MPC for Trajectory Tracking

Agus Hasan

University of Southern Denmark

October 6, 2020

Outline

- Nonlinear Programming Problem (NLP)
- CasADi (an open-source tool for nonlinear optimization and algorithmic differentiation)
- Model Predictive Control
- Trajectory Tracking

Nonlinear Programming Problem (NLP)

- NLP is the process of solving an optimization problem where some of the constraints or the objective function are nonlinear.
- A standard problem formulation in numerical optimization

$$
\begin{aligned}
& \min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) \\
s.t. \quad & \boldsymbol{g}_1(\boldsymbol{w}) \leq 0 \\
& \boldsymbol{g}_2(\boldsymbol{w}) = 0
\end{aligned}
$$

- Φ, $\boldsymbol{g}_1$, and $\boldsymbol{g}_2$ are usually assumed to be differentiable.
- Linear Programming: if Φ, $\boldsymbol{g}_1$, and $\boldsymbol{g}_2$ are linear.
- Quadratic Programming: if Φ is quadratic, while $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are linear.

Nonlinear Programming Problem (NLP)

Minimization or Maximization

$$\begin{aligned} &\min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) \\ s.t. \quad &\boldsymbol{g}_1(\boldsymbol{w}) \leq 0 \\ &\boldsymbol{g}_2(\boldsymbol{w}) = 0 \end{aligned}$$

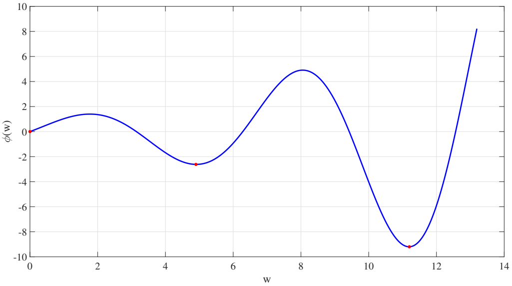
is equivalent to

$$\begin{aligned} &\max_{\boldsymbol{w}} -\Phi(\boldsymbol{w}) \\ s.t. \quad &\boldsymbol{g}_1(\boldsymbol{w}) \leq 0 \\ &\boldsymbol{g}_2(\boldsymbol{w}) = 0 \end{aligned}$$

Nonlinear Programming Problem (NLP)

Local vs Global

$$
\begin{aligned}
& \min_{w} e^{0.2w} \sin(w) \\
s.t. \quad & w \geq 0 \\
& x \leq 4\pi
\end{aligned}
$$



Nonlinear Programming Problem (NLP)

Solution of the optimization problem

$$\begin{aligned} & \min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) \\ s.t. \quad & \boldsymbol{g}_1(\boldsymbol{w}) \leq 0 \\ & \boldsymbol{g}_2(\boldsymbol{w}) = 0 \end{aligned}$$

Normally we are looking at the value of $\boldsymbol{w}$ that minimizes our objective

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \Phi(\boldsymbol{w})$$

By direct substitution we can get the corresponding value of the objective function

$$\begin{aligned} \Phi(\boldsymbol{w}^*) &= \Phi(\boldsymbol{w})|_{\boldsymbol{w}^*} \\ &= \min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) \end{aligned}$$

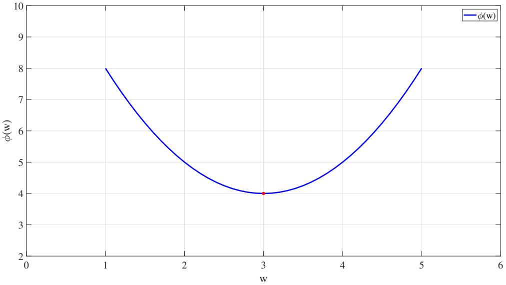
Nonlinear Programming Problem (NLP)

Find the local minimum of the following function

$$\mathbf{\Phi}(w) = w^2 - 6w + 13$$

Writing this problem as an NLP, we have

$$\min_{w} w^2 - 6w + 13$$



CasADi

- `https://web.casadi.org/get/`
- Has a general scope of numerical optimization
- In particular, it facilitates the solution of NLP
- Free & open-source (LGPL), also for commercial use
- 4 standard problems can be handled by CasADi
  - QP
  - NLP
  - Root finding
  - Initial value Problem (IVP) for ODE and DAE

CasADi

```
% CasADi v3.4.5
addpath('C:\Users\mehre\OneDrive\Desktop\CasADi\casadi-windows-matlabR2016a-v3.4.5')
import casadi.*

x = SX.sym('w'); % Decision variables
obj = x^2-6*x+13 ; % calculate obj

g = [];  % Optimization constraints - empty (unconstrained)
P = [];  % Optimization problem parameters - empty (no parameters used here)

OPT_variables = x;  %single decision variable
nlp_prob = struct('f', obj, 'x', OPT_variables, 'g', g, 'p', P);
```

SX data type is used to represent matrices whose elements consist of **symbolic expressions**

```
>> x
x =
w
```

```
>> obj
obj =
((sq(w)-(6*w))+13)
```

```
>> nlp_prob
  struct with fields:
    f: [1×1 casadi.SX]
    x: [1×1 casadi.SX]
    g: []
    p: []
```

CasADi

```
opts = struct;
opts.ipopt.max_iter = 100;
opts.ipopt.print_level = 0; %0,3
opts.print_time = 0; %0,1
opts.ipopt.acceptable_tol =1e-8;
% optimality convergence tolerance
opts.ipopt.acceptable_obj_change_tol = 1e-6;

solver = nlpsol('solver', 'ipopt', nlp_prob,opts);

args = struct;
args.lbx = -inf;  % unconstrained optimization
args.ubx = inf;   % unconstrained optimization
args.lbg = -inf;  % unconstrained optimization
args.ubg = inf;   % unconstrained optimization

args.p   =  [];  % There are no parameters in this optimization problem
args.x0  = -0.5; % initialization of the optimization variable

sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
    'lbg', args.lbg, 'ubg', args.ubg,'p',args.p);
x_sol = full(sol.x)            % Get the solution
min_value = full(sol.f)   % Get the value function
```

```
>>
x_sol =
     3
min_value =
     4
```

Ipopt (Interior Point Optimizer)* is an open source software package for **large-scale nonlinear optimization**. It can be used to solve general nonlinear programming problems (NLPs)

* Check **IPOPT manual** for more details about the options you can set.

$$\begin{array}{llcl} \underset{x}{\text{minimize:}} & & f(x,p) & \\ \text{subject to:} & x_{\text{lb}} \leq & x & \leq x_{\text{ub}} \\ & g_{\text{lb}} \leq & g(x,p) & \leq g_{\text{ub}} \end{array}$$

Remarks:
- **Single optimization variable**
- **Unconstrained optimization**
- **Local minimum = Global minimum**

CasADi

Use CasADi to solve the following problems

$$\begin{aligned} & \min_{w} e^{0.2w} \sin(w) \\ s.t. \quad & w \geq 0 \\ & x \leq 4\pi \end{aligned}$$

- **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

- **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

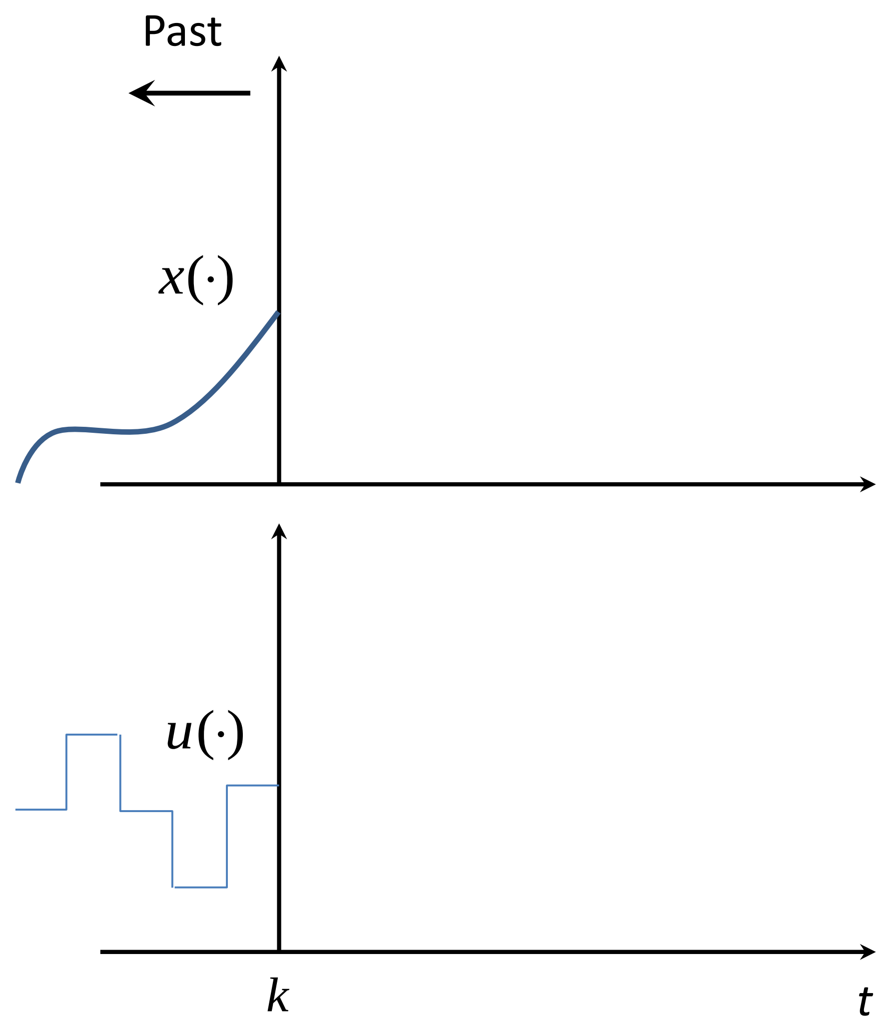
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \dots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• **Model Predictive Control (MPC)** (aka Receding/Moving Horizon Control)

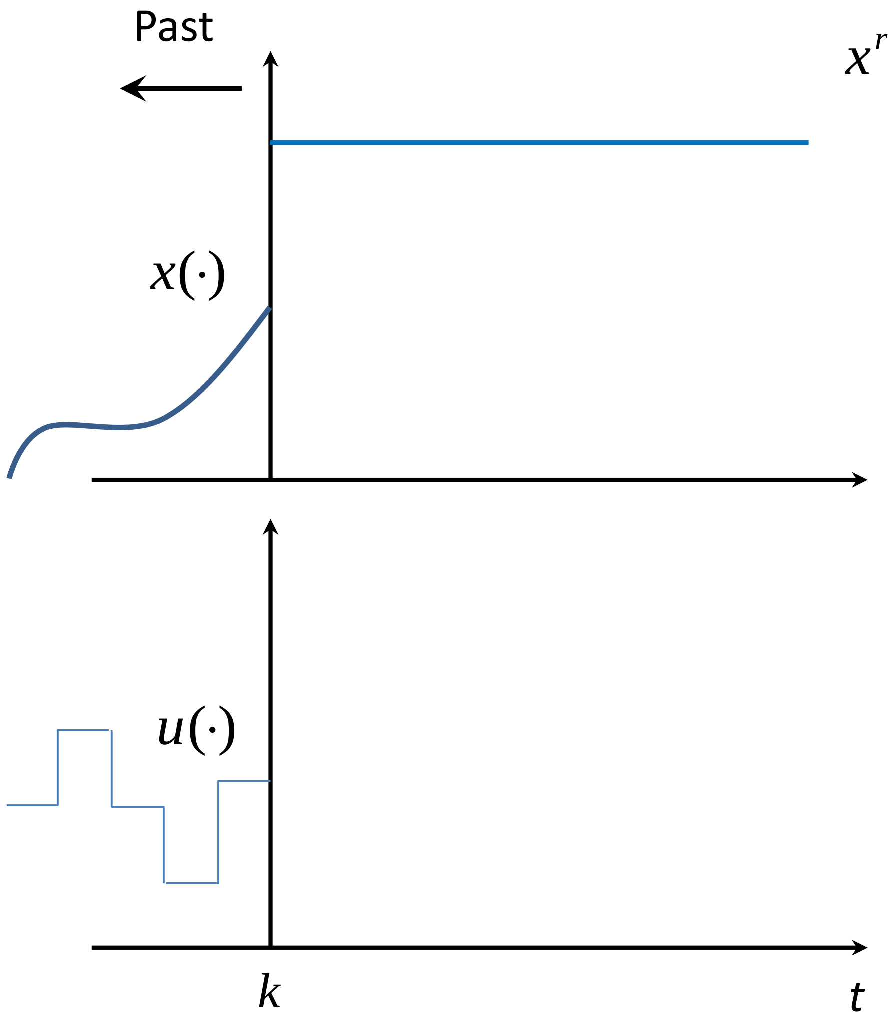
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

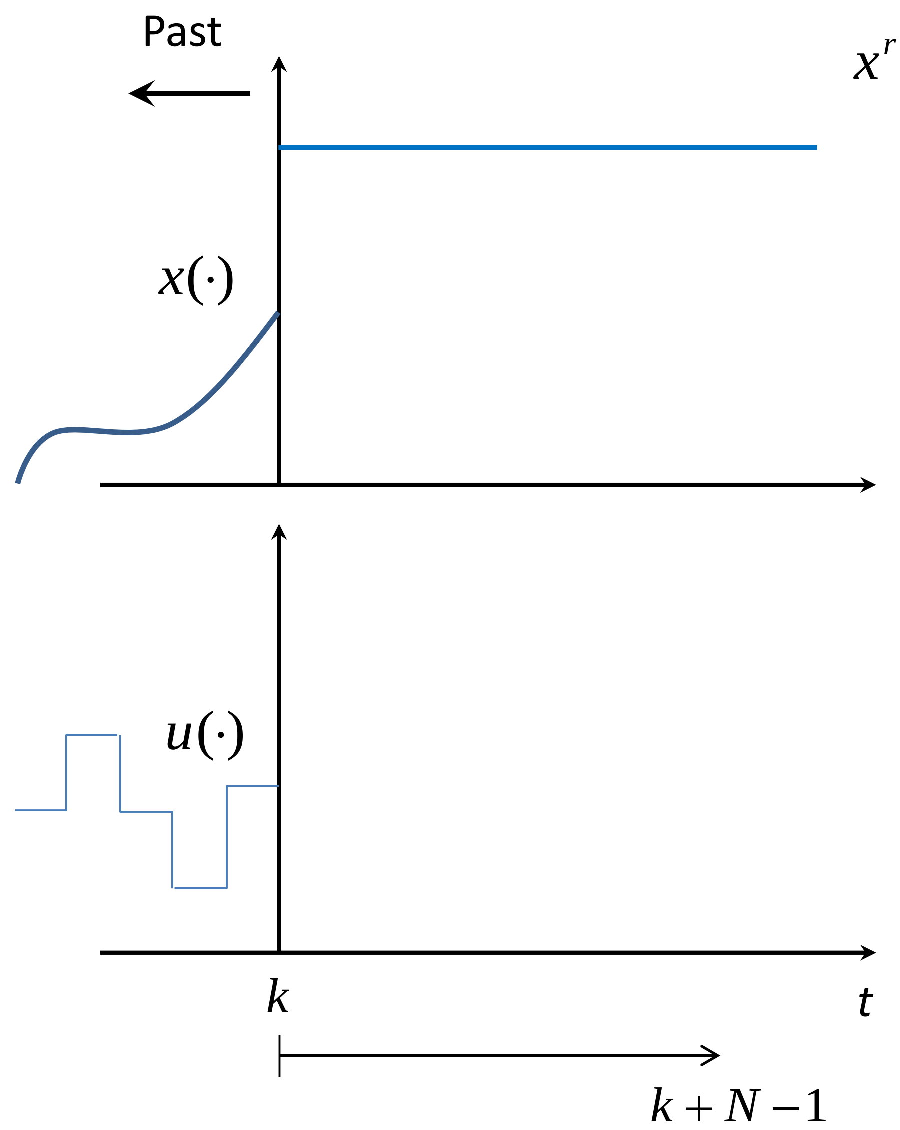
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\,,\, u^*(k+1)\,,\, \ldots\, u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• **Model Predictive Control (MPC)** (aka Receding/Moving Horizon Control)

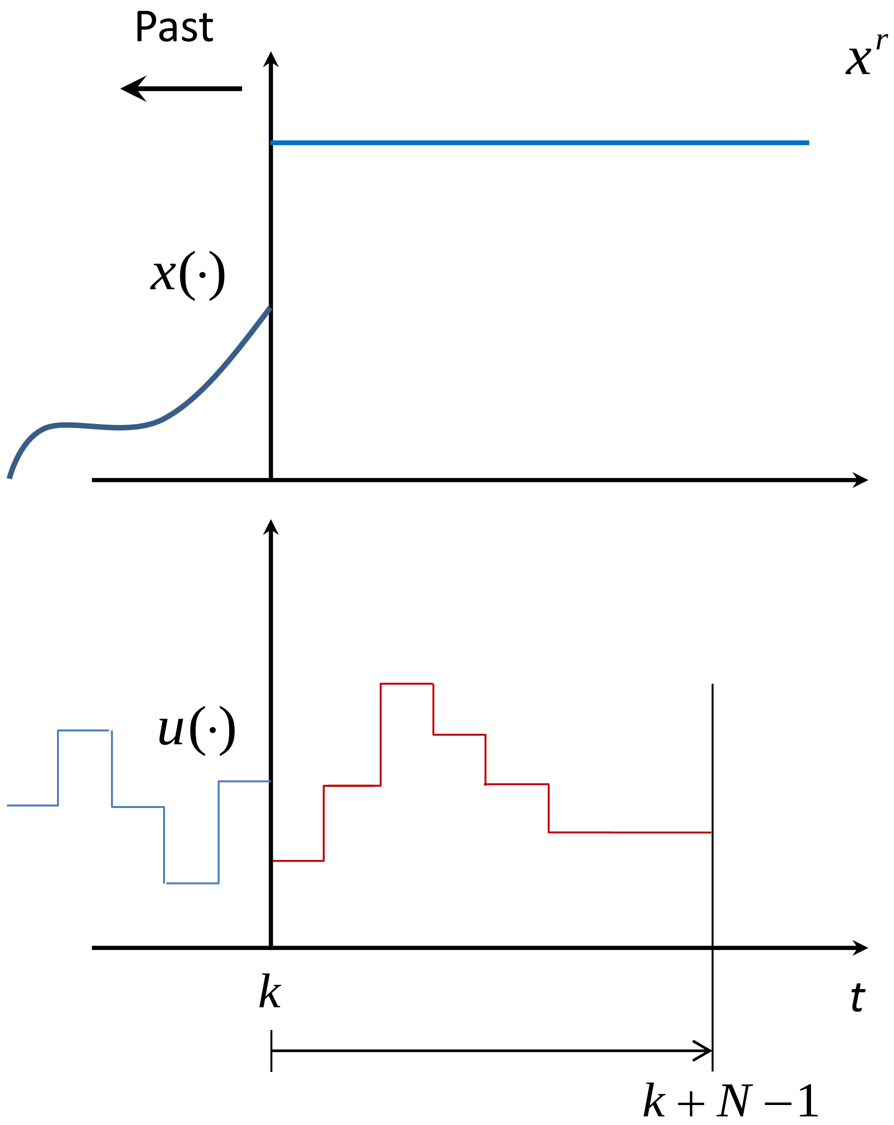
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

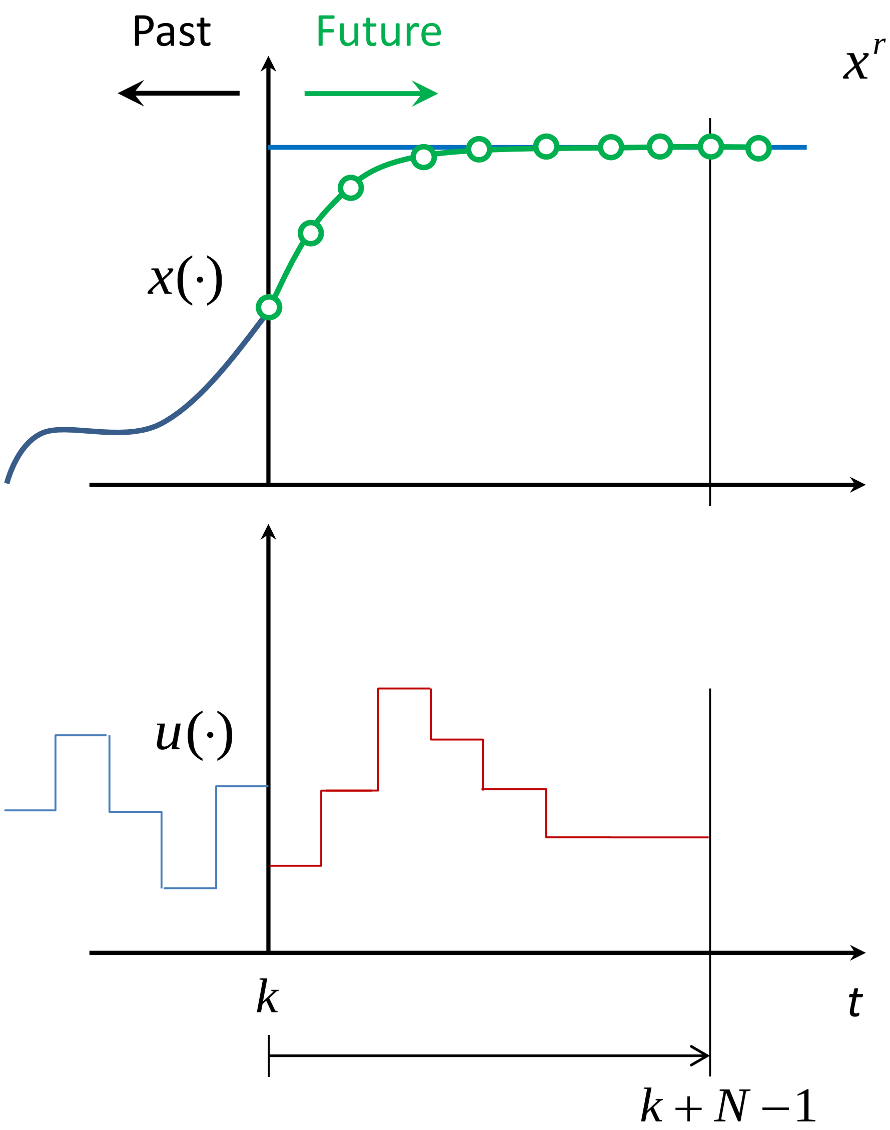
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

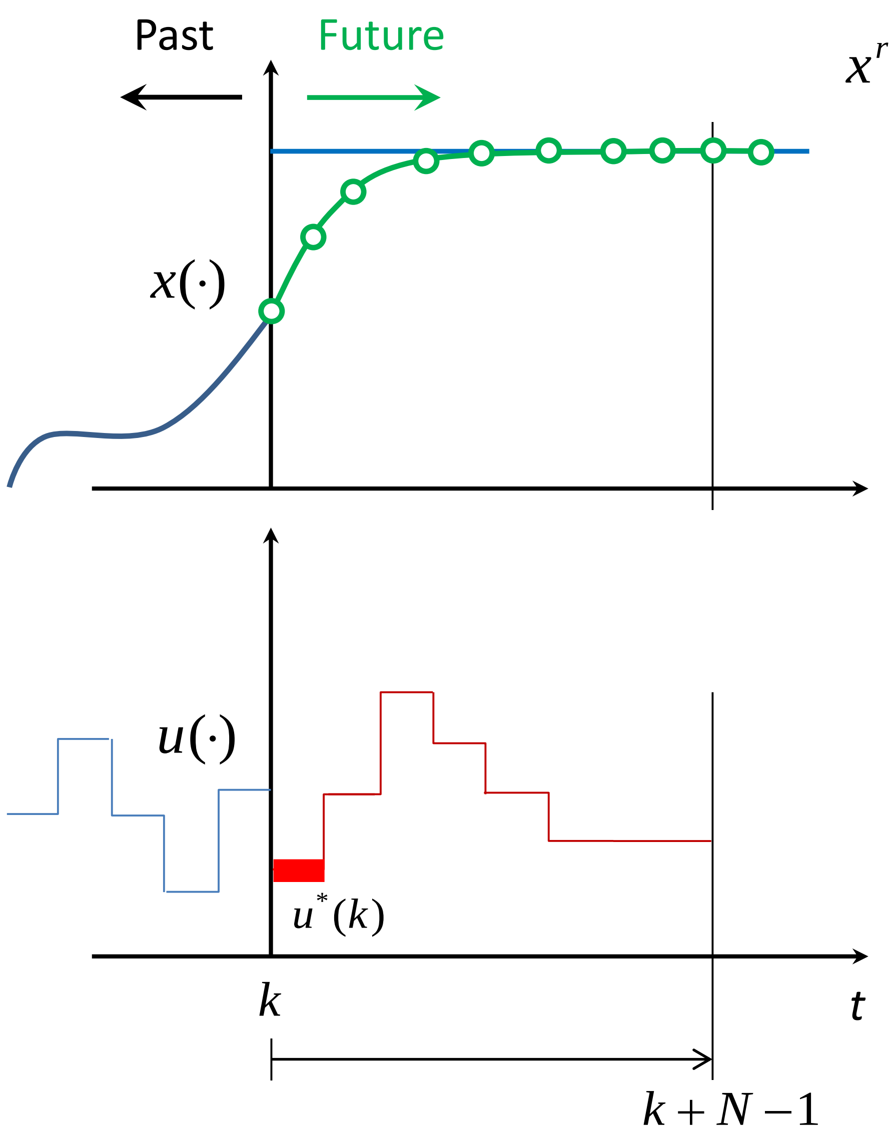
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\,,\, u^*(k+1)\,,\, \ldots\, u^*(k+N\text{-}1))$

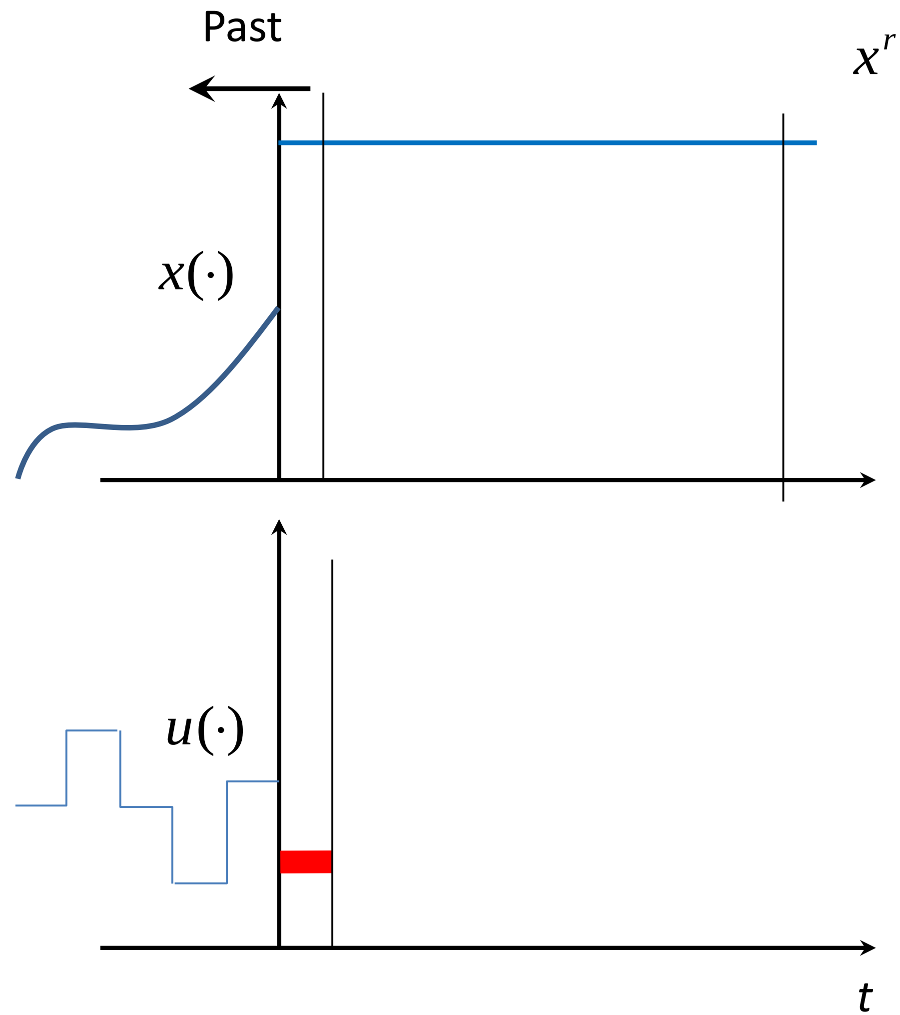
- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.



• **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

Single input single output simple example

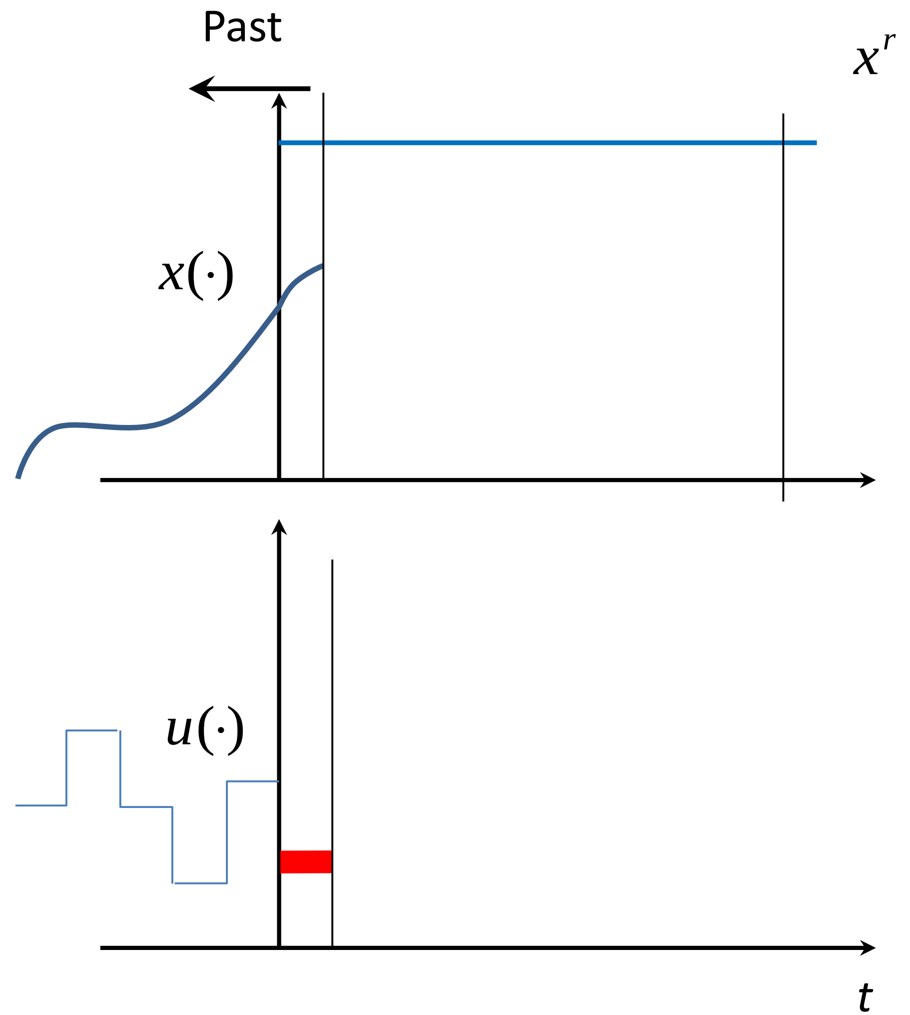
$$x(k+1) = f(x(k), u(k))$$



- **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$



• **Model Predictive Control (MPC) (**aka Receding/Moving Horizon Control**)**

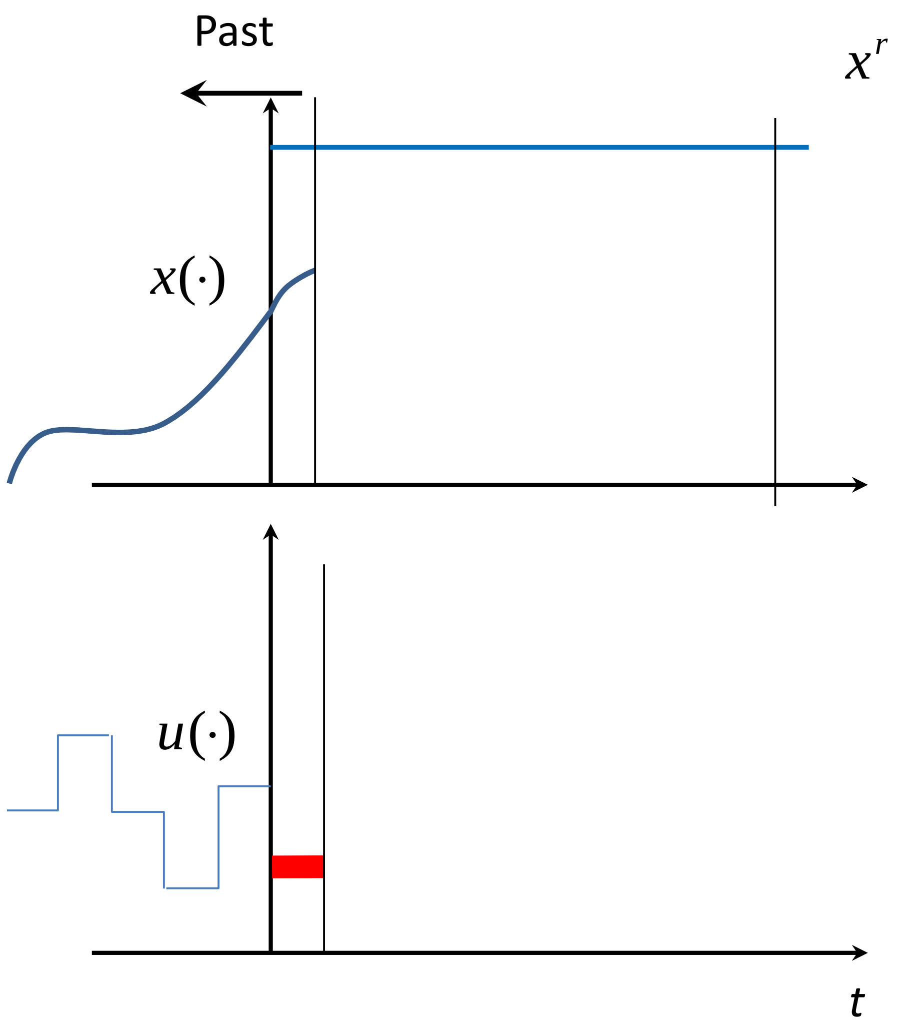
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.
- Repeat the same steps at the next decision instant



• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

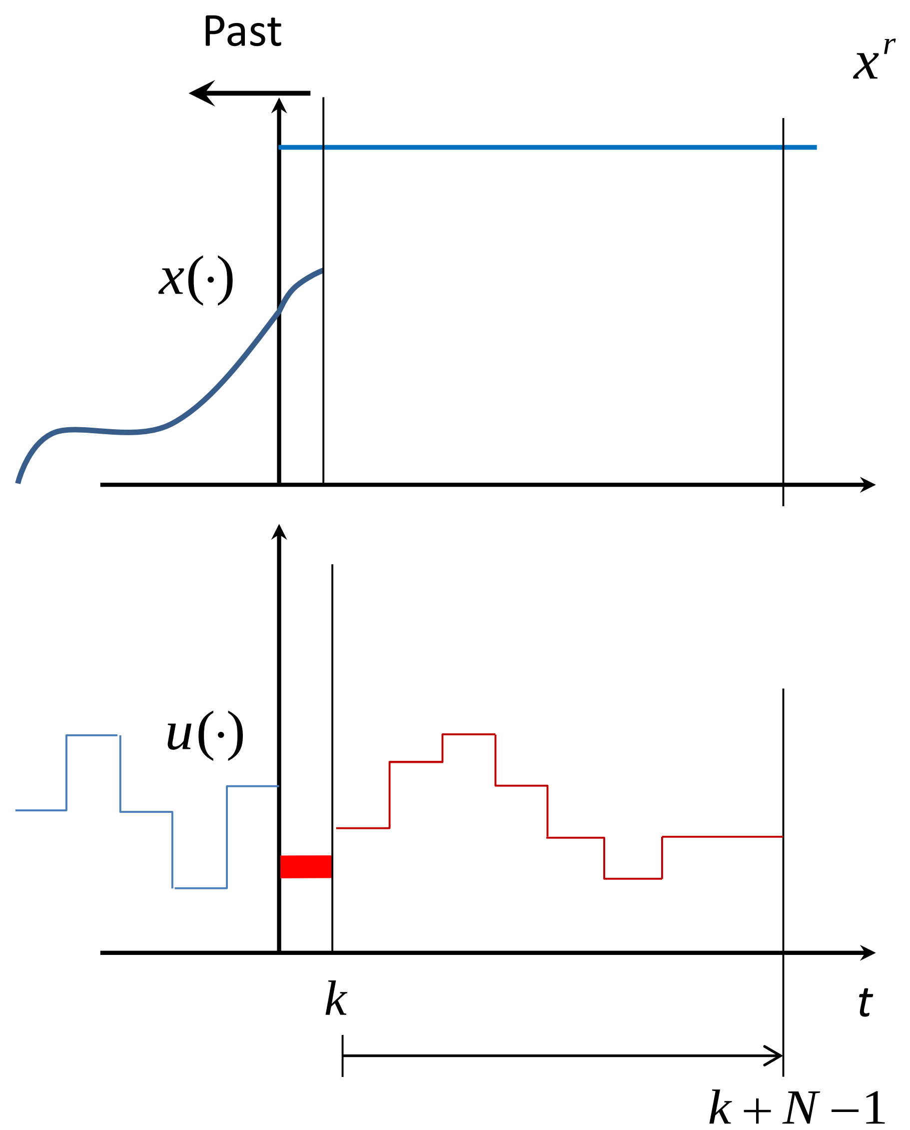
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.
- Repeat the same steps at the next decision instant



• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

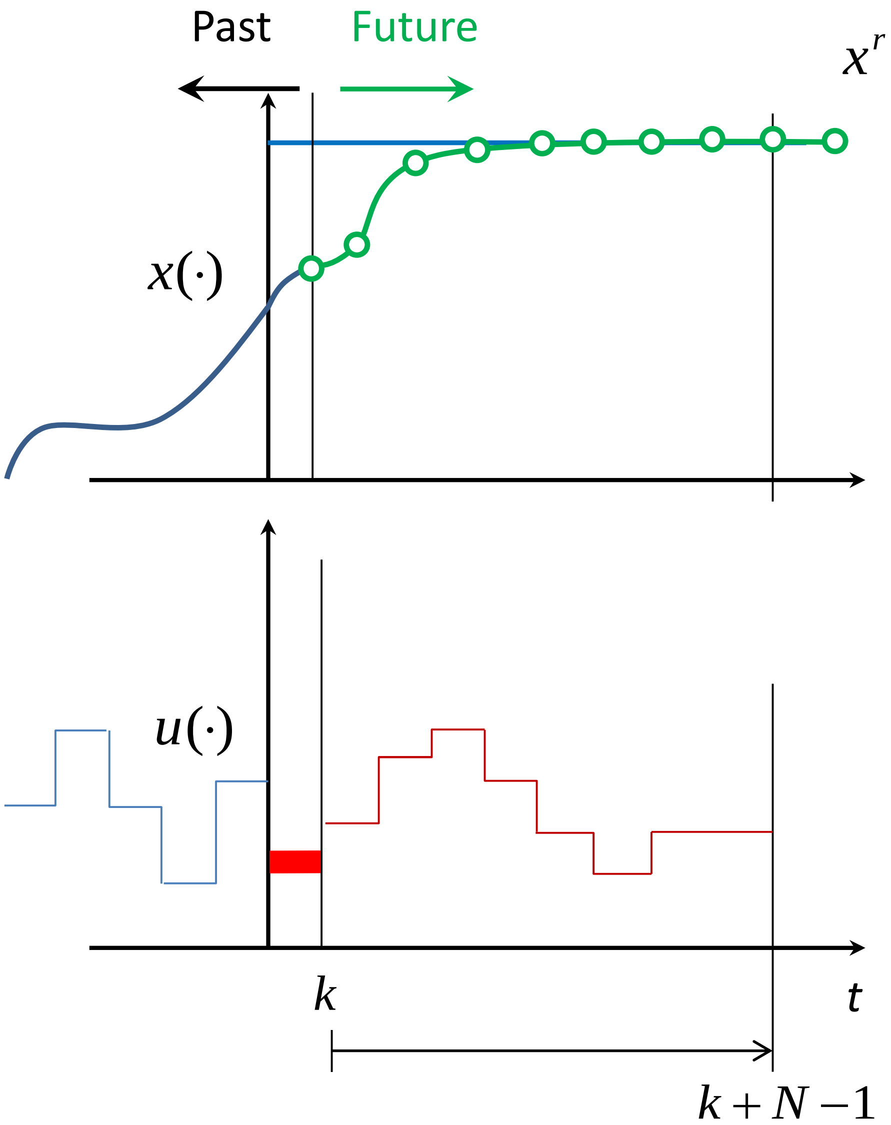
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \dots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.
- Repeat the same steps at the next decision instant



• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

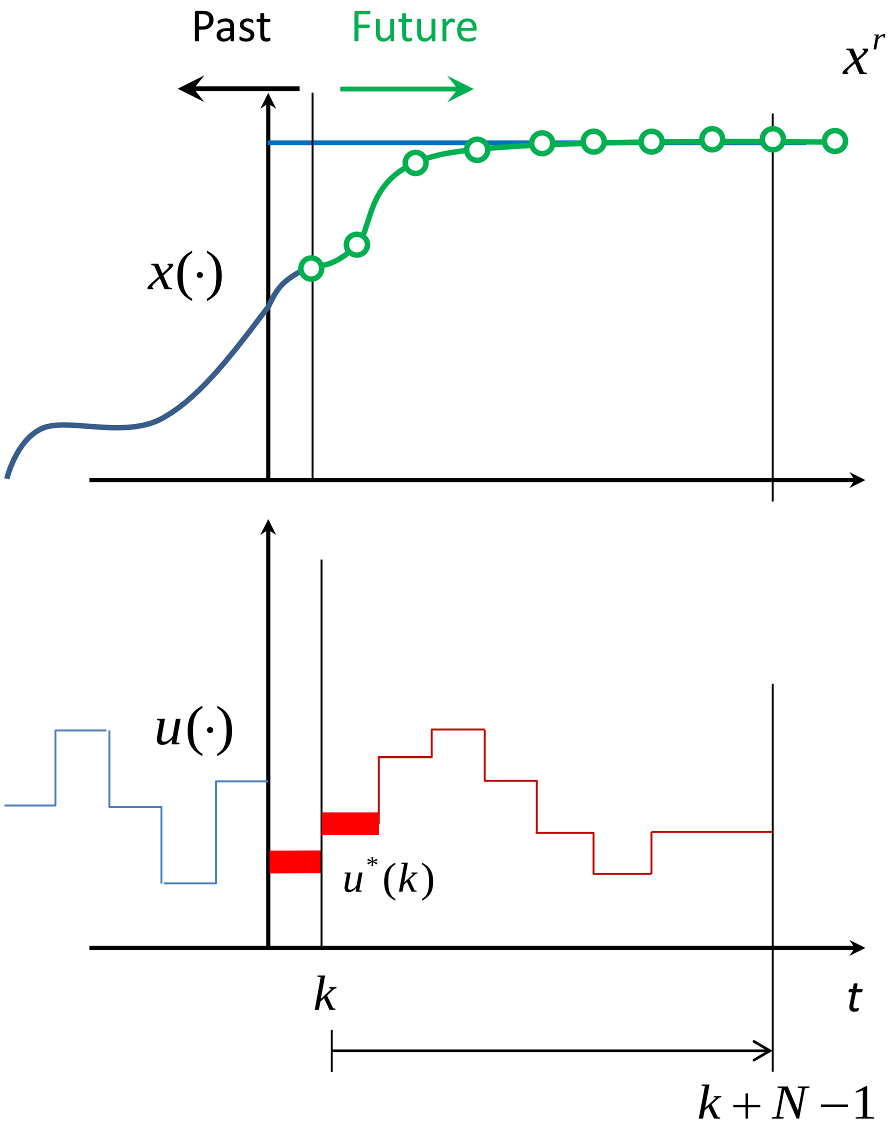
Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.
- Repeat the same steps at the next decision instant



• Model Predictive Control (MPC) (aka Receding/Moving Horizon Control)

Single input single output simple example

$$x(k+1) = f(x(k), u(k))$$

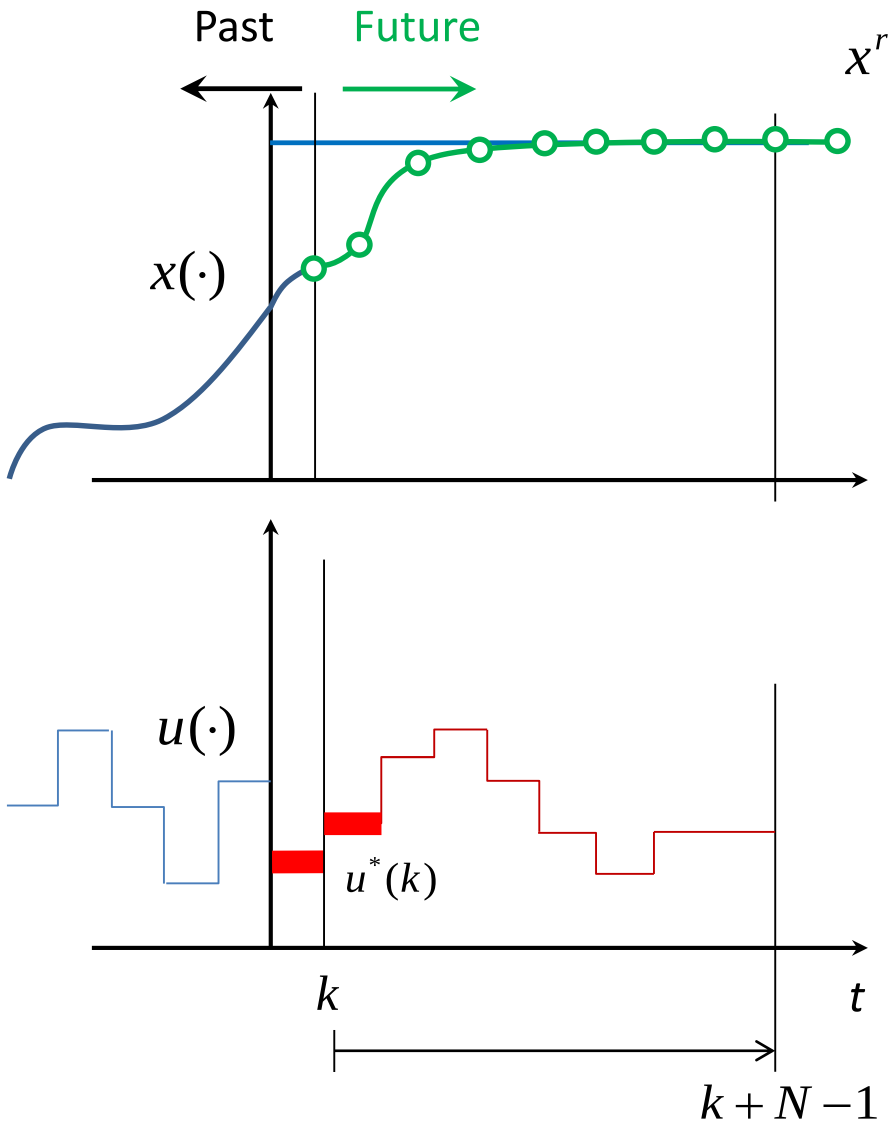
- At **decision instant** k, measure the state $x(k)$
- Based on $x(k)$, compute the (**optimal**) **sequence of controls** over a **prediction horizon** N:

$u^*(x(k)) := (u^*(k)\ ,\ u^*(k+1)\ ,\ \ldots\ u^*(k+N\text{-}1))$

- **Apply** the control $u^*(k)$ on the sampling period $[k, k+1]$.
- Repeat the same steps at the next decision instant

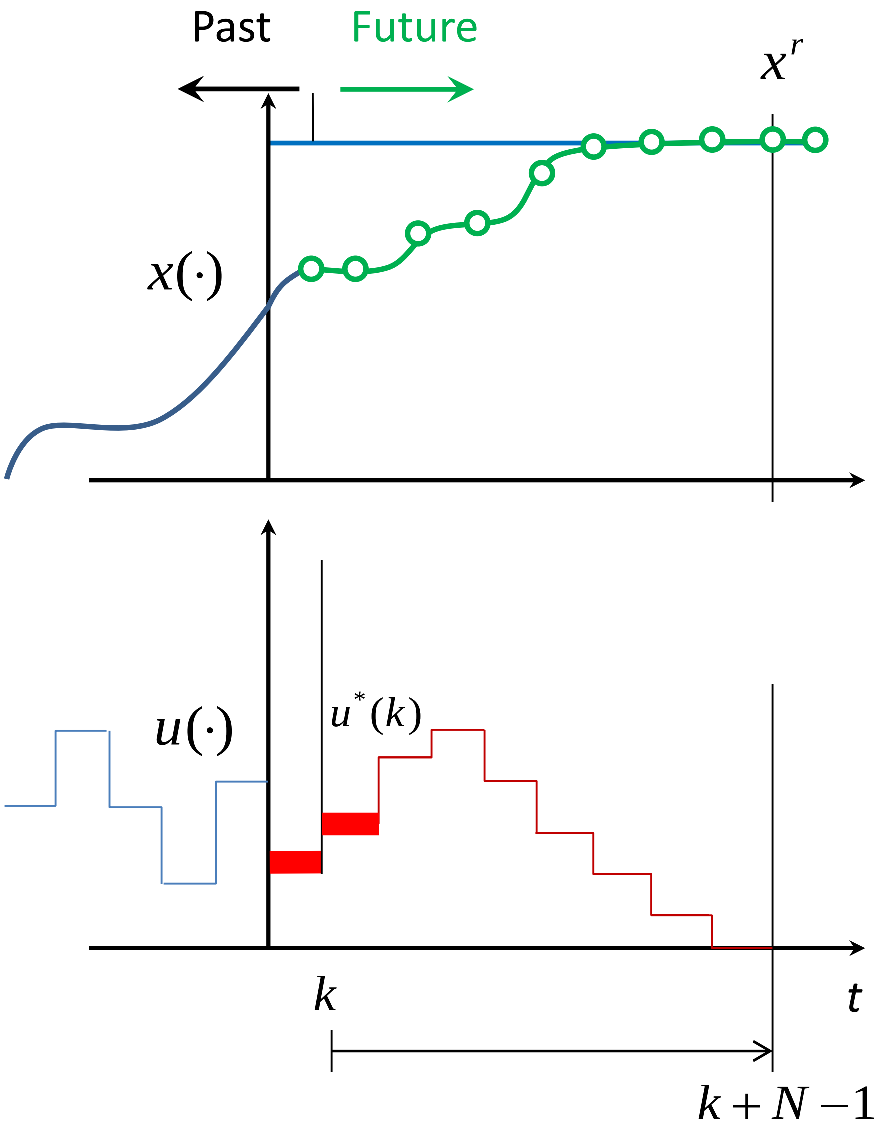
MPC Strategy Summary[1]:

1. Prediction
2. Online optimization
3. Receding horizon implementation



[1]Mark Cannon (2016)

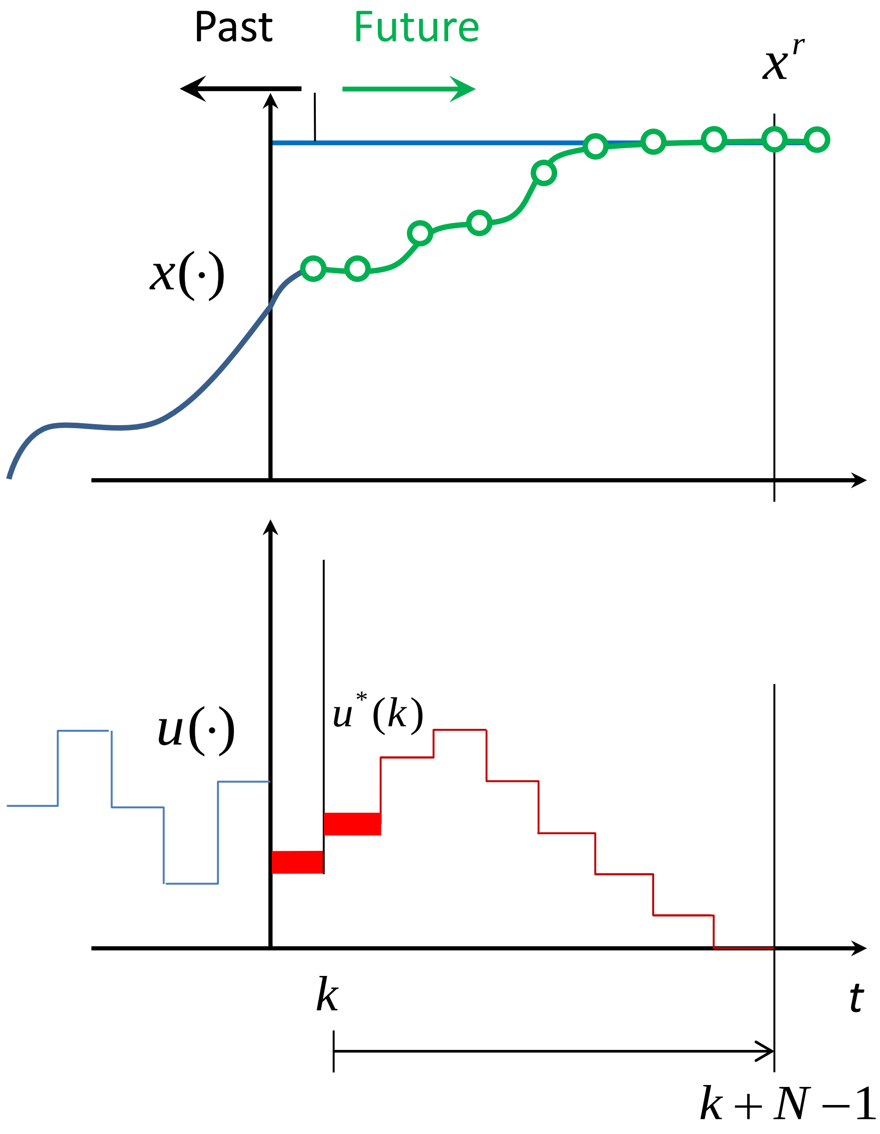
• **MPC Mathematical Formulation**



- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

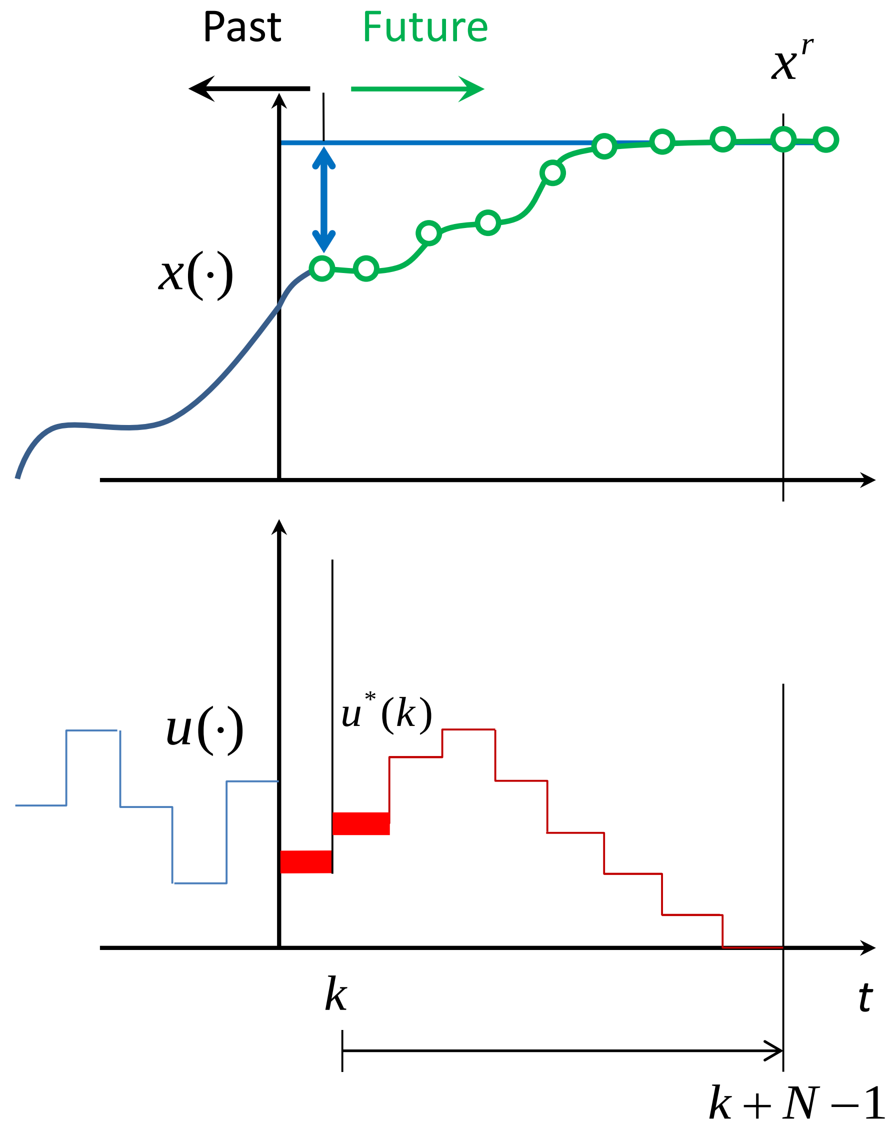
$$\ell(\mathbf{x},\mathbf{u}) = \left\|\mathbf{x}_{\mathbf{u}} - \mathbf{x}^r\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u} - \mathbf{u}^r\right\|_{\mathbf{R}}^2$$



- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

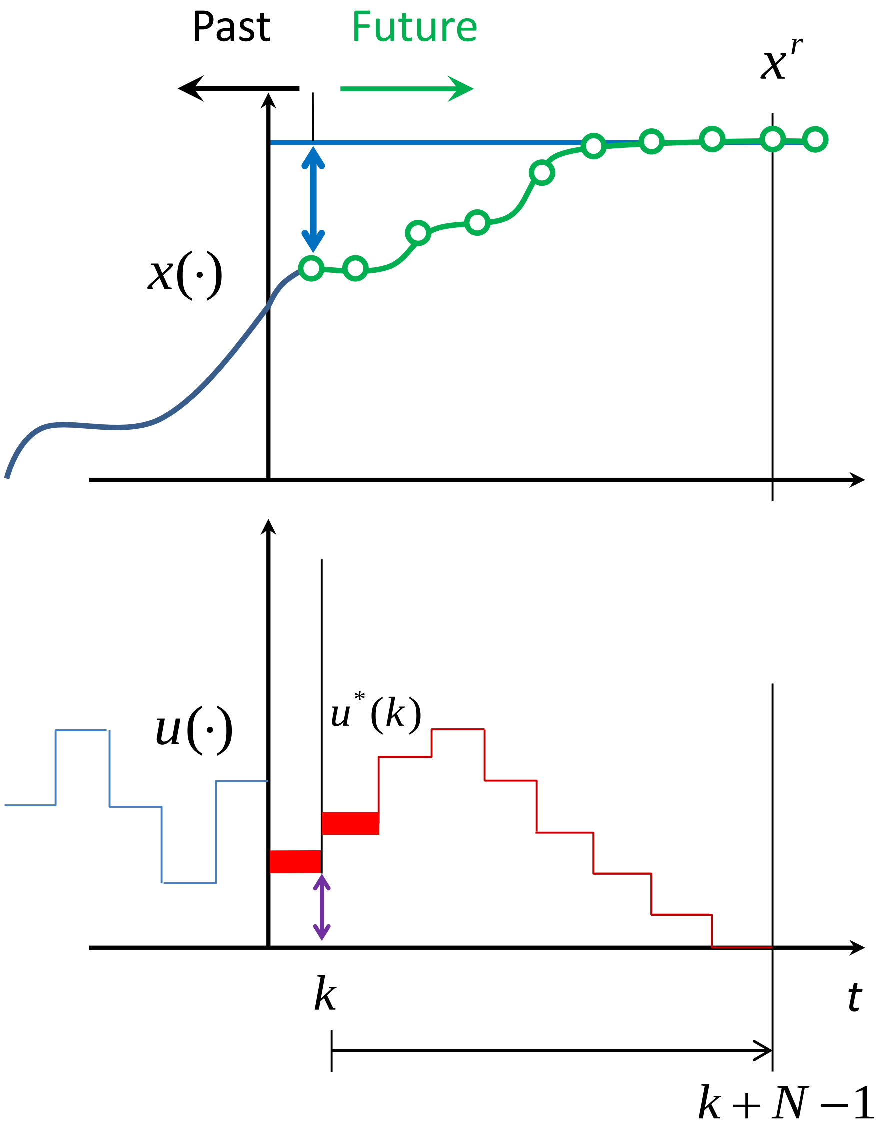
$$\ell(\mathbf{x},\mathbf{u}) = \left\| \mathbf{x}_{\mathbf{u}} - \mathbf{x}^{r} \right\|_{\mathbf{Q}}^{2} + \left\| \mathbf{u} - \mathbf{u}^{r} \right\|_{\mathbf{R}}^{2}$$



- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

$$\ell(\mathbf{x},\mathbf{u}) = \left\|\mathbf{x_u} - \mathbf{x}^r\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u} - \mathbf{u}^r\right\|_{\mathbf{R}}^2$$



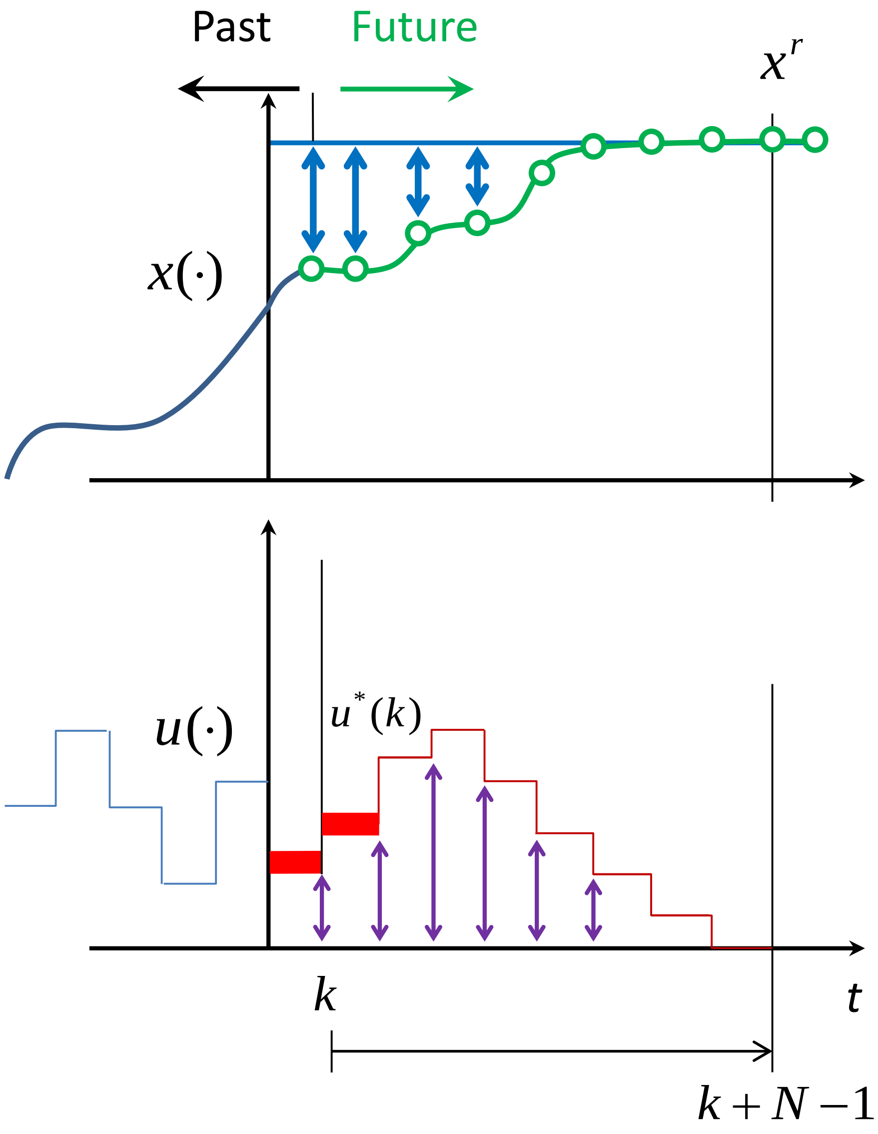
- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

$$\ell(\mathbf{x},\mathbf{u}) = \left\|\mathbf{x}_{\mathbf{u}} - \mathbf{x}^{r}\right\|_{\mathbf{Q}}^{2} + \left\|\mathbf{u} - \mathbf{u}^{r}\right\|_{\mathbf{R}}^{2}$$

Cost Function: Evaluation of the running costs along the whole prediction horizon

$$J_N(\mathbf{x},\mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k))$$



- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

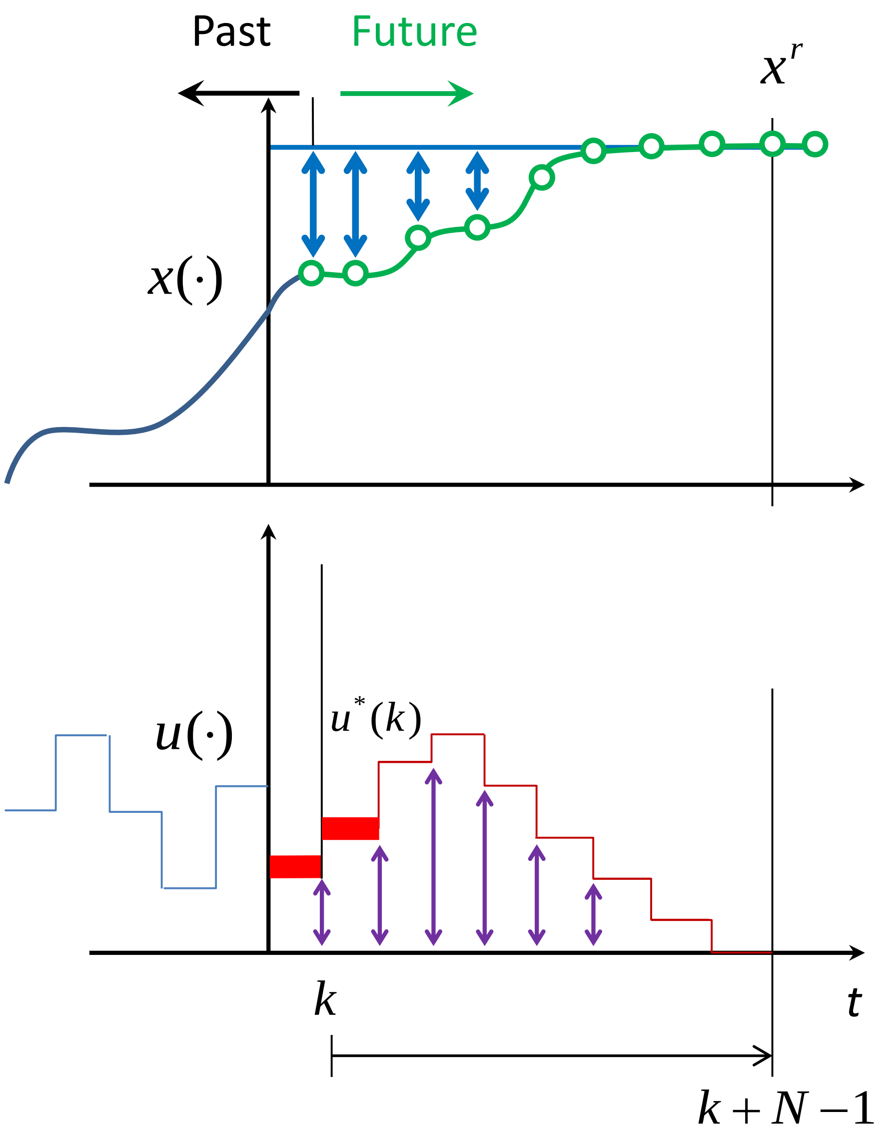
$$\ell(\mathbf{x},\mathbf{u}) = \left\|\mathbf{x}_{\mathbf{u}} - \mathbf{x}^r\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u} - \mathbf{u}^r\right\|_{\mathbf{R}}^2$$

Cost Function: Evaluation of the running costs along the whole prediction horizon

$$J_N(\mathbf{x},\mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k))$$

Optimal Control Problem (OCP): to find a minimizing control sequence

$$\begin{aligned}
&\underset{\mathbf{u}}{\text{minimize}}\, J_N(\mathbf{x}_0,\mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k))\\
&\text{subject to}: \mathbf{x}_{\mathbf{u}}(k+1) = \mathbf{f}(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k)),\\
&\qquad\qquad \mathbf{x}_{\mathbf{u}}(0) = \mathbf{x}_0,\\
&\qquad\qquad \mathbf{u}(k) \in U, \ \forall k \in [0, N-1]\\
&\qquad\qquad \mathbf{x}_{\mathbf{u}}(k) \in X, \ \forall k \in [0, N]
\end{aligned}$$



- **MPC Mathematical Formulation**

Running (stage) Costs: characterizes the control objective

$$\ell(\mathbf{x},\mathbf{u}) = \left\| \mathbf{x}_{\mathbf{u}} - \mathbf{x}^r \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u} - \mathbf{u}^r \right\|_{\mathbf{R}}^2$$

Cost Function: Evaluation of the running costs along the whole prediction horizon

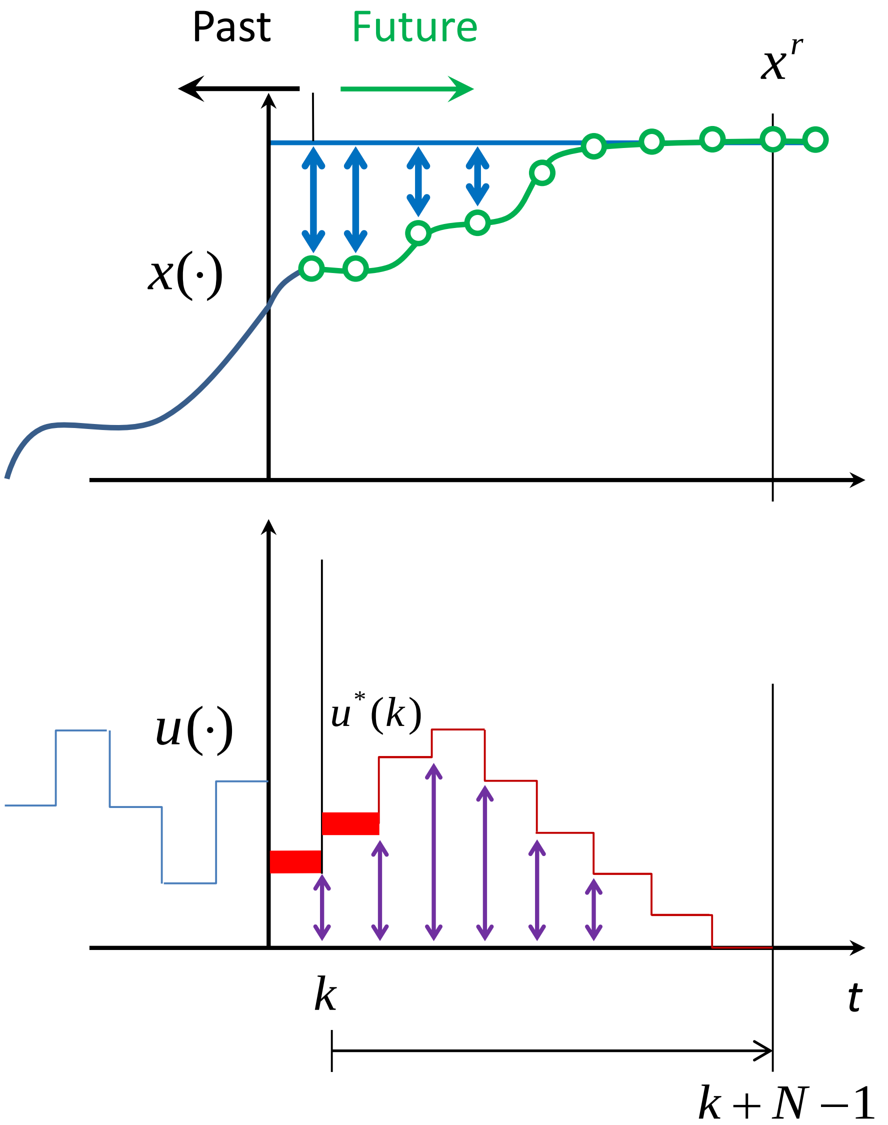
$$J_N(\mathbf{x},\mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k))$$

Optimal Control Problem (OCP): to find a minimizing control sequence

$$\begin{aligned}
\underset{\mathbf{u}}{\text{minimize}}\, J_N(\mathbf{x}_0,\mathbf{u}) &= \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k)) \\
\text{subject to}: \mathbf{x}_{\mathbf{u}}(k+1) &= \mathbf{f}(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k)), \\
\mathbf{x}_{\mathbf{u}}(0) &= \mathbf{x}_0, \\
\mathbf{u}(k) &\in U, \quad \forall k \in [0, N-1] \\
\mathbf{x}_{\mathbf{u}}(k) &\in X, \quad \forall k \in [0, N]
\end{aligned}$$

Value Function: minimum of the cost function

$$V_N(\mathbf{x}) = \min_{\mathbf{u}} J_N(\mathbf{x}_0,\mathbf{u})$$



- **Considered System and Control Problem** (Differential drive robots)

From Input/output point of view, robot as a system can be viewed as

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}v\cos\theta\\ v\sin\theta\\ \omega\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}$$

Linear velocity input

v

ω

Angular velocity input

$x, \dot{x}$

$y, \dot{y}$

$\theta, \dot{\theta}$

Control Signals

System States

- **Considered System and Control Problem** (Differential drive robots)

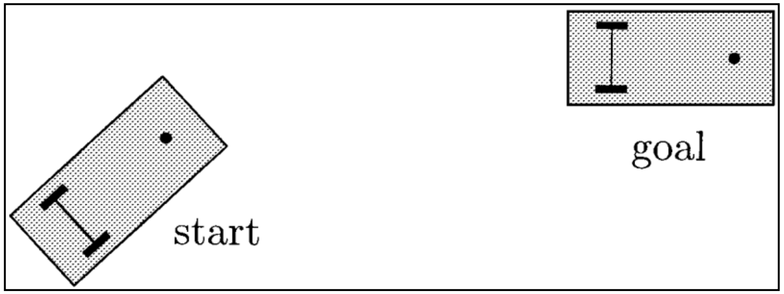
  - **Control objectives**

point stabilization

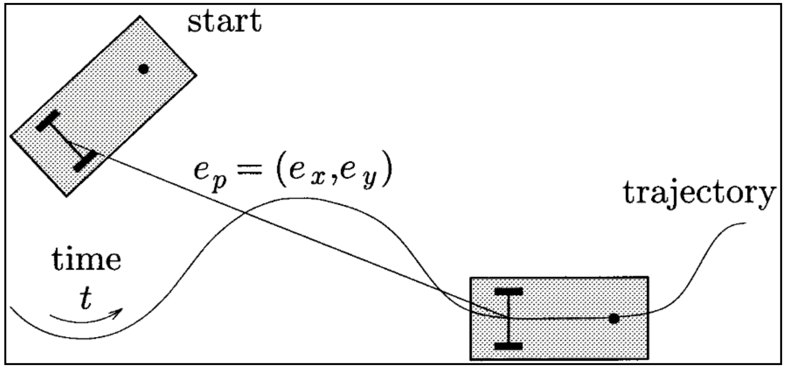
$$\begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta_r(t) \end{bmatrix} = \left\{ \begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix}, \forall t \right\}$$

- reference values of the state vector are constant over the control period

Point stabilization



Trajectory tracking



trajectory tracking

$$\begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta_r(t) \end{bmatrix} = \begin{bmatrix} x_d(t) \\ y_d(t) \\ \theta_d(t) \end{bmatrix}$$

- time varying reference values of the state vector

- **Model Predictive Control for** (Differential drive robots – point stabilization)

system model

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}(t), \mathbf{u}(t)) \qquad\qquad \mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \xrightarrow[\text{Sampling Time } (\Delta T)]{\text{Euler Discretization}} \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \Delta T \begin{bmatrix} v(k)\cos\theta(k) \\ v(k)\sin\theta(k) \\ \omega(k) \end{bmatrix}$$

MPC controller

Running (stage) Costs: $\ell(\mathbf{x}, \mathbf{u}) = \left\| \mathbf{x}_{\mathbf{u}} - \mathbf{x}^{ref} \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u} - \mathbf{u}^{ref} \right\|_{\mathbf{R}}^2$

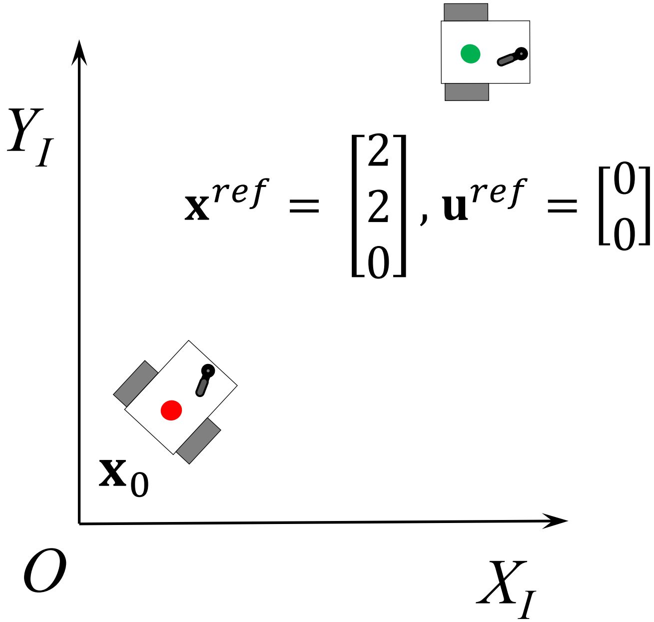
Optimal Control Problem (OCP):

$$\underset{\mathbf{u}\,\text{admissible}}{\text{minimize}}\, J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_{\mathbf{u}}(k), \mathbf{u}(k))$$

$$\begin{aligned} \text{subject to: } & \mathbf{x}_{\mathbf{u}}(k+1) = \mathbf{f}(\mathbf{x}_{\mathbf{u}}(k), \mathbf{u}(k)), \\ & \mathbf{x}_{\mathbf{u}}(0) = \mathbf{x}_0, \\ & \mathbf{u}(k) \in U, \ \ \forall k \in [0, N-1] \\ & \mathbf{x}_{\mathbf{u}}(k) \in X, \ \ \forall k \in [0, N] \end{aligned}$$

- **Point Stabilization Recap**

$$\ell(\mathbf{x},\mathbf{u}) = \|\mathbf{x_u} - \mathbf{x}^{ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u} - \mathbf{u}^{ref}\|_{\mathbf{R}}^2$$



$$\mathbf{x}^{ref} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}, \mathbf{u}^{ref} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(OCP):

$$\underset{\mathbf{u}\,\text{admissible}}{\text{minimize}}\, J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{N-1} \ell(\mathbf{x_u}(k), \mathbf{u}(k))$$

$$\text{subject to}: \mathbf{x_u}(k+1) = \mathbf{f}(\mathbf{x_u}(k), \mathbf{u}(k)),$$

$$\mathbf{x_u}(0) = \mathbf{x}_0,$$

$$\mathbf{u}(k) \in U, \quad \forall k \in [0, N-1]$$

$$\mathbf{x_u}(k) \in X, \quad \forall k \in [0, N]$$

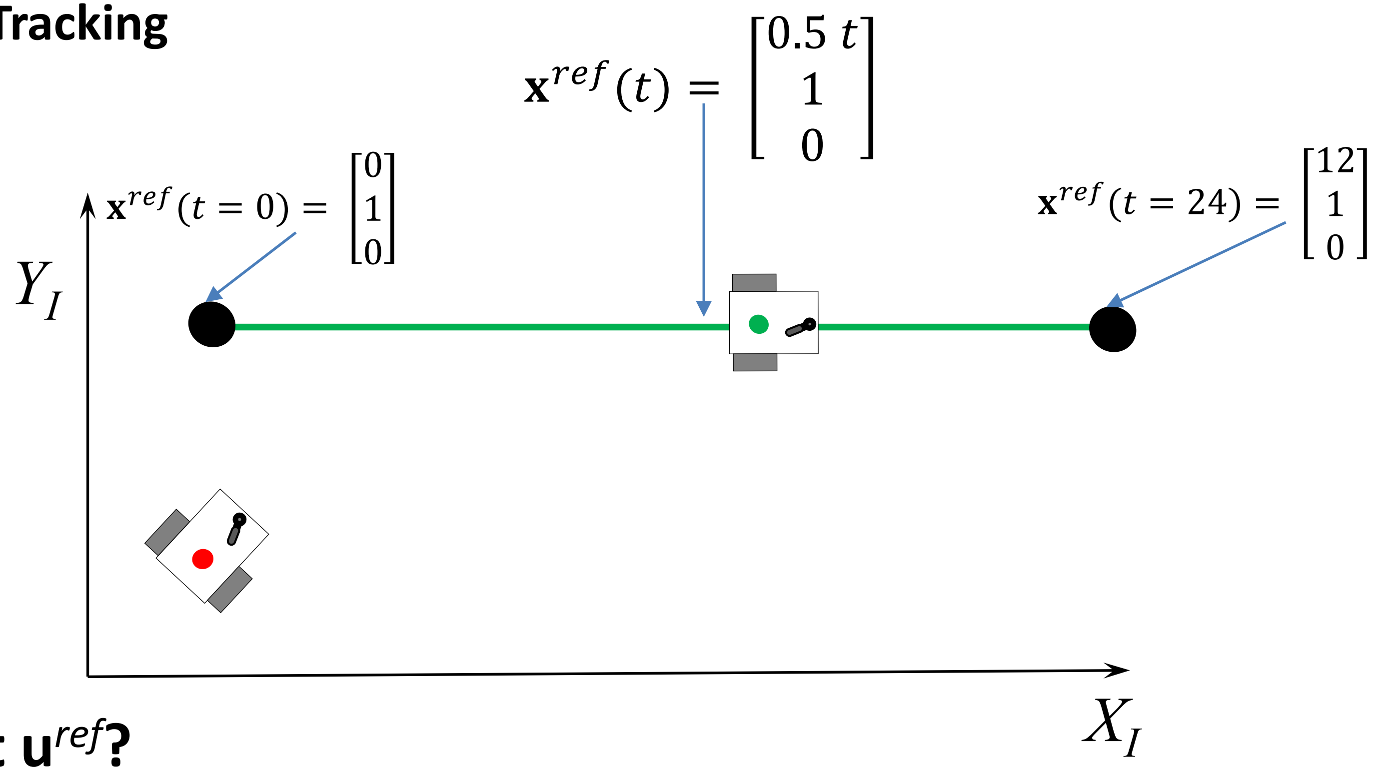
Let's expand J_N for N = 3

$$J_N(\mathbf{x}_0, \mathbf{u}) = \sum_{k=0}^{2} \ell(\mathbf{x_u}(k), \mathbf{u}(k)) = \left\|\mathbf{x_u}(0) - \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(0) - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$

$$+ \left\|\mathbf{x_u}(1) - \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(1) - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$

$$+ \left\|\mathbf{x_u}(2) - \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(2) - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$

- **Trajectory Tracking**



How to get $\mathbf{u}^{ref}$?

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$$

Square and add the first two equations, we get

$$\dot{x}^2 + \dot{y}^2 = v^2 \rightarrow v = 0.5\ [m/s]$$

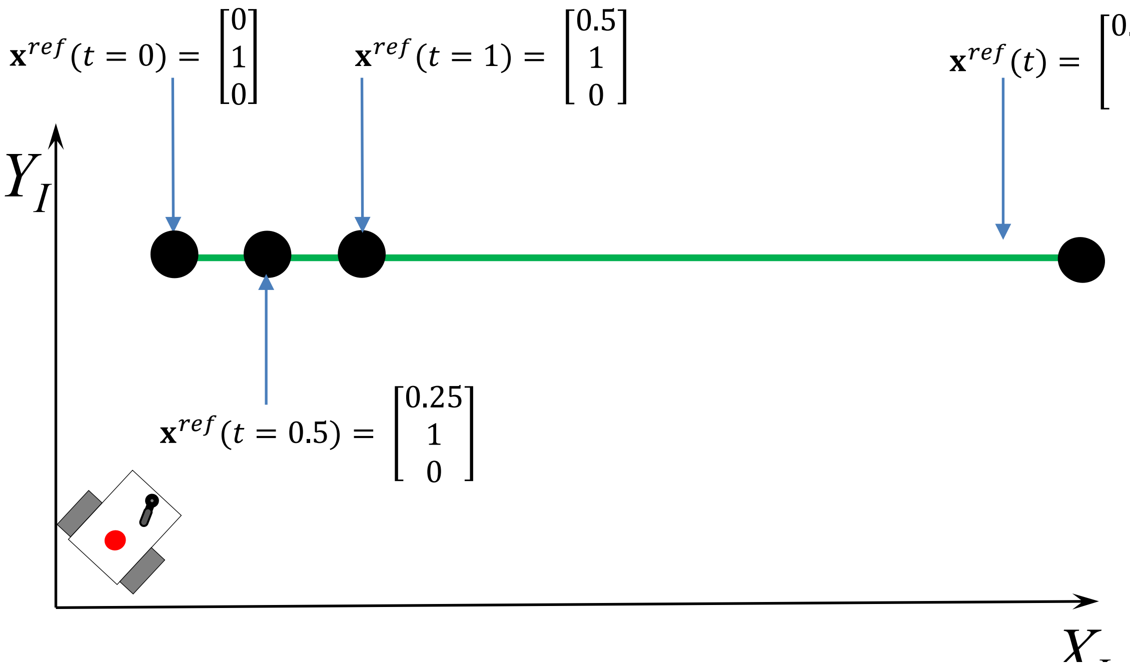
From the third equation we have

$$\omega = 0\ [rad/s]$$

$$\mathbf{u}^{ref} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

Feedforward Control Actions

- **Trajectory Tracking**



Let's expand J_N for N = 3

$$J_N(\mathbf{x}_0,\mathbf{u}) = \sum_{k=0}^{2} \ell\big(\mathbf{x}_{\mathbf{u}}(k),\mathbf{u}(k)\big) = \left\|\mathbf{x}_{\mathbf{u}}(0) - \begin{bmatrix} 0.00 \\ 1 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(0) - \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$
$$+ \left\|\mathbf{x}_{\mathbf{u}}(1) - \begin{bmatrix} 0.25 \\ 1 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(1) - \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$
$$+ \left\|\mathbf{x}_{\mathbf{u}}(2) - \begin{bmatrix} 0.50 \\ 1 \\ 0 \end{bmatrix}\right\|_{\mathbf{Q}}^2 + \left\|\mathbf{u}(2) - \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}\right\|_{\mathbf{R}}^2$$