SAPIENZA
UNIVERSITÀ DI ROMA

# Controllo di un robot mobile in presenza di saturazioni sugli ingressi di controllo


Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Laurea in Ingegneria Informatica e Automatica

**Emanuele Coletta**
Matricola 2001600

Relatore
Prof. Giuseppe Oriolo



Anno Accademico 2023/2024

**Controllo di un robot mobile in presenza di saturazioni sugli ingressi di controllo**
Tesi sperimentale retrospettiva monocentrica. Sapienza Università di Roma

© 2024 Emanuele Coletta. Tutti i diritti riservati

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: coletta.2001600@studenti.uniroma1.it

*A Ginevra*

# Capitolo 1

# Introduzione

## 1.1 Scopo del progetto

Nell'ambito della robotica mobile è spesso di interesse far seguire ad un robot una traiettoria desiderata. Quest'azione prende il nome di *trajectory tracking*. Si differenzia dall'inseguimento di un percorso (*path following*) in quanto una traiettoria implica, oltre l'inseguimento del percorso, il rispetto di una legge oraria con cui questo deve avvenire. Tuttavia, in un caso reale, non è detto che tutte le traiettorie che si desidera seguire possano essere effettivamente seguite con successo: alcune traiettorie potrebbero richiedere sforzi di controllo non fisicamente realizzabili dagli attuatori utilizzati sul robot, facendoli così entrare in regime di saturazione. La saturazione è un regime del sistema fortemente non lineare e non linearizzabile. Far operare attuatori in regime di saturazione, soprattutto per tempi prolungati, ne può causare il deterioramento e la rottura ed è quindi desiderabile evitarla.

In questa tesi viene affrontato il problema di eseguire *trajectory tracking* con robot di tipo uniciclo e differential drive in presenza di limiti di saturazione sugli attuatori. Come detto, i limiti di saturazione limitano le traiettorie fisicamente realizzabili dal robot. L'obiettivo è la realizzazione di un controllore che applichi un'*azione correttiva* a quella di tracking al fine di assicurarsi che venga richiesto uno sforzo di controllo che rientri nei limiti di saturazione. Lo studio viene fatto in assenza di ostacoli sul percorso da seguire.

Il controllore che realizza il tracking viene ottenuto tramite linearizzazione input-output del sistema. La formulazione di questo controllore viene esposta nel capitolo 2. Non è garantito che questo controllore generi sforzi di controllo che non saturino gli attuatori. In caso accadesse, il tracking della traiettoria potrebbe essere errato e non prevedibile. In questo progetto, per evitare che gli attuatori entrino in regime di saturazione viene aggiunta un'azione correttiva, ottenuta con un controllore basato su MPC (MPC-based), che modifica l'azione di tracking per imporre che l'azione compiuta dagli attuatori, somma di quella correttiva e quella di tracking, rientri nei limiti di saturazione.

Il controllore MPC-based minimizza una funzione obiettivo desiderata all'interno di un dominio limitato da determinati vincoli, prevedendo il comportamento del sistema in istanti di tempo che si susseguono all'interno di un orizzonte finito. Il controllore è quindi intrinsecamente a tempo discreto. L'azione correttiva, per costruzione, previene il tracking ideale e quindi la funzione obiettivo dovrà essere scelta in modo da minimizzare l'errore di tracking causato dalla correzione. La formulazione esatta del controllore MPC viene esposta nel capitolo 3.

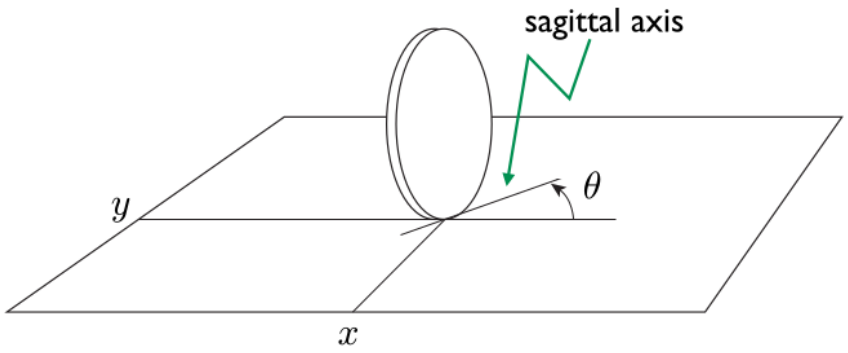



**Figura 1.1** Il robot uniciclo [1]

Verrà analizzato il comportamento del sistema sotto l'azione di tre controllori diversi: il primo senza azione correttiva; il secondo con azione correttiva e un orizzonte di predizione di un singolo step; l'ultimo con azione correttiva e un orizzonte di predizione pari a quindici step (1,5 secondi). In tutti e tre i casi il tracking viene realizzato da un controllore ottenuto per linearizzazione input-output come specificato nel capitolo 2. Il comportamento del robot sotto l'azione dei tre controllori è stato simulato in MATLAB eseguendo prove su una varietà di traiettorie e in una varietà di condizioni. Sono di particolare interesse le traiettorie che possono portare il sistema in saturazione, ad esempio quelle che richiedono velocità elevate, superiori a quelle realizzabili dagli attuatori, le traiettorie che presentano curvatura molto strette o errori iniziali elevati.

## 1.2 Robot uniciclo

L'uniciclo è un robot che presenta una sola ruota, attuata e orientabile, in grado di ruotare attorno all'asse verticale che passa per il suo centro. Nella pratica un tale robot ha bisogno di un controllore per il mantenimento dell'equilibrio. Il modello dell'uniciclo ignora questa necessità e assume che il robot possa rimanga sempre in equilibrio senza bisogno di un controllore apposito. Per questo è considerato un modello puramente teorico. Sotto queste condizioni, l'uniciclo può sia traslare avanti e indietro lungo l'asse definito sagittale, che cambiare orientamento ruotando attorno all'asse perpendicolare al piano e passante per il centro della ruota. Nonostante l'uniciclo ideale non sia praticamente realizzabile, i modelli cinematici di altri robot praticamente realizzabili possono essere ricondotti all'uniciclo tramite un cambio di input. Nel capitolo 4 viene affrontato il caso del robot differential drive.

Inoltre la ruota è sottoposta ad un vincolo di puro rotolamento: non essendo omnidirezionale essa non può traslare perpendicolarmente all'asse sagittale senza slittare. Per questo, l'asse normale a quello sagittale viene definito anche *zero-motion line*. Dell'uniciclo viene definito un modello cinematico, ovvero in cui compaiono solo le velocità generalizzate (derivate prime delle coordinate generalizzate). Questo è possibile perché nell'ambito della robotica mobile gli attuatori spesso hanno dei controllori PID di basso livello che prendono come riferimento una velocità e gestiscono internamente la dinamica dell'attuatore.

Quello di puro rotolamente è un vincolo cinematico non-olonomico. Sotto di esso si deve avere

$$\begin{pmatrix} \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = n^T \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = 0 \tag{1.1}$$

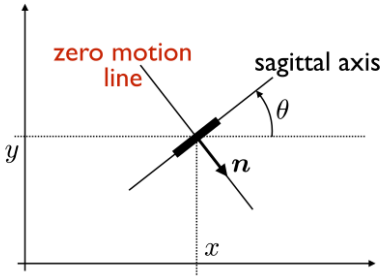



**Figura 1.2** La *zero-motion line* del robot uniciclo [1]

Dove $\begin{pmatrix}\sin\theta & -\cos\theta\end{pmatrix}$ è la zero-motion line: l'asse normale a quello sagittale. Imporre l'uguaglianza a zero indica che i due vettori devono rimanere perpendicolari fra loro e che quindi il movimento può avvenire solo lungo l'asse sagittale e non lungo quello normale, per il quale la ruota dovrebbe slittare.

Per ottenere un modello cinematico a partire dal vincolo, lo si riscrive in maniera tale da far comparire tutte le coordinate generalizzate. Come indicato in figura 1.1, per l'uniciclo queste sono $x, y$ e $\theta$. L'equazione (1.1) viene riscritta come

$$\begin{pmatrix}\sin\theta & -\cos\theta & 0\end{pmatrix}\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = A^T(q)\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = 0 \tag{1.2}$$

Tutte le velocità ammissibili appartengono quindi a $ker(A^T(q))$, per cui una base valida è

$$\left\{\begin{pmatrix}cos\theta\\ sin\theta\\ 0\end{pmatrix}, \begin{pmatrix}0\\ 0\\ 1\end{pmatrix}\right\}$$

Da cui si ottiene il modello cinematico dell'uniciclo

$$\begin{cases}\dot{x} = v\cos\theta\\ \dot{y} = v\sin\theta\\ \dot{\theta} = \omega\end{cases} \tag{1.3}$$

Il nome dei parametri è stato scelto per evidenziare il loro ruolo: $\omega$ è la velocità angolare con cui il robot ruota attorno al all'asse passante per il proprio centro, mentre $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ è la velocità lineare con cui il robot procede.

# Capitolo 2

# Trajectory tracking con uniciclo con linearizzazione input-output

Si consideri un sistema privo di evoluzione libera (*driftless*) del tipo

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

allora $\dot{y} = \dfrac{\partial h}{\partial x}\dot{x} = \dfrac{\partial h}{\partial x}G(x)\ u = T(x)\ u$. Nel caso la *matrice di disaccoppiamento* $T(x)$ sia invertibile si può scegliere un nuovo input $v$ e impostare $u = T^{-1}(x)\ v$ e quindi $\dot{y} = T(x)T^{-1}(x)\ v = v$ stabilendo così una mappa lineare fra gli input $v$ e le derivate temporali degli output.

Il sistema si comporta dunque come un integratore il cui ingresso è $v = \dot{y}$ e la cui uscita è $y$.

Un integratore può essere stabilizzato con feedback proporzionale, a cui viene aggiunto un termine di feedforward per poter seguire traiettorie arbitrarie. Si definisce l'errore come la differenza fra la traiettoria desiderata e quella attuale

$$e = y_d - y$$

la cui dinamica è

$$\dot{e} = \dot{y_d} - \dot{y} = \dot{y_d} - v$$

Scegliendo $v$ per imporre il termine di feedback proporzionale e quello di feedforward

$$v = \dot{y_d} + Ke \tag{2.1}$$

la dinamica dell'errore diventa

$$\dot{e} = -Ke$$

La stabilità asintotica è dunque garantita fintanto che gli autovalori di $K$ appartengono al semipiano positivo. Ne risulta che l'ingresso del sistema è

$$u = T^{-1}(x)(\dot{y_d} + Ke)$$

È ragionevole inoltre scegliere $K$ diagonale per evitare fenomeni di ri-accoppiamento.

Nel caso dell'uniciclo è di interesse tracciare la posizione cartesiana, quindi si sceglie come uscita il vettore $\begin{pmatrix} y_1 & y_2 \end{pmatrix}^T = \begin{pmatrix} x & y \end{pmatrix}^T$, le cui derivate sono $\begin{pmatrix} \dot{y_1} & \dot{y_2} \end{pmatrix}^T = \begin{pmatrix} \dot{x} & \dot{y} \end{pmatrix}^T$

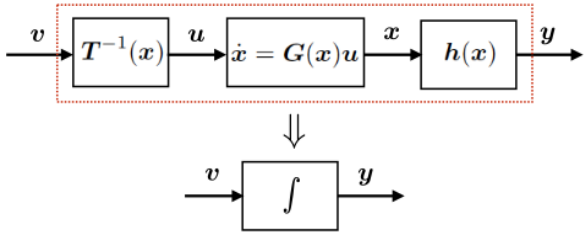



**Figura 2.1** Il sistema si comporta come un integratore [1]

Da, (1.3), questo rende la matrice di disaccoppiamento $T(x) = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{pmatrix}$: singolare e quindi non invertibile. Per ovviare a questo problema si esegue il tracking non del centro esatto del robot, ma di un punto $B$ a distanza $b$ da esso, rendendo le uscite desiderate

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x + b\cos\theta \\ y + b\sin\theta \end{pmatrix} \tag{2.2}$$

per cui

$$\begin{pmatrix} \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} \dot{x} - b\sin\theta\dot{\theta} \\ \dot{y} + b\cos\theta\dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta - b\sin\theta\omega \\ v\sin\theta + b\cos\theta\omega \end{pmatrix} \tag{2.3}$$

e quindi la matrice di disaccoppiamento diventa

$$T(\theta) = \begin{pmatrix} \cos\theta & -b\sin\theta \\ \sin\theta & b\cos\theta \end{pmatrix} \tag{2.4}$$

la cui inversa è

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

Scegliendo inoltre gli ingressi come (2.1):

$$\boldsymbol{u^{track}} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \dot{x}_{ref} + k_x(x_{ref} - x_B) \\ \dot{y}_{ref} + k_y(y_{ref} - y_B) \end{pmatrix} \tag{2.5}$$

Gli input del sistema uniciclo sono

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = T^{-1}(\theta)\ \boldsymbol{u^{track}} \tag{2.6}$$

da cui risulta infine che il sistema linearizzato input-output è

$$\begin{cases} \dot{x} = u_1 = \dot{x}_{ref} + k_x(x_{ref} - x_P) \\ \dot{y} = u_2 = \dot{y}_{ref} + k_y(y_{ref} - y_P) \\ \dot{\theta} = \dfrac{u_2\cos\theta - u_1\sin\theta}{b} \end{cases} \tag{2.7}$$

Il cui schema di controllo è raffigurato in figura 2.2.

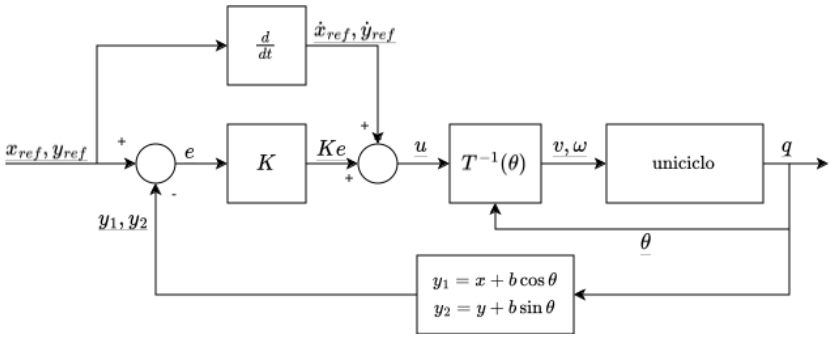



**Figura 2.2** Schema di controllo per trajectory tracking con uniciclo con linearizzazione input-output

# Capitolo 3

# Azione correttiva con controllore MPC-based

Lo scopo dell'azione correttiva $\boldsymbol{u^{corr}}$ è modificare l'azione di tracking in maniera tale che la loro somma non superi i limiti di saturazione. Nello schema di controllo utilizzato, l'azione correttiva viene aggiunta a quella di tracking [2] come in figura 3.1.

In questo progetto, l'azione correttiva è realizzata con un controllore MPC-based. Un controllore MPC(*Model Predictive Control*)-based utilizza il modello matematico del sistema per prevederne l'evoluzione futura, ad istanti discreti di un *orizzonte di predizione*. L'evoluzione predetta, e la sequenza di input che la realizzano, devono minimizzare una *funzione di costo*. Lo stato e gli ingressi del sistema devono inoltre rispettare dei vincoli. Questo imposta un problema di ottimizzazione le cui variabili di decisioni sono la sequenza di input da applicare in ogni istante dell'orizzonte. Generalmente, l'evoluzione dello stato del sistema è calcolata on-line durante la risoluzione del problema di ottimizzazione. Della soluzione, solo l'ingresso relativo al primo istante viene utilizzato come ingresso del sistema.

La forma generale di un controllore MPC è

$$\begin{aligned} u^* = \arg\min \quad & \sum_{i=0}^{C-1} L(\boldsymbol{x}_i, \boldsymbol{u}_i) + E(\boldsymbol{x}_N) \\ \text{t.c.} \quad & x_0 = \hat{x}_0 \\ & x_{i+1} = F(\boldsymbol{x}_i, \boldsymbol{u}_i), i = 0, \cdots, C-1 \\ & G(\boldsymbol{x}_i, \boldsymbol{u}_i) \leq 0, i = 0, \cdots, C-1 \end{aligned}$$

In questa scrittura, il secondo vincolo esprime come il controllore predice l'evoluzione del sistema; il primo vincolo indica la condizione iniziale; il terzo vincolo invece esprime uno o più vincoli generici su stato ed ingressi.

**Funzione di costo**

$$\sum_{i=0}^{C} (\boldsymbol{u_i^{corr}})^T \ \boldsymbol{u_i^{corr}} \tag{3.2}$$

La funzione di costo scelta è (3.2): la somma delle norme quadre della correzione effettuata ad ogni istante nell'orizzonte. L'obiettivo è quindi la minimizzazione della correzione stessa, dato che, per costruzione, questa perturba il tracking esatto della traiettoria [2]. La correzione non viene dunque applicata se non ve ne è bisogno, ovvero se nessun vincolo è attivo.

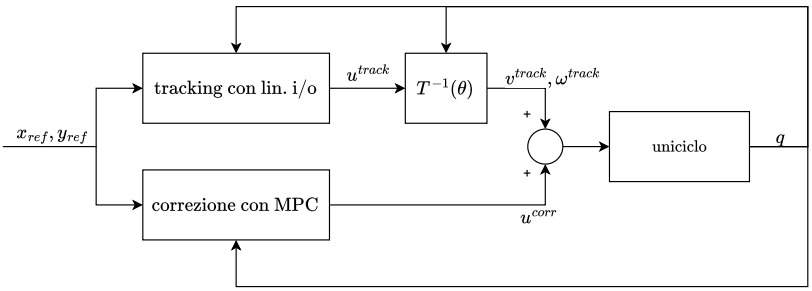



**Figura 3.1** Schema di controllo con azione correttiva

**Vincoli** I vincoli di interesse sono quelli relativi alla saturazione degli attuatori, ovvero

$$\left| \begin{pmatrix} v \\ \omega \end{pmatrix} \right| \leq \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} \text{ oppure } -\begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} \leq \begin{pmatrix} v \\ \omega \end{pmatrix} \leq \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} \tag{3.3}$$

Il vincolo così definisce una regione ammissibile per gli input $v$, $\omega$ a forma di rettangolo. Per questo motivo questo tipo di vincoli assume anche il nome di *box constraints* (figura 3.2). Per come sono stati definiti gli input dell'uniciclo in (2.6) e tenendo in considerazione l'aggiunta dell'azione correttiva come mostrato in figura 3.1, il vincolo può essere riscritto come

$$-\begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} \leq T^{-1}(\theta)\ \boldsymbol{u^{track}} + \boldsymbol{u^{corr}} \leq \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} \tag{3.4}$$

Per quanto riguarda MPC, ne risulta che ad ogni istante dell'orizzonte di predizione, a partire da $t_k$, debbano valere i limiti inferiori e superiori

$$\begin{cases} u_i^{corr} \leq \begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix} - T_i^{-1}(\theta)\boldsymbol{u_i^{track}} \\ u_i^{corr} \geq -\begin{pmatrix} v^{max} \\ \omega^{max} \end{pmatrix} - T_i^{-1}(\theta)\boldsymbol{u_i^{track}} \end{cases} \quad \text{per } i = 0, ..., C \tag{3.5}$$

**Implementazione** La struttura del vincolo (3.5) evidenza la dipendenza di ogni vincolo da $T_{k+i}^{-1}(\theta)$ e $\boldsymbol{u_{k+i}^{track}}$, le quali possono essere calcolate a partire dallo stato attuale del sistema $\boldsymbol{q_{k+i}}$. Tuttavia quest'ultimo dipende dalla correzione effettuate all'istante precedente $\boldsymbol{u_{k+i-1}^{corr}}$, che è una variabile di decisione. Il fatto che $T_{k+i}^{-1}(\theta)$ dipende da $\boldsymbol{q_{k+i}}$ attraverso funzioni trigonometriche rende la dipendenza non lineare e il problema difficile da costruire e risolvere.

In questo progetto l'evoluzione del sistema viene invece calcolata offline, prima della risoluzione del problema di ottimizzazione, integrando numericamente il sistema secondo Runge-Kutta al secondo ordine e usando i valori di $\boldsymbol{u^{corr}}$ come provenienti dall'orizzonte di predizione calcolato dall'esecuzione di MPC all'istante precedente $t_{k-1}$. Di questi, il primo elemento deve essere scartato, in quanto si riferisce all'istante passato $t_{k-1}$, e si deve predisporre un nuovo elemento alla fine dell'orizzonte, che può essere impostato a zero per convenienza. Nel fare questo, si sta di fatto facendo "slittare" l'orizzonte in avanti nel tempo di un istante. Calcolare l'evoluzione del sistema in questo modo costituisce di fatti un passo di linearizzazione.

La scelta della funzione di costo rende il problema di ottimizzazione un problema di programmazione quadratica (QP - Quadratic Programming). MATLAB fornisce il risolutore quadprog, che prende in input un problema nella forma

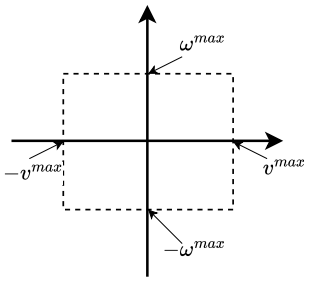



**Figura 3.2** Box constraints per input del robot uniciclo

$$\min\left(\frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} + f^T\boldsymbol{x}\right) \text{ con i vincoli } \begin{cases} A\boldsymbol{x} \leq b \\ A_{eq}\boldsymbol{x} = b_{eq} \\ lb \leq \boldsymbol{x} \leq ub \end{cases} \tag{3.6}$$

L'incognita $\boldsymbol{x}$ deve includere ogni vettore $\boldsymbol{u_i^{corr}}$, perciò assume la forma di un vettore di vettori:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{u}_i^{corr} & \boldsymbol{u}_{i+1}^{corr} & \cdots & \boldsymbol{u}_{i+C}^{corr} \end{pmatrix} = \begin{pmatrix} v_i^{corr} & \omega_i^{corr} & v_{i+1}^{corr} & \omega_{i+1}^{corr} & \cdots & v_{i+C}^{corr} & \omega_{i+C}^{corr} \end{pmatrix}^T \tag{3.7}$$

La cui dimensione risulta essere $1 \times 2C$. È vettore colonna, di cui ogni indice dispari è un ingresso $v_i^{corr}$, mentre ogni indice pari è un input $\omega_i^{corr}$. Le altre matrici assumono le opportune dimensioni in accordo con la costruzione del vettore $\boldsymbol{x}$: $H$ è la matrice identità di dimensione $2C$ (moltiplicata per 2 per tener conto del fattore di 1/2) e $\boldsymbol{f}^T$ è il vettore nullo di dimensione $2C \times 1$, essendo assenti termini lineari.

I vincoli vengono imposti usando gli argomenti di limiti inferiori e superiori come in (3.3) dove, $T_i^{-1}(\theta)$ e $\boldsymbol{u_i^{track}}$ sono ottenute per integrazioni successive del sistema secondo Runge-Kutta al secondo ordine, utilizzando le $\boldsymbol{u_i^{corr}}$ ottenute dall'orizzonte prodotto dalla precedente iterazione di MPC.

Da questo risulta un algoritmo per il calcolo del termine correttivo in un dato istante $t_i$:

1. Si calcola e memorizza $T_i^{-1}(\theta)$.
2. Si calcola e memorizza $\boldsymbol{u_i^{track}}$.
3. Si calcola l'input per il sistema $T_i^{-1}(\theta)\boldsymbol{u_i^{track}} + \boldsymbol{u_i^{corr}}$. $\boldsymbol{u_i^{corr}}$ viene dall'orizzonte di predizione della precedente esecuzione di MPC.
4. Si integrano secondo Eulero/Runge-Kutta le equazioni del sistema.
5. Si ottiene lo stato predetto all'istante $t_{i+1}$.
6. Si itera il processo da 1 a 5 fino alla fine dell'orizzonte (l'istante $t_{i+C}$).
7. Si costruisce il QP con le $T_i^{-1}(\theta)$e $\boldsymbol{u_i^{track}}$memorizzate, usando le $\boldsymbol{u_i^{corr}}$come variabili di decisioni.
8. Si risolve il QP, ottenendo una sequenza di $\boldsymbol{u_i^{corr}}$. La prima viene applicata all'istante corrente, le restanti vengono usati al passo 3 della successiva esecuzione di MPC.

## 3.1 Tracking e correzione

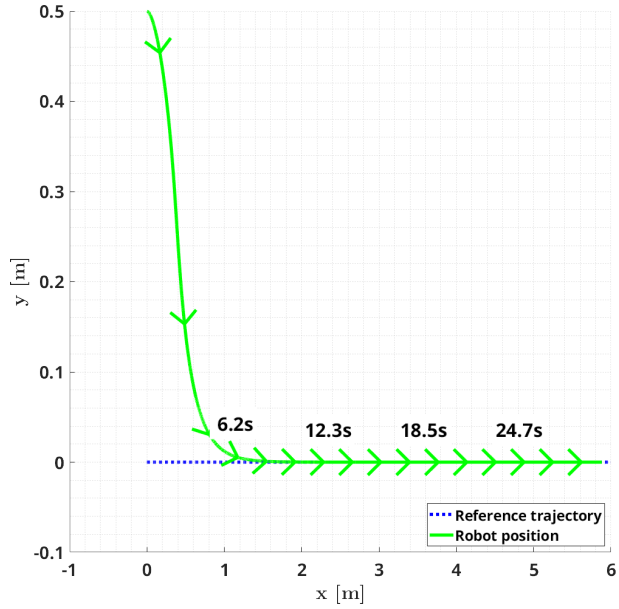
In questa sezione sono riportati dei risultati di trajectory tracking con robot uniciclo con applicata la correzione MPC-based. In tabella sono riportati i parametri della simulazione. I controllori sono implementati a tempo discreto, tuttavia il robot è un sistema a tempo continuo e la sua evoluzione viene simulata usando il risolutore ode45, simulando un organo di ritenuta di ordine zero fra il controllore e il sistema. Il controllore MPC è stato simulato in due varianti: con $C = 1$ (1-step) e $C = 15$ (multistep).

| Grandezza | Parametro | Valore | Unità di misura |
|---|---|---|---|
| Limite di saturazione su $v$ | $v^{max}$ | 0.5 | m/s |
| Limite di saturazione su $\omega$ | $\omega^{max}$ | 0.5 | rad/s |
| Matrice dei guadagni proporzionali | K | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ | - |
| Distanza fra il punto tracciato e il centro del robot | b | 0.12 | m |
| Tempo di campionamento del controllore | $T_c$ | 0.1 | s |
| Timestep per MPC | $dt$ | 0.1 | s |
| Lunghezza orizzonte di predizione per MPC multistep | $C$ | 15 | $dt$ |
| Durata delle simulazioni | $T_{fin}$ | 30 | s |

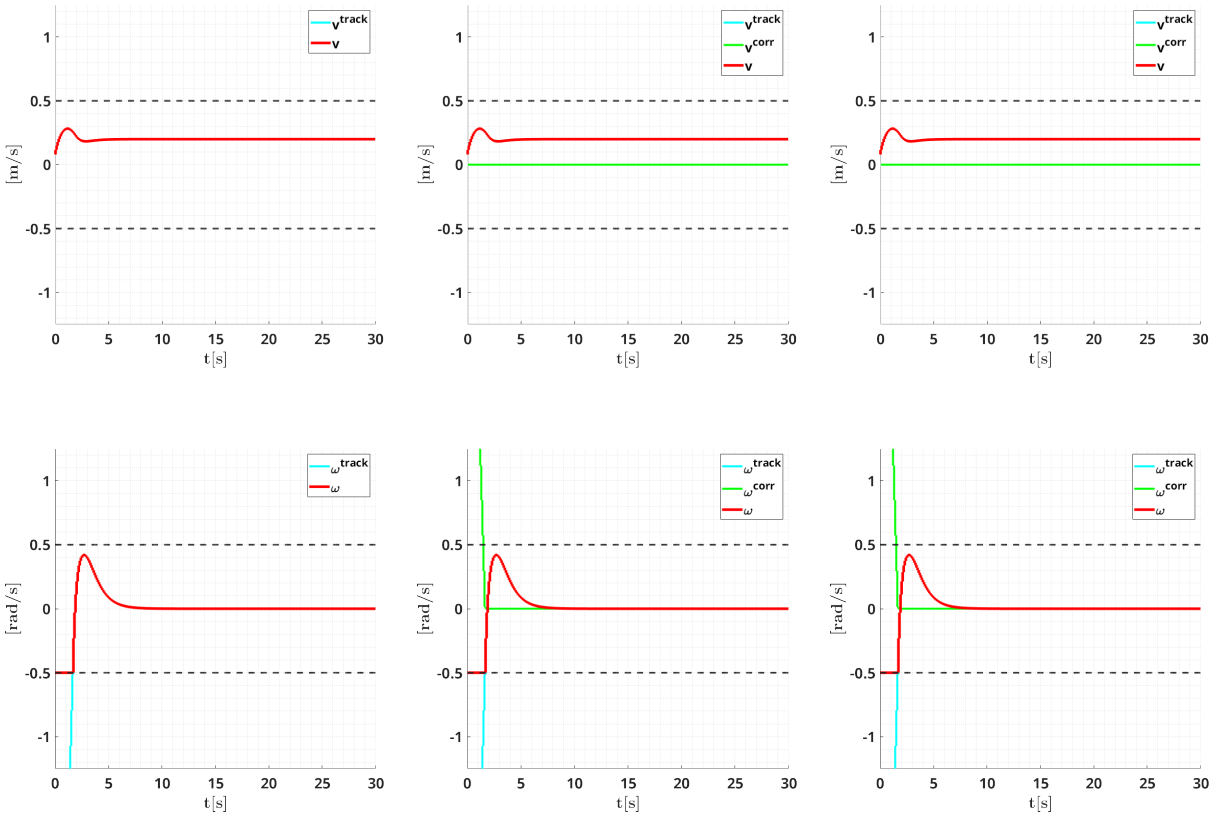
### 3.1.1 Retta con errore iniziale

Tracking di una retta di equazione $\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$ con $q_0 = \begin{pmatrix} 0 & 0.5 & 0 \end{pmatrix}^T$

L'errore iniziale fa saturare $\omega^{track}$. Quando $\omega^{track}$ è in saturazione viene correttamente generata una correzione che fa rientrare $\omega$ nei limiti di saturazione. I profili degli ingressi generati dai tre controllori sono uguali, quindi la traiettoria seguita dal robot è la stessa in tutti e tre i casi.




(a) Traiettoria del robot



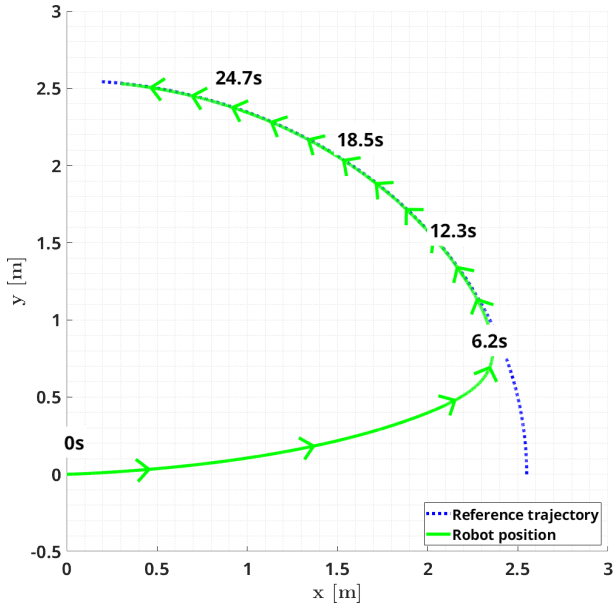

(b) Input solo tracking (c) Input tracking + MPC 1-step (d) Input tracking + MPC multistep

**Figura 3.3** Simulazione 1: tracking di una retta con errore iniziale

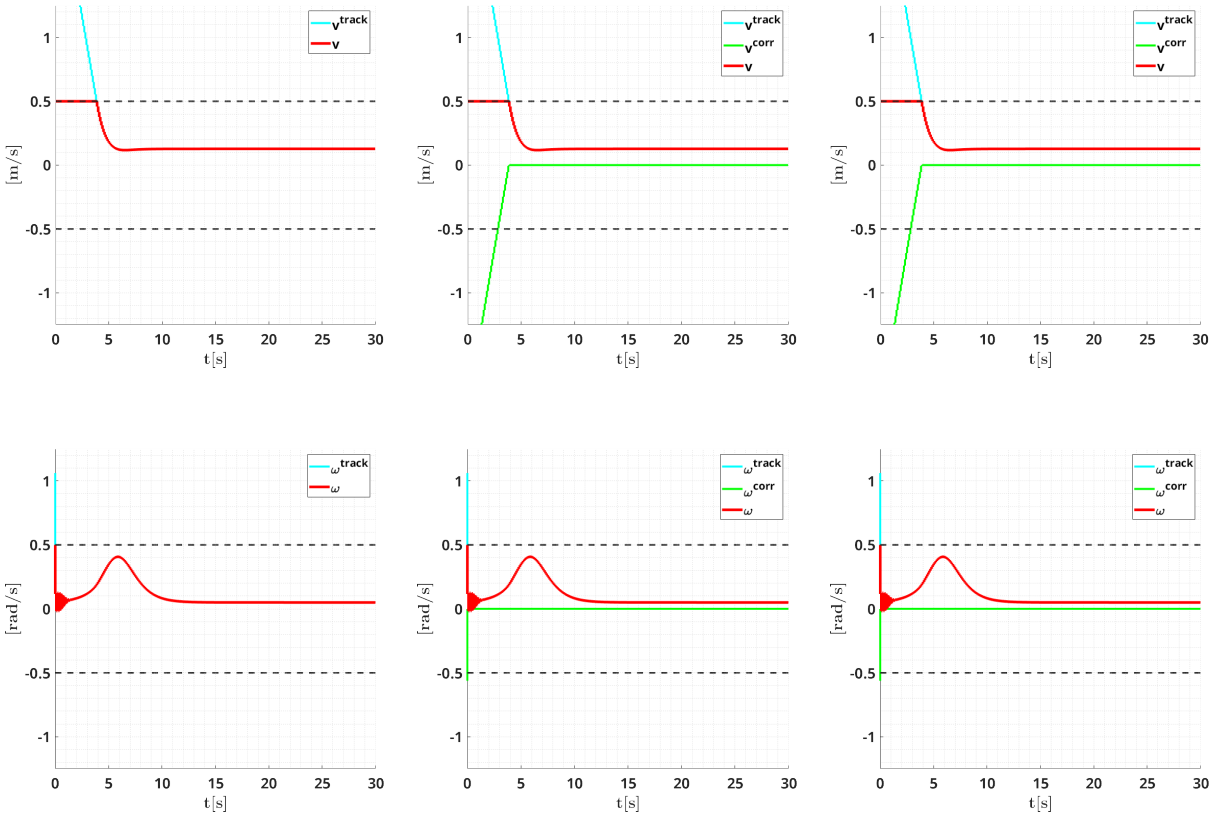
### 3.1.2 Cerchio

Tracking di un archo di cerchio di equazione $\begin{cases} x_{ref} = 2.55\cos(0.05t) \\ y_{ref} = 2.55\cos(0.05t) \end{cases}$ con $q_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$

Il comportamento è simile a quello della retta con errore iniziale, ma in questo caso sia $v^{track}$ che $\omega^{track}$ sono in saturazione a causa dell'errore iniziale. Viene generata una correzione che fa rientrare $v$ e $\omega$ totali nei limiti di saturazione. Anche in questo caso i profili degli ingressi generati dai tre controllori sono uguali, quindi la traiettoria seguita dal robot è la stessa in tutti e tre i casi. L'effetto di chattering all'inizio è invece dovuto al tempo di campionamento troppo alto. Abbassarlo a 0.05s elimina il fenomeno senza impattare notevole sui tempi di simulazione.




(a) Traiettoria del robot




(b) Input solo tracking (c) Input tracking + MPC 1-step (d) Input tracking + MPC multistep

**Figura 3.4** Simulazione 2: tracking di un arco di cerchio con errore iniziale

### 3.1.3 Quadrato

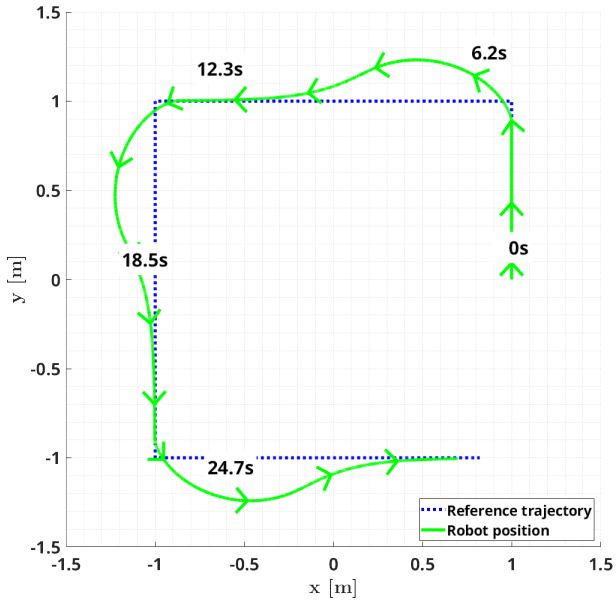
Tracking di un quadrato di lato 2m con $q_0 = \begin{pmatrix} 1 & 0 & -\pi/2 \end{pmatrix}$

Il comportamento della correzione è analogo a quello del cerchio e della retta con errore iniziale. In questo caso però $v^{track}$ non satura mai e $\omega^{track}$ satura in corrispondenza delle curve ad angolo retto. Viene correttamente generata una correzione che mantiene $\omega$ nei limiti di saturazione. Anche in questo caso i profili degli ingressi generati dai tre controllori sono uguali e la traiettoria seguita dal robot è la stessa in tutti e tre i casi.

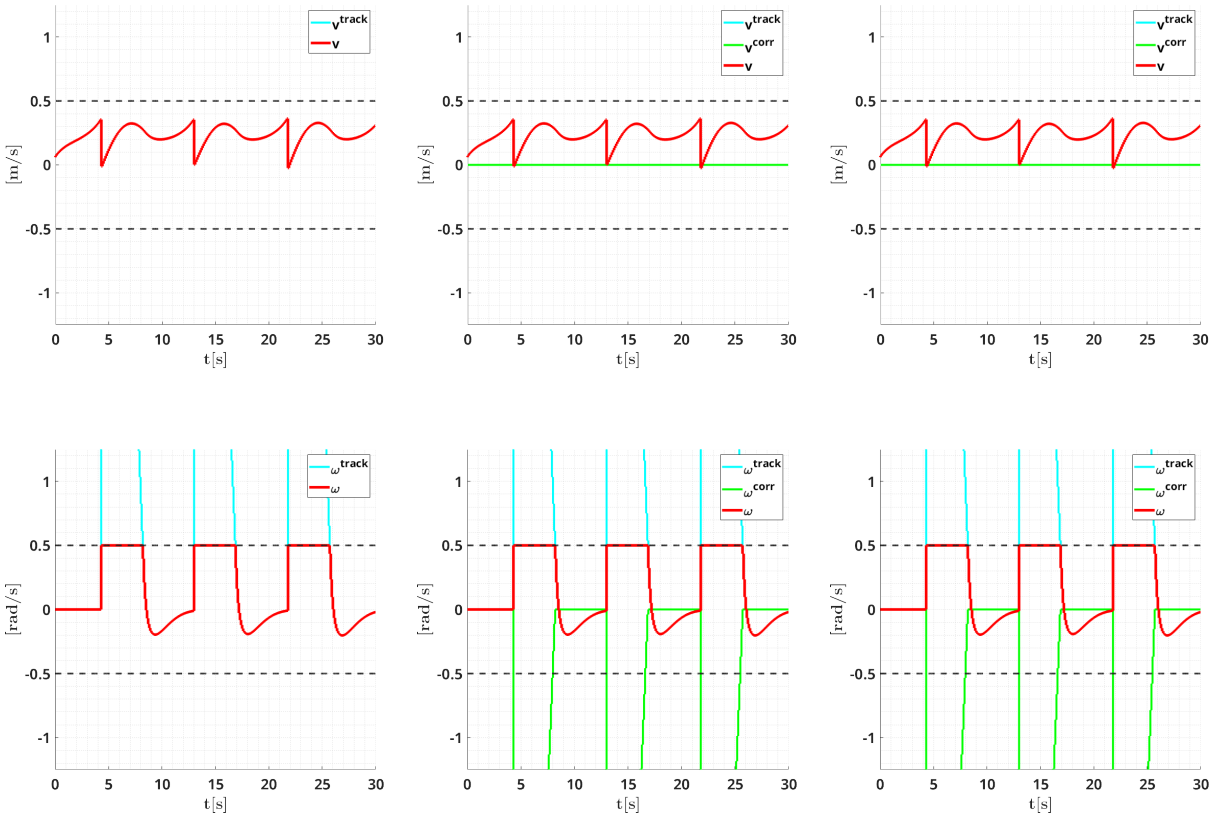
### 3.1.4 Differenze fra i tre controllori

In tutti i casi si osservano differenze praticamente ignorabili fra gli ingressi prodotti dal controllore con solo tracking (senza correzione) e quelli prodotti dai due controllori con correzione MPC-based. Nei casi in cui la traiettoria non necessita di correzioni, le differenze sono in ordini di grandezza minori di $10^{-10}$m e sono imputabili ad errori di precisione nella rappresentazione dei numeri a virgola mobile. Nei casi in cui la traiettoria necessiti di correzioni, si osserva che la correzione ottima trovata sia da MPC 1-step che MPC multistep è quella che fa comportare la correzione come un organo di saturazione. Questo è conseguenza dalla formulazione della funzione di costo e del fatto che la correzione desidera è la minima possibile. La correzione minima è, infatti, quella che si comporta come un saturatore. Dunque gli ingressi corretti a monte della saturazione sono uguali a quelli non corretti a valle della saturazione.

Anche le differenze fra le correzioni applicate da MPC 1-step e MPC multistep sono pressoché ignorabili. Questo deriva non solo dalla formulazione della funzione di costo ma anche dalla formulazione dei vincoli. Come conseguenza del predirre l'evoluzione del sistema offline, prima della risoluzione del QP, ogni variabile di decisione è indipendente dalle altre. Ne risulta che, ad un dato istante $t_k$, MPC 1-step e MPC multistep producono la stessa correzione. Le differenze, che sono nell'ordine di $10^{-8}$m/s (per $v$) e $10^{-8}$rad/s (per $\omega$) , sono imputabili all'accumulo di errori di integrazione in MPC multistep nella fase di predizione. Questi fanno predirre alle esecuzioni successive dell'algoritmo una traiettoria leggermente sbagliata e l'errore viene recuperato ad ogni istante dalla nuova correzione.




(a) Traiettoria del robot




(b) Input solo tracking




(c) Input tracking + MPC 1-step

(d) Input tracking + MPC multistep

**Figura 3.5** Simulazione 3: tracking di una quadrato di lato 2m

# Capitolo 4

# Estensione a differential drive

Il modello del robot uniciclo è stato descritto nel capitolo 1 come puramente teorico. Tuttavia, modelli più complessi di robot praticamente realizzabili sono riducibili ad un uniciclo "equivalente" tramite un cambio di input. In questo capitolo viene presentato il caso del robot differential drive. Esso presenta due ruote poste sullo stesso asse, fisse ed attuate indipendentemente, ed un terzo ruotino libero (*caster wheel*) per assicurare equilibrio senza bisogno di un controllore apposito. Il ruotino libero non è attuato ma per costruzione si orienta automaticamente nella direzione di movimento del robot. Componendo opportunamente le velocità di rotazione del motore destro e quello sinistro, rispettivamente $\omega_r$ e $\omega_l$, si impone la traiettoria al robot.

La posizione dell'uniciclo equivalente è il punto medio fra la posizione delle due ruote

$$\begin{cases} x = \dfrac{x_r + x_l}{2} \\ y = \dfrac{y_r + y_l}{2} \end{cases}$$

Ogni ruota contribuisce alla velocità lungo ogni asse cartesiano, dipendentemente dall'orientamento

$$\begin{cases} \dot{x} = \dfrac{\dot{x_r} + \dot{x_l}}{2} = \dfrac{v_r + v_l}{2} \cos\theta \\ \dot{y} = \dfrac{\dot{y_r} + \dot{y_l}}{2} = \dfrac{v_r + v_l}{2} \sin\theta \end{cases}$$

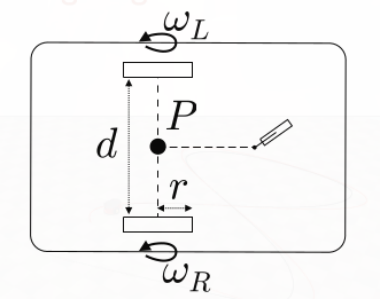
Le velocità lineari imposte da ogni motore sono direttamente proporzionali alle velocità angolari dell'asse del motore stesso attraverso il raggio della ruota

$$v_r = r \cdot \omega_r, v_l = r \cdot \omega_l \implies \begin{cases} \dot{x} = \dfrac{r(\omega_r + \omega_l)}{2} \cos\theta \\ \dot{y} = \dfrac{r(\omega_r + \omega_l)}{2} \sin\theta \end{cases}$$

Ricordando che nello schema dell'uniciclo vale $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ risulta

$$v = \frac{r(w_r + w_l)}{2} \tag{4.1}$$

La rotazione avviene attorno all'asse perpendicolare al piano e passante il punto medio fra le ruote. La velocità angolare di rotazione dell'uniciclo equivalente è la differenza delle velocità angolari imposte da ogni motore indipendentemente. Dato che i motori impongono rotazioni in versi opposti attorno allo stesso asse le due vanno sottratte.




(a) Lo schema di un robot differential drive nel piano

(b) Il Pioneer P3DX, un robot differential drive usato a livello accademico

**Figura 4.1**

$$\omega = \frac{v_r}{d} - \frac{v_l}{d} = \frac{r(\omega_r - \omega_l)}{d} \tag{4.2}$$

La matrice

$$W = \begin{pmatrix} r/2 & r/2 \\ r/d & -r/d \end{pmatrix} \tag{4.3}$$

esprime il cambio di input dagli input del differential drive agli input dell'uniciclo.

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} \tag{4.4}$$

Da (1.3) e (4.4) si ottiene il modello cinematico del robot differential drive

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = T(\theta) \begin{pmatrix} v \\ \omega \end{pmatrix} = T(\theta)\ W \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} \tag{4.5}$$

Per applicare lo schema di controllo in figura 2.2 la matrice $T(\theta)$ deve essere presa non singolare come in (2.2) e quindi fare il tracking di un punto spostato rispetto al centro del robot. Gli input per il differential drive risultano essere

$$\begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} = W^{-1}T^{-1}(\theta) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{4.6}$$

Per quanto riguarda il controllore MPC, è sufficiente riscrivere i vincoli in (3.3) per utilizzare l'inversa della matrice di disaccoppiamento appropriata per il robot differential drive $W^{-1}T^{-1}(\theta)$.

I *box constraints* in questo caso vengono applicati agli input del robot differential drive $\begin{pmatrix} \omega_r & \omega_l \end{pmatrix}^T$ e non a quelli dell'uniciclo $\begin{pmatrix} v & \omega \end{pmatrix}^T$. La regione ammissibile per gli input del robot differential drive rimane un quadrato, mentre a causa del cambio di input la regione ammissibile per gli input dell'uniciclo equivalente diventa un rombo.

In questa sezione sono riportati dei risultati di trajectory tracking con robot differential drive con applicata la correzione MPC-based. In tabella sono riportati i parametri della simulazione. La metodologia di simulazione e l'implementazione dei controllori è la stessa che in sezione 3.1

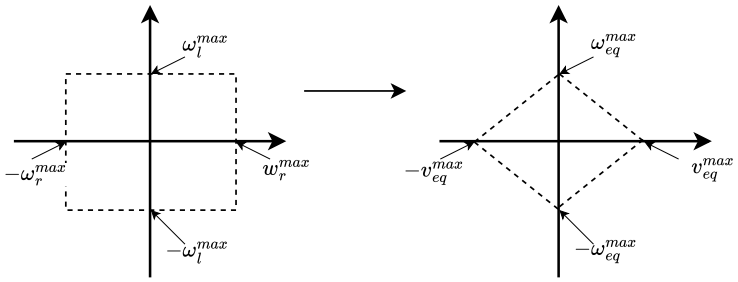


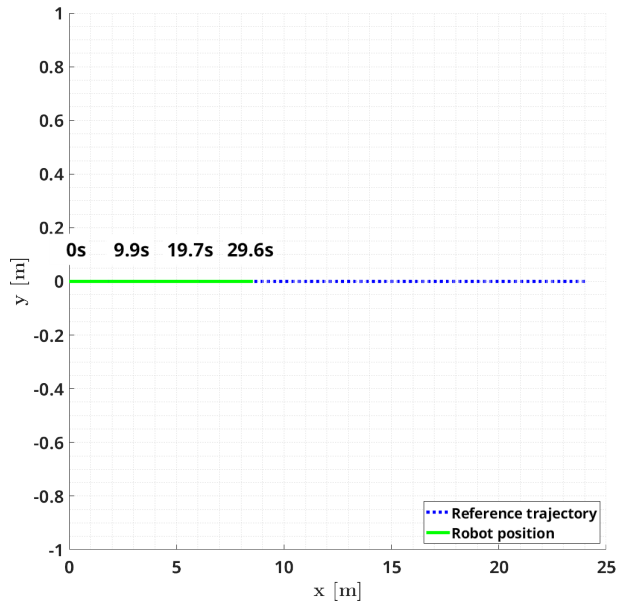

**Figura 4.2** Box constraints per robot differential drive

| Grandezza | Parametro | Valore | Unità di misura |
|---|---|---|---|
| Limite di saturazione su $\omega_r$ | $\omega_l^{max}$ | 2.85 | rad/s |
| Limite di saturazione su $\omega_l$ | $\omega_l^{max}$ | 2.85 | rad/s |
| Matrice dei guadagni proporzionali | K | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ | - |
| Distanza fra il punto tracciato e il centro del robot | b | 0.12 | m/s |
| Raggio delle ruote | r | 0.1 | m/s |
| Distanza fra le ruote | d | 0.15 | m/s |
| Tempo di campionamento del controllore | $T_c$ | 0.1 | s |
| Timestep per MPC | $dt$ | 0.1 | s |
| Lunghezza orizzonte di predizione per MPC multistep | $C$ | 15 | $dt$ |
| Durata delle simulazioni | $T_{fin}$ | 30 | s |

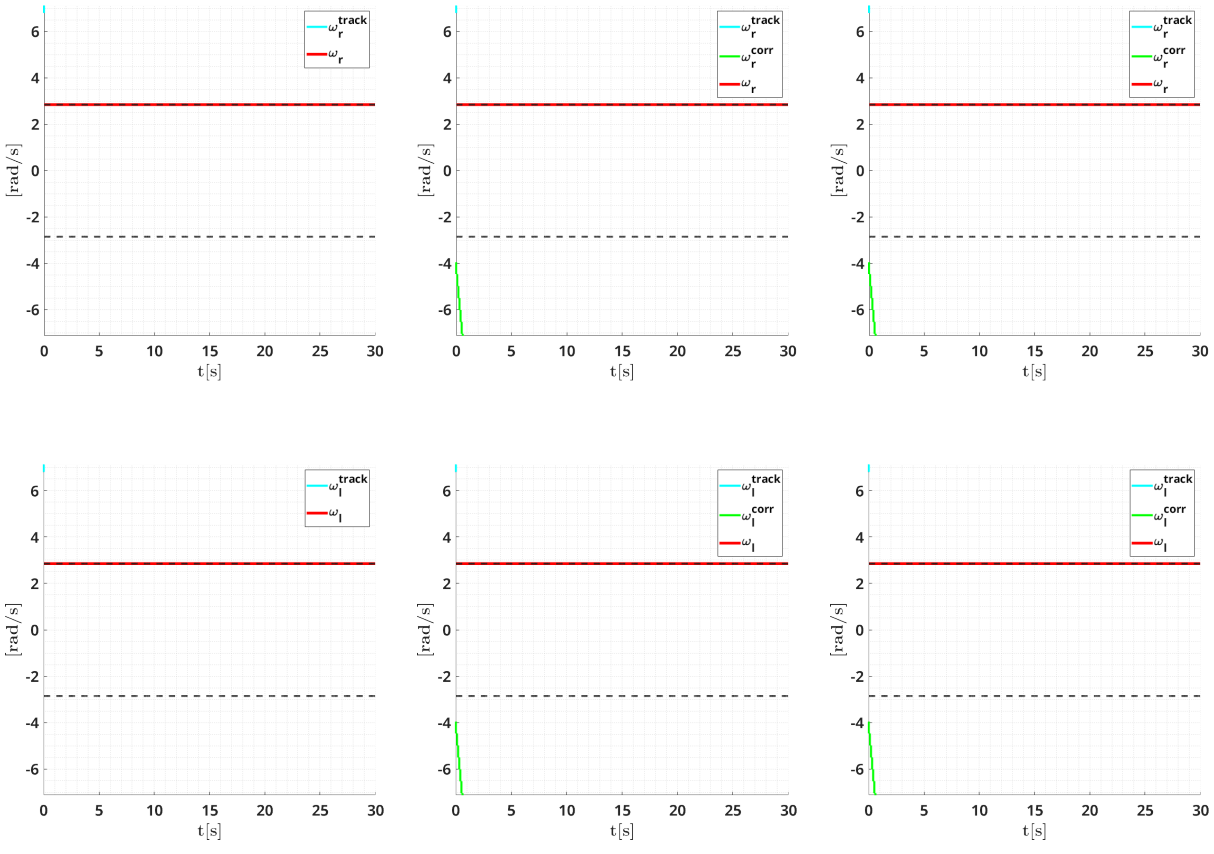
### 4.0.1 Retta troppo veloce

Tracking di una retta di equazione $\begin{cases} x_{ref} = 0.8t \\ y_{ref} = 0 \end{cases}$ con condizioni iniziali $q_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$

La traiettoria è troppo veloce da inseguire ed entrambi gli attuatori saturano instantaneamente e rimangono in saturazione. Questo fa aumentare costantemente l'errore e, di conseguenza, l'input di tracking. Viene generata una correzione che non fa saturare gli attuatori, ma anche questa aumenta costantemente come conseguenza del comportamento dell'input di tracking. I profili degli ingressi generati dai tre controllori sono uguali, quindi la traiettoria seguita dal robot è la stessa in tutti e tre i casi.




(a) Traiettoria del robot



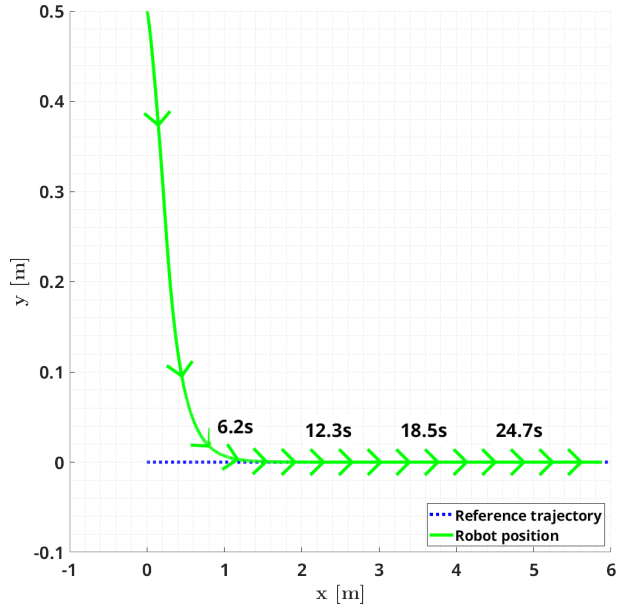

(b) Input solo tracking (c) Input tracking + MPC 1-step (d) Input tracking + MPC multistep

**Figura 4.3** Simulazione 4: tracking di una retta troppo veloce con differential drive

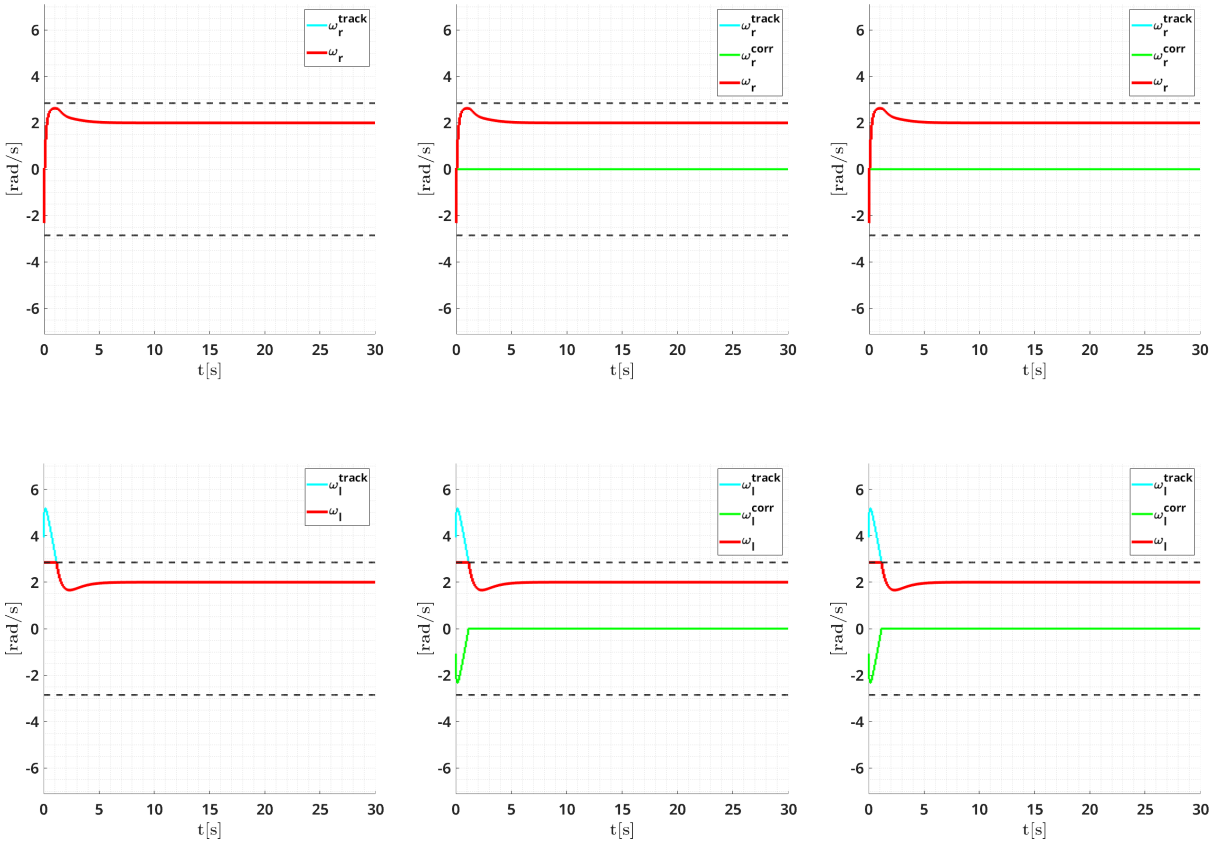
### 4.0.2 Retta con errore iniziale

Tracking di una retta di equazione $\begin{cases} x_{ref} = 0.2t \\ y_{ref} = 0 \end{cases}$ con condizioni iniziali $q_0 = \begin{pmatrix} 0 & 0.5 & 0 \end{pmatrix}^T$

L'errore iniziale fa saturare entrambi gli attuatori ma poi viene recuperato. Con i controllori MPC-based viene correttamente generata una correzione che non fa saturare gli attuatori. Anche in questo caso i profili degli ingressi generati dai tre controllori sono uguali e la traiettoria seguita dal robot è la stessa in tutti e tre i casi.




(a) Traiettoria del robot




(b) Input solo tracking

(c) Input tracking + MPC 1-step

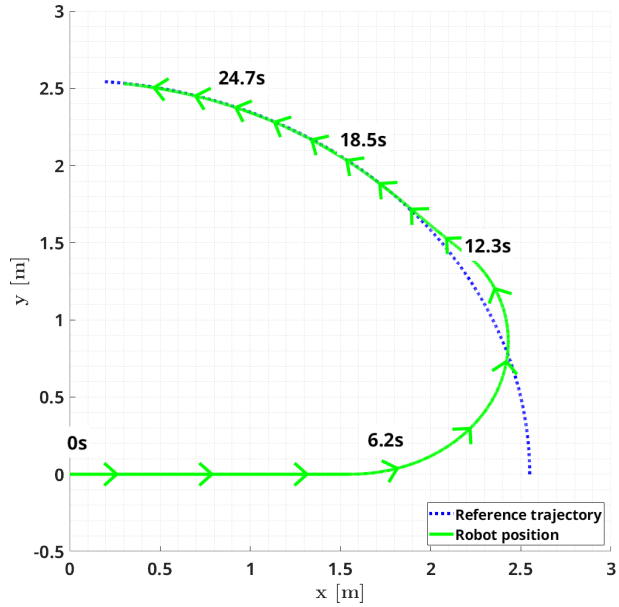
(d) Input tracking + MPC multistep

**Figura 4.4** Simulazione 5: tracking di una retta con errore iniziale con differential drive

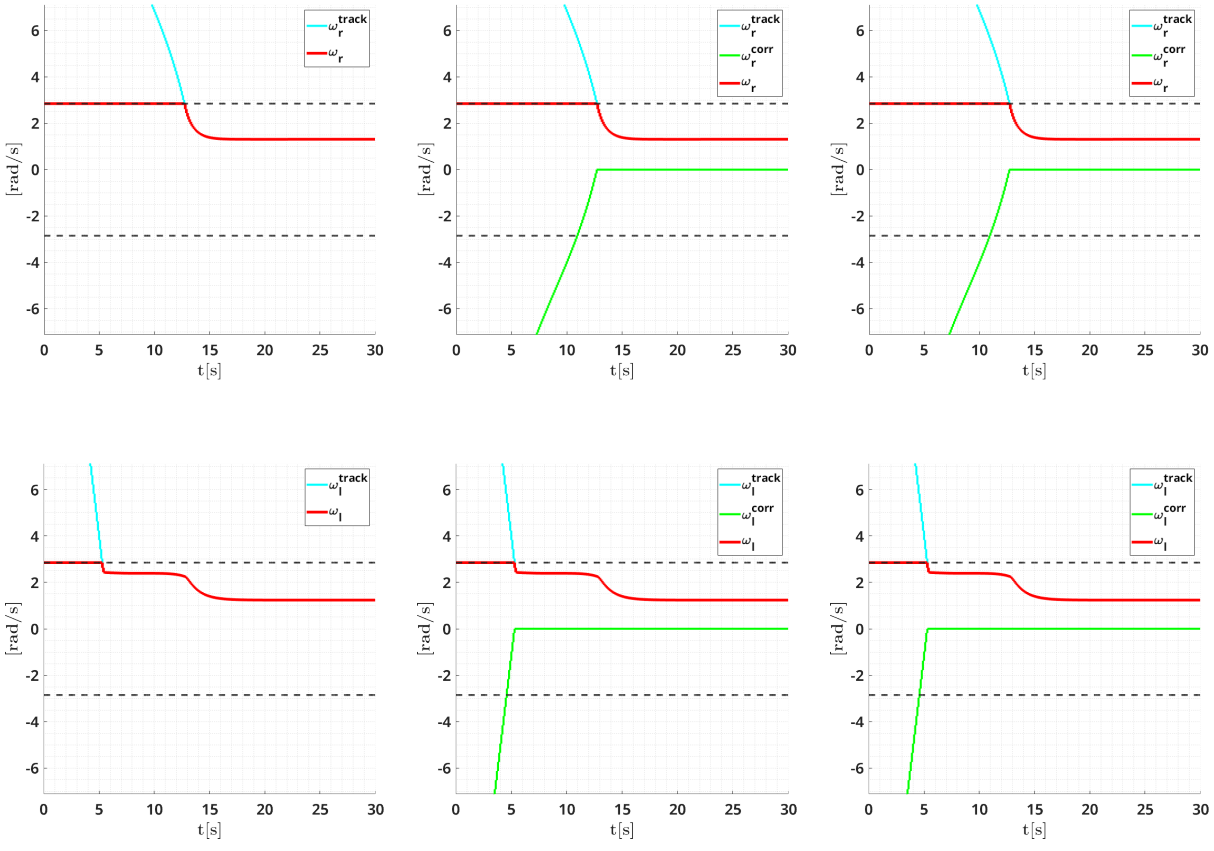
### 4.0.3 Cerchio

Tracking di un cerchio di equazione $\begin{cases} x_{ref} = 2.55\cos(0.05t) \\ y_{ref} = 2.55\sin(0.05t) \end{cases}$ con condizioni iniziali $q_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$

Analogamente alla retta con errore iniziale, l'errore fa saturare entrambi gli attuatori ma poi viene recuperato. Con i controllori MPC-based viene correttamente generata una correzione che non fa saturare gli attuatori. Anche in questo caso i profili degli ingressi generati dai tre controllori sono uguali, quindi la traiettoria seguita dal robot è la stessa in tutti e tre i casi.




(a) Traiettoria del robot



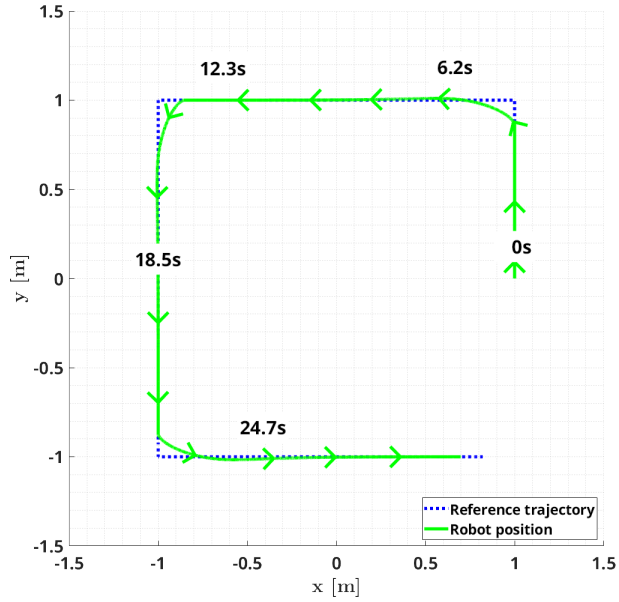

(b) Input solo tracking (c) Input tracking + MPC 1-step (d) Input tracking + MPC multistep

**Figura 4.5** Simulazione 6: tracking di un arco di cerchio con errore iniziale con differential drive

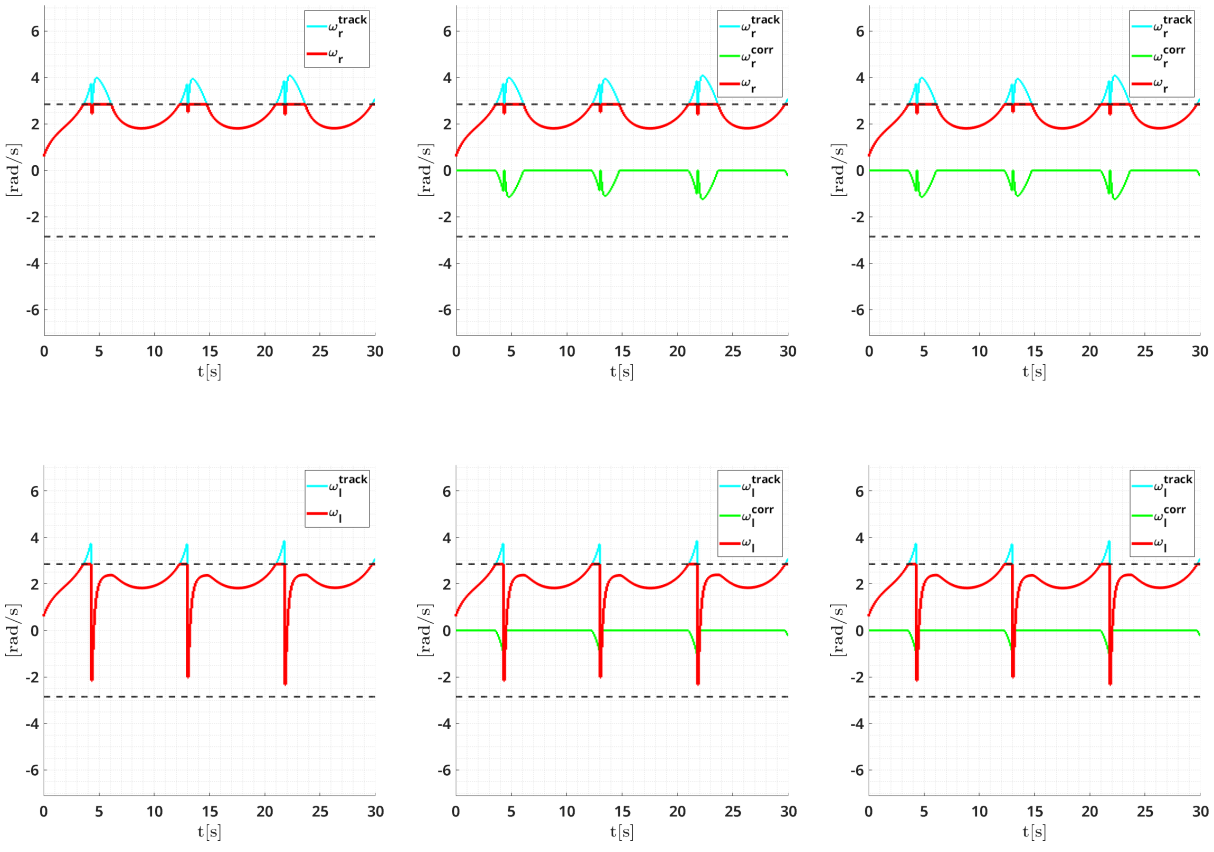
### 4.0.4 Quadrato

Tracking di un quadrato di lato 2m e centro l'origine, con velocità $0.18m/s$ e condizione iniziale $q_0 = \begin{pmatrix} 1 & 0 & -\pi/2 \end{pmatrix}$

Analogamente a quanto succede per l'uniciclo (sezione 3.1.3), le curve ad angolo retto sono troppo strette per il robot e fanno saturare gli attuatori. I due controllori MPC generano correttamente una correzione che non fa saturare gli attuatori. Anche in questo caso i profili degli ingressi generati dai tre controllori sono uguali, quindi la traiettoria seguita dal robot è la stessa in tutti e tre i casi.




(a) Traiettoria del robot




(b) Input solo tracking (c) Input tracking + MPC 1-step (d) Input tracking + MPC multistep

**Figura 4.6** Simulazione 7: tracking di un quadrato di lato 2m con differential drive

### 4.0.5 Differenze fra i tre controllori e differenze rispetto all'uniciclo

Il comportamento della correzione è lo stesso che per il robot uniciclo, come spiegato in sezione 3.1.4: i tre controllori generano profili di input uguali. Per come è stato definito il QP, la correzione minima (e quindi ottima) è quella che fa comportare la correzione come un saturatore. Ne risulta, di nuovo, che gli input corretti a monte della saturazione siano uguali a quelli non corretti a valle della stessa. Le differenze sono imputabili a errori di precisione nella rappresentazione floating point e all'accumulo di errori di integrazione.

Le differenze che si osservano nelle traiettoria tracciate dal robot uniciclo e dal robot differential drive sono conseguenza della scelta dei parametri. In particolare, il robot differential drive presenta una velocità lineare leggermente minore e una velocità angolare leggermente maggiore rispetto all'uniciclo.

# Capitolo 5

# Funzioni di costo alternative

Nei precedenti capitoli si è visto come la struttura del problema porta la correzione MPC-based a comportarsi come la saturazione. In questo capitolo si esplora il comportamento di funzioni di costo diverse da quella descritta in (3.2) e ne si studia l'effetto sul tracking della traiettoria. Si può pensare, ad esempio, a minimizzare solo una fra $(v^{corr})^2$e $(\omega^{corr})^2$.

Per la struttura definita in figura 3.1, nel caso del robot uniciclo è impossibile minimizzare solo una fra $(v^{corr})^2$e $(\omega^{corr})^2$, poichè una delle due variabili di decisione scomparirebbe dalla funzione di costo. Nel caso del robot differential drive invece è possibile definire $v$ e $\omega$ (e quindi le relative correzioni) in termini di $\omega_r$ e $\omega_l$ attraverso il cambio di input mostrato in (4.4). Si vede facilmente che minimizzare $(v^{corr})^2$porta l'Hessiano (matrice H in (3.6)) a essere singolare. Lo stesso accade per $(\omega^{corr})^2$. Questo rende necessario utilizzare un algoritmo diverso rispetto a `interior-point-convex`, l'algoritmo utilizzato di predefinito da quadprog. L'algoritmo `active-set` riesce a risolvere il problema correttamente.

### 5.0.1 Cerchio, differential drive, minimizzando correzione su $v$

Da (4.4) si ottiene che

$$v^2 = \frac{r^2}{2}(\omega_r^2 + \omega_l^2 + 2\omega_r\omega_l)$$

Basta quindi impostare

$$H = 2\frac{r^2}{4}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} = \frac{r^2}{2}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} \tag{5.1}$$

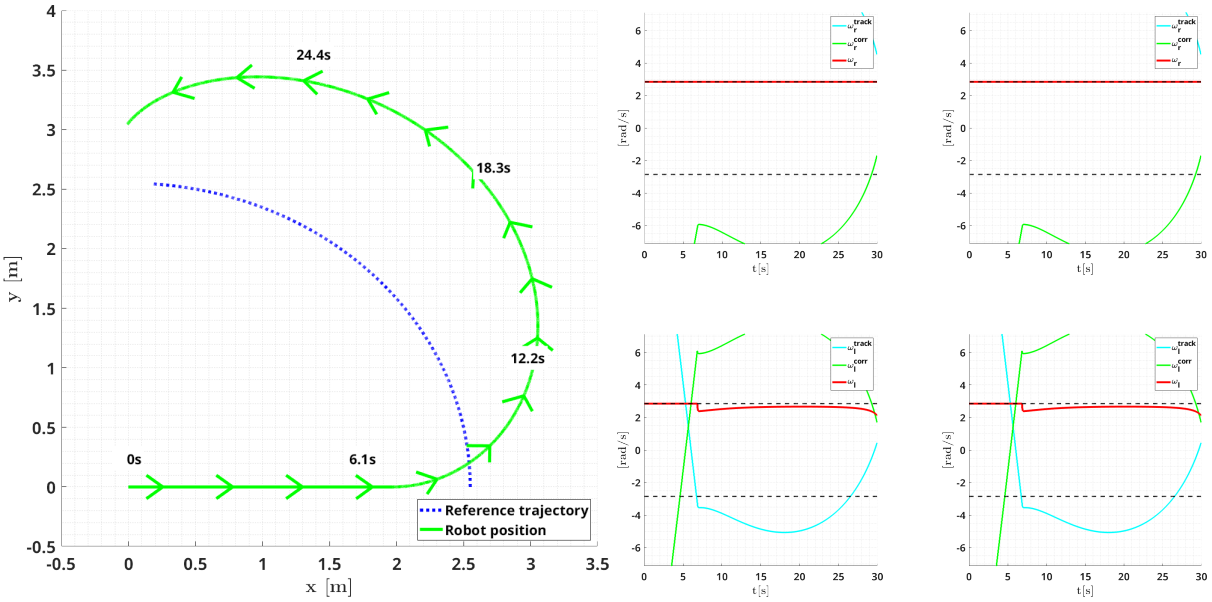
per minimizzare solo la velocità lineare del robot. Per MPC multistep $H$ è una matrice diagonale a blocchi di grandezza $C$, con ogni blocco come (5.1). Il 2 serve per tener conto del fattore $1/2$ nella formulazione di quadprog, ma poi viene semplificato con il fattore $1/4$ che viene dall'elevazione al quadrato di $v$. Per questa simulazione il tempo di campionamento è stato abbassato a 0.05s per evitare fenomeni di chattering.

Si osserva che minimizzando solo $v$ il tracking della traiettoria peggiora di molto.

### 5.0.2 Cerchio, differential drive, minimizzando correzione su $\omega$

Da (4.4) si ottiene che

$$\omega^2 = \frac{r^2}{d^2}(\omega_r^2 + \omega_l^2 - 2\omega_r\omega_l)$$




(a) Traiettoria del robot con correzione

(b) Input tracking + MPC 1-step

(c) Input tracking + MPC multistep

**Figura 5.1** Simulazione 8: tracking di un arco di cerchio con errore iniziale con differential drive, minimizzando la correzione sulla sola velocità lineare.

Basta quindi impostare

$$H = 2\frac{r^2}{d^2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{5.2}$$

per minimizzare solo la velocità angolare del robot. Per MPC multistep $H$ è una matrice diagonale a blocchi di grandezza $C$, con ogni blocco come (5.2). Il 2 serve per tener conto del fattore 1/2 nella formulazione di quadprog.

Per questa simulazione il tempo di campionamento è stato abbassato a 0.05s per evitare fenomeni di chattering.

Si osserva che minimizzando solo $\omega$ l'errore iniziale viene recuperato prima, evitando la sovraelongazione che si osservava in sezione 4.0.3.

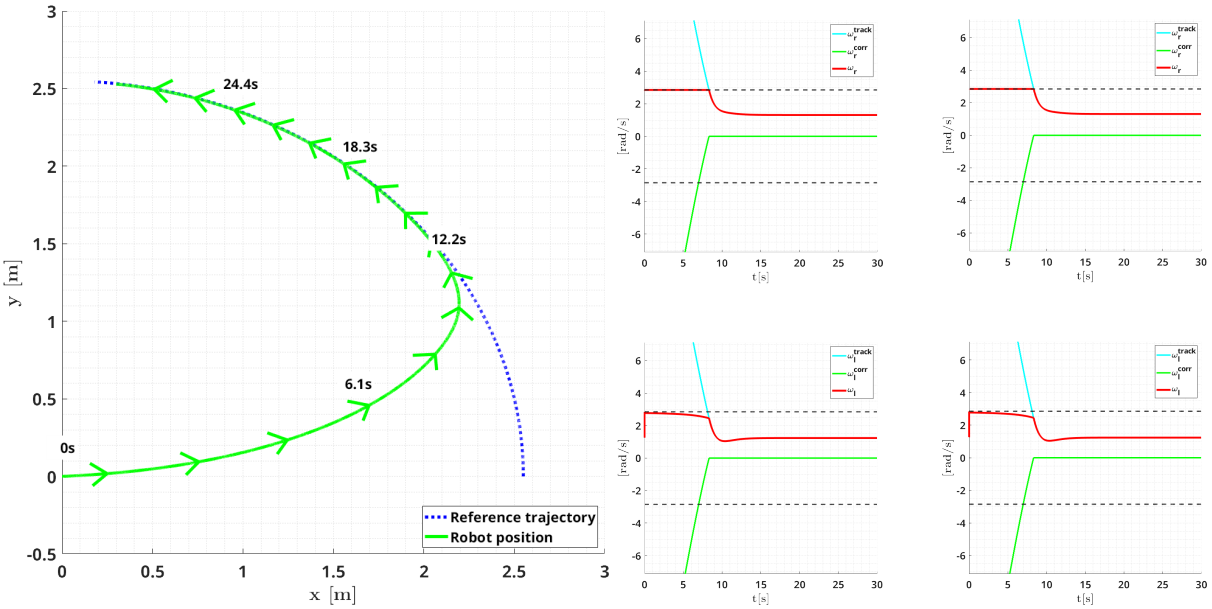
## 5.1 Correzione sugli input di tracking

La struttura del controllore definita in figura 3.1 applica la correzione direttamente agli ingressi del robot. È possibile modificare la struttura come in figura 5.3 per applicare la correzione agli input di tracking prima della matrice di disaccoppiamento.

Con questa struttura si possono ottenere gli stessi risultati dei capitoli 3 e 4 impostando H in maniera tale da minimizzare la correzione vista sugli input del robot

$$H = \begin{pmatrix} T_i^{-1}(\theta) & 0 & \cdots & 0 \\ 0 & T_{i+1}^{-1}(\theta) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & T_{i+C}^{-1}(\theta) \end{pmatrix} \tag{5.3}$$

e cambiando i vincoli nella forma $A\boldsymbol{x} \leq b$ per esprimere i vincoli di saturazione.




(a) Traiettoria del robot con correzione

(b) Input tracking + MPC 1-step

(c) Input tracking + MPC multistep

**Figura 5.2** Simulazione 9: tracking di un arco di cerchio con errore iniziale con differential drive, minimizzando la correzione sulla sola velocità angolare.

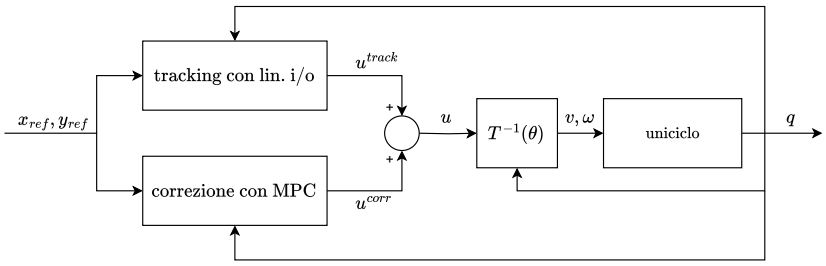
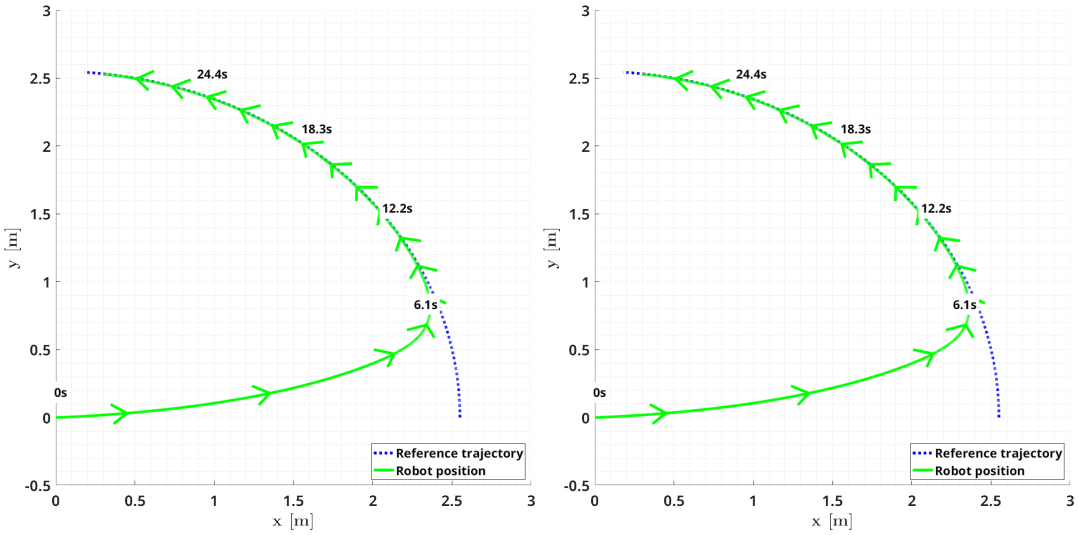



**Figura 5.3** Schema di controllo con azione correttiva su input di tracking

$$A = \begin{pmatrix} T_i^{-1}(\theta) & 0 & \cdots & 0 \\ -T_i^{-1}(\theta) & 0 & \cdots & 0 \\ 0 & T_{i+1}^{-1}(\theta) & \cdots & 0 \\ 0 & -T_{i+1}^{-1}(\theta) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & T_{i+C}^{-1}(\theta) \\ 0 & 0 & \cdots & -T_{i+C}^{-1}(\theta) \end{pmatrix} \quad b = \begin{pmatrix} \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} - T_i^{-1}(\theta)\boldsymbol{u_i^{track}} \\ \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} + T_i^{-1}(\theta)\boldsymbol{u_i^{track}} \\ \vdots \\ \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} - T_{i+C}^{-1}(\theta)\boldsymbol{u_i^{track}} \\ \begin{pmatrix} v_{max} \\ \omega_{max} \end{pmatrix} + T_{i+C}^{-1}(\theta)\boldsymbol{u_i^{track}} \end{pmatrix} \tag{5.4}$$

Ma soprattutto questa formulazione può essere usata per minimizzare in maniera isolata $v$ e $\omega$ nel caso dell'uniciclo, continuando comunque a far comparire tutte le variabili di decisione (entrambe le componenti di ogni $\boldsymbol{u_i^{corr}}$) nella funzione di costo. Notando che




(a) Traiettoria con correzione solo su $(v^{corr})^2$ (b) Traiettoria con correzione solo su $(\omega^{corr})^2$

**Figura 5.4** Simulazioni 9, 10: Inseguimento di un arco di cerchio con errore iniziale con uniciclo, minimizzando la correzione sulla velocità lineare (a) e angolare (b)

$$\begin{aligned} v &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} T^{-1}(\theta) \boldsymbol{u^{corr}} \\ \omega &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} T^{-1}(\theta) \boldsymbol{u^{corr}} \end{aligned} \tag{5.5}$$

Per minimizzare $(v^{corr})^2$e $(\omega^{corr})^2$si può impostare H (o un blocco di H) rispettivamente a

$$H = T^{-1}(\theta)^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} T^{-1}(\theta) \tag{5.6}$$

per $v$ ed a

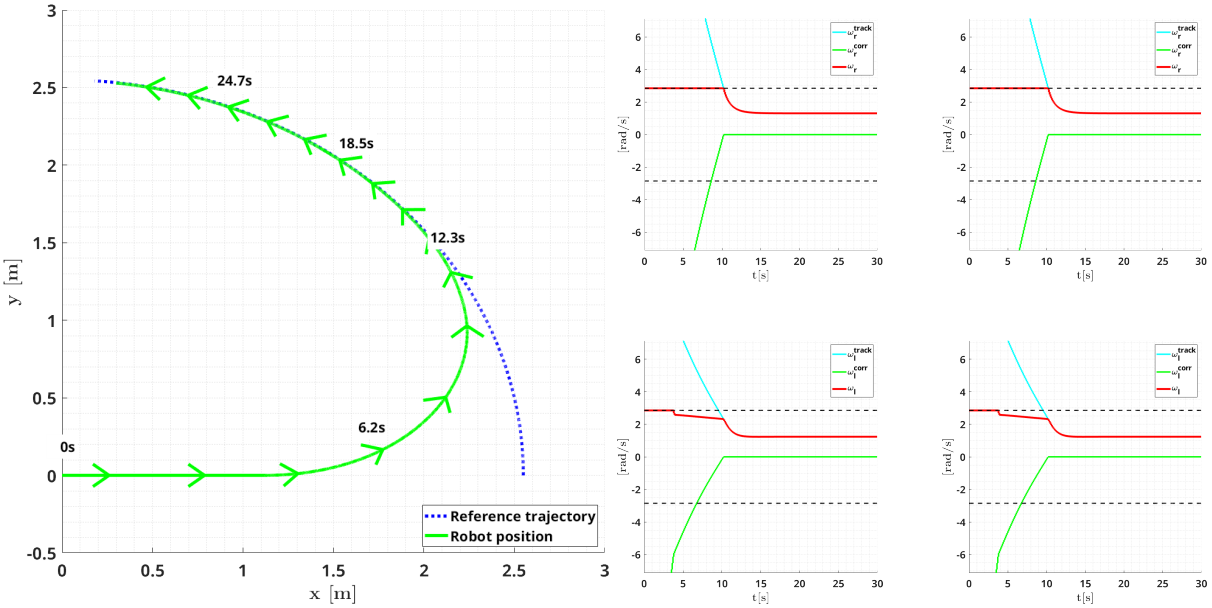
$$H = T^{-1}(\theta)^T \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} T^{-1}(\theta) \tag{5.7}$$

per $\omega$. Sviluppi simili possono essere fatti per il robot differential drive, arrivando agli stessi risultati delle sezioni 5.0.1 e 5.0.2.

Nel caso dell'uniciclo tuttavia non si apprezzano differenze significative nel minimizzare una o l'altra componente della correzione, come riportato in figura 5.4.

In ultimo, questa struttura permette anche un ulteriore tipo di correzione. Infatti, se si eseguisse il tracking del centro esatto del robot, applicare una correzione significherebbe modificare direttamente $\dot{x}$ e $\dot{y}$. Ponendo poi $H$ pari alla matrice identità si starebbe minimizzando proprio $(\dot{x}^{corr})^2 + (\dot{y}_i^{corr})^2 = (v_i^{corr})^2$. Tuttavia, dato che si sta eseguendo il tracking di un punto spostato rispetto al centro del robot (come espresso in (2.3)), modificare direttamente l'input di tracking modifica non solo $v_i^{corr}$ ma anche parzialmente $\omega_i^{corr}$. In effetti, data la forma degli ingressi di tracking (2.3), $\dot{x}^2 + \dot{y}^2 = v^2 + b^2\omega^2$.

Ne risulta, per il differential drive, un comportamento vicino a quello visto in sezione 5.0.2, con una variazione della traiettoria che evita la grande sovraelongazione che invece si vede in




(a) Traiettoria del robot con correzione

(b) Input tracking + MPC 1-step

(c) Input tracking + MPC multistep

**Figura 5.5** Simulazione 10: tracking di un arco di cerchio con errore iniziale con differential drive, con correzione sugli ingressi di tracking.

4.0.3. I risultati sono esposti in figura 5.5. Visto quanto appena detto, lo stesso risultato può essere ottenuto con lo schema di controllo in figura 3.1 e impostando $H$ come

$$H = 2 \cdot \left[ \frac{r^2}{4} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + b^2 \frac{r^2}{d^2} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \right] \tag{5.8}$$

Tutte queste situazioni mettono il luce come il comportamento osservato nei capitoli 3 e 4 sia piuttosto peculiare e attribuile alla scelta della funzione di costo e come funzioni di costo diverse portino a comportamenti molto diversi fra loro.

Inoltre, il fatto che con il robot uniciclo non si apprezzino differenze particolari dalla minimizzazione di una sola delle componenti della correzione deriva dalla forma della regione ammissibile: essendo un rettangolo, la scelta del valore per una variabile non vincola ulteriormente l'altra. Nel caso del robot differential drive invece, essendo la regione ammissibile un rombo, la scelta di una valore per una delle due variabili di decisione vincola il valore dell'altra, che ha come conseguenza la notevole differenza fra le due traiettorie.

In tutte le simulazione si osserva comunque che MPC 1-step e MPC multistep producono gli stessi ingressi. Questo è conseguenza della struttura del problema e in particolare dello step di linearizzazione e dei vincoli, come illustrato in sezione 3.1.4.

# Capitolo 6

# Conclusione

Il *trajectory tracking* è l'azione di inseguimento di una traiettoria con un robot. Tuttavia alcune traiettorie possono far saturare gli attuatori del robot, rendendone impossibile il tracking esatto. Questo avviene se la traiettoria richiede una velocità eccessiva per gli attuatori, presenta curvature elevate oppure errori iniziali elevati fra la posizione del robot e la traiettoria.

Lo scopo di questo progetto era fare trajectory tracking con robot di tipo uniciclo e differential drive, aggiungendo un termine generato da un controllore MPC-based per correggere l'azione di tracking ed assicurare che lo sforzo di controllo rientrasse nei limiti di saturazione degli attuatori. Il problema di ottimizzazione del controllore MPC-based è stato impostato per minimizzare la correzione necessaria a far rientrare il controllo nei limiti di saturazione.

Il funzionamento del termine correttivo è stato verificato simulando in MATLAB l'inseguimento di diverse traiettoria con diverse condizioni iniziali. È stato trovato che il controllore MPC-based riesce con successo a generare un'azione correttiva che non fa saturare gli attuatori, tuttavia la scelta di minimizzare la correzione la porta a comportarsi come la saturazione stessa. Ne risulta che gli ingressi corretti a monte della saturazione sono uguali a quelli non corretti a valle della saturazione, sia per il robot uniciclo che per il robot differential drive. Inoltre, è stato trovato che la formulazione del problema di ottimizzazione rende le variabili di decisione indipendenti fra loro, rendendo indifferente la lunghezza dell'orizzonte di predizione. Per un'implementazione più computazionalmente efficiente ci si può quindi limitare ad un controllore MPC 1-step.

È comunque importante notare che questi due fenomeni non si estendono, in generale, a modelli di robot più complessi o a formulazioni con vincoli aggiuntivi. Ne è un esempio [2], che usa la stessa funzione di costo usata in questo progetto ma introduce vincoli di stabilità del sistema nella formulazione di MPC. Per un ulteriore esempio, si pensi ad un robot car-like per cui si introduce un ulteriore vincolo sull'angolo di sterzo. In questo caso, la saturazione degli ingressi violerebbe il vincolo, mentre l'uso di MPC troverebbe una soluzione diversa in grado di soddisfarli tutti.

Nell'ultimo capitolo è stato visto come questo fenomeno sia inoltre dovuto alla funzione di costo, e come funzioni di costo diverse portino a comportamenti anche molto diversi fra loro. In particolare, è stato esplorato l'impatto sul tracking della minimizzazione di solo una componente dell'azione correttiva. In sviluppi futuri di questo lavoro verranno provate formulazione diverse della funzione di costo, che prendano in considerazione anche l'errore di tracking, ad esempio minimizzandone la norma quadra, oltre che il termine correttivo stesso.

# Bibliografia

[1] Giuseppe Oriolo. AdA - Introduction to mobile robotics, 2024. URL `https://www.diag.uniroma1.it/oriolo/ada-imr/index.html`.

[2] Manuel Beglini, Tommaso Belvedere, Leonardo Lanari, and Giuseppe Oriolo. An Intrisically Stable MPC Approach for Anti-Jackknifing Control of Tractor-Trailer Vehicles. *IEEE/ASME Transactions on Mechatronics*, 27(6):4417–4428, 2022. doi: https://doi.org/10.1109/TMECH.2022.3154295.