Controllo di un robot con ingressi limitati

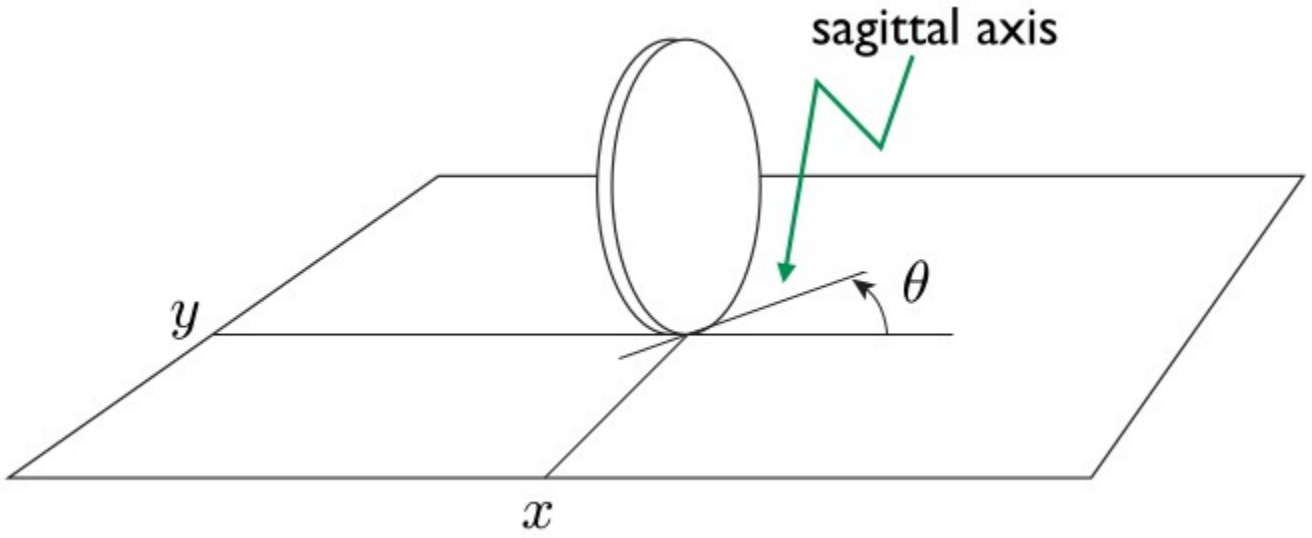
Emanuele Coletta 2001600

SAPIENZA
UNIVERSITÀ DI ROMA

Obiettivi

- Fare trajectory tracking con un robot uniciclo/differential drive
- Nella realtà gli attuatori che vengono utilizzati hanno limiti fisici che non possono essere superati (es. velocità di un motore)
- Generalmente, i controllori che progettiamo non evitano a prescindere che gli attuatori entrino in saturazione
- Sta a noi progettare il controllore e l'hardware adeguatamente
- Allora vogliamo *correggere* il comando per evitare di generare input che manderebbero gli attuatori in saturazione
- La correzione è apportata con un controllore MPC-based

Robot uniciclo

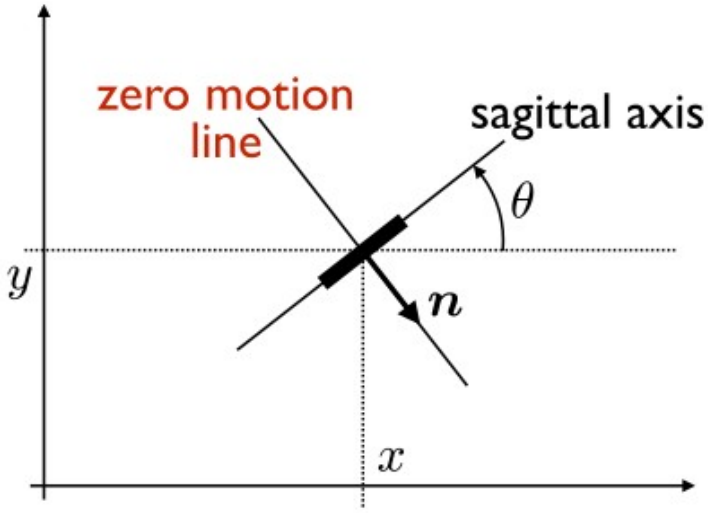


Solo teorico! Però può essere espanso a differential drive

Robot uniciclo (cont.)

- È sottoposto a un vincolo di puro rotolamento

$$\begin{pmatrix} \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = 0$$



- Riscrivendo per far comparire tutte le coordinate generalizzate e trovando il kernel

$$\begin{pmatrix} \sin\theta & -\cos\theta & 0 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \mathrm{A}^T(q)\dot{q} = 0$$

- Si ottiene il modello cinematico dell'uniciclo

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

Trajectory tracking con lin. i/o

Dato un sistema *driftless*

$$\begin{cases} \dot{x} = G(x)u \\ y = h(x)x \end{cases}$$

Derivando l'uscita

$$\dot{y} = \frac{\partial h(x)}{\partial x}\dot{x} = \frac{\partial h(x)}{\partial x}G(x)u = T(x)u$$

Definendo l'input come

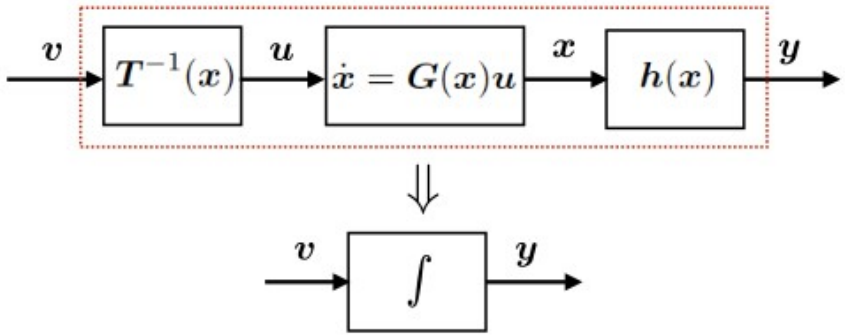
$$u = T^{-1}(x)v$$

Si ha

$$\dot{y} = v$$

Trajectory tracking con lin. i/o (cont.)

Il sistema diventa un integratore



Che può essere stabilizzato con un proporzionale + eventuale feedforward, definendo

$$\dot{y} = v = \dot{y}_d + K(y_d - y)$$

Finchè gli autovalori di K appartengono al semipiano destro l'errore converge asintoticamente a zero

Trajectory tracking con lin. i/o per uniciclo

Per l'uniciclo l'output è la posizione cartesiana

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \implies \begin{pmatrix} \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}$$

$$T(x) = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{pmatrix}$$

mat. disaccoppiamento singolare!

$$u = \begin{pmatrix} v \\ \omega \end{pmatrix}$$

input uniciclo

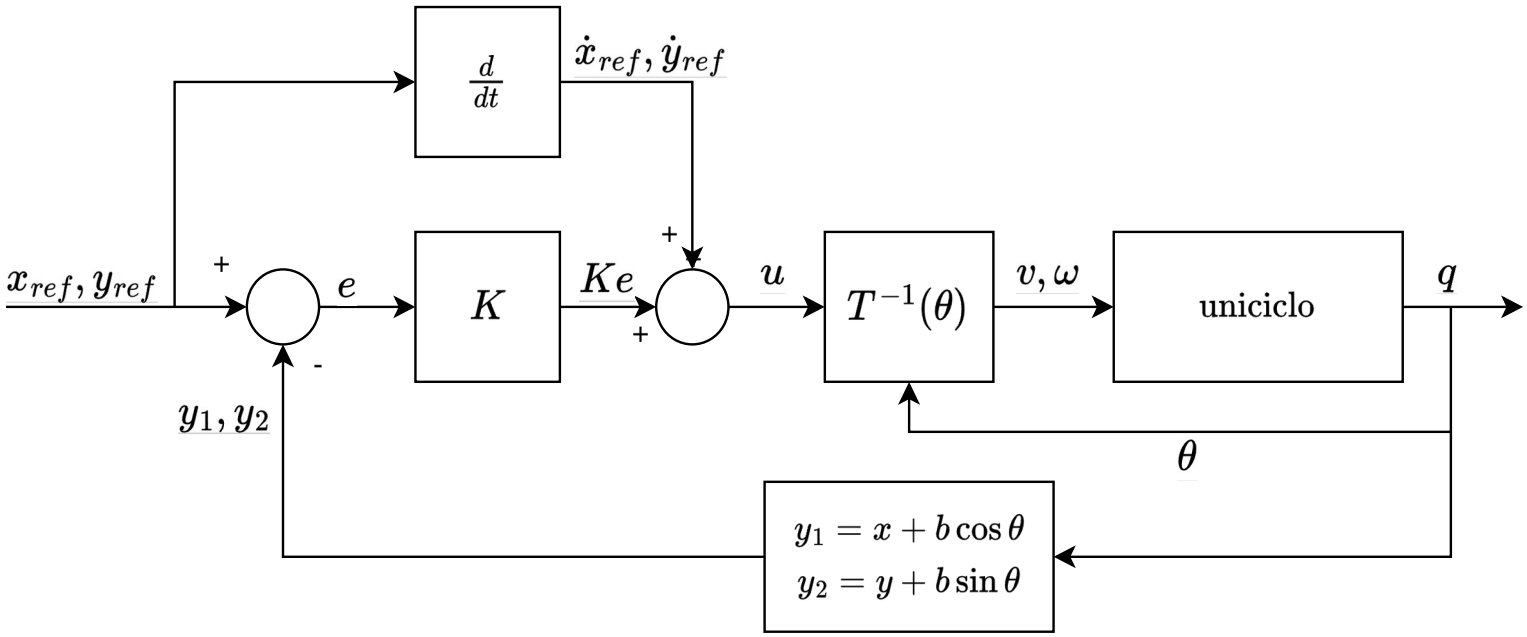
Tracciando però un punto B a distanza b dal centro del robot, la matrice di disaccoppiamento diventa invertibile

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x + b\cos\theta \\ y + b\sin\theta \end{pmatrix} \implies \begin{pmatrix} \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} \dot{x} - b\sin\theta\dot{\theta} \\ \dot{y} + b\cos\theta\dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta - \omega b\sin\theta \\ v\sin\theta + \omega b\cos\theta \end{pmatrix}$$

$$T(x) = \begin{pmatrix} \cos\theta & -b\sin\theta \\ \sin\theta & b\cos\theta \end{pmatrix} \implies T^{-1}(x) = \begin{pmatrix} \cos\theta & -sin\theta \\ -\sin\theta/b & \cos\theta/b \end{pmatrix}$$

Trajectory tracking con lin. i/o per uniciclo

Schema di controllo

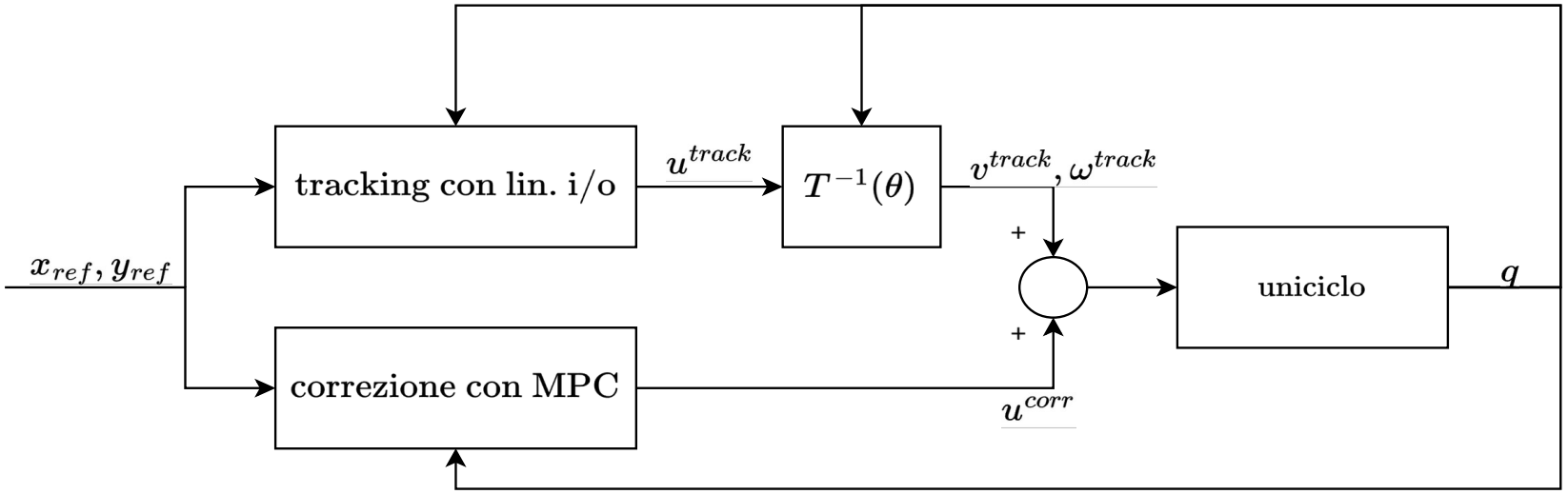


Azione correttiva

- Il controllore ottenuto per linearizzazione i/o può generare comandi che portano gli attuatori in saturazione
- Questo avviene quando si richiedono velocità troppo elevate, curve troppo strette o si hanno errori iniziali elevati
- Si vuole *correggere* l'azione di tracking per evitare che questo avvenga
- L'azione correttiva è realizzata tramite un controllore MPC-based
- L'azione correttiva potrebbe disturbare il tracking! Questo effetto indesiderato va evitato quanto più possibile

Azione correttiva MPC-based

Schema di controllo



La correzione viene applicata input dell'uniciclo

Azione correttiva MPC-based

Definizione del problema

- Un controllore MPC(Model Predictive Control)-based usa il modello del sistema per *predirne* l'evoluzione futura
- Lavora a tempo discreto, su ogni istante di un *orizzonte temporale* (parametro)
- Ad ogni istante, predice l'evoluzione dello stato del sistema lungo l'orizzonte
- Basandosi su questa, sceglie la sequenza di input da impartire ad ogni istante dell'orizzonte, tale che:
  - Minimizzino una *funzione di costo,* dipendente da input e stato
  - Rispettino dei *vincoli*

Azione correttiva MPC-based

Definizione del problema (cont.)

- L’azione correttiva potrebbe disturbare il tracking
- Va minimizzata per evitare questo fenomeno
- Scegliamo come funzione di costo la somma delle norme quadre delle azioni correttive ad ogni istante dell’orizzonte

$$L(x,u) = \sum_{i=0}^{C} (u_i^{corr})^T u_i^{corr}$$

- Così facendo non ci sarà correzione quando non necessario, per non disturbare il tracking

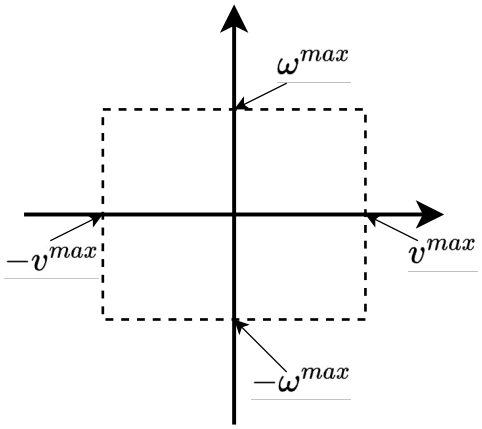
Azione correttiva MPC-based

Definizione del problema (cont.)

- I vincoli devono evitare di portare gli attuatori in saturazione

$$\left| T^{-1}(\theta) u^{track} + u^{corr} \right| \leq \begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix}$$

- La regione degli input ammissibili per l'uniciclo è un rettangolo (per questo si chiamano *box constraints)*



Azione correttiva MPC-based

Implementazione in MATLAB

- L'uso della norma quadra come funzione di costo rende il problema di minimizzazione un problema di programmazione quadratica
- Si usa il solver quadprog di MATLAB, che risolve problemi nella forma

$$\min\left(\frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} + f^T\boldsymbol{x}\right) \text{ con i vincoli } \begin{cases} A\boldsymbol{x} \leq b \\ A_{eq}\boldsymbol{x} = b_{eq} \\ lb \leq \boldsymbol{x} \leq ub \end{cases}$$

- quadprog è un risolutore solver based. Applica un algoritmo di soluzione data la forma del problema
- La forma del problema non rende praticamente utilizzabile l'uso di un risolutore problem-based (in cui si definisce il problema in forma simbolica)

Azione correttiva MPC-based

Implementazione in MATLAB

- Il vettore delle incognite x è un vettore di vettori contenente tutte le u^{corr} dell'orizzonte

$$x = \begin{pmatrix} v_i^{corr} & \omega_i^{corr} & v_{i+1}^{corr} & \omega_{i+1}^{corr} & \cdots & v_{i+C}^{corr} & \omega_{i+C}^{corr} \end{pmatrix}^T$$

- Per scrivere la funzione di costo nella forma necessaria basta porre

$$H = 2 \cdot I_{2C \times 2C}$$

- I vincoli possono essere presi nella forma upper bound/lower bound

$$\begin{cases} u_i^{corr} \leq \begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix} - T_i^{-1}(\theta) u_i^{track} \\ u_i^{corr} \geq - \begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix} - T_i^{-1}(\theta) u_i^{track} \end{cases}$$

Azione correttiva MPC-based

Implementazione in MATLAB

Ad ogni istante t_k

a) Si calcola e memorizza la matrice di disaccoppiamento $T_i^{-1}(\theta)$

b) Si calcola e memorizza u^{track}

c) Si usa u^{track} appena calcolato e u^{corr} proveniente dall'orizzonte calcolalato a t_{k-1} per calcolare l'input del sistema

d) Il sistema viene integrato con Eulero

e) Si ottiene lo stato predetto al tempo t_{k+1}

f) Si itera da (a) fino ad arrivare a t_{k+C}

g) Si costruisce il QP usando u_i^{corr} come incognite e costruendo i vincoli con u_i^{track} e $T_i^{-1}(\theta)$, come visto prima

h) Si chiama quadprog, u_k^{corr} corrente si usa come input del sistema, le restati vengono usati agli istanti successivi

Trajectory tracking + correzione MPC-based (uniciclo)

Simulazioni

- In tutti gli esempi i limiti di saturazione dei comandi sono $\begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$
- Altri parametri $K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, b = 0.12\text{m}$
- Tempo di campionamento 0.1s
- Il timestep per MPC è pari al tempo di campionamento. Per MPC multistep l'orizzonte di predizione è pari a 15 step (1.5s)

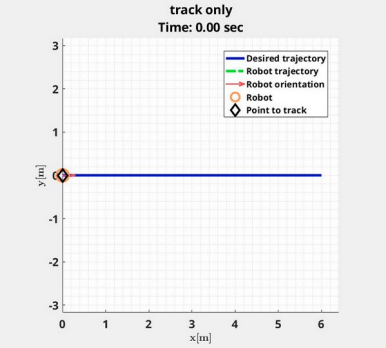
Trajectory tracking + correzione MPC-based

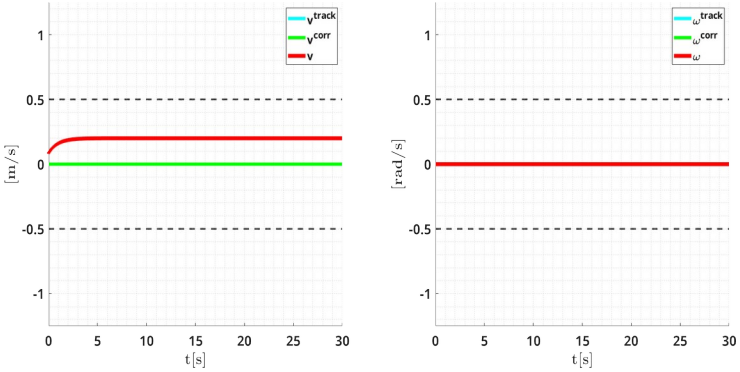
Tracking di una retta

$$\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$$

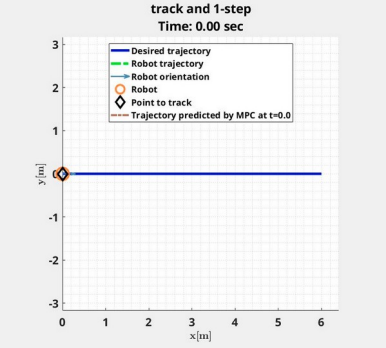
$$q_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$$

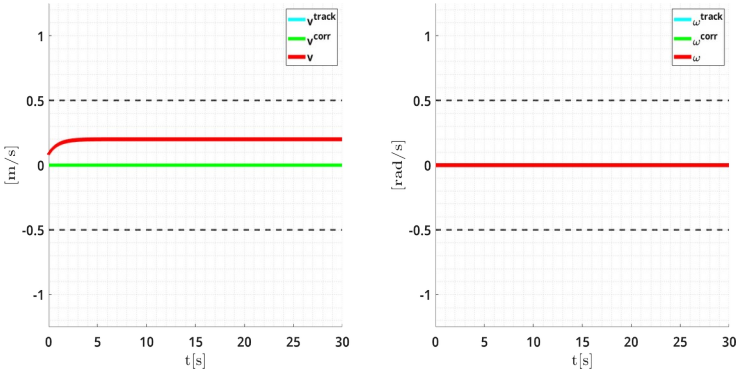
Solo tracking:



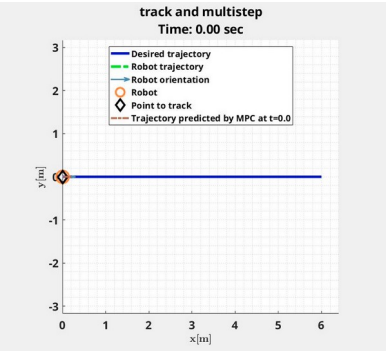


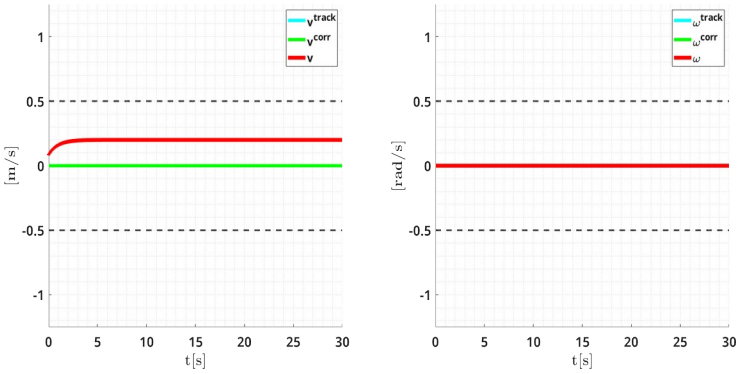
Tracking +
MPC 1-step:





Tracking +
MPC multistep:





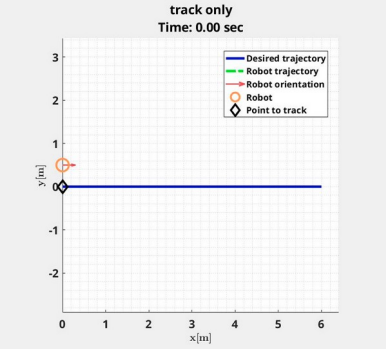
Trajectory tracking + correzione MPC-based

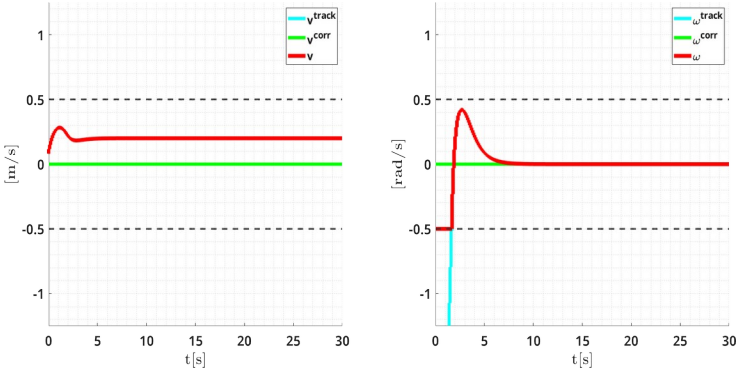
Tracking di una retta con errore iniziale

$$\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$$

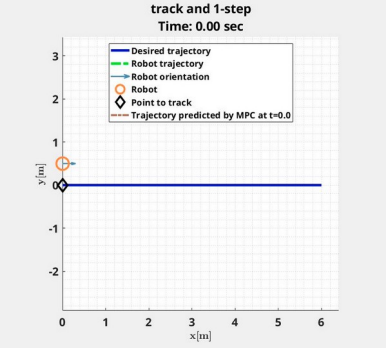
$$q_0 = \begin{pmatrix} 0 & 0.5 & 0 \end{pmatrix}^T$$

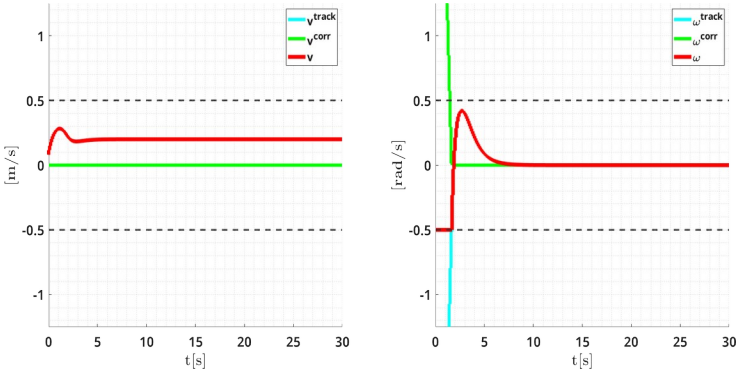
Solo tracking:



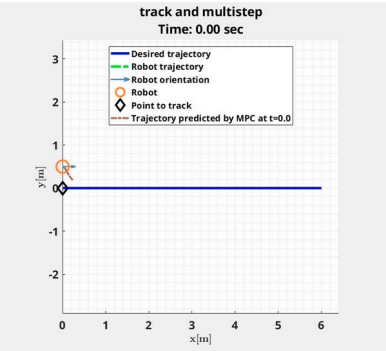


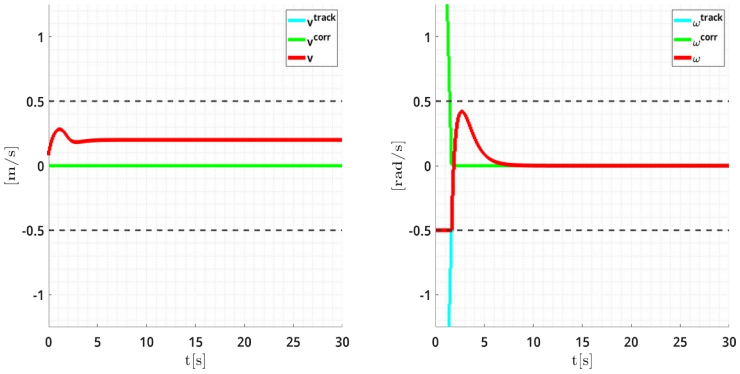
Tracking +
MPC 1-step:





Tracking +
MPC multistep:

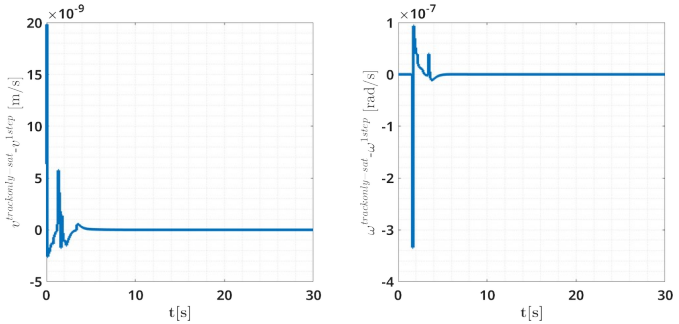




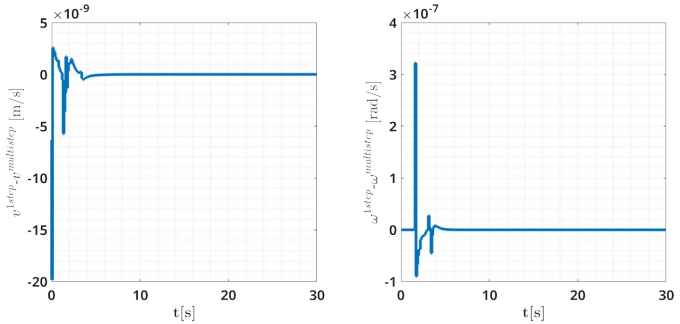
Tracking di una retta con errore iniziale

Differenze fra i controllori

- L’errore iniziale fa saturare ω



Differenze input fra solo tracking e 1-step



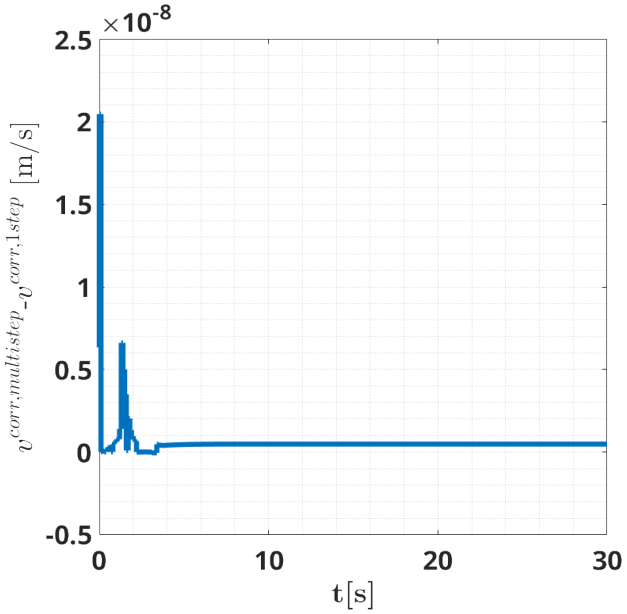
Differenze input fra solo tracking e multistep

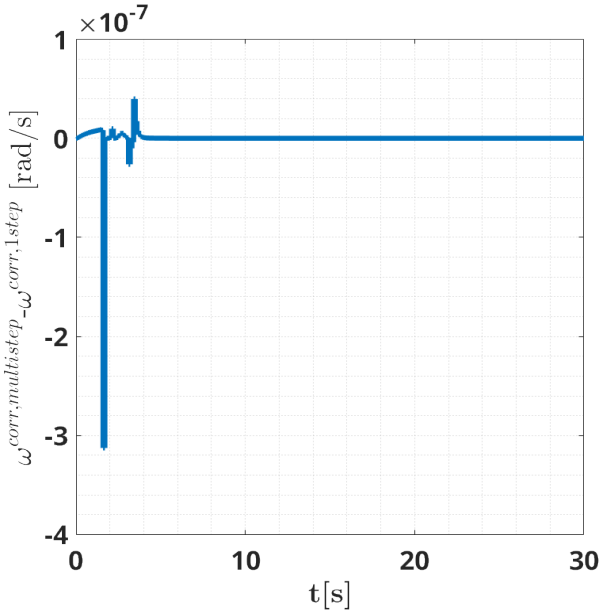
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di una retta con errore iniziale

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a v e ω da 1-step e multistep sono irrisorie





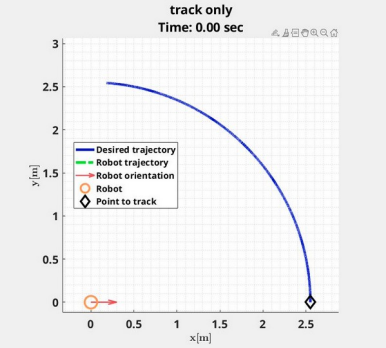
Trajectory tracking + correzione MPC-based

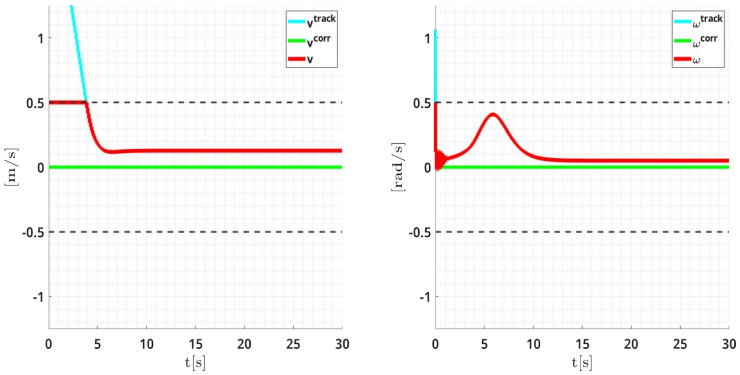
Tracking di un arco di cerchio con errore iniziale

$$\begin{cases} x_{ref} = 2.55\cos(0.05t) \\ y_{ref} = 2.55\sin(0.05t) \end{cases}$$

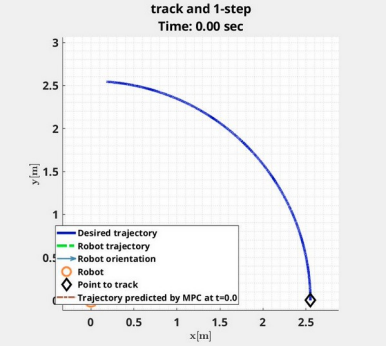
$$q_0 = \begin{pmatrix} 0 & 0 & -\pi/2 \end{pmatrix}^T$$

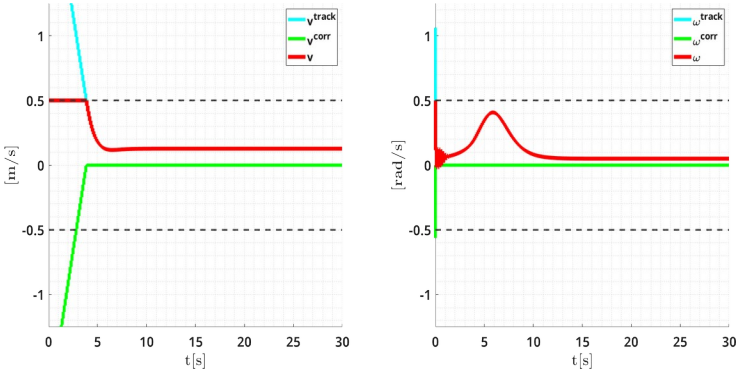
Solo tracking:



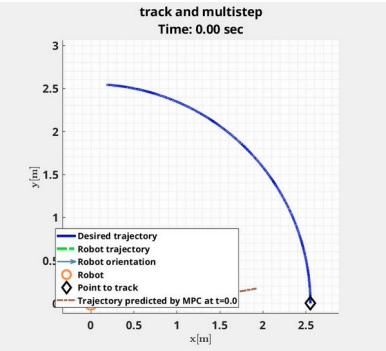


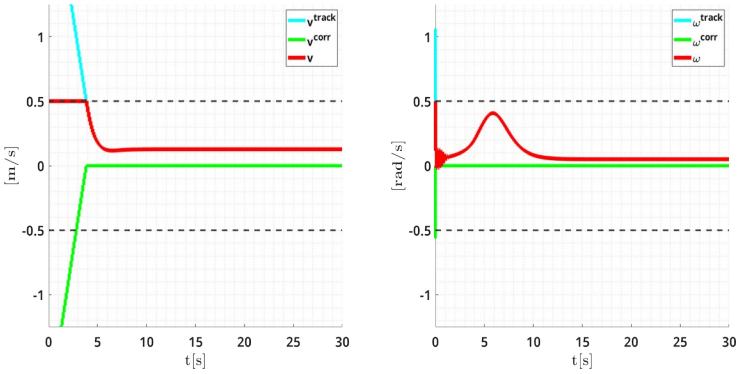
Tracking +
MPC 1-step:





Tracking +
MPC multistep:

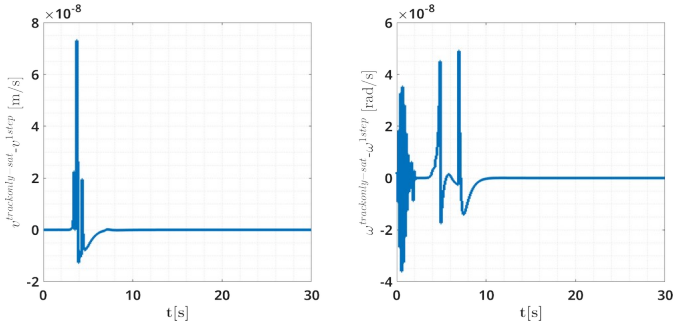




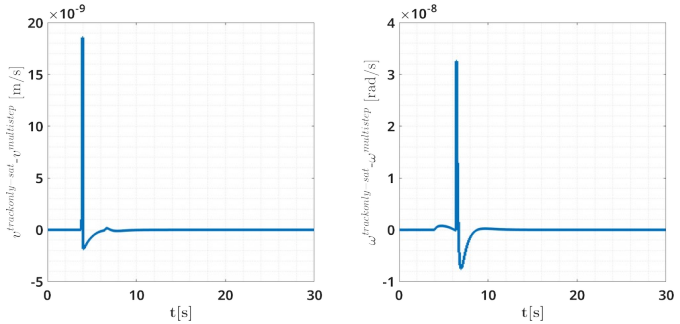
Tracking di un arco di cerchio con errore iniziale

Differenze fra i controllori

- L'errore iniziale fa saturare v e ω. ω inizialmente oscilla



Differenze input fra solo tracking e 1-step



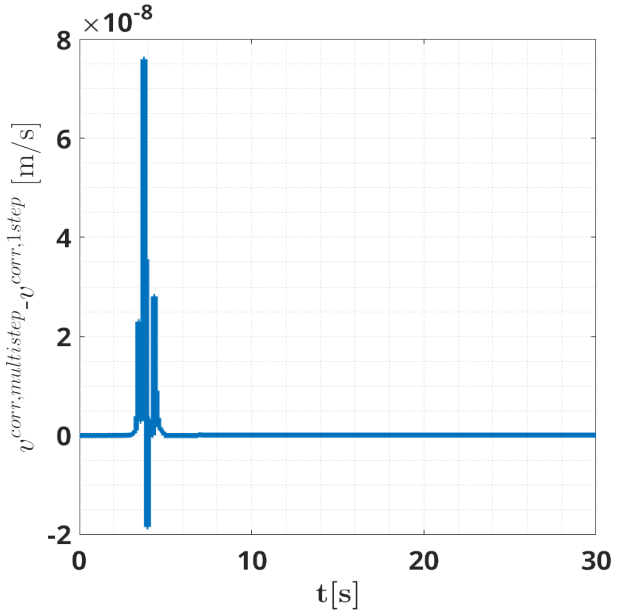
Differenze input fra solo tracking e multistep

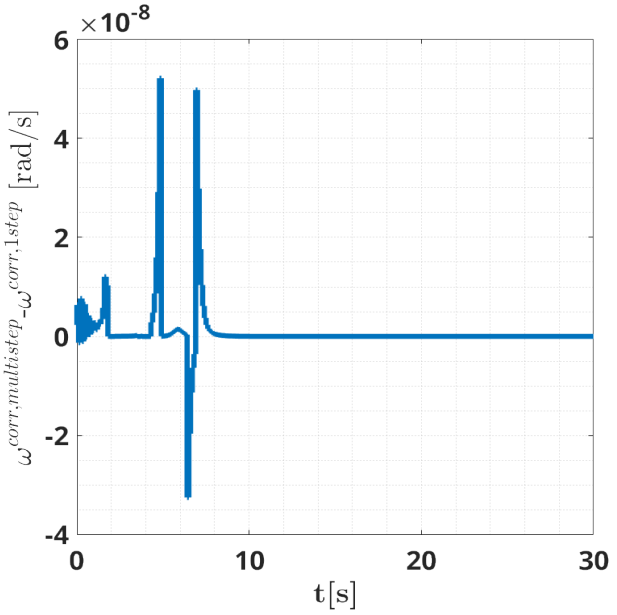
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di un arco di cerchio con errore iniziale

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a v e ω da 1-step e multistep sono irrisorie





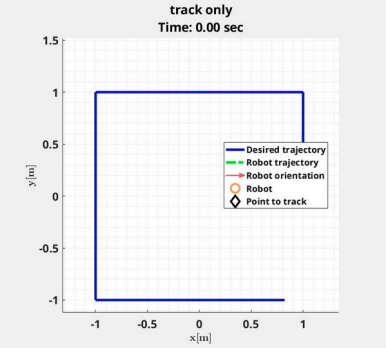
- Le differenze maggiori sono in corrispondenza del punto in cui gli ingressi smettono di saturare

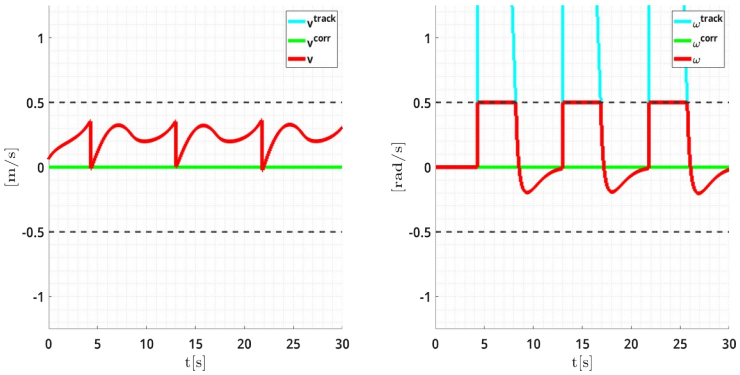
Trajectory tracking + correzione MPC-based

Tracking di un quadrato di lato 2m

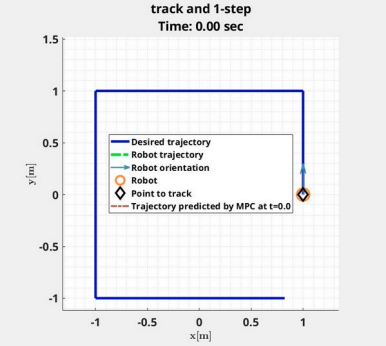
$$q_0 = \begin{pmatrix} 1 & 0 & -\pi/2 \end{pmatrix}^T$$

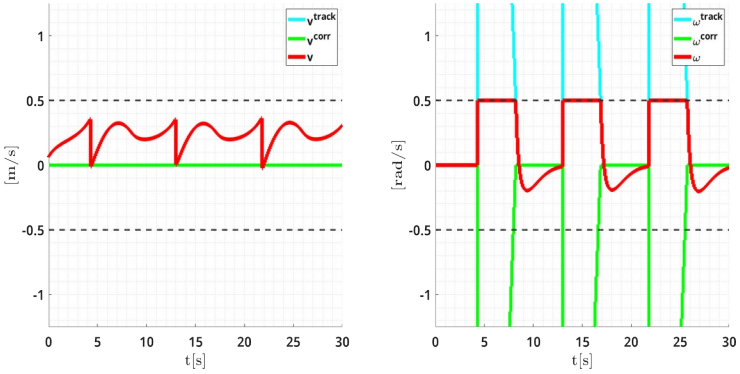
Solo tracking:



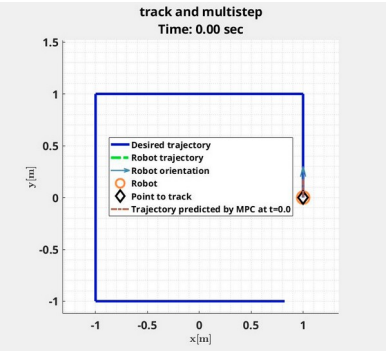


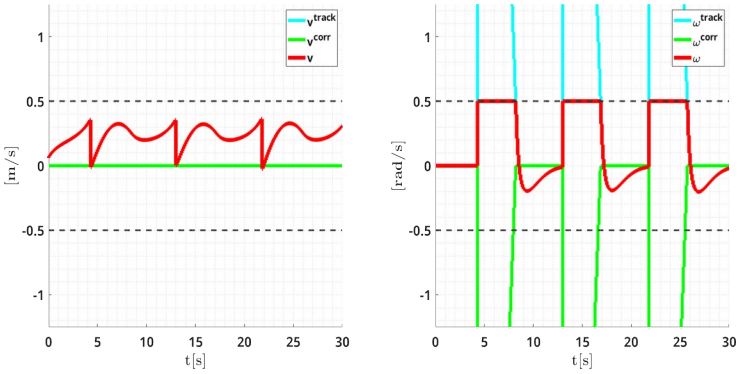
Tracking +
MPC 1-step:





Tracking +
MPC multistep:

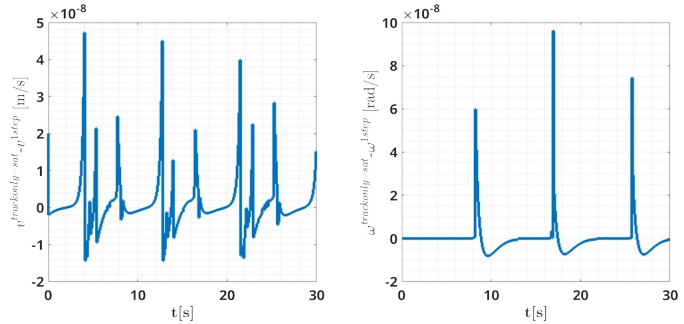




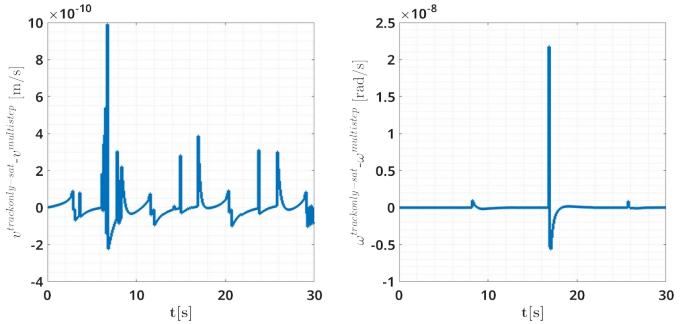
Tracking di una quadrato

Differenze fra i controllori

- La curva ad angolo retto fa saturare ω.



Differenze input fra solo tracking e 1-step



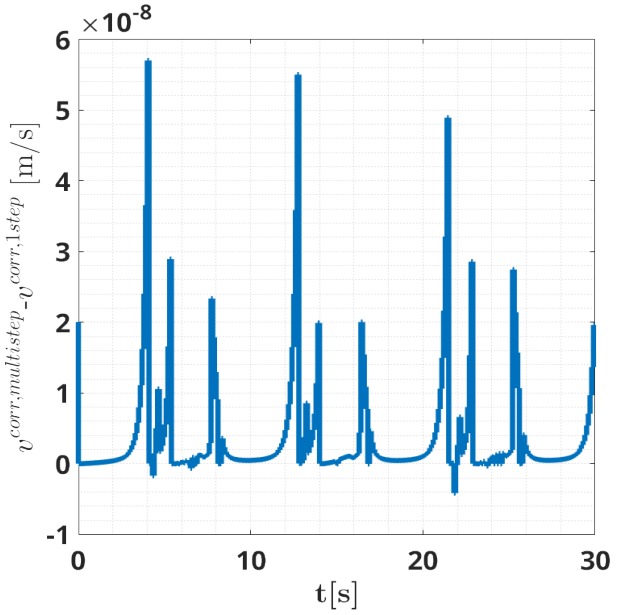
Differenze input fra solo tracking e multistep

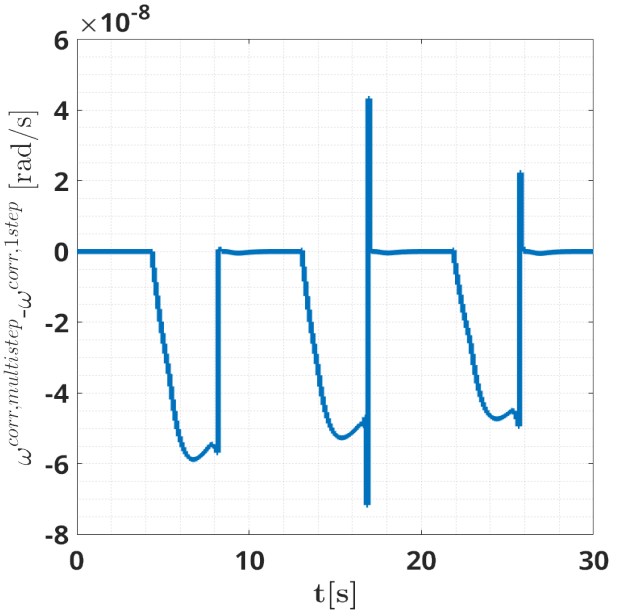
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di una quadrato

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a v e ω da 1-step e multistep sono irrisorie





- Le differenze maggiori sono in corrispondenza dei vertici del quadrato, quando viene effettuata la curva brusca

Trajectory tracking + correzione MPC-based

Differenze fra i controllori

- La correzione ottima (quella a norma quadra minima) è quella che si comporta come la saturazione
- Gli ingressi corretti a monte della saturazione e quelli non corretti a valle della saturazione sono uguali

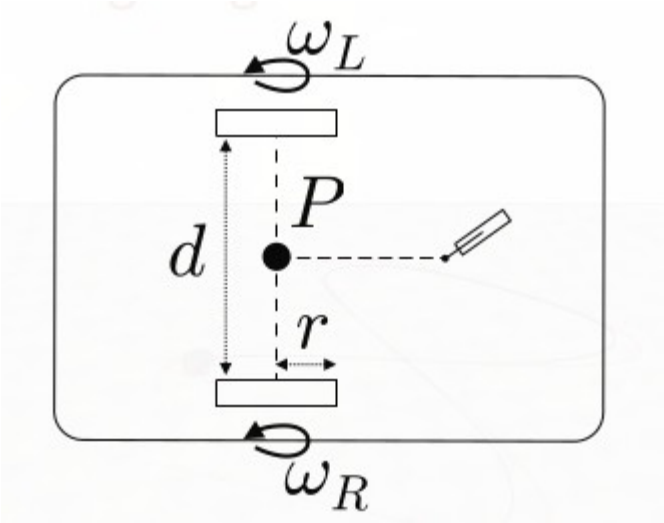
Trajectory tracking + correzione MPC-based

Differenze fra le correzioni

- Inoltre la struttura del problema di ottimizzazione rende le correzioni effettuate ad ogni istante indipendenti fra loro
- In multistep, solo la correzione all'istante corrente viene utilizzata come input, le altre vengono solo usate nelle iterazioni future dell'algoritmo
- Ad un dato istante t_k, 1-step e multistep producono la stessa correzione
- Altre differenze sono imputabili all'accumulo di errori di integrazione in multistep, che vengono corretti dalle correzioni successive

Estensione a differential drive

- Un robot differential drive ha due ruote fisse poste sullo stesso asse attuate indipendentemente + un terzo ruotino libero (caster wheel) usato per mantenere l'equilibrio
- Variando le velocità angolari con cui i motori ruotano si stabiliscono velocità angolare e tangenziale del robot



Estensione a differential drive (cont.)

Si riduce ad un uniciclo equivalente con il cambio di input

$$v = r\frac{\omega_r + \omega_l}{2} \qquad w = r\frac{\omega_r - \omega_l}{d}$$

Che può essere visto in forma matriciale come

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} r/2 & r/2 \\ r/d & -r/d \end{pmatrix} \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} = W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix}$$

Partendo dall'uniciclo, il modello cinematico del differential drive è

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix}$$

Estensione a differential drive (cont.)

Le uscite per il tracking

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = T(\theta) \begin{pmatrix} v \\ \omega \end{pmatrix} = T(\theta) W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix}$$

Scegliendo il punto spostato e la matrice di disaccoppiamento non singolare

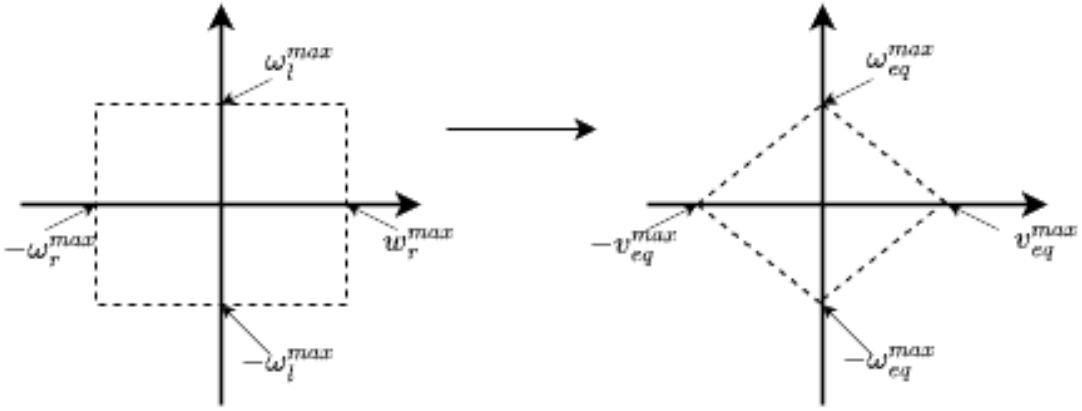
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x + b\cos\theta \\ y + b\sin\theta \end{pmatrix} \implies \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = T(\theta) W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix}$$

Gli input del robot differential drive sono

$$\begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} = \underbrace{W^{-1} T^{-1}(\theta)}_{\text{Usare questa nei vincoli di MPC!}} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Estensione a differential drive (cont.)

Il cambio di input ha effetto sulla regione ammissibile: per (ω_r, ω_l) rimane un rettangolo, mentre per gli input dell'uniciclo equivalente diventa un rombo.



Trajectory tracking + correzione MPC-based (differential drive)

Simulazioni

- In tutti gli esempi i limiti di saturazione dei comandi sono $\begin{pmatrix} \omega_r^{max} \\ \omega_l^{max} \end{pmatrix} = \begin{pmatrix} 2.85 \\ 2.85 \end{pmatrix}$
- Altri parametri $K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, r = 0.1\text{m}, d = 0.15\text{m}, b = 0.12\text{m}$
- Tempo di campionamento 0.1s
- Il timestep per MPC è pari al tempo di campionamento. Per MPC multistep l'orizzonte di predizione è pari a 15 step (1.5s)

- I limiti di saturazione risultano in una velocità lineare massima minore e una velocità angolare maggiore rispetto l'uniciclo

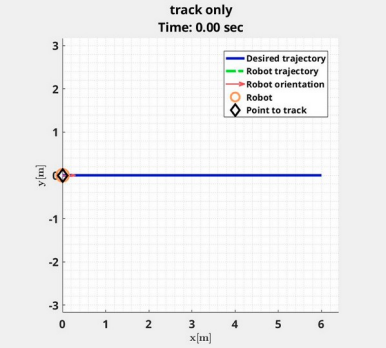
Trajectory tracking + correzione MPC-based

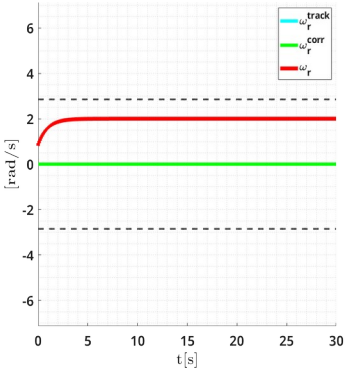
Tracking di una retta

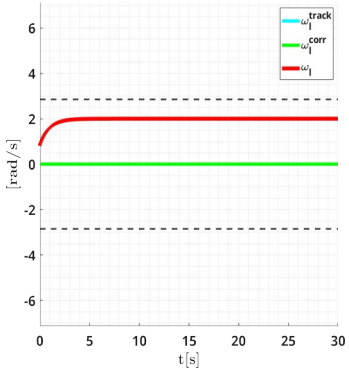
$$\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$$

$$q_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$$

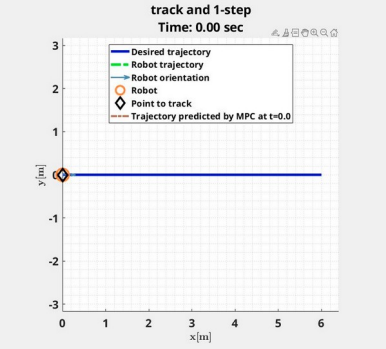
Solo tracking:

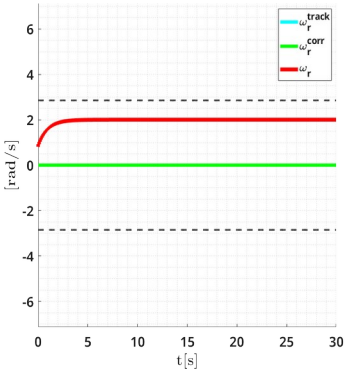


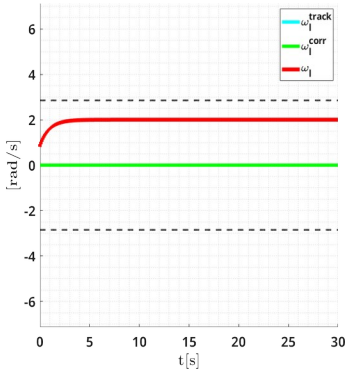




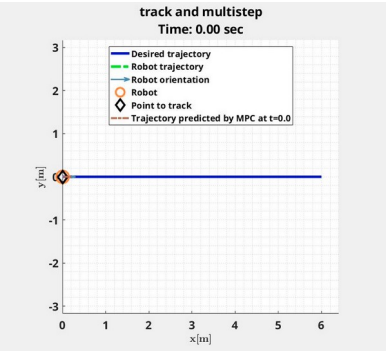
Tracking +
MPC 1-step:

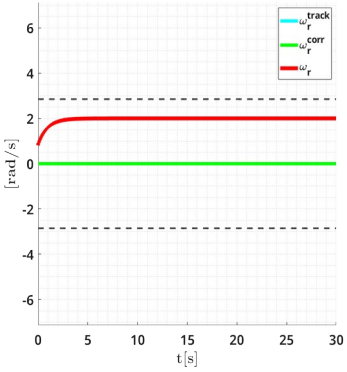


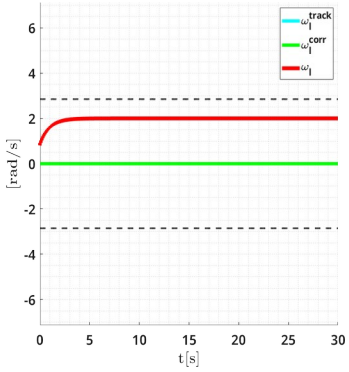




Tracking +
MPC multistep:







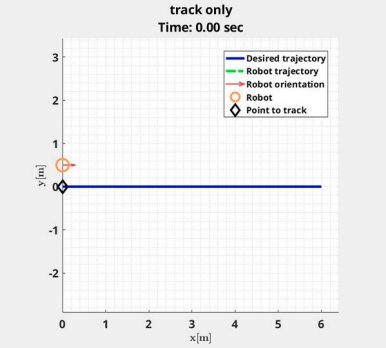
Trajectory tracking + correzione MPC-based

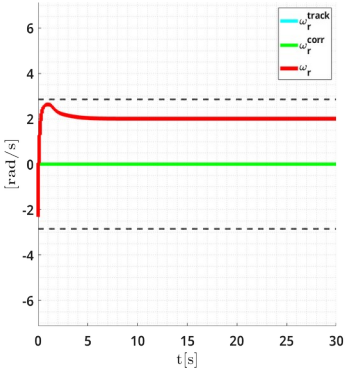
Tracking di una retta con errore iniziale

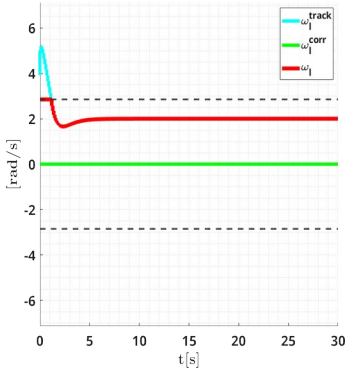
$$\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$$

$$q_0 = \begin{pmatrix} 0 & 0.5 & 0 \end{pmatrix}^T$$

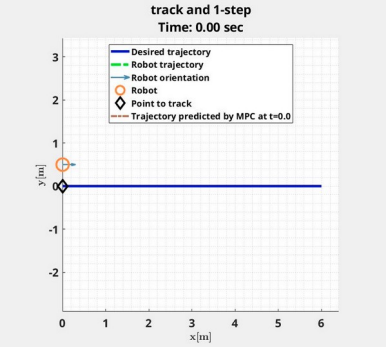
Solo tracking:

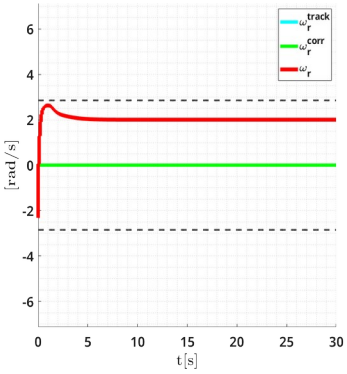


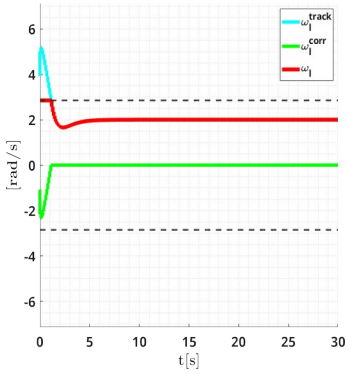




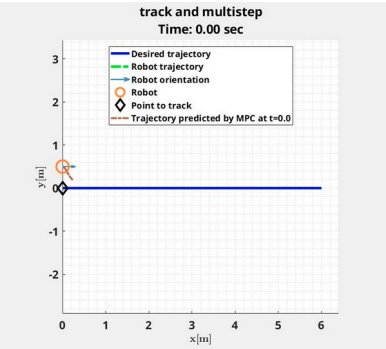
Tracking +
MPC 1-step:

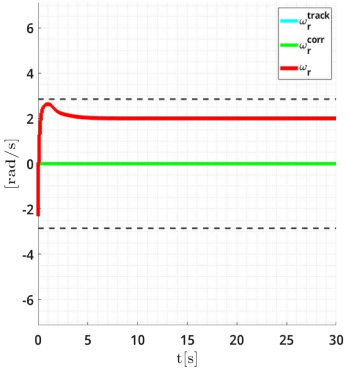


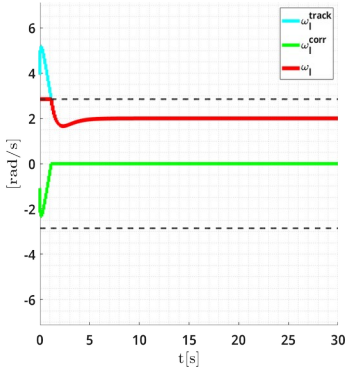




Tracking +
MPC multistep:



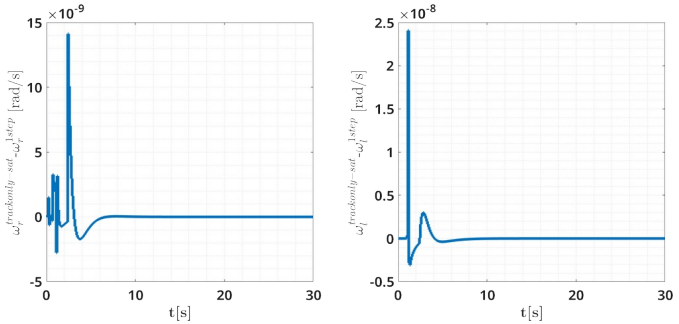




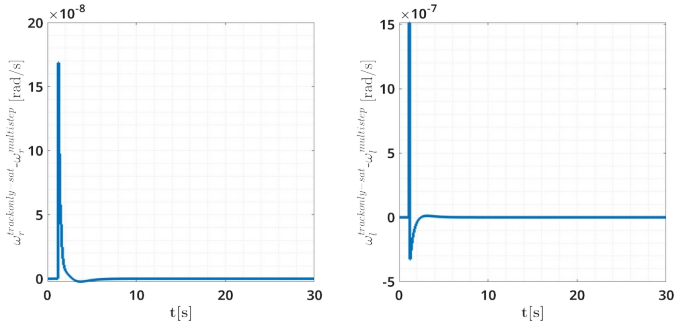
Tracking di una retta con errore iniziale

Differenze fra i controllori

- L'errore iniziale fa saturare ω_l



Differenze input fra solo tracking e 1-step



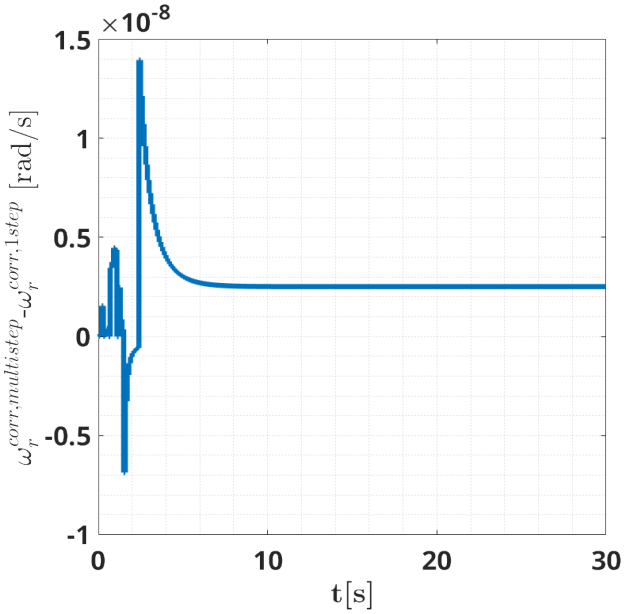
Differenze input fra solo tracking e multistep

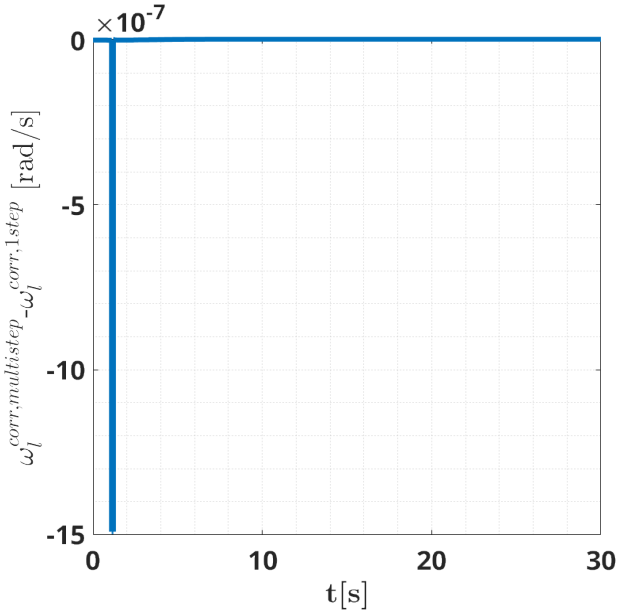
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di una retta con errore iniziale

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a ω_r e ω_l da 1-step e multistep sono irrisorie





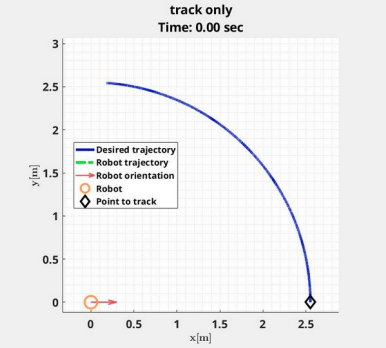
Trajectory tracking + correzione MPC-based

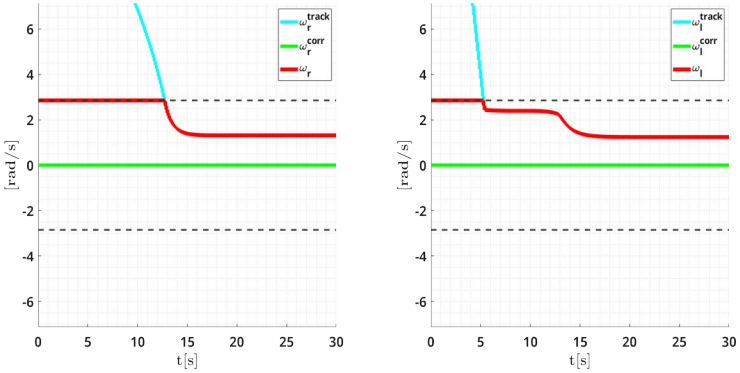
Tracking di un arco di cerchio con errore iniziale

$$\begin{cases} x_{ref} = 2.55\cos(0.05t) \\ y_{ref} = 2.55\sin(0.05t) \end{cases}$$

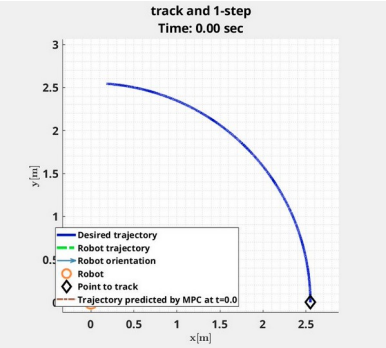
$$q_0 = \begin{pmatrix} 0 & 0 & -\pi/2 \end{pmatrix}^T$$

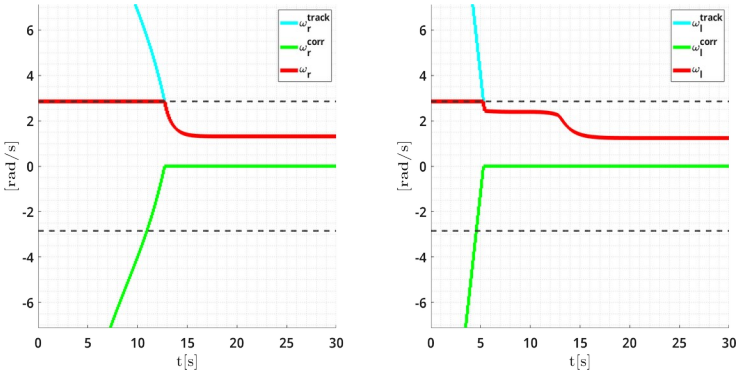
Solo tracking:



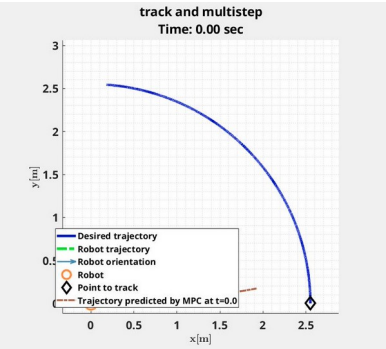


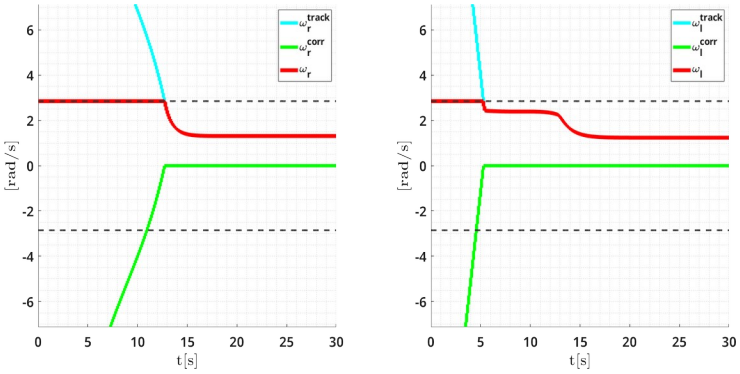
Tracking +
MPC 1-step:





Tracking +
MPC multistep:

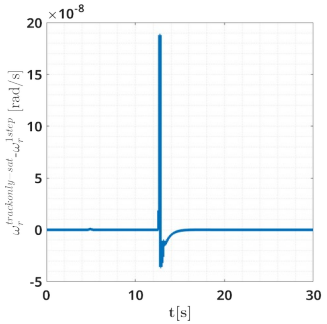


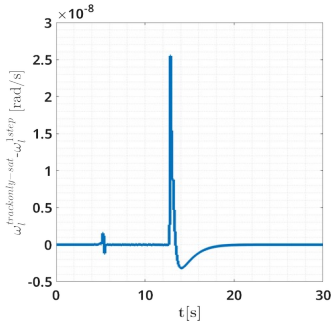


Tracking di un arco di cerchio con errore iniziale

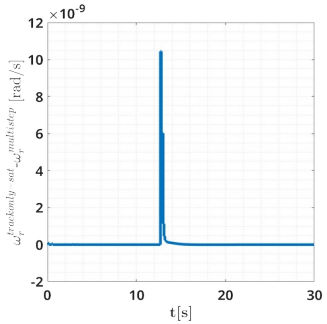
Differenze fra i controllori

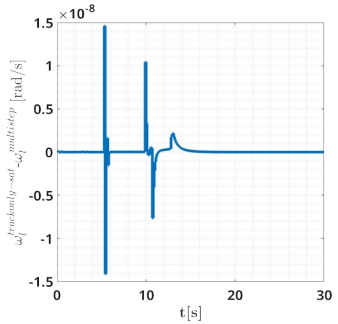
- L'errore iniziale fa saturare ω_r e ω_l





Differenze input fra solo tracking e 1-step





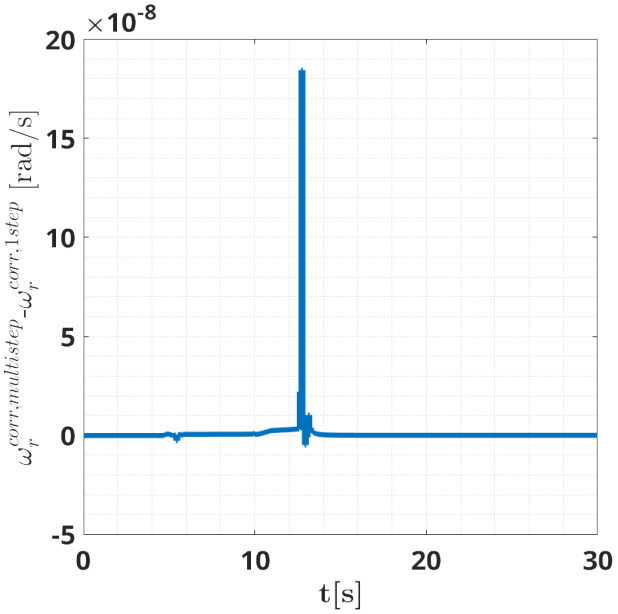
Differenze input fra solo tracking e multistep

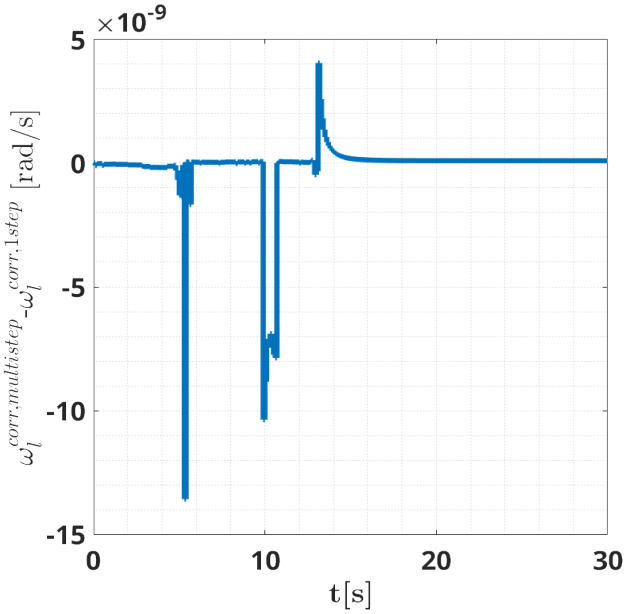
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di una retta con errore iniziale

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a ω_r e ω_l da 1-step e multistep sono irrisorie



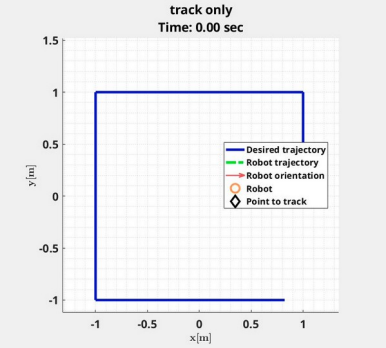


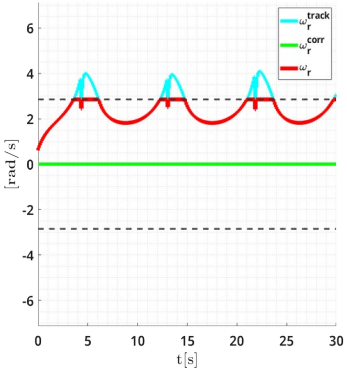
Trajectory tracking + correzione MPC-based

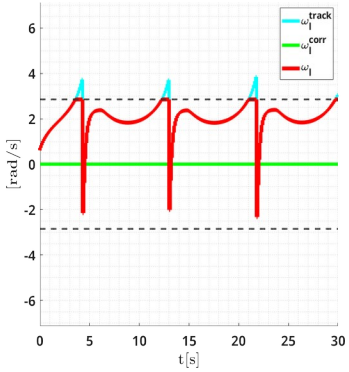
Tracking di un quadrato di lato 2m

$$q_0 = \begin{pmatrix} 1 & 0 & -\pi/2 \end{pmatrix}^T$$

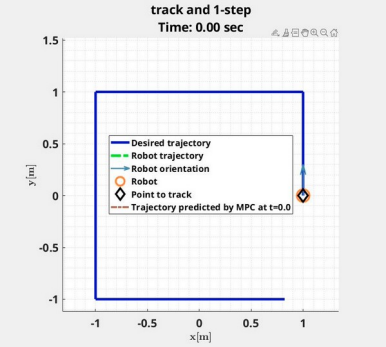
Solo tracking:

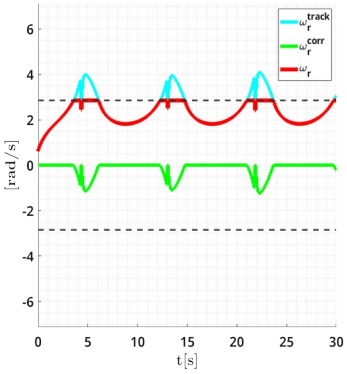


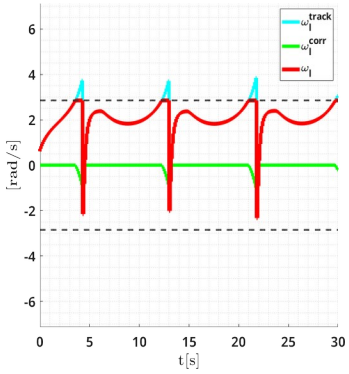




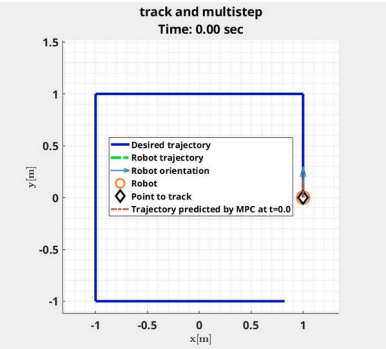
Tracking +
MPC 1-step:

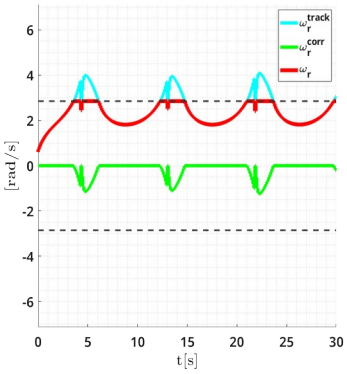


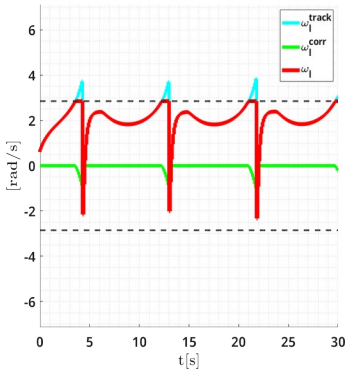




Tracking +
MPC multistep:



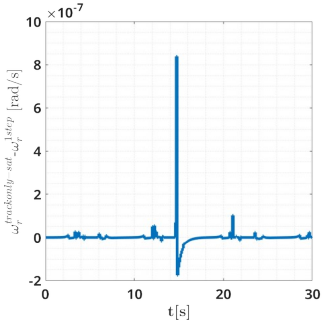


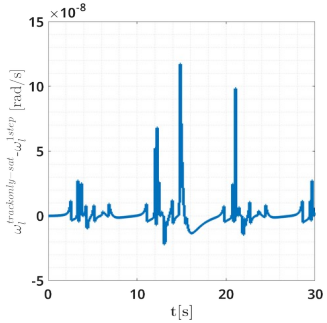


Tracking di un arco di cerchio con errore iniziale

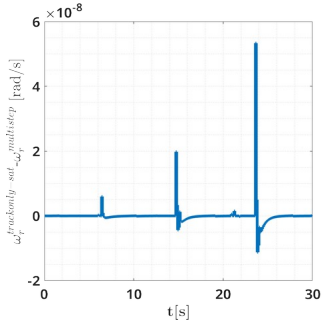
Differenze fra i controllori

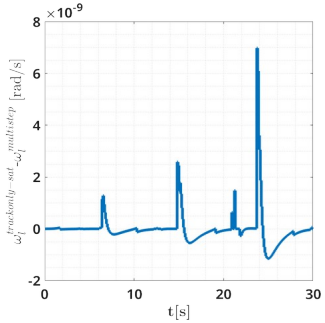
- Al contrario dell'uniciclo, gli attuatori non saturano





Differenze input fra solo tracking e 1-step





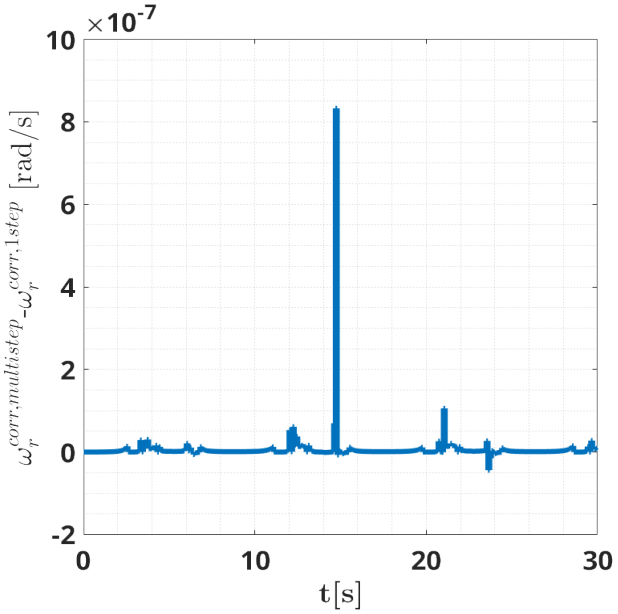
Differenze input fra solo tracking e multistep

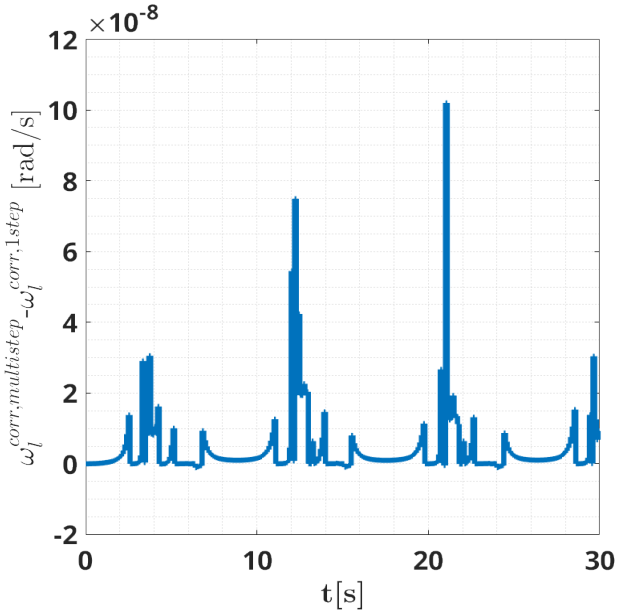
- La correzione ottima (minima) è quella che si comporta come la saturazione
- Le differenze sono ignorabili, dovuti ad errori di integrazione fra Eulero/ode45 e di rappresentazione dei numeri a virgola mobile

Tracking di una retta con errore iniziale

Differenze fra i controllori

- Anche le differenze fra le correzioni apportate a ω_r e ω_l da 1-step e multistep sono irrisorie





Trajectory tracking + correzione MPC-based

Differenze con uniciclo

- Come nell’uniciclo, la correzione ottima (quella a norma quadra minima) è quella che si comporta come la saturazione
- Il comportamento è analogo all’uniciclo: tutti e tre i controllori hanno lo stesso comportamento
- Le differenze nelle traiettorie sono dovute a limiti di saturazione minori per il differential drive, dovuti alla scelta dei parametri

Conclusioni

- L’aggiunta di una correzione MPC-based evita la saturazione degli attuatori
- In un caso reale, i vincoli di MPC possono essere resi più ancora stringenti dei limiti di saturazione, per essere sicuri di tenersene fuori
- Le differenze fra MPC multi-step e 1-step sono pressoché ignorabili, questo deriva dalla forma particolare del problema
- Ci si può limitare a 1-step per un’implementazione più computazionalmente efficiente