) + \mathcal{U}(k)^T\mathcal{B}^T\mathcal{Q}\mathcal{A}\hat{x}(k|k) \\ &\quad\quad + \mathcal{U}(k)^T\mathcal{B}^T\mathcal{Q}\mathcal{B}\mathcal{U}(k) + \mathcal{U}^T(k)\mathcal{R}\mathcal{U}(k) \end{aligned}$$

- Since we cannot modify the initial state, $\hat{x}(k|k)^T\mathcal{A}^T\mathcal{Q}\mathcal{A}\hat{x}(k|k)$ can be discarded from the optimization problem.

- From basic matrix properties, the index can be written in the QP-form as

$$J := 2\underbrace{\hat{x}(k|k)^T\mathcal{A}^T\mathcal{Q}\mathcal{B}}_{\text{vector}}\mathcal{U}(k) + \mathcal{U}(k)^T \underbrace{\left(\mathcal{B}^T\mathcal{Q}\mathcal{B} + \mathcal{R}\right)}_{\text{positive definite matrix}} \mathcal{U}(k)$$

Constraints condensing: dependence on input only

- The constraints can be written in compact form as

$$\underbrace{\begin{bmatrix} F_k & 0 & & \\ 0 & F_{k+1} & & \\ & & \ddots & \\ & & & F_{k+N-1} \end{bmatrix}}_{\mathcal{F}} \mathcal{U}(k) - \underbrace{\begin{bmatrix} f_k \\ f_{k+1} \\ \vdots \\ f_{k+N-1} \end{bmatrix}}_{f} < 0 \qquad \underbrace{\begin{bmatrix} G_{k+1} & 0 & & \\ 0 & G_{k+2} & & \\ & & \ddots & \\ & & & G_{k+N} \end{bmatrix}}_{\mathcal{G}} \hat{\mathcal{X}}(k) - \underbrace{\begin{bmatrix} g_{k+1} \\ g_{k+2} \\ \vdots \\ g_{k+N} \end{bmatrix}}_{g} < 0$$

$$\mathcal{F}\mathcal{U}(k) - f < 0 \qquad \mathcal{G}\hat{\mathcal{X}}(k) - g < 0$$

- By recalling that $\hat{\mathcal{X}}(k) = \mathcal{A}\hat{x}(k|k) + \mathcal{B}\mathcal{U}(k)$, state constraints can be written as a function on input only as

$$\mathcal{G}\mathcal{B}\mathcal{U}(k) - (g + \mathcal{G}\mathcal{A}\hat{x}(k|k)) < 0$$

Basic of optimization: quadratic programming (QP)

- Consider the following optimization problem

$$\begin{array}{ll} \min_{x} & \frac{1}{2}x^T Qx + c^T x \\ \text{s.t.} & Ax - b \leq 0 \end{array}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $Q^T = Q \succ 0 \in \mathbb{R}^{n \times n}$.

- Why is QP problem interesting?
  - Well-posed optimization problem: **convex problem**, convergence guaranteed
  - **Efficient algorithms** are known
  - General formulation for **optimization on linear system** (shown in next slides)

MPC problem as QP optimization

- **MPC can be written as a QP** $\begin{array}{ll} \min & \frac{1}{2}x^T Q x + c^T x \\ x \text{ s.t.} & Ax - b \leq 0 \end{array}$ **problem**

$$\begin{array}{ll} \min & \mathcal{U}(k)^T \left(\mathcal{B}^T \mathcal{Q} \mathcal{B} + \mathcal{R}\right) \mathcal{U}(k) + 2\hat{x}(k|k)^T \mathcal{A}^T \mathcal{Q} \mathcal{B} \mathcal{U}(k) \\ \mathcal{U}(k) \text{ s.t.} & \\ \multicolumn{2}{l}{\begin{bmatrix} \mathcal{F} \\ \mathcal{G}\mathcal{B} \end{bmatrix} \mathcal{U}(k) - \begin{bmatrix} f \\ g - \mathcal{G}\mathcal{A}\hat{x}(k|k) \end{bmatrix} < 0} \end{array}$$

- Optimization problem in $\mathcal{U}(k)$ only
- Constraints both consider input and state cases (state constraints need to be feasible)
- If only input constraints are present, a solution always exists

Solving the QP problem

- A QP problem

$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x + c^T x \\ x \text{ s.t.} & Ax - b \leq 0 \end{array}$$

  has **no closed form solution**

- Solution obtained via iterative methods (active set, interior point)

- Trivial case: **if no constraint is present**

$$\mathcal{U}^{\text{opt}}(k) = -\underbrace{\left(\mathcal{B}^T \mathcal{Q} \mathcal{B} + \mathcal{R}\right)^{-1} \mathcal{B}^T \mathcal{Q} \mathcal{A}}_{\mathcal{K}} \hat{x}(k|k)$$

  Same solution obtained with LQR! (e.g. segway)

+

MATLAB Pseudocode for Linear MPC

- Matlab can solve a QP problem with the command

```
x = quadprog(Q,c,A,b)
```

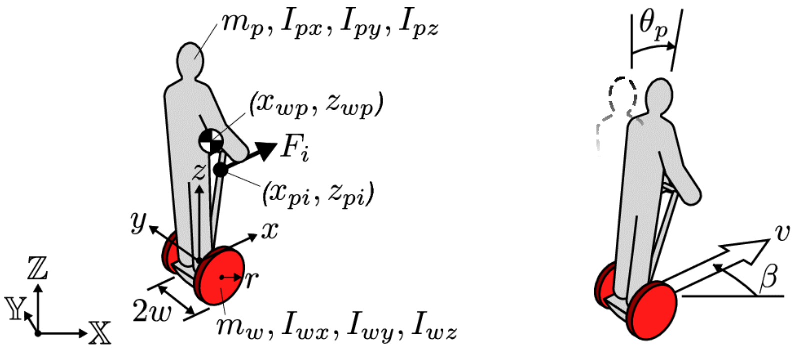
$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x + c^T x \\ x \text{ s.t.} & Ax - b \leq 0 \end{array}$$

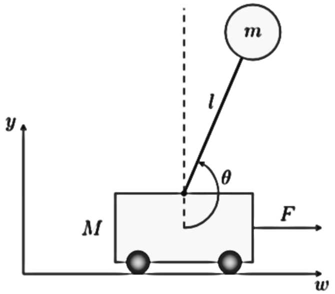
- Pseudocode: at each time step

1. *Obtain the state (measured or estimated):* $\hat{x}(k|k)$
2. *Build matrices* $\mathcal{A}_C$, $\mathcal{B}_C$, $\mathcal{M}_C$, $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{F}$, f, $\mathcal{G}$ *and* g
3. *Set*
$$Q \longleftarrow \mathcal{B}_C^T \mathcal{Q} \mathcal{B}_C + \mathcal{R}$$
$$c \longleftarrow \left(\left(\mathcal{A}_C \hat{x}(k|k) + \mathcal{M}_C \mathcal{D}(k) - \mathcal{Y}_0(k) \right)^T \mathcal{Q} \mathcal{B}_C - \mathcal{U}_0^T \mathcal{R} \right)^T$$
$$A \longleftarrow \begin{bmatrix} \mathcal{F} \\ \mathcal{G}\mathcal{B} \end{bmatrix}$$
$$f \longleftarrow \begin{bmatrix} f \\ g - \mathcal{G}\mathcal{A}\hat{x}(k|k) \end{bmatrix}$$
4. *Solve* `x = quadprog(Q,c,A,b)`
5. *Apply the first control input to the system*

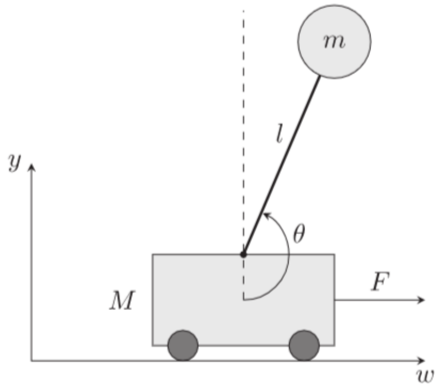
+ Linear MPC application: inverted pendulum

Why inverted pendulum?





Inverted pendulum: model



$$\ddot{w}_0 = \frac{ml\sin(\theta)\dot{\theta}^2 + mg\cos(\theta)\sin(\theta) + u}{M + m - m(\cos(\theta))^2},$$

$$\ddot{\theta} = -\frac{ml\cos(\theta)\sin(\theta)\dot{\theta}^2 + u\cos(\theta) + (M+m)g\sin(\theta)}{l(M + m - m(\cos(\theta))^2)}$$

- The equations have to be linearized around π, leading to

$$\begin{bmatrix} \dot{w}_0 \\ \ddot{w}_0 \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(m+M)g}{Ml} & 0 \end{bmatrix} \begin{bmatrix} w_0 \\ \dot{w}_0 \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml} \end{bmatrix} u$$

Inverted pendulum: Operative Example

- The cost function is

$$\min_{u,x} \sum_{i=1}^{N} \left(\hat{x}(i+1) - x_r(i+1)\right)^T Q \left(\hat{x}(i+1) - x_r(i+1)\right) + u(i)^T R u(i)$$
$$s.t.\, x(0) = \hat{x}$$
$$\hat{x}(k+1) = \mathrm{F}(\hat{x}(\mathrm{k}), \mathrm{u}(\mathrm{k}))$$
$$u_{lb} < u < u_{ub}$$
$$x_{lb} < \hat{x} < x_{ub}$$

- Model is discretized with ZOH with sampling time T=0.1s

- Control task: position tracking of the cart. Control depends on:
  1. **Tuning of weight matrices**
  2. **Constraints on input and state**
  3. **Horizon length**
  4. Discretization time and control frequency

Inverted pendulum: MPC tuning

- How to tune the cost function
  - Most common choice: diagonal matrices

Kart position

Kart velocity

$$Q = \begin{bmatrix} ? & 0 & 0 & 0 \\ 0 & ? & 0 & 0 \\ 0 & 0 & ? & 0 \\ 0 & 0 & 0 & ? \end{bmatrix} \qquad R = ?$$

Input force

Pendulum position

Pendulum angular velocity

- Better to set $R > 0$ for several reasons
  - $R = 0$ would allow an infinite amount of force (Sampling time?)
  - Numerical problems
  - Typical choice $R \ll 1$, $R = 0.001$

Inverted pendulum: MPC tuning

- First attempt

Kart position

Kart velocity

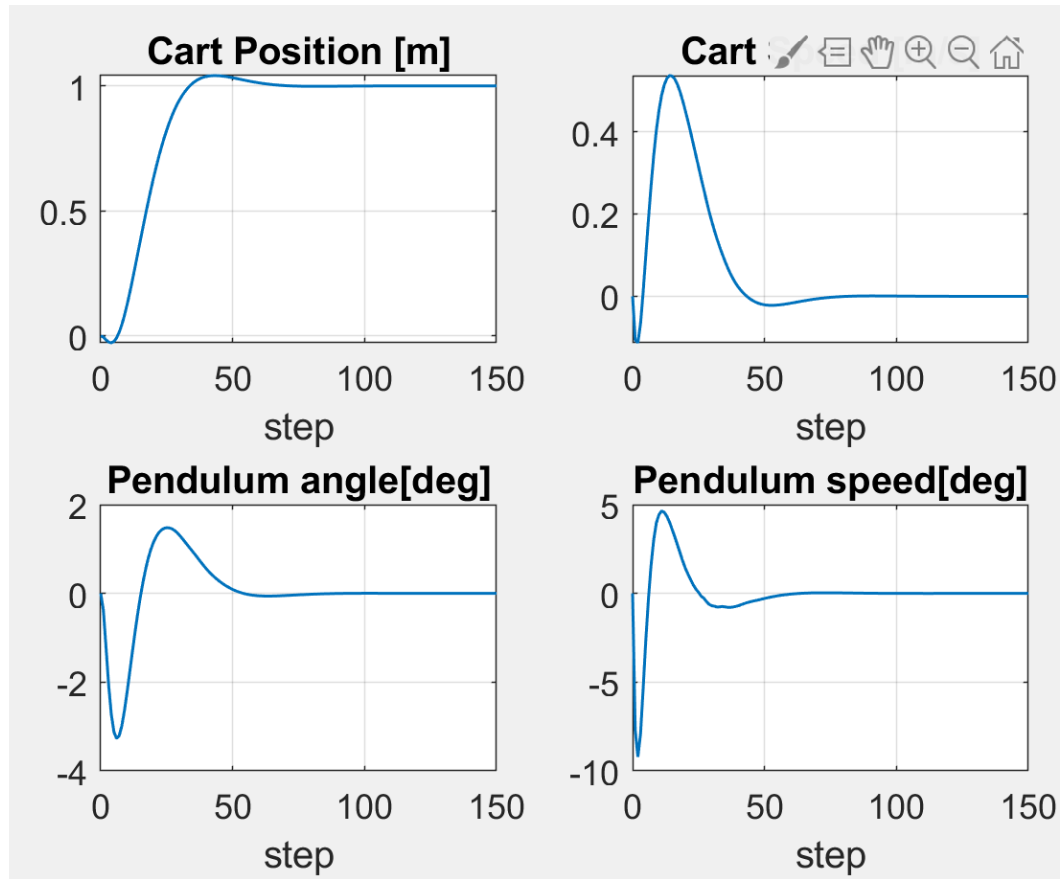
$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

Pendulum position

Pendulum angular velocity

$R = 0.01$

$N = 50$



Inverted pendulum: MPC tuning

Kart position

Kart velocity

$$Q = \begin{bmatrix} \mathbf{10} & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

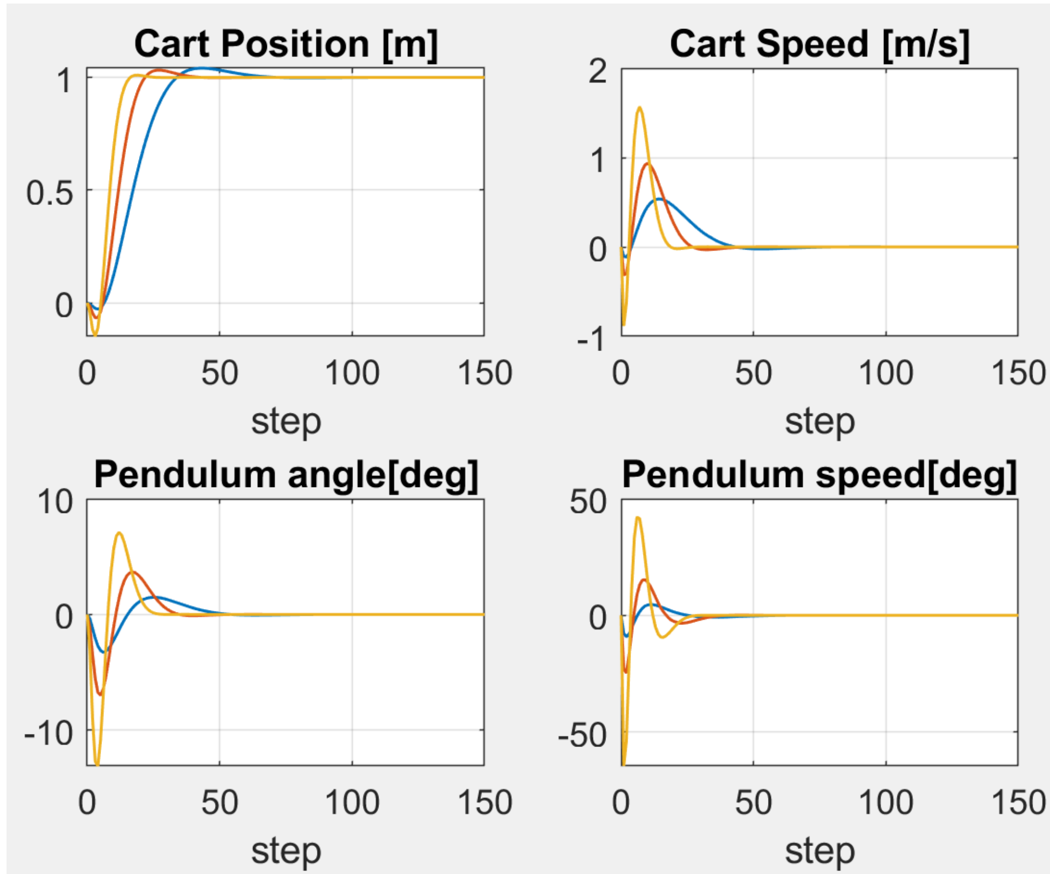
Pendulum position

Pendulum angular velocity

$$Q = \begin{bmatrix} \mathbf{100} & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$R = 0.01$

$N = 50$



Inverted pendulum: input and state constraint

- Recalling the MPC problem:

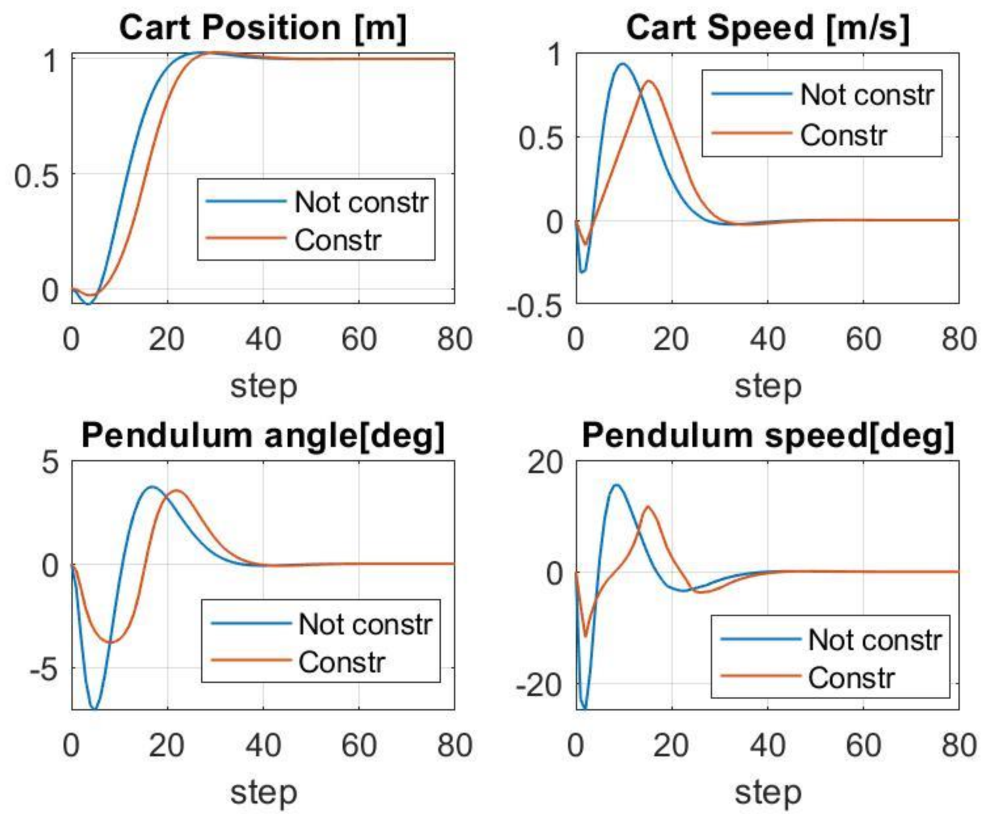
$$\min_{u,x} \sum_{i=1}^{N} \big(\hat{x}(i+1) - x_r(i+1)\big)^T Q \big(\hat{x}(i+1) - x_r(i+1)\big) + u(i)^T R u(i)$$

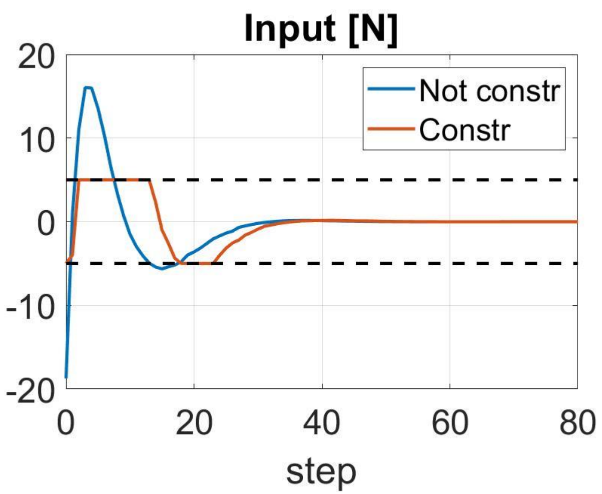
$$\begin{aligned} s.t.\ & x(0) = \hat{x} \\ & \hat{x}(k+1) = \mathrm{F}(\hat{x}(\mathrm{k}), \mathrm{u}(\mathrm{k})) \\ & \boldsymbol{u_{lb}} < \boldsymbol{u} < \boldsymbol{u_{ub}} \\ & \boldsymbol{x_{lb}} < \boldsymbol{\hat{x}} < \boldsymbol{x_{ub}} \end{aligned}$$

- Consider the following constraints on angle and input:
  - $\boldsymbol{-5 < u < +5}$ limitation on input (e.g. saturation)
  - $\boldsymbol{-5 < u < +5}$, $\boldsymbol{-1 < \hat{\theta} < 1}$: strong limitation on tilt angle

Inverted pendulum: input constraint

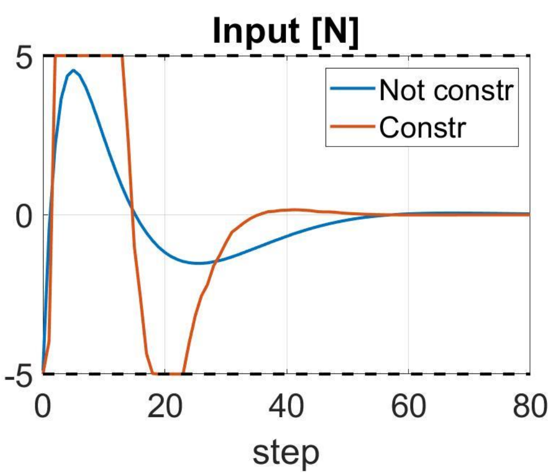
- The framework is the one as before
- **We explicitly consider controller saturation**

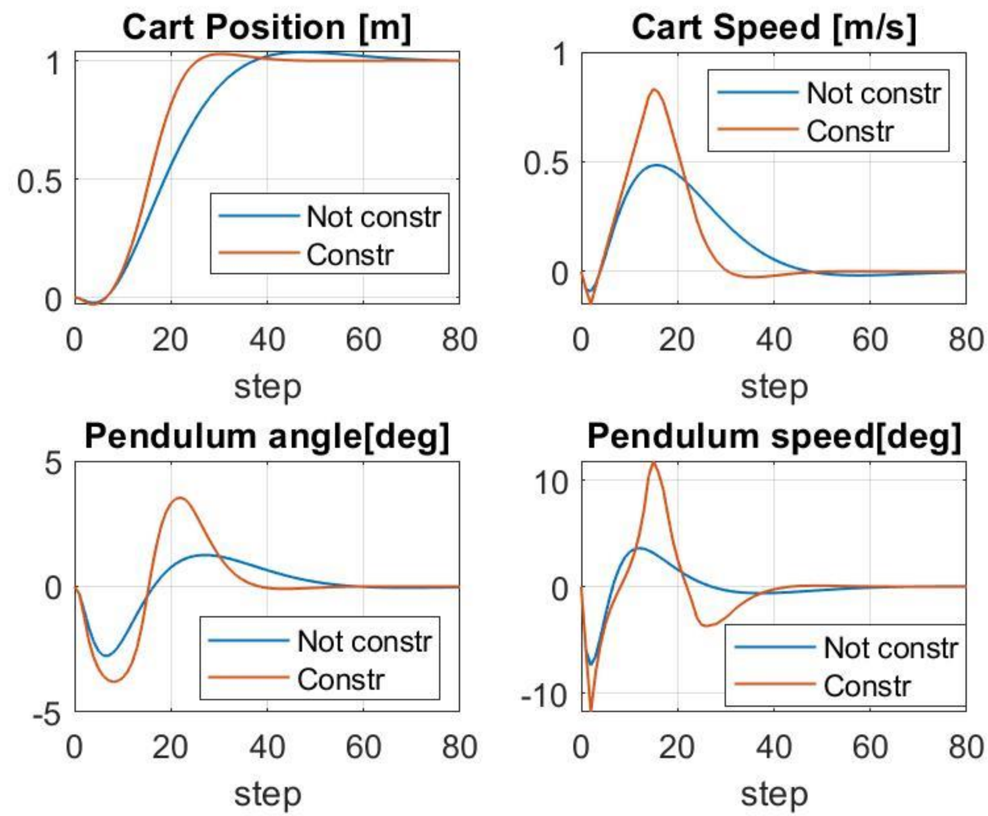




Inverted pendulum: conservative tuning for fulfilling constraints

- We want the controller to be limited but without saturation
- Slow response!

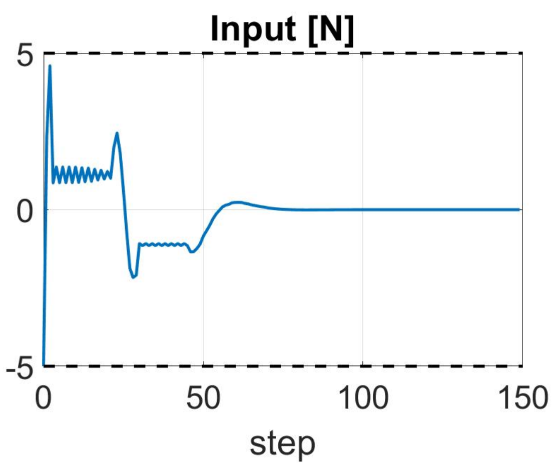


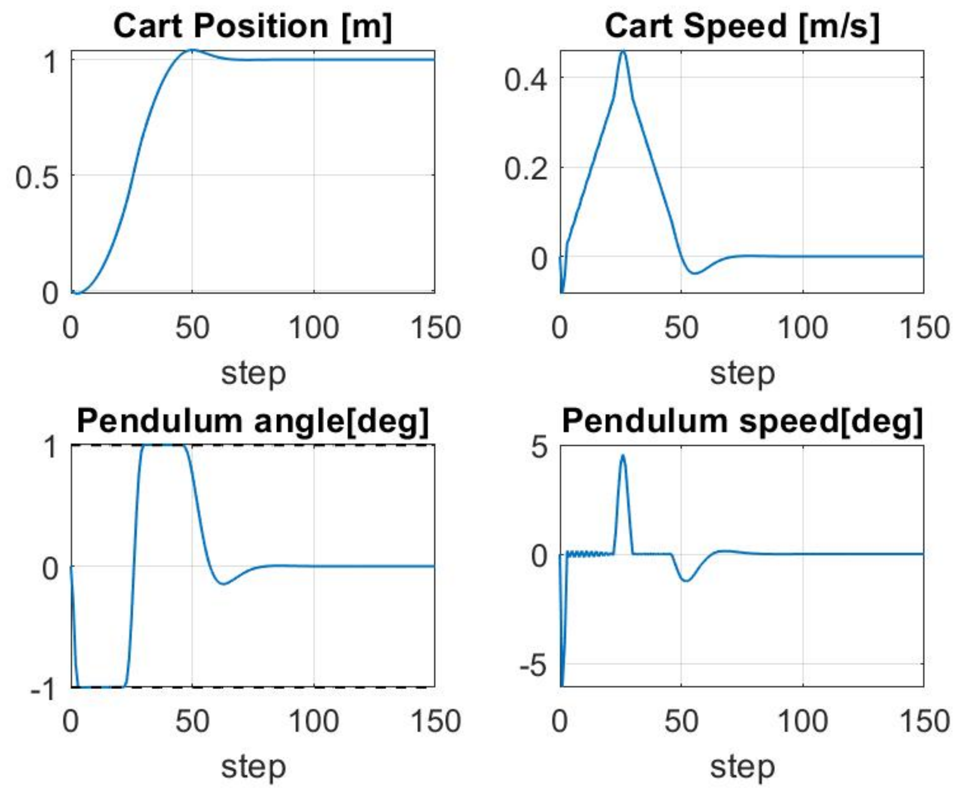


Inverted pendulum: input and state constraint

- **Constraint on pitch angle $|\theta| < 1°$**

- System is slower in stabilizing, but constraint is satisfied





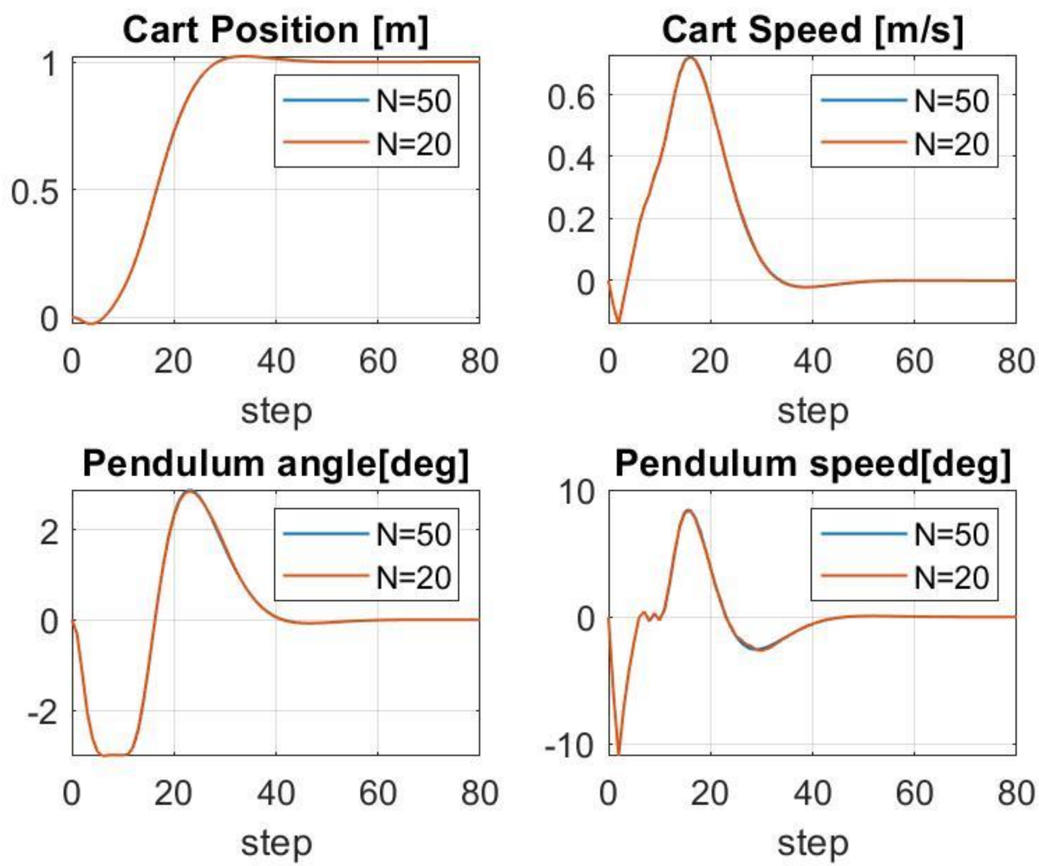
Inverted pendulum: horizon length considerations

- Why do we need a tradeoff on the horizon length?
  - Long N:
    - Good dynamics preview
    - High computational burden
  - Short N
    - Less knowledge about future dynamics
    - Less calculation

- **General rule: horizon length combined with sampling time (N*T) has to cover the main dynamics of the system**

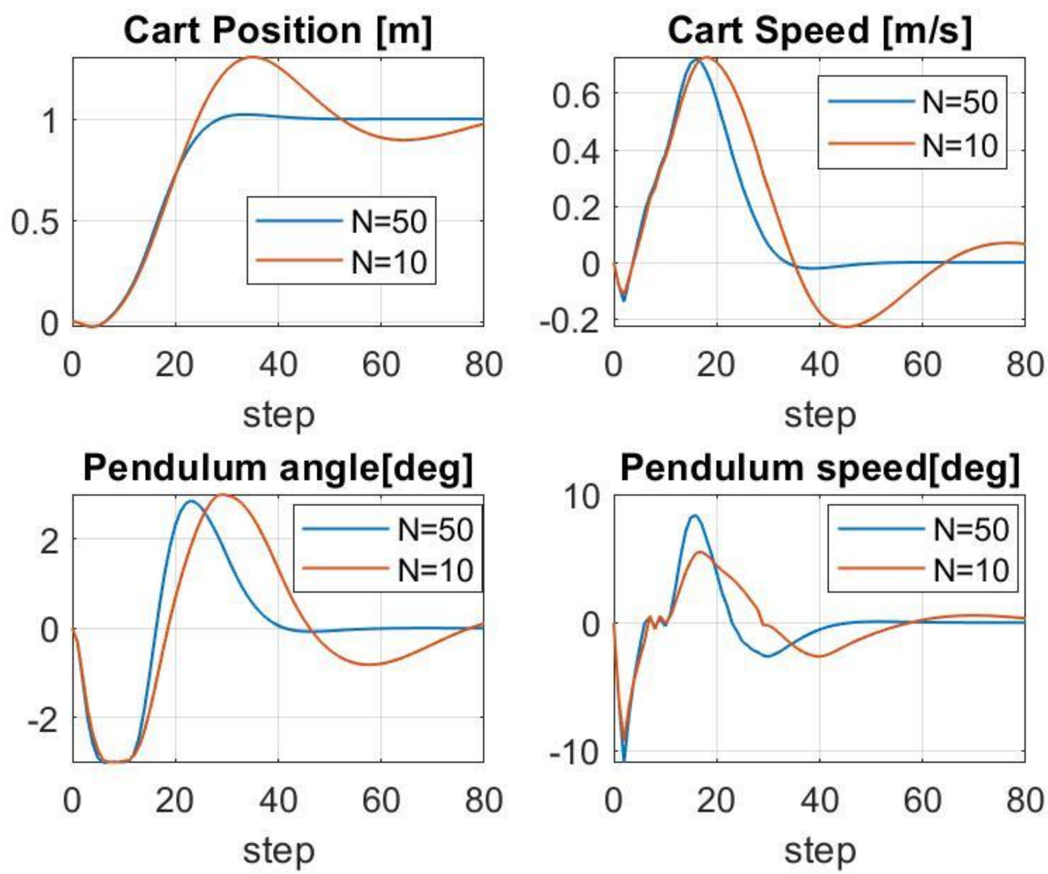
Inverted pendulum: horizon length N=50 vs N=20

- With N=20 the time span is 2s, enough for describe pendulum dynamics
- Moving to N=20 reduces computational burden



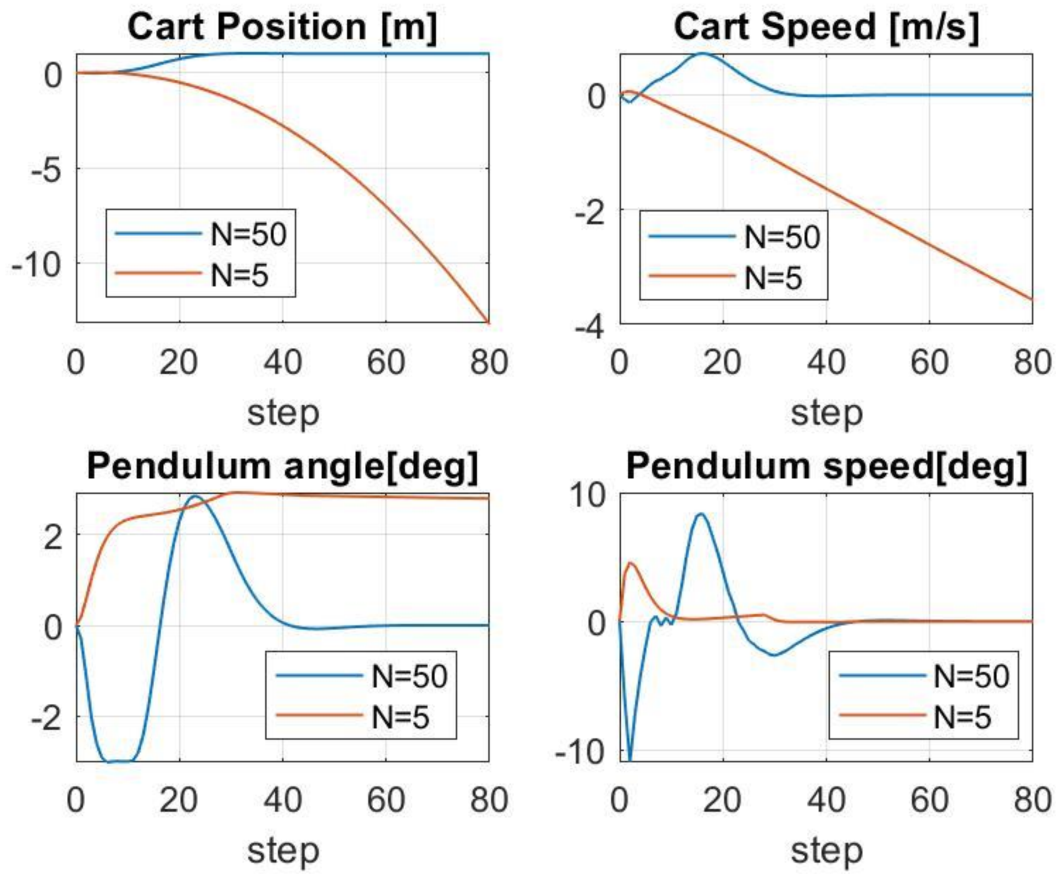
Inverted pendulum: horizon length N=50 vs N=10

- With N=10 the time span is 1s,
- Slower response
- Tends to oscillate more
- Dynamics is not entirely previewed



Inverted pendulum: horizon length N=50 vs N=5

- With N=10 the time span is 0.5s
- Horizon is too short for stabilizing the system
- **System diverges!**



+ State of the art: Non-linear MPC, autonomous virtual driver

State-of-the-art: Non linear MPC

$$\min_{x,u} \sum_{k=0}^{N-1} \|x_k - \bar{x}\|_Q^2 + \|u_k - \bar{u}\|_R^2 + \|x_N - \bar{x}\|_S^2$$

$$s.t.\quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$$

$$\boldsymbol{g}_{\boldsymbol{min}} \le \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) \le \boldsymbol{g}_{\boldsymbol{max}}$$

- State-of-the-art framework is **non-linear MPC**
  - Better characterization of dynamics
  - Combination of nonlinear dynamics and nonlinear constraints achieves better performances

- Difficulties?
  - Solving the (non-linear) optimization problem
  - Using in real-time on fast systems (e.g. vehicles)

Virtual driver: motivations and problem statement

- Virtual prototyping tools are nowadays widely used in automotive industry
  - Reduction of time-to-market
  - Reduction of costs for real prototypes
  - Increase safety
- Reproducing a realistic driver behavior strongly affects reliability of virtual simulation tools
  - Need for a human-like Virtual Driver

[1] J. A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on. IEEE, 2013, pp. 2335–2340
[2] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," Optimal Control Applications and Methods, vol. 36, pp. 628–647, 09 2015

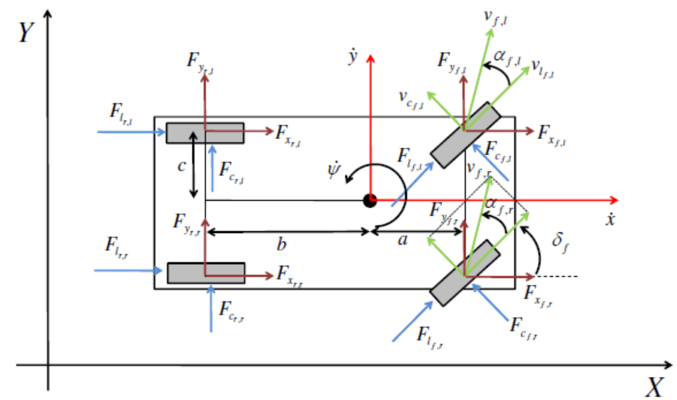
Car dynamical model: inputs and outputs

- Input vector $u = [\delta_f, \gamma]$
  - δ_f is the steering angle
  - γ is the normalized throttle\braking effort

- Output vector/optimization variables

$$h = [e_v, \quad e_\psi + \beta, \quad \dot{e}_\psi, \quad \beta\ ]$$

  - e_v is the velocity error
  - $\dot{e}_\psi$ is the heading error velocity
  - β is the sideslip angle

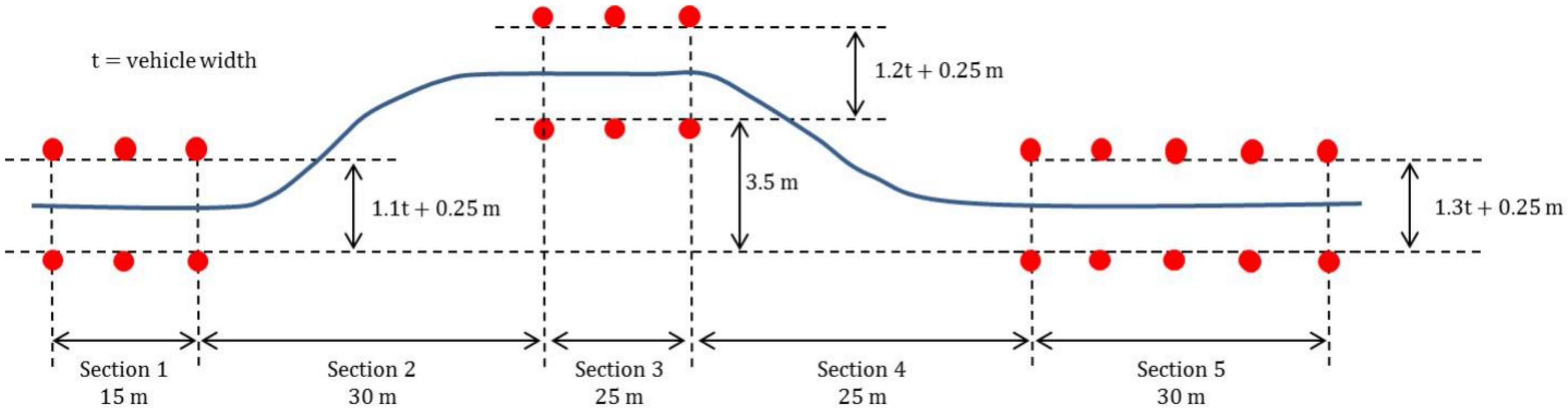


Cost function

$$J = \sum_{k=0}^{N_p} (h)^T Q (h) + \sum_{k=0}^{N_p-1} u^T R u$$

Test case: Double Lane Change, as the one of the illustrative example

- Constant reference velocity ($\bar{v} = 30.5\ \frac{m}{s^2}$)
- Controller maintains velocity while staying within the cones



Live Animation

MPC setup:

- Sampling space

  $T_s = 1m$

- Prediction horizon

  $N = 50m$

- Control frequency

  $F_c = 100\ Hz$

Research activity

Research activity

- **Applicative** projects
  - Learning-based NMPC for
    - Racing four-wheel vehicles
    - Autnomous kart
    - Racing motorcycles
  - NMPC for
    - Driving Assistance in Human Machine Interaction strategies
    - Hydrofoils electric ships
    - Driving simulators
    - Electric drives
    - Anti-wheelie for Motorcycles
    - Warfarin
    - Photo Bioreactor

- **Methodological** projects
  - Learning-based NMPC using GP regression models
  - NMPC computational effort reduction
  - NMPC Automatic Tuning based on genetic algorithms
  - NMPC robustness

- Contacts

  - Ph.D. Mattia Bruschetta mattia.bruschetta@dei.unipd.it
  - Prof. Alessandro Beghi alessandro.beghi@unipd.it