Università degli Studi di Roma "La Sapienza"

Facoltà di Ingegneria

Tracking Adattativo di un Robot Car-Like mediante *Feedback Linearization*

Lorenzo Magliocchetti Arrigo Marchiori Michele Marino

Ottobre 2006

Indice

Elenco delle figure

Capitolo 1

Introduzione

Scopo di questo lavoro è la progettazione e la simulazione di un sistema di controllo per un robot mobile anolonomo su ruote che si muove in un piano. Tale sistema di controllo si basa sulla tecnica del *feedback*.

Nel controllo automatico, il feedback migliora la performance del sistema controllato permettendo l'esecuzione di un task assegnato anche in presenza di disturbi esterni e/o errori iniziali. A tal fine, viene ricostruito lo stato attuale del robot utilizzando misure effettuate in tempo reale tramite sensori, i quali possono essere propriocettivi (calcolo odometrico mediante encoder) o esterocettivi (sonar, laser).

Limiteremo la nostra analisi al caso in cui l'ambiente in cui opera il robot è privo di ostacoli. Nel caso — più realistico — in cui siano presenti ostacoli, consideriamo implicitamente il controllore calato in una architettura gerarchica in cui un pianificatore risolve a monte il problema dell'aggiramento dell'ostacolo. In questa prospettiva, il controllore deve far inseguire al robot una data traiettoria, supposta priva di ostacoli.

Il sistema robotico considerato è un veicolo il cui modello cinematico approssima quello di un'autovettura. La configurazione di questo robot è rappresentata dalla posizione e l'orientamento del suo corpo principale nel piano nonché dall'angolo di sterzata delle ruote anteriori. Per il controllo di movimento sono disponibili due comandi di velocità. Tale modello copre in modo alquanto realistico una grande varietà di veicoli robotici.

La natura anolonoma del robot *Car-Like* è collegata all'assunzione che le ruote del robot possono ruotare senza slittare. Ciò vuol dire che sono presenti vincoli che limitano i movimenti istantanei che il robot può effettuare. Questo problema può essere aggirato con un'opportuna scelta delle grandezze utilizzate per il controllo.

In questo documento abbiamo utilizzato una particolare notazione tipografica, per distinguere "a colpo d'occhio" grandezze scalari e vettoriali:

grandezze scalari sono indicate in carattere corsivo. Ad esempio: x.

grandezze vettoriali sono indicate in grassetto. Esempio: $\mathbf{x}$.

matrici sono indicate in grassetto, spesso da lettera maiuscola. Esempio: $\mathbf{A}$.

funzioni sono indicate a seconda del tipo di valore che ritornano. Esempio: $\mathbf{f}(\mathbf{x})$ è una funzione vettoriale di variabile vettoriale.

1.1 Classificazione del problema

Il problema che si andrà ad affrontare riguarda il *tracciamento di una traiettoria* preassegnata. Il robot deve raggiungere e seguire un cammino nello spazio cartesiano secondo una determinata legge oraria. In modo equivalente, il robot deve inseguire un *robot di riferimento*.

Il problema di tracciamento consiste nell'annullamento dell'errore cartesiano bidimensionale[1] $\mathbf{e} = \begin{bmatrix} z_{d1} - z_1 & z_{d2} - z_2 \end{bmatrix}^\top$ usando entrambi gli ingressi di controllo. Per il robot *Car-Like* imporre $\mathbf{e} \to 0$ per $t \to \infty$ implica il controllo sulle quattro variabili di configurazione (la posizione del punto rappresentativo del robot — che verrà scelta *ad hoc* —, l'orientamento del robot e l'angolo di sterzata).

1.2 Generalità sul controllo

Da un punto di vista controllistico, il compito sopra descritto è definito dal sistema non-lineare

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v} \tag{1.1}$$

che rappresenta il modello cinematico del robot. $\mathbf{q}$ è il vettore 4-dimensionale di coordinate generalizzate, $\mathbf{v}$ è il vettore bidimensionale delle velocità di ingresso (velocità di trazione delle ruote posteriori e velocità di sterzata), mentre le colonne $\mathbf{g_1}$ e $\mathbf{g_2}$ della matrice $\mathbf{G}$ sono campi vettoriali.
Il sistema (1.1) è senza deriva, una caratteristica dei modelli cinematici del primo ordine. La sua natura non-lineare è intrinsecamente correlata all'anolonomia dei vincoli *Pfaffiani* originali. A differenza di quanto accade con i manipolatori articolati, stabilizzare un robot mobile intorno ad un punto è più difficile che stabilizzarlo lungo una traiettoria. Questo avviene perchè provando a linearizzare il sistema (1.1) intorno ad una configurazione fissa, si ottiene un sistema non controllabile. Di converso, la linearizzazione intorno ad una traiettoria dà luogo ad un sistema lineare tempo-variante controllabile.

[1] Indichiamo la posizione attuale e quella "desiderata" del robot rispettivamente con i vettori $\mathbf{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top$ e $\mathbf{z_d} = \begin{bmatrix} z_{d1} & z_{d2} \end{bmatrix}^\top$

Capitolo 2

Modellazione e analisi del robot

In questo capitolo verranno derivate le equazioni cinematiche di un robot *Car-Like* e verranno analizzate le proprietà fondamentali del sistema corrispondente da un punto di vista controllistico.

2.1 Modellazione cinematica

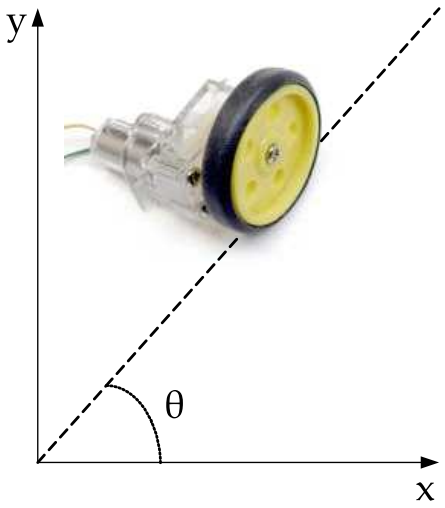

Figura 2.1: Modello dell'uniciclo

La caratteristica principale del modello cinematico di un robot mobile su ruote è la presenza dei vincoli di anolonomia dovuti alla rotazione delle ruote *senza slittamento* sul suolo. Per semplicità, consideriamo una singola ruota che mantiene il suo corpo verticale su un piano, come mostrato in figura 2.1. Questo tipo di modello è chiamato *uniciclo*. La sua configurazione può essere descritta da un vettore $\mathbf{q}$ di tre

coordinate generalizzate, ovvero la posizione del punto di contatto fra ruota e suolo (x, y) misurata in un frame fisso e l'angolo di orientamento della ruota θ rispetto all'asse fisso x.

Le velocità generalizzate $\dot{\mathbf{q}}$ non possono assumere qualunque valore; in particolare, devono rispettare la relazione:

$$\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0 \tag{2.1}$$

la quale mostra che la velocità lineare del punto di contatto è perpendicolare all'asse della stessa (velocità laterale nulla).

L'equazione (2.1) è un tipico esempio di vincolo *Pfaffiano* $\mathbf{C}(\mathbf{q})\dot{\mathbf{q}} = 0$, lineare nelle velocità generalizzate. Di conseguenza, tutte le velocità ammissibili sono contenute nel kernel della matrice $\mathbf{C}(\mathbf{q})$. In questo caso, si ottiene

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} \rho\omega_r + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_s \tag{2.2}$$

dove ω_r e ω_s sono rispettivamente la velocità di rotolamento sul piano e la velocità angolare riferita all'orientamento della ruota, mentre ρ è il raggio. Definiamo il vettore

$$\mathbf{v} = \begin{bmatrix} \omega_r \\ \omega_s \end{bmatrix}. \tag{2.3}$$

In questo modo si ritrova la forma dell'equazione (1.1). L'equazione (2.2) rappresenta il *modello cinematico* dell'uniciclo.

Consideriamo adesso il modello di un robot *Car-Like*, come mostrato in figura 2.2. Per semplicità, assumeremo che le due ruote di ogni asse collassino in una singola ruota al centro dell'asse in questione (approssimazione *"bicycle"*). Le coordinate generalizzate sono $\mathbf{q} = \begin{bmatrix} x & y & \theta & \phi \end{bmatrix}^\top$, dove (x, y) sono le coordinate cartesiane del punto di controllo, θ misura l'orientamento del corpo del robot rispetto all'asse x e ϕ è l'angolo di sterzata.

Il sistema è soggetto ai seguenti vincoli di anolonomia, uno per ogni ruota:

$$\dot{x}_f s_{\theta\phi} - \dot{y}_f c_{\theta\phi} = 0 \tag{2.4}$$

$$\dot{x} s_\theta - \dot{y} c_\theta = 0 \tag{2.5}$$

dove è stata utilizzata l'espressione compatta per le funzioni trigonometriche[1], $x_f = x + \ell c_\theta$ e $y_f = y + \ell s_\theta$ denotano le coordinate della ruota anteriore ed ℓ è la lunghezza del robot (ovvero la distanza fra gli assi).

Effettuando la sostituzione $(x_f, y_f) \to (x, y)$, il primo vincolo diventa:

$$\dot{x} s_{\theta\phi} - \dot{y} c_{\theta\phi} - \dot{\theta}\ell c_\phi = 0 \tag{2.6}$$

[1] Da adesso in poi verrà usata questa notazione: $s_{\theta\phi} = sin(\theta + \phi)$, e così via.

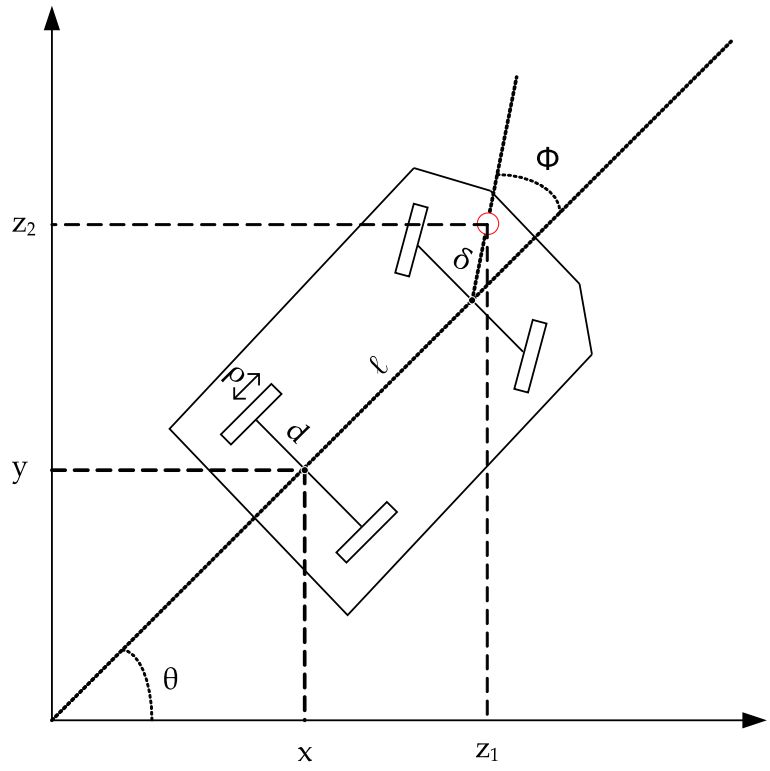

Figura 2.2: Modello del robot *Car-Like*

Si può a questo punto costruire la matrice Pfaffiana dei vincoli:

$$\mathbf{C}(\mathbf{q}) = \begin{bmatrix} s_{\theta\phi} & -c_{\theta\phi} & -\ell c_\phi & 0 \\ s_\theta & -c_\theta & 0 & 0 \end{bmatrix} \tag{2.7}$$

la quale ha rango 2 e da cui deriva il modello cinematico:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} c_\theta \\ s_\theta \\ t_\phi/\ell \\ 0 \end{bmatrix} \rho\omega_r + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega_s \tag{2.8}$$

con la stessa denominazione delle variabili usata nella (2.2).

Si registra una singolarità in $\phi = \pm\pi/2$, in cui il primo campo vettoriale ha una discontinuità. Una tale situazione si verifica solo teoricamente, allorquando le ruote anteriori sono normali all'asse longitudinale. Nella pratica, nella maggioranza dei casi, l'angolo di sterzata ϕ appartiene ad un intervallo centrato in $\phi = 0$ e di estensione inferiore a π.

Occorre considerare che un modello cinematico più completo dovrebbe includere gli angoli di rotazione di ogni ruota come coordinata generalizzata per tener conto

della presenza di attuatori e sensori sugli assi delle ruote nonché di non-idealità tipiche come ad esempio la deformazione degli pneumatici.

Tuttavia, il modello sopra descritto cattura l'essenza della cinematica del veicolo ed è adeguato alla progettazione della strategia di controllo.

Capitolo 3

Inseguimento di traiettoria

In questo capitolo, consideriamo il problema di far seguire al robot una traiettoria cartesiana mediante un controllo a retroazione. Tale controllo linearizza staticamente la dinamica del robot — non-lineare, come vedremo — e successivamente applica un ingresso basato sul controllo proporzionale e derivativo. Questo approccio di *feedback linearizzazione* assicura stabilità asintotica per uno stato iniziale arbitrario.

3.1 Controllo mediante *feedback-linearizzazione* esatta

È noto che se il numero delle coordinate generalizzate eguaglia quello degli input di comando, è possibile utilizzare un feedback non-lineare dallo stato che permetta di cancellare totalmente le non-linearità del sistema e di renderlo lineare. In generale, la linearità delle equazioni del sistema si visualizza solamente dopo una trasformazione di coordinate nello spazio di stato. Una volta linearizzato il sistema, è semplice completare la sintesi di un controllore lineare stabilizzante. Accanto alla possibilità di trasformare mediante retroazione l'intero set di equazioni differenziali in un sistema lineare (*full-state linearization*), è possibile trovare un risultato più debole in cui è reso lineare soltanto il trasferimento ingresso-uscita (*input-output linearization*). Esistono condizioni necessarie e sufficienti per la risolubilità di entrambi i problemi mediante una retroazione *statica*.

Consideriamo il sistema non-lineare descritto dalle equazioni:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\,\mathbf{v} \tag{3.1}$$

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \tag{3.2}$$

dove $\mathbf{x}$ è lo stato del sistema, $\mathbf{v}$ è l'ingresso e $\mathbf{z}$ è l'uscita cui vogliamo assegnare un comportamento desiderato, ad esempio percorrere una determinata traiettoria.

Indichiamo la d-esima derivata di Lie di $\mathbf{h}(\mathbf{x})$ rispetto a $\mathbf{g}(\mathbf{x})$: $L_g^d\mathbf{h}(\mathbf{x})$. Al variare di d, la sua espressione è:

$$L_g\mathbf{h}(\mathbf{x}) = \left(\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right)^{\top} \mathbf{g}(\mathbf{x}) \tag{3.3}$$

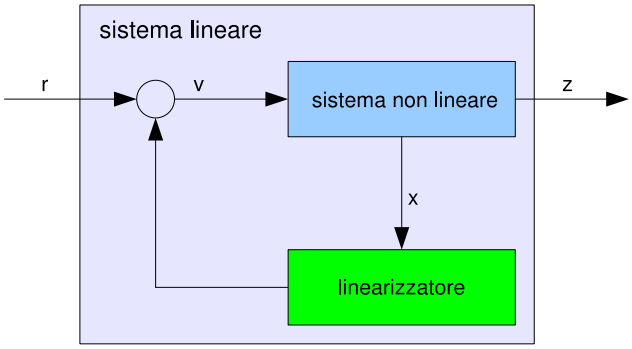

Figura 3.1: Sistema di controllo con *feedback-linearizzazione*

$$L_g^2\mathbf{h}(\mathbf{x}) = L_g(L_g\mathbf{h}(\mathbf{x})) \tag{3.4}$$
$$L_g^3\mathbf{h}(\mathbf{x}) = \dots. \tag{3.5}$$

Si noti che $L_g^d\mathbf{h}(\mathbf{x})$ ha le dimensioni di una matrice. Allo stesso modo si definiscono le derivate di Lie di $\mathbf{h}(\mathbf{x})$ rispetto a $\mathbf{f}(\mathbf{x})$: $L_f^d\mathbf{h}(\mathbf{x})$; queste hanno le dimensioni di un vettore.

Derivando $\mathbf{z}$ rispetto al tempo si ottiene:

$$\dot{\mathbf{z}} = \left(\frac{\partial\mathbf{h}(\mathbf{x})}{\partial\mathbf{x}}\right)^\top \dot{\mathbf{x}} \tag{3.6}$$

Sostituendo le espressioni di $\dot{\mathbf{x}}$ dalla (3.1) e della derivata prima di Lie nell'espressione di $\dot{\mathbf{z}}$ (3.6) si ottiene:

$$\dot{\mathbf{z}} = L_f\mathbf{h}(\mathbf{x}) + L_g\mathbf{h}(\mathbf{x})\,\mathbf{v} \tag{3.7}$$

Supponiamo che $L_g\mathbf{h}(\mathbf{x})$ sia invertibile. Considerando $\mathbf{r} = \dot{\mathbf{z}}$ e risolvendo la (3.7) per $\mathbf{v}$ si ottengono le equazioni del sistema di controllo (figura 3.1):

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{r} \\ \mathbf{v} = (L_g\mathbf{h}(\mathbf{x}))^{-1}(-L_f\mathbf{h}(\mathbf{x}) + \mathbf{r}) \end{cases} \tag{3.8}$$

Dal momento che la derivata dell'uscita è pari all'ingresso, il sistema di controllo si comporta come un integratore.

In generale, sussiste una *dinamica interna* non-lineare che non influisce sulla risposta ingresso-uscita. Tuttavia bisogna controllare la sua stabilità per garantire un buon inseguimento dell'output.

3.1.1 Linearizzazione I/O mediante retroazione statica

Si faccia riferimento alla figura 2.2. Per un robot *Car-Like*, la scelta naturale dell'uscita per un compito di inseguimento di traiettoria è

$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{3.9}$$

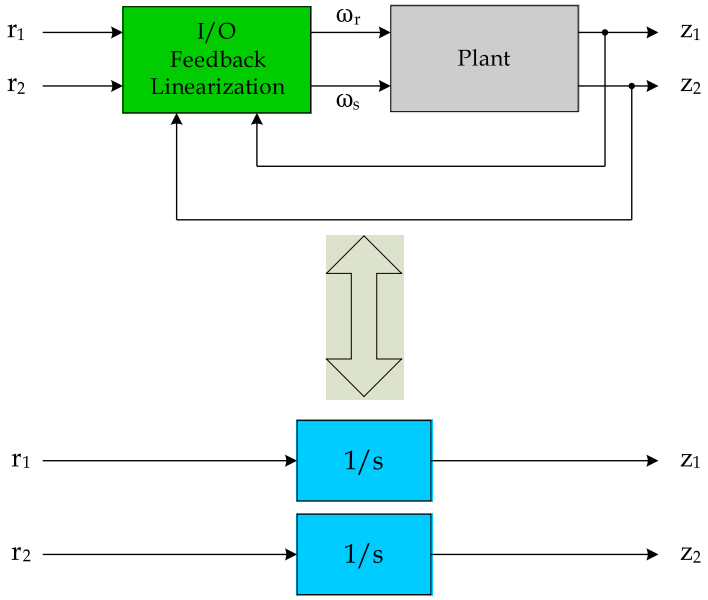

Figura 3.2: Schema di equivalenza *feedback-linearizzazione*

L'algoritmo linearizzante inizia calcolando

$$\dot{\mathbf{z}} = \begin{bmatrix} \rho c_\theta & 0 \\ \rho s_\theta & 0 \end{bmatrix} \mathbf{v} = \mathbf{A}(\theta)\mathbf{v} \tag{3.10}$$

Poichè questa matrice è singolare, l'algoritmo fallisce nella linearizzazione.

Si può aggirare il problema considerando un altro punto rappresentativo del robot. Sempre con riferimento alla figura 2.2, si può scegliere come uscita

$$\mathbf{z} = \begin{bmatrix} x + \ell c_\theta + \delta c_{\theta\phi} \\ y + \ell s_\theta + \delta s_{\theta\phi} \end{bmatrix} \tag{3.11}$$

dove δ è la distanza fra il nuovo punto rappresentativo del robot e il centro dell'asse anteriore, $\delta \neq 0$. Differenziando la nuova uscita si ottiene

$$\dot{\mathbf{z}} = \begin{bmatrix} \left(c_\theta - t_\phi \left(s_\theta + \frac{\delta s_{\theta\phi}}{\ell}\right)\right)\rho & -\delta s_{\theta\phi} \\ \left(s_\theta + t_\phi \left(c_\theta + \frac{\delta c_{\theta\phi}}{\ell}\right)\right)\rho & \delta c_{\theta\phi} \end{bmatrix} \mathbf{v} = \mathbf{A}(\theta,\phi)\mathbf{v} \tag{3.12}$$

Dal momento che $\det \mathbf{A}(\theta,\phi) = \rho\delta/c_\phi \neq 0$ per le motivazioni espresse a commento della (2.8), possiamo porre $\dot{\mathbf{z}} = \mathbf{r}$ e risolvere per gli ingressi $\mathbf{v}$:

$$\begin{aligned} \mathbf{v} &= \mathbf{A}^{-1}(\theta,\phi)\mathbf{r} \\ &= \begin{bmatrix} -\frac{c_\phi}{\delta}\left(s_\theta + t_\phi\left(c_\theta + \frac{\delta c_{\theta\phi}}{\ell}\right)\right) & \frac{c_\phi c_{\theta\phi}}{\rho} \\ \frac{c_\phi}{\delta}\left(c_\theta - t_\phi\left(s_\theta + \frac{\delta s_{\theta\phi}}{\ell}\right)\right) & \frac{c_\phi s_{\theta\phi}}{\rho} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \end{aligned} \tag{3.13}$$

Nelle nuove coordinate generalizzate $\begin{bmatrix} z_1 & z_2 & \theta & \phi \end{bmatrix}^\top$, le equazioni che descrivono la dinamica del sistema ad anello chiuso si trovano a partire dalla (3.13)[1]:

$$\dot{z}_1 = r_1 \tag{3.14}$$

$$\dot{z}_2 = r_2 \tag{3.15}$$

$$\dot{\theta} = s_\phi \frac{c_{\theta\phi} r_1 + s_{\theta\phi} r_2}{\ell} \tag{3.16}$$

$$\dot{\phi} = -\left(\frac{c_{\theta\phi} s_\phi}{\ell} + \frac{s_{\theta\phi}}{\delta}\right) r_1 - \left(\frac{s_{\theta\phi} s_\phi}{\ell} - \frac{c_{\theta\phi}}{\delta}\right) r_2. \tag{3.17}$$

Questa dinamica è lineare I/O e disaccoppiata (un integratore per ogni canale, vedi figura 3.2).

Si nota l'esistenza di una dinamica interna bidimensionale espressa dalle due equazioni differenziali che coinvolgono θ e ϕ.

Per risolvere il problema di inseguimento della traiettoria, scegliamo un controllo di tipo PD con derivata in *feed-forward*:

$$\begin{aligned} r_1 &= \dot{z}_{d1} + k_{p1}(z_{d1} - z_1) \\ r_2 &= \dot{z}_{d2} + k_{p2}(z_{d2} - z_2) \end{aligned} \tag{3.18}$$

dove k_{p1} e k_{p2} sono due guadagni, mentre z_{d1} e z_{d2} sono le coordinate di riferimento disponibili all'ingresso del controllore.

Il sistema di controllo, quindi, impone che l'errore di inseguimento converga esponenzialmente a zero per ogni condizione iniziale $\begin{bmatrix} z_1(t_0) & z_2(t_0) & \theta(t_0) & \phi(t_0) \end{bmatrix}^\top$. Le equazioni che lo descrivono si ricavano a partire dalla (3.13), sostituendo le espressioni (3.18):

$$\omega_r = \frac{1}{\rho}\left(c_\phi c_{\theta\phi}\left(\dot{z}_{d1} + k_{p1}(z_{d1} - z_1)\right) + c_\phi s_{\theta\phi}\left(\dot{z}_{d2} + k_{p2}(z_{d2} - z_2)\right)\right) \tag{3.19}$$

$$\begin{aligned} \omega_s = &-\left(\frac{c_{\theta\phi} s_\phi}{\ell} + \frac{s_{\theta\phi}}{\delta}\right)\left(\dot{z}_{d1} + k_{p1}(z_{d1} - z_1)\right) + \\ &-\left(\frac{s_{\theta\phi} s_\phi}{\ell} - \frac{c_{\theta\phi}}{\delta}\right)\left(\dot{z}_{d2} + k_{p2}(z_{d2} - z_2)\right). \end{aligned} \tag{3.20}$$

Si noti che:

- Mentre le due variabili di uscita convergono alla traiettoria desiderata — nel tempo e nei modi dettati dalla scelta dei guadagni nelle (3.18) — il comportamento delle variabili $\theta(t)$ e $\phi(t)$ è svincolato dalla specifica e dettato dalle equazioni (3.16) e (3.17).
- Una completa analisi richiederebbe lo studio della stabilità del sistema ad anello chiuso, per monitorare il comportamento nel tempo di $\theta(t)$ e $\phi(t)$ ed accertarne la limitatezza sulla traiettoria nominale.

[1] Per esprimere $\dot{\theta}$ in funzione di ω_r si utilizza l'equazione cinematica del modello (2.8).

Capitolo 4

Adattamento

La strategia di controllo vista nel capitolo precedente richiede una conoscenza approfondita del robot. Siccome ciò non è possibile a priori, in questo capitolo studiamo il problema di identificare "in linea" quei parametri del robot che sono necessari per controllarlo. Definiremo quindi un algoritmo che ci permetta di aggiornare le stime in modo iterativo.

4.1 Formulazione del problema

I parametri meccanici del robot non sono noti con precisione infinita. I componenti che formano la struttura derivano da processi industriali non perfetti. E' possibile quindi che si verifichi uno scostamento, seppur lieve, dal valore nominale di tali parametri. Il controllore del robot sfrutta queste grandezze per calcolare l'entità del controllo. Nello specifico, i due parametri ρ ed ℓ, che vengono utilizzati nella (3.13), non sono altro che *stime*, a priori diverse dai valori *veri*. Effettuando ora una differenziazione fra i parametri *veri* ρ ed ℓ e quelli *stimati* ρ_c ed ℓ_c, la (3.13) deve essere modificata in:

$$\mathbf{v} = \mathbf{A^{-1}}(\theta, \phi, \rho_c, \ell_c)\mathbf{r} \tag{4.1}$$

Se i parametri ρ_c ed ℓ_c fossero mantenuti costanti, ad esempio a seguito di un'inizializzazione all'interno del controllore, il controllo non lavorerebbe al meglio. Per ottimizzare la sua efficacia, pertanto, si procede alla progettazione di un sistema adattativo indiretto discreto che stimi in modo *dinamico* il valore (costante) di questi parametri:

1. viene praticata una stima ricorsiva a partire da un valore iniziale
2. ad ogni passo vengono fornite al controllore le stime aggiornate

Il sistema adattativo indiretto, chiamato anche *identificatore di sistema*, deve servirsi di uno stimatore che legga i comandi impartiti al robot e la risposta del robot stesso. Utilizzando un algoritmo di adattamento, ad ogni passo di calcolo il

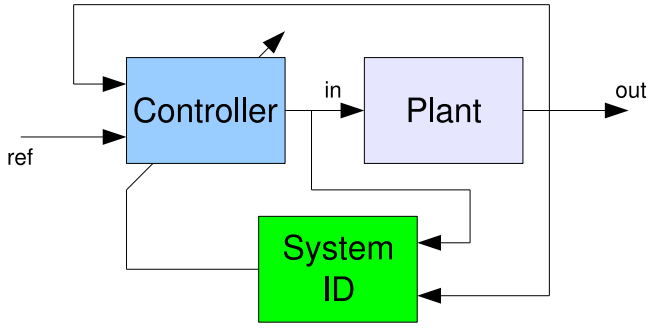

Figura 4.1: Identificatore di sistema

controllore viene tarato con i nuovi valori. Il sistema risultante è mostrato in figura 4.1.

4.2 Predisposizione alla stima

Per poter applicare un metodo di stima tanto veloce quanto efficace, occorre manipolare le equazioni differenziali che regolano la dinamica del robot affinché le relazioni risultanti siano *lineari* nei parametri da stimare.

A tal fine riscriviamo la (3.12) come un sistema di equazioni:

$$\dot{z}_1 = \left(c_\theta - t_\phi \left(s_\theta + \frac{\delta s_{\theta\phi}}{\ell}\right)\right)\omega_r\rho - \delta s_{\theta\phi}\omega_s \tag{4.2}$$

$$\dot{z}_2 = \left(s_\theta + t_\phi \left(c_\theta + \frac{\delta c_{\theta\phi}}{\ell}\right)\right)\omega_r\rho + \delta c_{\theta\phi}\omega_s \tag{4.3}$$

Per il momento, prendiamo in considerazione soltanto l'equazione (4.2). Sviluppandola, si ottiene:

$$\dot{z}_1 = (c_\theta - t_\phi s_\theta)\omega_r\rho - \delta s_{\theta\phi}\omega_s - \delta s_{\theta\phi} t_\phi \omega_r \frac{\rho}{\ell} \tag{4.4}$$

Il secondo termine non dipende nè da ρ nè da ℓ; il primo termine invece è lineare in ρ. A questo punto, si possono percorrere due strade alternative: linearizzare ρ ed ℓ mediante uno sviluppo in serie, oppure applicare una trasformazione dei parametri stimati senza ricorrere ad approssimazioni.

4.2.1 Approssimazione mediante sviluppo in serie

Per rendere l'espressione totalmente lineare in ρ e ℓ, si sviluppa il terzo termine della (4.4), cioè l'espressione ρ/ℓ, mediante uno sviluppo di Taylor multidimensionale arrestato al primo ordine:

$$\frac{\rho}{\ell} \approx \frac{\rho}{\ell}\Big|_{\rho_0,\ell_0} + \frac{\partial(\rho/\ell)}{\partial\rho}\Big|_{\rho_0,\ell_0}(\rho - \rho_0) + \frac{\partial(\rho/\ell)}{\partial\ell}\Big|_{\rho_0,\ell_0}(\ell - \ell_0) \tag{4.5}$$

dove ρ_0 e ℓ_0 sono dei valori ragionevoli per ρ ed ℓ.

Effettuando i calcoli si giunge alla seguente espressione:

$$\frac{\rho}{\ell} \approx \frac{1}{\ell_0}(\rho_0 - \frac{\rho_0 \ell}{\ell_0} + \rho) \tag{4.6}$$

I termini a destra sono lineari in ρ ed ℓ.

Sostituendo quest'ultima espressione nella (4.4) e riarrangiando si ottiene:

$$\dot{\tilde{z}}_1 = -\left[\delta s_{\theta\phi}\left(\omega_s + \frac{\omega_r t_\phi \rho_0}{\ell_0}\right)\right] + \left[\omega_r\left(c_\theta - t_\phi s_\theta - \frac{\delta s_{\theta\phi} t_\phi}{\ell_0}\right)\right]\rho + \frac{\omega_r \delta s_{\theta\phi} t_\phi \rho_0}{\ell_0^2}\ell \tag{4.7}$$

con $\dot{\tilde{z}}_1$ approssimazione di $\dot{z}_1$ tanto migliore quanto i valori ρ_0 ed ℓ_0 si avvicinano a quelli veri.

Mediante lo stesso procedimento applicato alla (4.3), si ottiene:

$$\dot{\tilde{z}}_2 = \left[\delta c_{\theta\phi}\left(\omega_s + \frac{\omega_r t_\phi \rho_0}{\ell_0}\right)\right] + \left[\omega_r\left(s_\theta + t_\phi c_\theta + \frac{\delta c_{\theta\phi} t_\phi}{\ell_0}\right)\right]\rho - \frac{\omega_r \delta c_{\theta\phi} t_\phi \rho_0}{\ell_0^2}\ell \tag{4.8}$$

con $\dot{\tilde{z}}_2$ approssimazione di $\dot{z}_2$ tanto migliore quanto i valori ρ_0 ed ℓ_0 si avvicinano a quelli veri.

Effettuando le sostituzioni:

$$\tilde{\mathbf{y}} = \begin{bmatrix} \dot{\tilde{z}}_1 + \delta s_{\theta\phi}\left(\omega_s + \frac{\omega_r t_\phi \rho_0}{\ell_0}\right) \\ \dot{\tilde{z}}_2 - \delta c_{\theta\phi}\left(\omega_s + \frac{\omega_r t_\phi \rho_0}{\ell_0}\right) \end{bmatrix} \tag{4.9}$$

$$\beta = \begin{bmatrix} \rho \\ \ell \end{bmatrix} \tag{4.10}$$

$$\phi_\rho(\mathbf{x}) = \begin{bmatrix} \omega_r\left(c_\theta - t_\phi s_\theta - \frac{\delta s_{\theta\phi} t_\phi}{\ell_0}\right) \\ \omega_r\left(s_\theta + t_\phi c_\theta + \frac{\delta c_{\theta\phi} t_\phi}{\ell_0}\right) \end{bmatrix} \tag{4.11}$$

$$\phi_\ell(\mathbf{x}) = \begin{bmatrix} \frac{\omega_r \delta s_{\theta\phi} t_\phi \rho_0}{\ell_0^2} \\ -\frac{\omega_r \delta c_{\theta\phi} t_\phi \rho_0}{\ell_0^2} \end{bmatrix} \tag{4.12}$$

dove $\mathbf{x} = \begin{bmatrix} \omega_r & \theta & \phi \end{bmatrix}^\top$, si riscrive semplicemente:

$$\tilde{\mathbf{y}} = \mathbf{F}(\mathbf{x}, \beta) = \phi_\rho(\mathbf{x})\rho + \phi_\ell(\mathbf{x})\ell \tag{4.13}$$

Definendo la matrice $\mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_\rho(\mathbf{x}) & \phi_\ell(\mathbf{x}) \end{bmatrix}$ si ottiene una forma ancora più compatta:

$$\tilde{\mathbf{y}} = \mathbf{F}(\mathbf{x}, \beta) = \mathbf{\Phi}(\mathbf{x})\beta \tag{4.14}$$

che è lineare nei parametri ρ e ℓ, ma non negli ingressi $\mathbf{x}$.

4.2.2 Trasformazione dei parametri

La linearizzazione effettuata nella precedente sezione, pur creando i presupposti per la stima diretta dei parametri fisici del robot, toglie allo stimatore preziose informazioni sul sistema da identificare. Si può quindi ritenere più conveniente operare una trasformazione preventiva dei parametri al fine di evitare approssimazioni nelle equazioni dinamiche. Si faccia riferimento alla (4.4). Definendo un nuovo parametro $\nu = \rho/\ell$, si nota che adesso la dipendenza dai parametri è lineare senza bisogno di approssimare. Sarà poi necessario risalire al parametro fisico, mediante la relazione $\ell = \rho/\nu$, al momento della presentazione delle stime al controllore, per rispettare la definizione di *"indirect adaptation"* data in [2].

Occorre adottare a questo punto una nuova notazione:

$$\frac{\rho}{\ell} = \nu \tag{4.15}$$

$$\tilde{\mathbf{y}} = \begin{bmatrix} \dot{z}_1 + \delta s_{\theta\phi}\omega_s \\ \dot{z}_2 - \delta c_{\theta\phi}\omega_s \end{bmatrix} \tag{4.16}$$

$$\beta = \begin{bmatrix} \rho \\ \nu \end{bmatrix} \tag{4.17}$$

$$\phi_\rho(\mathbf{x}) = \begin{bmatrix} \omega_r (c_\theta - t_\phi s_\theta) \\ \omega_r (s_\theta + t_\phi c_\theta) \end{bmatrix} \tag{4.18}$$

$$\phi_\nu(\mathbf{x}) = \begin{bmatrix} -\omega_r \delta s_{\theta\phi} t_\phi \\ \omega_r \delta c_{\theta\phi} t_\phi \end{bmatrix} \tag{4.19}$$

A questo punto, si ottiene:

$$\tilde{\mathbf{y}} = \mathbf{F}(\mathbf{x}, \beta) = \phi_\rho(\mathbf{x})\rho + \phi_\nu(\mathbf{x})\nu \tag{4.20}$$

Definendo la matrice $\mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \phi_\rho(\mathbf{x}) & \phi_\nu(\mathbf{x}) \end{bmatrix}$ si ottiene la forma compatta:

$$\tilde{\mathbf{y}} = \mathbf{F}(\mathbf{x}, \beta) = \mathbf{\Phi}(\mathbf{x})\,\beta \tag{4.21}$$

che è lineare in ρ e ν ma non in ℓ, e tantomeno negli ingressi $\mathbf{x}$.

4.3 Algoritmo di ottimizzazione

Una volta definito lo stimatore, ovvero la funzione $\mathbf{\Phi}(\mathbf{x})$, è possibile impostare l'algoritmo di identificazione del sistema. Consideriamo dapprima un approccio di tipo *batch* per poi estenderlo al caso *ricorsivo*. Verranno riportati gran parte dei passaggi per rendere più chiara l'esposizione.

4.3.1 Sistema e stimatore

Svincoliamoci per il momento dal problema specifico e consideriamo un generico sistema con ingressi vettoriali $\mathbf{x}$ di dimensione qualunque e uscite $\mathbf{y} \in \mathbb{R}^m$.

L'obiettivo, come precedentemente spiegato, è quello di stimare alcuni parametri $\beta \in \mathbb{R}^n$ di questo sistema. A tale scopo, si utilizza uno stimatore che sia caratterizzato da una dipendenza lineare dal vettore dei parametri da stimare:

$$\tilde{\mathbf{y}} = \mathbf{F}(\mathbf{x}, \beta) = \mathbf{\Phi}(\mathbf{x})\beta$$

dove $\mathbf{\Phi}$ è una funzione non necessariamente lineare dell'ingresso $\mathbf{x}$.

La "bontà" dei valori assegnati ai parametri dello stimatore deve essere giudicata in base ad un criterio; normalmente si definisce una funzione di penalità da minimizzare.

4.3.2 Criterio dei minimi quadrati *batch*

Si effettuano j "esperimenti", cioè misure dell'ingresso e dell'uscita del sistema. I valori si organizzano in vettori di $\mathbb{R}^{jm}$:

$$\mathbf{x_m} = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_j} \end{bmatrix} \qquad \mathbf{y_m} = \begin{bmatrix} \mathbf{y_1} \\ \mathbf{y_2} \\ \vdots \\ \mathbf{y_j} \end{bmatrix}$$

Per ogni esperimento si considera l'errore tra l'uscita del sistema e quella dello stimatore:

$$\mathbf{e_i} = \mathbf{y_i} - \tilde{\mathbf{y}}_\mathbf{i} = \mathbf{y_i} - \mathbf{\Phi}(\mathbf{x_i})\beta \tag{4.22}$$

Incolonnando gli errori di tutti gli esperimenti, si ottiene il vettore $\mathbf{e_m} \in \mathbb{R}^{jm}$:

$$\mathbf{e_m} = \begin{bmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \vdots \\ \mathbf{e_j} \end{bmatrix} = \mathbf{y_m} - \begin{bmatrix} \mathbf{\Phi}(\mathbf{x_1}) \\ \mathbf{\Phi}(\mathbf{x_2}) \\ \vdots \\ \mathbf{\Phi}(\mathbf{x_j}) \end{bmatrix} \beta = \mathbf{y_m} - \bar{\mathbf{\Phi}}(\mathbf{x_m})\beta \tag{4.23}$$

con $\bar{\mathbf{\Phi}}(\mathbf{x_m}) \in \mathbb{R}^{jm \times n}$.

La funzione di penalità che si considera è la seguente:

$$V_j(\beta) = \frac{1}{2}\mathbf{e_m}^\top \mathbf{R} \mathbf{e_m} \tag{4.24}$$

dove $\mathbf{R}$ è una matrice simmetrica e definita positiva che ha la funzione di pesare opportunamente gli esperimenti. Con $\mathbf{R} = \mathbf{I}$, ad esempio, tutti gli esperimenti hanno lo stesso peso.

Sostituendo la (4.23) nella (4.24), derivando $V_j(\beta)$ rispetto a β, annullando il risultato e risolvendo rispetto a β, si ottiene:

$$\beta_{opt} = \left(\bar{\mathbf{\Phi}}(\mathbf{x_m})^\top \mathbf{A} \bar{\mathbf{\Phi}}(\mathbf{x_m})\right)^{-1} \bar{\mathbf{\Phi}}(\mathbf{x_m})^\top \mathbf{A} \mathbf{y_m} \tag{4.25}$$

dove $\mathbf{A} = \left(\mathbf{R} + \mathbf{R}^\top\right)$, anch'essa simmetrica e definita positiva.

Perché questo criterio funzioni, la matrice $\bar{\mathbf{\Phi}}(\mathbf{x_m})^\top \mathbf{A} \bar{\mathbf{\Phi}}(\mathbf{x_m})$ deve essere invertibile; questo accade se l'ingresso $\mathbf{x_m}$ del sistema è sufficientemente eccitante.

4.3.3 Criterio dei minimi quadrati ricorsivi

Per poter aggiornare "in linea" i parametri dello stimatore, bisogna modificare il criterio dei minimi quadrati, fornendone una formulazione ricorsiva. In questo modo, ad ogni "passo" di computazione k è possibile aggiornare la stima, tenendo conto di ciò che si è calcolato nei passi precedenti.

Si desidera quindi adattare la relazione (4.25). Omettendo la dipendenza di $\bar{\mathbf{\Phi}}$ da $\mathbf{x_m}$ e riscrivendo l'espressione in funzione del passo k si ottiene:

$$\beta(k) = \left(\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k)\,\bar{\mathbf{\Phi}}(k)\right)^{-1}\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k)\,\mathbf{y_m}(k) \tag{4.26}$$

dove:

$$\bar{\mathbf{\Phi}}(k) \in \mathbb{R}^{km\times n}, \qquad \mathbf{y_m}(k) \in \mathbb{R}^{km}, \qquad \mathbf{A}(k) \in \mathbb{R}^{km\times km}$$

La struttura di $\bar{\mathbf{\Phi}}(k)$ è la stessa dell'eq. (4.23), cioè ad ogni passo si aggiunge in basso una nuova sottomatrice. Di conseguenza, la matrice al passo $k+1$ si può scrivere:

$$\bar{\mathbf{\Phi}}(k+1) = \begin{bmatrix} \bar{\mathbf{\Phi}}(k) \\ \mathbf{\Phi_{k+1}} \end{bmatrix} \quad \in \mathbb{R}^{(k+1)\times m} \tag{4.27}$$

dove $\mathbf{\Phi_{k+1}} \in \mathbb{R}^{m\times n}$.

Anche la matrice $\mathbf{A}(k)$ ed il vettore $\mathbf{y_m}(k)$ si aggiornano ad ogni passo:

$$\mathbf{y_m}(k+1) = \begin{bmatrix} \mathbf{y_m}(k) \\ \mathbf{y_{k+1}} \end{bmatrix} \quad \in \mathbb{R}^{(k+1)m} \tag{4.28}$$

Nell'aggiornamento di $\mathbf{A}$, però, la parte che si riferisce al passato viene moltiplicata per un coefficiente λ:

$$\mathbf{A}(k+1) = \begin{bmatrix} \lambda\mathbf{A}(k) & \\ & \mathbf{A_{k+1}} \end{bmatrix} \quad \in \mathbb{R}^{(k+1)m\times(k+1)m}, \qquad 0<\lambda\leq 1 \tag{4.29}$$

λ è detto *"fattore di dimenticanza"*: si può vedere, ad esempio, che al passo $k+j$ la matrice $\mathbf{A}(k+j)$ contiene la sottomatrice $\mathbf{A}(k)$ moltiplicata per λ^j. Se $\lambda = 1$, non si effettua alcuna dimenticanza.

Definiamo la matrice $\mathbf{P}(k)$ attraverso la sua inversa:

$$\mathbf{P}^{-1}(k) = \bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k)\,\bar{\mathbf{\Phi}}(k) \quad \in \mathbb{R}^{n\times n} \tag{4.30}$$

Al passo $k+1$, secondo le definizioni appena date (4.27) e (4.29), ponendo $\lambda = c^2$, si ottiene:

$$\begin{aligned}
\mathbf{P}^{-1}(k+1) &= \bar{\mathbf{\Phi}}^\top(k+1)\mathbf{A}(k+1)\bar{\mathbf{\Phi}}(k+1) = \\
&= \begin{bmatrix} \bar{\mathbf{\Phi}}^\top(k) & \mathbf{\Phi}_{\mathbf{k+1}}^\top \end{bmatrix}\begin{bmatrix} \lambda\mathbf{A}(k) & \\ & \mathbf{A_{k+1}} \end{bmatrix}\begin{bmatrix} \bar{\mathbf{\Phi}}(k) \\ \mathbf{\Phi_{k+1}} \end{bmatrix} = \\
&= \begin{bmatrix} \lambda\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k) & \mathbf{\Phi}_{\mathbf{k+1}}^\top\mathbf{A_{k+1}} \end{bmatrix}\begin{bmatrix} \bar{\mathbf{\Phi}}(k) \\ \mathbf{\Phi_{k+1}} \end{bmatrix} = \\
&= \lambda\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k)\,\bar{\mathbf{\Phi}}(k) + \mathbf{\Phi}_{\mathbf{k+1}}^\top\mathbf{A_{k+1}}\mathbf{\Phi_{k+1}} = \\
&= \lambda\mathbf{P}^{-1}(k) + \mathbf{\Phi}_{\mathbf{k+1}}^\top\mathbf{A_{k+1}}\mathbf{\Phi_{k+1}}
\end{aligned} \tag{4.31}$$

Ora si consideri l'espressione di β (4.26) al passo $k+1$:

$$\beta(k+1) = \left(\bar{\mathbf{\Phi}}^\top(k+1)\mathbf{A}(k+1)\bar{\mathbf{\Phi}}(k+1)\right)^{-1}\bar{\mathbf{\Phi}}^\top(k+1)\mathbf{A}(k+1)\mathbf{y_m}(k+1)$$

riconoscendo nella prima parentesi l'espressione di $\mathbf{P}^{-1}(k+1)$ dalla (4.30) e sostituendo le espressioni di $\bar{\mathbf{\Phi}}(k+1)$ (4.27), $\mathbf{A}(k+1)$ (4.29) e $\mathbf{y_m}(k+1)$ (4.28) si ottiene:

$$\begin{aligned}
\beta(k+1) &= \mathbf{P}(k+1)\bar{\mathbf{\Phi}}^\top(k+1)\mathbf{A}(k+1)\mathbf{y_m}(k+1) = \\
&= \mathbf{P}(k+1)\left[\begin{array}{cc}\bar{\mathbf{\Phi}}^\top(k) & \mathbf{\Phi_{k+1}^\top}\end{array}\right]\left[\begin{array}{cc}\lambda\mathbf{A}\,(k) & \\ & \mathbf{A_{k+1}}\end{array}\right]\mathbf{y_m}(k+1) = \\
&= \mathbf{P}(k+1)\left[\begin{array}{cc}\lambda\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}\,(k) & \mathbf{\Phi_{k+1}^\top A_{k+1}}\end{array}\right]\left[\begin{array}{c}\mathbf{y_m}\,(k) \\ \mathbf{y_{k+1}}\end{array}\right] = \\
&= \mathbf{P}(k+1)\left(\lambda\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}\,(k)\,\mathbf{y_m}\,(k) + \mathbf{\Phi_{k+1}^\top A_{k+1}y_{k+1}}\right) \qquad (4.32)
\end{aligned}$$

L'espressione di $\mathbf{P}(k+1)$ si ottiene invertendo la (4.31):

$$\mathbf{P}(k+1) = \left(\lambda\mathbf{P}^{-1}(k) + \mathbf{\Phi_{k+1}^\top A_{k+1}\Phi_{k+1}}\right)^{-1}$$

siccome:

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right)^{-1}\mathbf{DA}^{-1} \qquad (4.33)$$

allora:

$$\mathbf{P}(k+1) = \frac{1}{\lambda}\mathbf{P}(k) - \frac{1}{\lambda}\mathbf{P}(k)\mathbf{\Phi_{k+1}^\top}\left(\frac{1}{\lambda}\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi_{k+1}^\top} + \mathbf{A_{k+1}^{-1}}\right)^{-1}\frac{1}{\lambda}\mathbf{\Phi_{k+1}P}(k) \qquad (4.34)$$

Per semplificare l'espressione si introduce il termine $\mathbf{\Gamma}(k) \in \mathbb{R}^{m\times m}$. Esso è invertibile, perché definito positivo.

$$\begin{aligned}
\mathbf{\Gamma}(k) &= \left(\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi_{k+1}^\top} + \lambda\mathbf{A_{k+1}^{-1}}\right)^{-1} \\
\mathbf{P}(k+1) &= \tfrac{1}{\lambda}\left(\mathbf{P}(k) - \mathbf{P}(k)\mathbf{\Phi_{k+1}^\top\Gamma}(k)\mathbf{\Phi_{k+1}P}(k)\right)
\end{aligned} \qquad (4.35)$$

Sostituendo questa espressione appena trovata in quella di $\beta(k+1)$ (4.32) si ottiene:

$$\begin{aligned}
\beta(k+1) &= \mathbf{P}(k)\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}\,(k)\,\mathbf{y_m}\,(k) + \frac{1}{\lambda}\mathbf{P}(k)\mathbf{\Phi_{k+1}^\top A_{k+1}y_{k+1}} + \\
&-\mathbf{P}(k)\mathbf{\Phi_{k+1}^\top\Gamma}(k)\mathbf{\Phi_{k+1}P}(k)\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}(k)\mathbf{y_m}\,(k) + \\
&-\frac{1}{\lambda}\mathbf{P}(k)\mathbf{\Phi_{k+1}^\top\Gamma}(k)\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi_{k+1}^\top A_{k+1}y_{k+1}} \qquad (4.36)
\end{aligned}$$

Abbandonando per un attimo questa espressione, si consideri che per definizione di $\beta(k)$ (4.26) e di $\mathbf{P}(k)$ (4.30):

$$\beta(k) = \mathbf{P}(k)\bar{\mathbf{\Phi}}^\top(k)\mathbf{A}\,(k)\,\mathbf{y_m}\,(k) \qquad (4.37)$$

e per la seconda delle (4.35):

$$\begin{aligned}\mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}} &= \frac{1}{\lambda}\left(\mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}} - \mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{\Gamma}(k)\mathbf{\Phi_{k+1}A_{k+1}}\right) = \\ &= \frac{1}{\lambda}\mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\left(\mathbf{A_{k+1}} - \mathbf{\Gamma}(k)\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}}\right) \quad (4.38)\end{aligned}$$

L'espressione tra parentesi è pari a $\lambda\mathbf{\Gamma}(k)$:

$$\mathbf{A_{k+1}} - \mathbf{\Gamma}(k)\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}} = \lambda\mathbf{\Gamma}(k)$$

infatti, raggruppando per $\mathbf{\Gamma}(k)$:

$$\mathbf{\Gamma}(k)\left(\lambda\mathbf{I} + \mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}}\right) = \mathbf{A_{k+1}}$$

moltiplicando a destra entrambi i membri per $\mathbf{A}_{\mathbf{k+1}}^{-1}$ si ottiene:

$$\mathbf{\Gamma}(k)\left(\lambda\mathbf{A}_{\mathbf{k+1}}^{-1} + \mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\right) = \mathbf{I}$$

questa si rivela un'identità se si considera l'espressione di $\mathbf{\Gamma}(k)$ (4.35):

$$\mathbf{\Gamma}(k)\mathbf{\Gamma}^{-1}(k) = \mathbf{I}$$

Di conseguenza, la (4.38) diventa:

$$\mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}} = \mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{\Gamma}(k) \quad (4.39)$$

Inserendo questa e la (4.37) nell'espressione di $\beta(k+1)$ che si stava calcolando (4.36) si ottiene:

$$\begin{aligned}\beta(k+1) &= \beta(k) + \frac{1}{\lambda}\mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}y_{k+1}} + \\ &\quad -\mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}\Phi_{k+1}P}(k)\bar{\mathbf{\Phi}}^{\top}(k)\mathbf{A}(k)\mathbf{y_m}(k) + \\ &\quad -\frac{1}{\lambda}\mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}y_{k+1}} \\ &= \beta(k) + \frac{1}{\lambda}\mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}\Gamma}^{-1}(k)\mathbf{A_{k+1}y_{k+1}} + \\ &\quad -\mathbf{P}(k-1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}\Phi_{k+1}P}(k)\Big(\bar{\mathbf{\Phi}}^{\top}(k)\mathbf{A}(k)\mathbf{y_m}(k) + \\ &\quad +\frac{1}{\lambda}\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}y_{k+1}}\Big) \\ &= \beta(k) + \mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}}\left(\frac{1}{\lambda}\mathbf{\Gamma}^{-1}(k)\mathbf{A_{k+1}y_{k+1}} + \right. \\ &\quad \left. -\mathbf{\Phi_{k+1}P}(k)\left(\bar{\mathbf{\Phi}}^{\top}(k)\mathbf{A}(k)\mathbf{y_m}(k) + \frac{1}{\lambda}\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}y_{k+1}}\right)\right) \\ &= \beta(k) + \mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}}\left(\frac{1}{\lambda}\left(\mathbf{\Gamma}^{-1}(k) + \right.\right. \\ &\quad \left.\left. -\mathbf{\Phi_{k+1}P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\right)\mathbf{A_{k+1}y_{k+1}} + \right. \\ &\quad \left. -\mathbf{\Phi_{k+1}}\beta(k)\right)\end{aligned}$$

Sostituendo l'espressione di $\mathbf{\Gamma}^{-1}(k)$ (4.35) si vede che l'espressione più interna, tra parentesi, diventa:

$$\mathbf{\Phi_{k+1}}\mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top} + \mathbf{A}_{\mathbf{k+1}}^{-1} - \mathbf{\Phi_{k+1}}\mathbf{P}(k)\mathbf{\Phi}_{\mathbf{k+1}}^{\top} = \lambda\mathbf{A}_{\mathbf{k+1}}^{-1}$$

Di conseguenza, l'espressione di $\beta(k+1)$ risulta:

$$\beta(k+1) = \beta(k) + \mathbf{P}(k+1)\mathbf{\Phi}_{\mathbf{k+1}}^{\top}\mathbf{A_{k+1}}\left(\mathbf{y_{k+1}} - \mathbf{\Phi_{k+1}}\beta(k)\right) \tag{4.40}$$

Rispetto al caso *batch*, un segnale di ingresso non eccitante non pregiudica la stabilità numerica di questo criterio. Nel caso in cui un parametro non sia stimabile in base agli ingressi e alle uscite misurate, esso semplicemente non viene aggiornato.

4.4 Algoritmo

Per applicare l'algoritmo ricorsivo, è necessario scegliere una frequenza di aggiornamento dei parametri. Si indichi con T il tempo che trascorre tra due iterazioni dell'algoritmo di stima; agli istanti $t = kT$ le grandezze β da stimare vengono aggiornate.

Tali grandezze entrano nella legge di controllo (3.19), (3.20), che qui riscriviamo utilizzando i parametri stimati $\rho_c(kT)$ e $\ell_c(kT)$:

$$\omega_r = \frac{1}{\rho_c(kT)}\left(c_\phi c_{\theta\phi}\left(\dot{z}_{d1} + k_{p1}(z_{d1} - z_1)\right) + c_\phi s_{\theta\phi}\left(\dot{z}_{d2} + k_{p2}(z_{d2} - z_2)\right)\right) \tag{4.41}$$

$$\begin{aligned}\omega_s = &-\left(\frac{c_{\theta\phi}s_\phi}{\ell_c(kT)} + \frac{s_{\theta\phi}}{\delta}\right)\left(\dot{z}_{d1} + k_{p1}(z_{d1} - z_1)\right) + \\ &-\left(\frac{s_{\theta\phi}s_\phi}{\ell_c(kT)} - \frac{c_{\theta\phi}}{\delta}\right)\left(\dot{z}_{d2} + k_{p2}(z_{d2} - z_2)\right)\end{aligned} \tag{4.42}$$

Nelle espressioni che seguono, si intenda:

$$\begin{aligned}\bar{c}_\theta &= \cos\left(\theta(kT)\right)\\ \bar{s}_\theta &= \sin\left(\theta(kT)\right)\\ \bar{t}_\phi &= \tan\left(\phi(kT)\right)\\ \bar{s}_{\theta\phi} &= \sin\left(\theta(kT) + \phi(kT)\right)\\ \bar{c}_{\theta\phi} &= \cos\left(\theta(kT) + \phi(kT)\right)\\ \bar{\omega}_r &= \omega_r(kT)\\ \bar{\omega}_s &= \omega_s(kT)\\ \bar{z}_{p1} &= \dot{z}_1(kT)\\ \bar{z}_{p2} &= \dot{z}_2(kT)\end{aligned}$$

4.4.1 Funzione linearizzata

Inserendo le espressioni (4.9)–(4.12) e (4.14) nelle (4.35) e (4.40), l'algoritmo consiste nella sequenza di calcoli:

$$
\begin{aligned}
\mathbf{\Phi_{k+1}} &= \begin{bmatrix} \bar{\omega}_r \left(\bar{c}_\theta - \bar{t}_\phi \bar{s}_\theta - \frac{\delta \bar{s}_{\theta\phi} \bar{t}_\phi}{\ell_0} \right) & \frac{\bar{\omega}_r \delta \bar{s}_{\theta\phi} \bar{t}_\phi \rho_0}{\ell_0^2} \\ \bar{\omega}_r \left(\bar{s}_\theta + \bar{t}_\phi \bar{c}_\theta + \frac{\delta c_{\theta\phi} \bar{t}_\phi}{l_0} \right) & -\frac{\bar{\omega}_r \delta \bar{c}_{\theta\phi} \bar{t}_\phi \rho_0}{\ell_0^2} \end{bmatrix} \\
\mathbf{y_{k+1}} &= \begin{bmatrix} \bar{z}_{p1} + \delta \bar{s}_{\theta\phi} \left(\bar{\omega}_s + \frac{\bar{\omega}_r \bar{t}_\phi \rho_0}{\ell_0} \right) \\ \bar{z}_{p2} - \delta \bar{c}_{\theta\phi} \left(\bar{\omega}_s + \frac{\bar{\omega}_r \bar{t}_\phi \rho_0}{\ell_0} \right) \end{bmatrix} \\
\mathbf{\Gamma}(k) &= \left(\mathbf{\Phi_{k+1}} \mathbf{P}(k) \mathbf{\Phi}_{\mathbf{k+1}}^\top + \lambda \mathbf{A}_{\mathbf{k+1}}^{-1} \right)^{-1} \\
\mathbf{P}(k+1) &= \frac{1}{\lambda} \left(\mathbf{P}(k) - \mathbf{P}(k) \mathbf{\Phi}_{\mathbf{k+1}}^\top \mathbf{\Gamma}(k) \mathbf{\Phi_{k+1}} \mathbf{P}(k) \right) \\
\beta(k+1) &= \beta(k) + \mathbf{P}(k+1) \mathbf{\Phi}_{\mathbf{k+1}}^\top \mathbf{A_{k+1}} \left(\mathbf{y_{k+1}} - \mathbf{\Phi_{k+1}} \beta(k) \right)
\end{aligned}
$$

4.4.2 Funzione non linearizzata

Inserendo le espressioni (4.15)–(4.19) e (4.21) nelle (4.35) e (4.40), l'algoritmo consiste nella sequenza di calcoli:

$$
\begin{aligned}
\mathbf{\Phi_{k+1}} &= \begin{bmatrix} \bar{\omega}_r \left(\bar{c}_\theta - \bar{t}_\phi \bar{s}_\theta \right) & -\bar{\omega}_r \delta \bar{s}_{\theta\phi} \bar{t}_\phi \\ \bar{\omega}_r \left(\bar{s}_\theta + \bar{t}_\phi \bar{c}_\theta \right) & \bar{\omega}_r \delta \bar{c}_{\theta\phi} \bar{t}_\phi \end{bmatrix} \\
\mathbf{y_{k+1}} &= \begin{bmatrix} \bar{z}_{p1} + \delta \bar{s}_{\theta\phi} \bar{\omega}_s \\ \bar{z}_{p2} - \delta \bar{c}_{\theta\phi} \bar{\omega}_s \end{bmatrix} \\
\mathbf{\Gamma}(k) &= \left(\mathbf{\Phi_{k+1}} \mathbf{P}(k) \mathbf{\Phi}_{\mathbf{k+1}}^\top + \lambda \mathbf{A}_{\mathbf{k+1}}^{-1} \right)^{-1} \\
\mathbf{P}(k+1) &= \frac{1}{\lambda} \left(\mathbf{P}(k) - \mathbf{P}(k) \mathbf{\Phi}_{\mathbf{k+1}}^\top \mathbf{\Gamma}(k) \mathbf{\Phi_{k+1}} \mathbf{P}(k) \right) \\
\beta(k+1) &= \beta(k) + \mathbf{P}(k+1) \mathbf{\Phi}_{\mathbf{k+1}}^\top \mathbf{A_{k+1}} \left(\mathbf{y_{k+1}} - \mathbf{\Phi_{k+1}} \beta(k) \right)
\end{aligned}
$$

Capitolo 5

Realizzazione

In questo capitolo, mostriamo la realizzazione al computer del modello del robot e del controllore.

5.1 Simulatore e schema generale

Per la simulazione del sistema è stato utilizzato *Simulink*© (versione 6.4), compreso nella versione 7.2.0.232 (r2006a) di *Matlab*© per WindowsXP. Lo schema simulink si presenta come in figura 5.1 a pagina 31.

5.2 Modellazione del Robot

Il robot è rappresentato come in figura 5.2 a pagina 32 ed è stato modellato come descritto nella sezione 3.1.1: un blocco che riceve in ingresso due comandi (il comando ω_r di velocità angolare delle ruote posteriori[1] e quello ω_s di velocità angolare dello sterzo) e restituisce le coordinate del punto rappresentativo[2] $\begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top$ con le sue derivate, l'angolo di orientamento del robot e l'angolo di sterzo. Cliccando sul blocco in questione si visualizzano i sottoblocchi che implementano le equazioni dinamiche (3.12), che qui riscriviamo:

$$\dot{z}_1 = \left(c_\theta - t_\phi \left(s_\theta + \frac{\delta s_{\theta\phi}}{\ell} \right) \right) \omega_r \rho - \delta s_{\theta\phi} \omega_s \tag{5.3}$$

[1] Le velocità delle ruote posteriori (motrici) devono essere diverse e dipendenti dall'angolo di sterzata, per evitare di perdere aderenza in curva. Per il controllo è stato sufficiente considerare l'approssimazione *"bicycle"* e ω_r come la velocità di rotazione di una ruota immaginaria posizionata all'incrocio fra l'asse del robot e l'asse delle ruote posteriori.

[2] Tali coordinate si possono facilmente calcolare mediante le relazioni dirette:

$$z_1 = x + \ell c_\theta + \delta c_{\theta\phi} \tag{5.1}$$
$$z_2 = y + \ell s_\theta + \delta s_{\theta\phi} \tag{5.2}$$

dove x e y sono le coordinate del robot.

$$\dot{z}_2 = \left(s_\theta + t_\phi \left(c_\theta + \frac{\delta c_{\theta\phi}}{\ell} \right) \right) \omega_r \rho + \delta c_{\theta\phi} \omega_s \tag{5.4}$$

ed un altro sottoblocco che esporta nel workspace le coordinate del robot (x,y) che servono non tanto per il controllo quanto per la visualizzazione dei grafici 2D e VRML. Il blocco `carlike` restituisce anche la velocità angolare di ciascuna delle ruote posteriori, dati che potrebbero risultare utili nel caso in cui si volessero simulare altre funzioni:

$$\omega_{sx} = (v/\rho)(1 - d(t_\phi/\ell)) \tag{5.5}$$
$$\omega_{dx} = (v/\rho)(1 + d(t_\phi/\ell)) \tag{5.6}$$

dove d ed ℓ sono rispettivamente la lunghezza del semiasse e del robot, ρ è il raggio delle ruote e v è la velocità del punto (x,y). Le derivate delle coordinate rappresentative del robot vengono fornite al sistema adattativo di stima.

Un commento a parte va fatto sul sottoblocco `saturation`, mostrato in figura 5.3 a pagina 33, che pone dei vincoli su:

- l'angolo massimo di sterzata ($|\phi| \leq \phi_{max}$) (vedi pag. 63)
- la velocità massima ottenibile dall'attuatore di sterzata e da quello di trazione ($|\omega_s| \leq \omega_{s,max}, |\omega_r| \leq \omega_{r,max}$) (vedi pag. 63)
- la massima banda passante[3]

Tali vincoli introducono una forte non-linearità nel modello — per quanto lo rendano più fedele al vero comportamento del sistema — e difatti una scelta troppo restrittiva sui parametri rende il controllo maggiormente difficile se non addirittura impossibile.

5.3 Progettazione del Controllore

Il controllore, il cui schema è mostrato in figura 5.4 a pagina 34 ha due funzioni:

1. rendere lineare la relazione che intercorre fra le coordinate (z_1, z_2) del punto rappresentativo ed un ingresso ausiliario costruito ad hoc
2. effettuare un controllo lineare di tipo PD con derivata in feed-forward in modo tale da annullare l'errore di tracking

Le equazioni che descrivono il controllore sono state ricavate nella sezione 3.1.1.

[3] Questo vincolo viene modellizzato come un filtro di Bessel del sesto ordine, la cui frequenza di taglio (in rad/sec) è contenuta nella variabile `motorbw`

5.4 Codice Matlab

È stato scelto di usare, ove possibile, un codice *Matlab EMBEDDED* per rendere l'esplorazione del progetto più snella e veloce. Dove questo non è stato possibile, sono state create funzioni esterne attivate da uno script principale (`startupXXXX06.m`), il quale si occupa delle inizializzazioni, dell'avvio della simulazione in Simulink, della raccolta e gestione dei dati e della visualizzazione delle informazioni.

5.4.1 Generazione della traiettoria di riferimento

Nel file `startupXXXX06.m` è possibile scegliere il cammino e la legge oraria inserendo nella matrice `punti = [t, zd1, zd2]` i punti da raggiungere al tempo t. Tale matrice viene passata ad una funzione esterna, `interpola_traiettoria` (vedi pag. 63), la quale restituisce due spline. Tali spline vengono rese globalmente accessibili da un blocco Simulink, `s_traiettoria`, che ha come ingresso un clock e come uscite z_{d1} e z_{d2}. Compito del blocco è quello di dare il punto della traiettoria desiderata corrispondente all'attuale istante del tempo di simulazione.

5.4.2 Inizializzazioni e gestione dati

Nel file `startupXXXX06.m` si eseguono le inizializzazioni di tutte le variabili (tra cui le condizioni iniziali degli integratori presenti nelle equazioni dinamiche del robot) e si avvia la simulazione (vedi pag. 64). I dati vengono visualizzati mediante due figure: nella prima viene riportata, oltre all'andamento nel tempo delle coordinate del punto rappresentativo, dell'orientamento del robot e dell'angolo di sterzata, anche degli *snapshot* del piano di lavoro sovrapposti temporalmente, in modo tale da avere una visualizzazione più intuitiva ed immediata dei dati raccolti. Nella seconda figura, invece, è presente un'animazione 2D del robot che può essere successivamente esportata in un file avi, mediante il comando `movie2avi`.
E' anche possibile attivare una visualizzazione 3D mediante il modulo VRML (vedi sezione 5.6).

5.5 Sistema adattativo

Lo schema del sistema adattativo è mostrato in figura 5.5 a pagina 35. Il sistema è di tipo discreto: ad ogni intervallo di tempo, guidato dalla variabile `upd_time`, vengono letti gli ingressi e le uscite del blocco `carlike` ed il sistema, servendosi di uno stimatore basato sulla linearizzazione delle equazioni dinamiche secondo i parametri ρ ed ℓ, effettua una stima ricorsiva **bidimensionale** ottenuta mediante il procedimento descritto nella sezione 4.3.3 e fornisce i valori aggiornati ρ_c ed ℓ_c al controllore (vedi pag 65).

Per poter inserire un sistema discreto in un sistema continuo sono stati utilizzati dei campionatori e degli interpolatori *sample&hold*. Per evitare il loop algebrico causato dal fatto che per il calcolo dei parametri è necessario conoscere lo stato

del robot governato dall'azione di controllo che a sua volta necessita dei parametri provenienti dall'identificatore, sono stati inseriti degli switch, mostrati nella figure 5.6 e 5.7 a pagina 36, i quali in $t = 0$ presentano valori preimpostati al controllore e all'identificatore di sistema. Dopo un determinato lasso di tempo, il loop si chiude e la simulazione entra a regime.

5.6 Simulazione 3D

Per la simulazione 3D del *Car-Like* è stato utilizzato l'ambiente di sviluppo *Virtual Reality Toolbox (VRT)* di MATLAB. Il tool consente di connettere un mondo virtuale, definito attraverso il linguaggio *VRML*, con Simulink e MATLAB. Virtual Reality Toolbox consente di interagire virtualmente con i modelli di sistemi dinamici tempovarianti e fornisce un'interfaccia MATLAB flessibile verso mondi virtuali 3D. Dopo aver creato gli oggetti MATLAB ed averli associati con il mondo virtuale, è possibile controllarli attraverso l'uso di funzioni e metodi. VRT fornisce inoltre blocchi che permettono di collegare direttamente segnali dell'ambiente Simulink al mondo virtuale. Tale connessione permette di visualizzare il modello del *Car-Like* attraverso un'animazione tridimensionale. Il modello 3D dell'oggetto e quindi del mondo virtuale è stato creato attraverso l'editor VRML V-Realm Builder. Il blocco utilizzato in Simulink per il controllo del *Car-Like* è chiamato *VR-sink*. Questo permette di inviare dati al Virtual Reality Toolbox. Una volta incluso questo blocco nel diagramma Simulink, è possibile selezionare il mondo virtuale e connettervi i segnali Simulink. Il Virtual Reality Toolbox esegue automaticamente una scansione del mondo virtuale al fine di identificare i nodi che Simulink può gestire. Tutte le proprietà dei nodi vengono visualizzate attraverso una struttura gerarchica ad albero. Dopo aver scelto i nodi da controllare, Simulink aggiorna il blocco VR-sink con le relative porte di ingresso e di uscita.

Per quanto riguarda il modello del *Car-Like*, i segnali di comando sono i seguenti:

- L'angolo di sterzata ϕ
- La velocità angolare delle ruote (prelevata dal differenziale)
- La posizione del punto rappresentativo del robot

All'interno del blocco denominato `3D_model` troviamo il blocco `VR-sink` e i relativi sottoblocchi per il controllo dei nodi. In particolare si notano gli ingressi per il controllo dell'angolo di sterzo, della velocità angolare delle ruote, e del movimento del *Car-Like* all'interno del mondo virtuale 3D. Ogni sottoblocco di comando posizione ha la struttura riportata in figura 5.8 a pagina 36. In particolare vengono inviate le coordinate della posizione (x,y,z) in ingresso ad un Multiplexer, la cui uscita va a pilotare il relativo ingresso del VR-sink.

Per quanto riguarda il sottoblocco di comando angolare, questo ha la struttura riportata in figura 5.9. In questo caso l'uscita `angolo` è composta da quattro segnali, cioè i tre assi di riferimento e il comando angolare. Gli assi di riferimento servono a

specificare rispetto a quale asse avviene la rotazione. Nel caso di figura 5.9, essendo [010], la rotazione avviene intorno all'asse y. Per il controllo dell'angolo di sterzo, in particolare, la sequenza di comandi è rappresentata dalla variazione nel tempo dell'angolo ϕ prelevata dal modello del *Car-Like*. All'avvio della la simulazione, viene subito aperto il viewer 3D come mostrato in figura 5.10. A questo punto è possibile cambiare la visuale attraverso il pannello di navigazione. Nell'esempio qui riportato lo scopo è quello di assegnare una traiettoria che porti il *Car-Like* nel parcheggio del fast food come mostrato in figura 5.11, a pagina 38. È possibile disattivare la simulazione 3D impostando la variabile `en_3D` = 0 nel file di startup. Non appena viene avviata la simulazione (e se `en_3D` = 1) viene visualizzata l'animazione 3D[4] del *Car-Like* che segue la traiettoria assegnata al fine di eseguire il compito specificato.

5.6.1 V-Realm Builder 2.0

L'editor VRML utilizzato per creazione del modello 3D è il V-Realm Builder versione 2.0 della Ligos. In figura 5.12 viene riportata la schermata iniziale del modello del *Car-Like*. Come si può vedere dalla struttura ad albero sulla sinistra, il mondo è composto principalmente da un ambiente circostante (background) e dal modello del *Car-Like*. I due nodi `Transform` rappresentano gli oggetti 3D (alberi, fast food ecc.), mentre il nodo `Fog` rappresenta un effetto nebbia del mondo virtuale. La nostra attenzione va focalizzata sul nodo `carlike` e in particolare nel nodo `children` di quest'ultimo (figura 5.13). All'interno del nodo `children` notiamo la presenza di altri nodi e in particolare:

- I nodi `Point Light` che rappresentano quattro punti luce nel mondo virtuale
- Il nodo `axle` che rappresenta l'asse tra le ruote anteriori e quelle posteriori
- Il nodo `F_axle` che rappresenta l'asse delle ruote anteriori
- Il nodo `R_axle` che rappresenta l'asse delle ruote posteriori
- Il nodo `car` che rappresenta le carene del *Car-Like*

Per quanto riguarda i nodi `Point Light` ciò che viene fatto è posizionare i punti luce nel mondo virtuale attraverso la proprietà `location` come mostrato in figura 5.14. Quindi ogni punto luce è rappresentato da un vettore tridimensionale. Per quanto riguarda il nodo `axle`, all'interno di `children` troviamo un nodo `shape` all'interno del quale è presente un oggetto cilindrico che rappresenta proprio l'asse del *Car-Like*. Attraverso le proprietà `height` e `radius` è possibile impostare, rispettivamente, la lunghezza ed il raggio dell'asse. Il nodo `appearance` (figura 5.15), invece, consente di impostare il materiale del cilindro nonché il colore nell'ambiente 3D. All'interno

[4]Nel caso in cui non fosse possibile visualizzare il mondo virtuale, il problema può essere dovuto ai driver della scheda video in uso. Per risolvere il problema, è possibile tentare di ridurre l'accelerazione hardware in "Risoluzione problemi" presente nelle impostazioni della scheda video.

del nodo `F_axle` troviamo il nodo `FW_axle` che rappresenta l'asse delle ruote anteriori, il nodo `FRW_or` che rappresenta la ruota anteriore destra e il nodo `FLW_or` che rappresenta la ruota anteriore sinistra. La definizione del nodo `FW_axle` è simile a quella del nodo `axle` vista in precedenza (figura 5.16). La struttura del nodo relativo alla ruota anteriore sinistra è riportata qui di seguito. I discorsi riportati di seguito sono validi anche per il nodo relativo alla ruota anteriore destra. Si nota subito la presenza di tre nodi `Transform` in cascata: `FLW_or`, `FLW_steer` e `FLW_rot`. All'interno dell'ultimo `children` viene realizzata la struttura vera e propria della ruota in esame. In questo è presente un nodo `shape` che è un cilindro e quindi la ruota anteriore sinistra e otto nodi `Transform` all'interno dei quali sono presenti altrettanti nodi `shape` che rappresentano i raggi della ruota. Il motivo per cui si utilizzano tre nodi `Transform` in cascata è il seguente:

- Attraverso le proprietà `rotation` e `translation` del nodo `FLW_steer` viene controllato l'angolo di sterzo della ruota. Viene selezionato anche l'asse secondo il quale avviene la rotazione durante la simulazione 3D.
- Attraverso le proprietà `rotation` e `translation` del nodo `FLW_rot` (figura 5.17) viene selezionato l'asse secondo il quale avviene la rotazione durante la simulazione 3D dettata dalla sequenza di ingresso proveniente dal diagramma Simulink.
- Attraverso le proprietà `rotation` e `translation` del nodo `FLW_or` viene ruotata nonché traslata la ruota al fine di orientarla e posizionarla correttamente all'interno dell'ambiente virtuale 3D.

I nodi `Transform` relativi ai raggi della ruota servono a dare la giusta posizione nonché l'orientamento ai singoli raggi mediante le proprietà `rotation` e `translation`. Per quanto riguarda l'asse e le ruote posteriori vale la stessa descrizione presentata per la struttura anteriore del *Car-Like*.

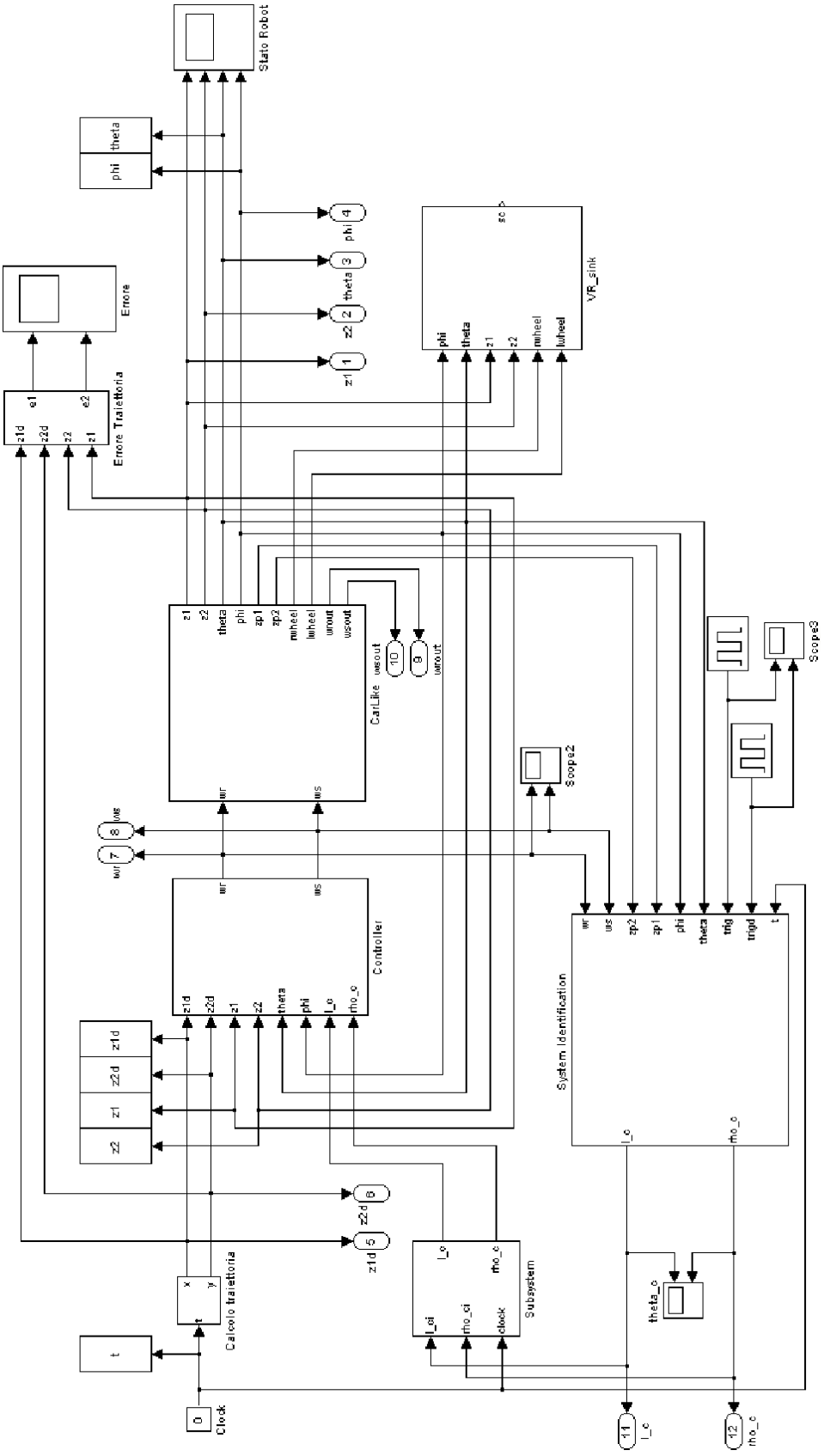

Figura 5.1: Schema Simulink

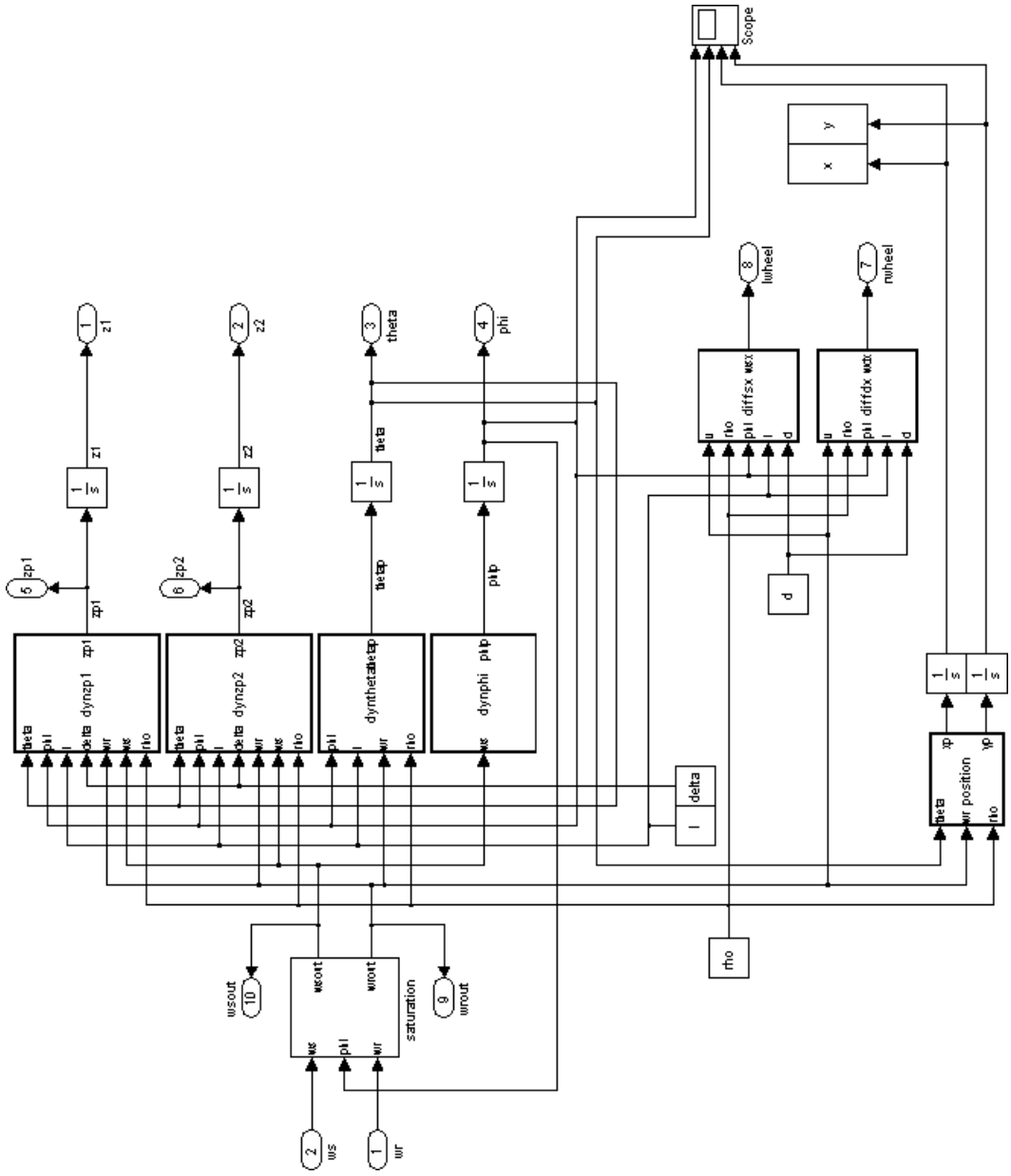

Figura 5.2: Schema Simulink: blocco `carlike`

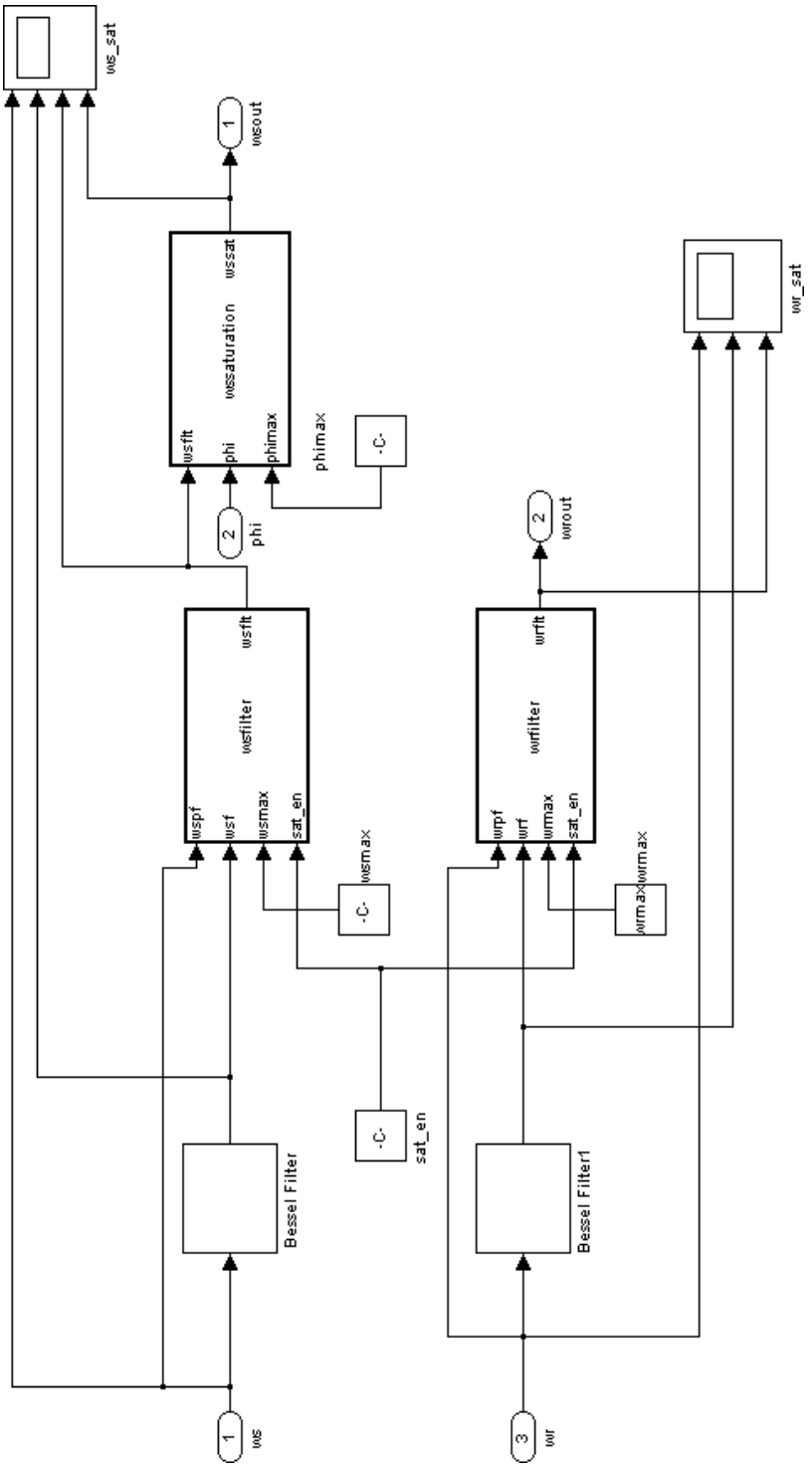

Figura 5.3: Schema Simulink: blocco `saturation`

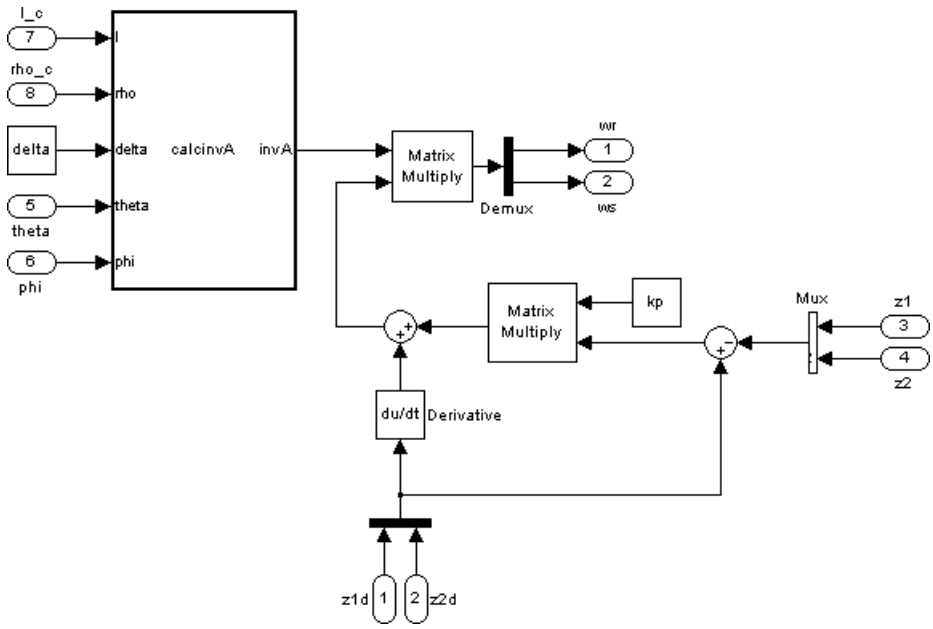

Figura 5.4: Schema Simulink: blocco `controller`

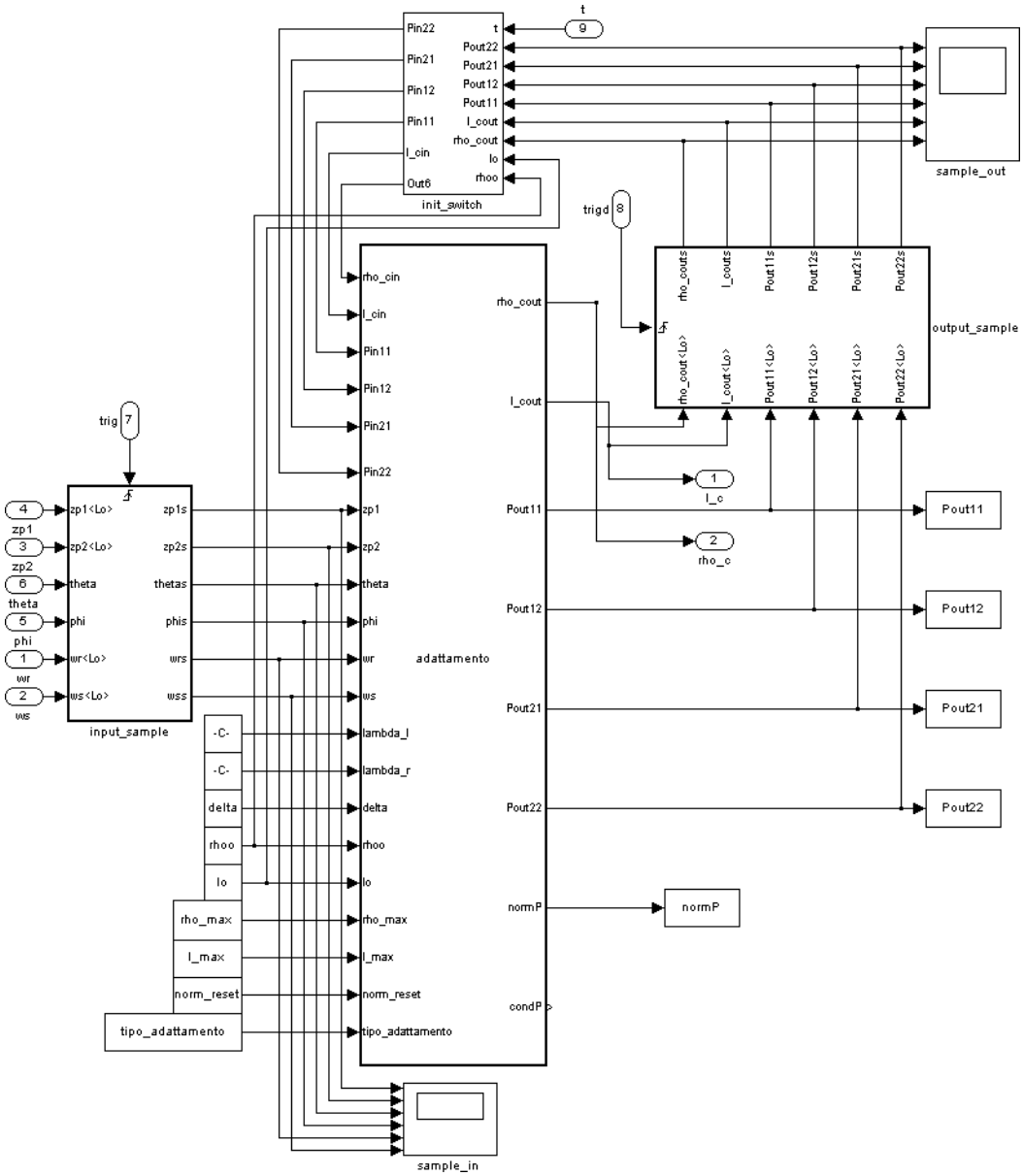

Figura 5.5: Schema Simulink: blocco `system_identification`

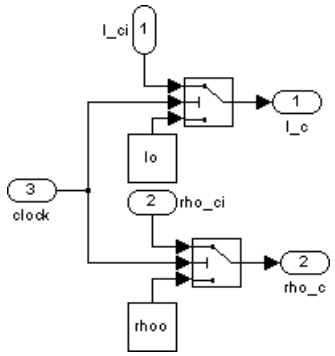

Figura 5.6: Schema Simulink: blocco `switch1`

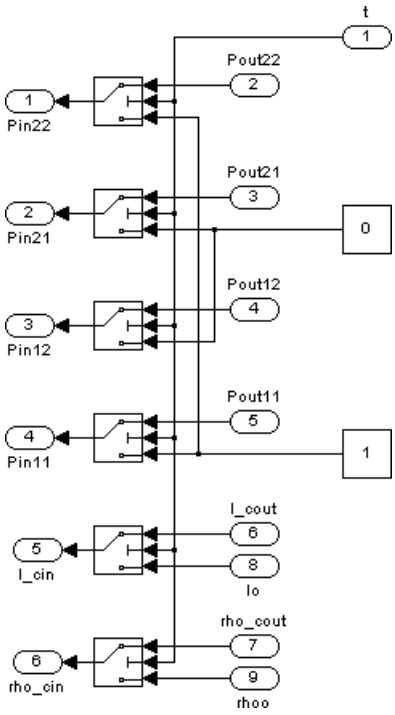

Figura 5.7: Schema Simulink: blocco `switch2`

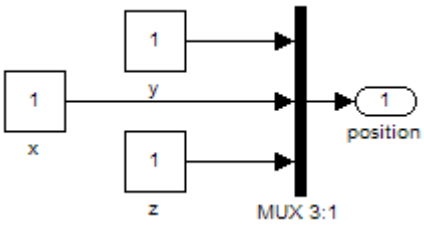

Figura 5.8: Sottoblocco comando posizione

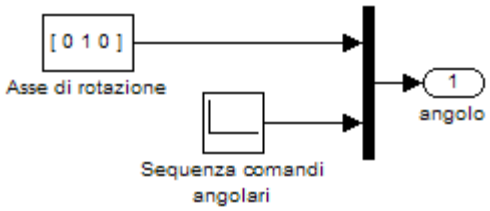

Figura 5.9: Sottoblocco comando angolare

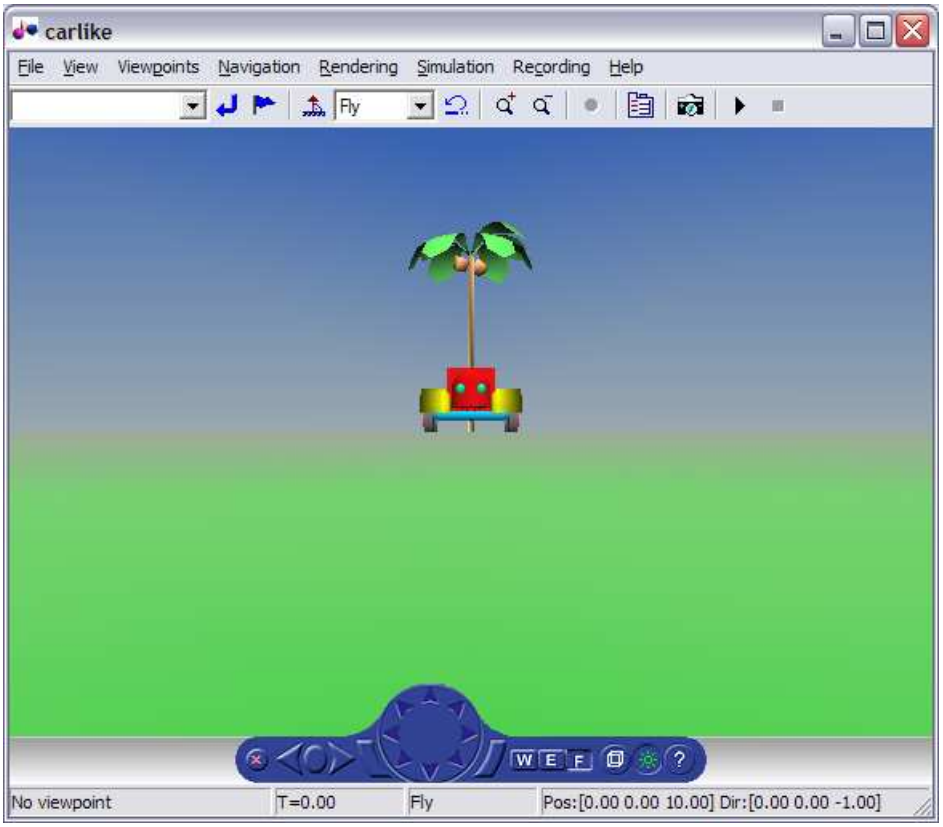

Figura 5.10: Visuale iniziale

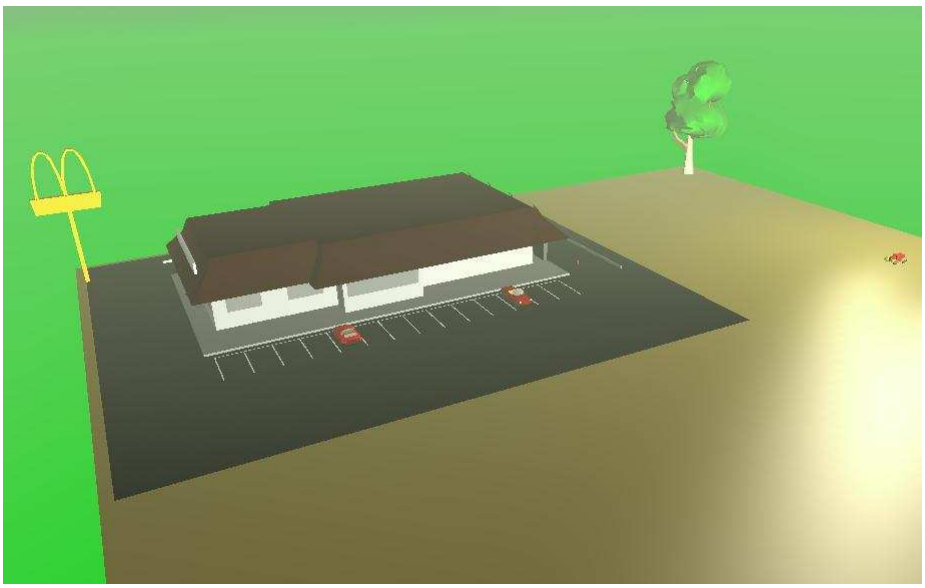

Figura 5.11: Schermata della simulazione attraverso il viewer 3D

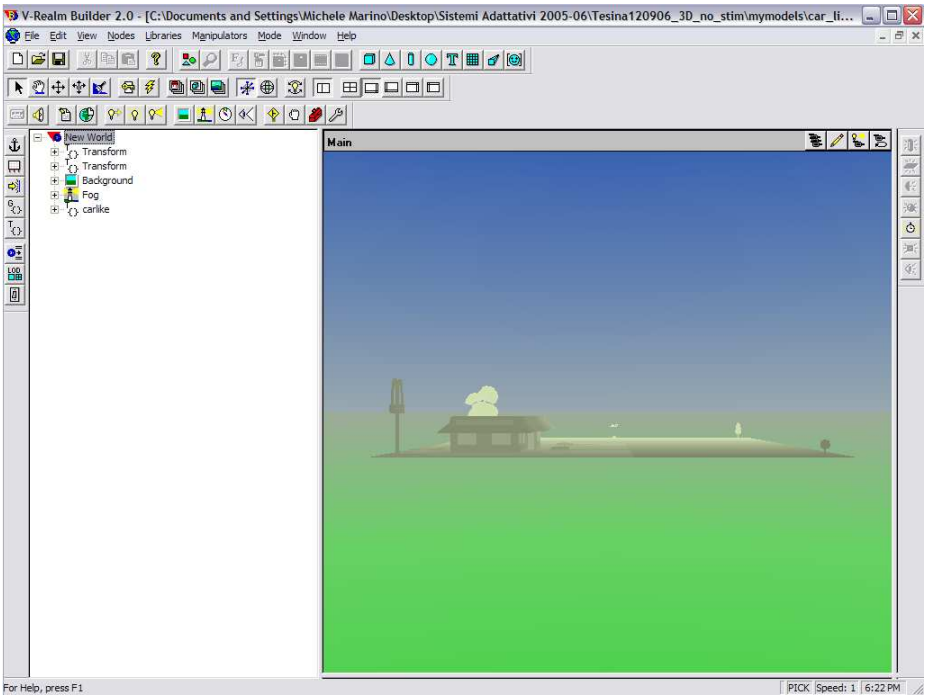

Figura 5.12: Schermata principale V-Realm Builder 2.0

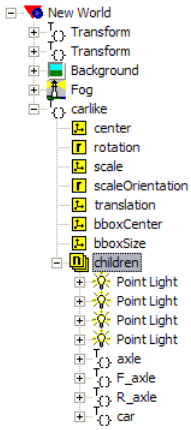

Figura 5.13: Struttura del nodo `carlike`

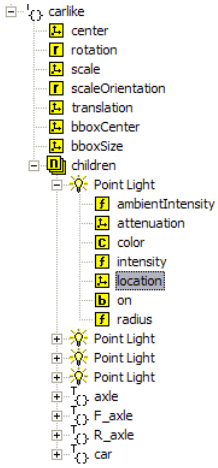

Figura 5.14: Struttura del nodo `Point Light`

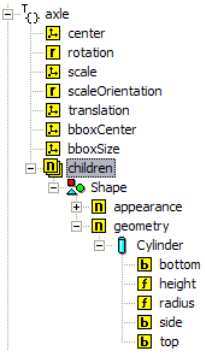

Figura 5.15: Struttura del nodo `axle`

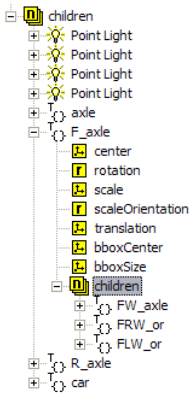

Figura 5.16: Struttura del nodo `FW_axle`

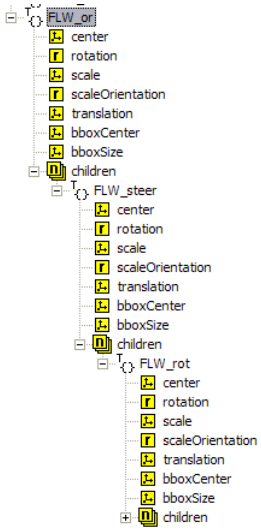

Figura 5.17: Struttura del nodo `FLW_or`

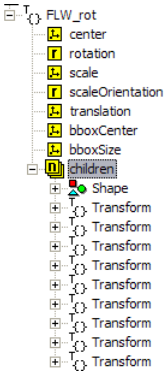

Figura 5.18: Struttura del nodo `children`

Capitolo 6

Test

La struttura dei seguenti test è stata ideata al fine di mettere in risalto gran parte degli aspetti del controllo nonché i margini di stabilità del controllo stesso. Il sistema, infatti, presenta non-linearità e dinamiche che non vengono considerate nel controllore. La feedback-linearizzazione agisce soltanto sulle equazioni differenziali (3.12) in cui ad esempio non compare la condizione sul massimo angolo di sterzata $|\phi| \leq \phi_{max} \leq \pi/2$. Ancora più stringente è il vincolo (non sempre attivo nelle nostre simulazioni) dato dalla non-idealità degli attuatori di trazione e di sterzo, modellati come un filtro di Bessel del sesto ordine e con banda passante definita nello script di inizializzazione.

Viene definita *traiettoria ammissibile* una qualsiasi traiettoria percorribile dal punto rappresentativo del robot. Nel caso di non-idealità degli attuatori si considera traiettoria ammissibile una traiettoria che non causa, a livello di comando[1], il superamento dei limiti meccanici del robot. In altre parole, una traiettoria è ammissibile se e solo se, per ogni t:

- per $\phi = \phi_{max}$, $\omega_{s,comando} \leq 0$ e per $\phi = \phi_{min} = -\phi_{max}$, $\omega_{s,comando} \geq 0$
- $|\omega_{r,comando}| \leq \omega_{r,max}$
- $|\omega_{s,comando}| \leq \omega_{s,max}$

Per mettere in risalto il comportamento del controllo e della identificazione di sistema in condizioni nominali, nei primi esperimenti gli attuatori vengono modellizzati in modo ideale, tenendo conto soltanto della saturazione dello sterzo. Successivamente vengono introdotte le dinamiche non modellate e la traiettoria di riferimento viene generata volutamente senza garanzia di ammissibilità — secondo la definizione data precedentemente — per sottoporre il controllore e l'identificatore a situazioni fuori dalla norma e per analizzare eventuali malfunzionamenti.

[1] per *comando* si intende l'uscita del controllore o, in modo equivalente, il segnale in velocità che viene dato agli attuatori. Per *forzamento*, invece, si intende il profilo effettivo in velocità seguito dagli attuatori. Nel caso in cui gli attuatori sono ideali comando e forzamento coincidono.

6.1 Primo esperimento

Nel primo esperimento è stata imposta al robot una traiettoria circolare di raggio pari a 5 metri centrata nell'origine, come si vede in figura 6.1. In tutti gli esperimenti, la posizione iniziale del punto rappresentativo del robot è $(\ell+\Delta, 0)$ mentre il retrotreno del robot si trova sull'origine degli assi.

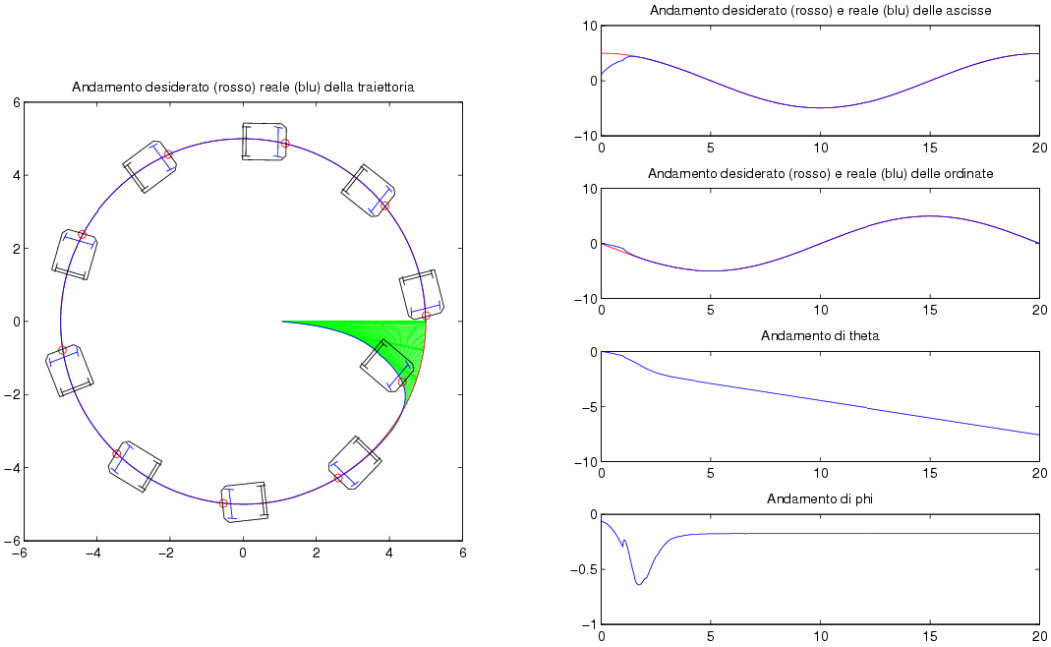

Figura 6.1: Primo esperimento - traiettoria e stato

La figura mostra la traiettoria desiderata (in rosso) e quella realmente percorsa (in blu); in verde viene espressa l'entità dell'errore di inseguimento, ovvero l'area fra le due traiettorie, mentre a destra vengono visualizzati i grafici che mostrano l'andamento nel tempo dello stato del robot. In questo caso l'"aggancio" è molto veloce e il robot segue perfettamente la traiettoria imposta annullando pertanto l'errore.

La figura 6.2 mostra l'andamento dei comandi di velocità dei due attuatori. Si nota una discontinuità nel profilo in $t = 1sec$, dovuta al fatto che in quell'istante il controllore riceve le due nuove stime dei parametri, molto diverse da quelle precedenti, come si evince dalla figura 6.3. Tale figura riguarda l'adattamento dei parametri ℓ e ρ. Come valori "veri" (incogniti) sono stati scelti $\ell = 0.87[m]$ e $\rho = 0.14[m]$ e la frequenza di aggiornamento dei parametri è stata fissata pari ad un secondo. La stima di ρ è la più semplice ed efficace. La mancata convergenza ai valori "veri" dei parametri non inficia il buon inseguimento della traiettoria: l'errore di posizione $\mathbf{e}$ e la sua derivata $\dot{\mathbf{e}}$ tendono a zero per $t \to \infty$ nonostante il vettore dei parametri $\beta_c = \begin{bmatrix} \rho_c & \ell_c \end{bmatrix}^\top$ si stabilizzi ad un valore $\beta^\star \neq \beta$. La convergenza sarebbe possibile

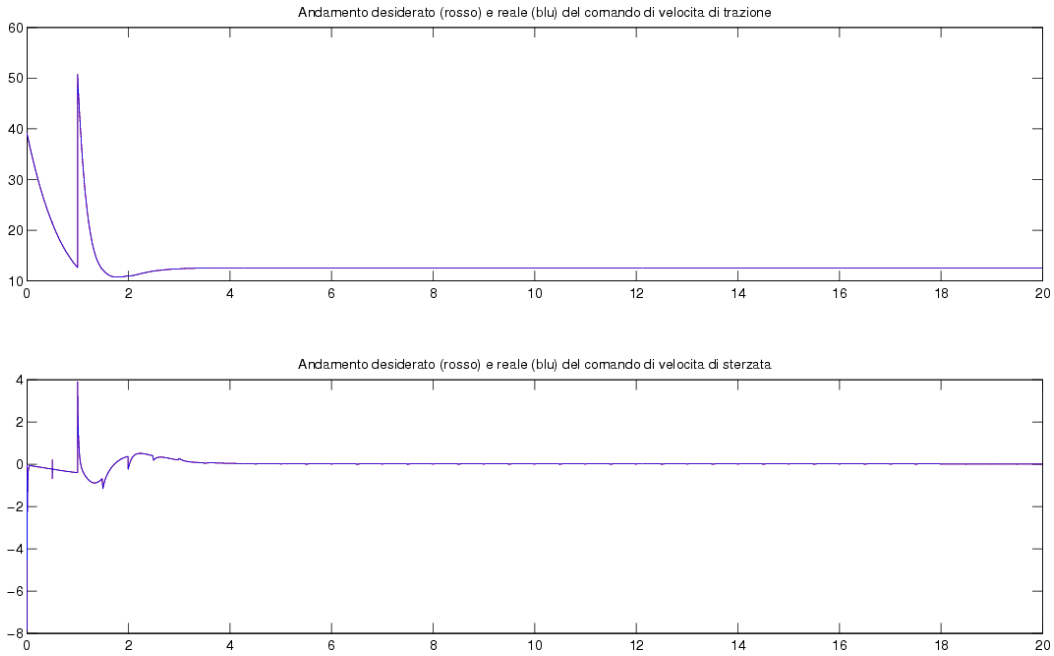

Figura 6.2: Primo esperimento - comandi di trazione e sterzo

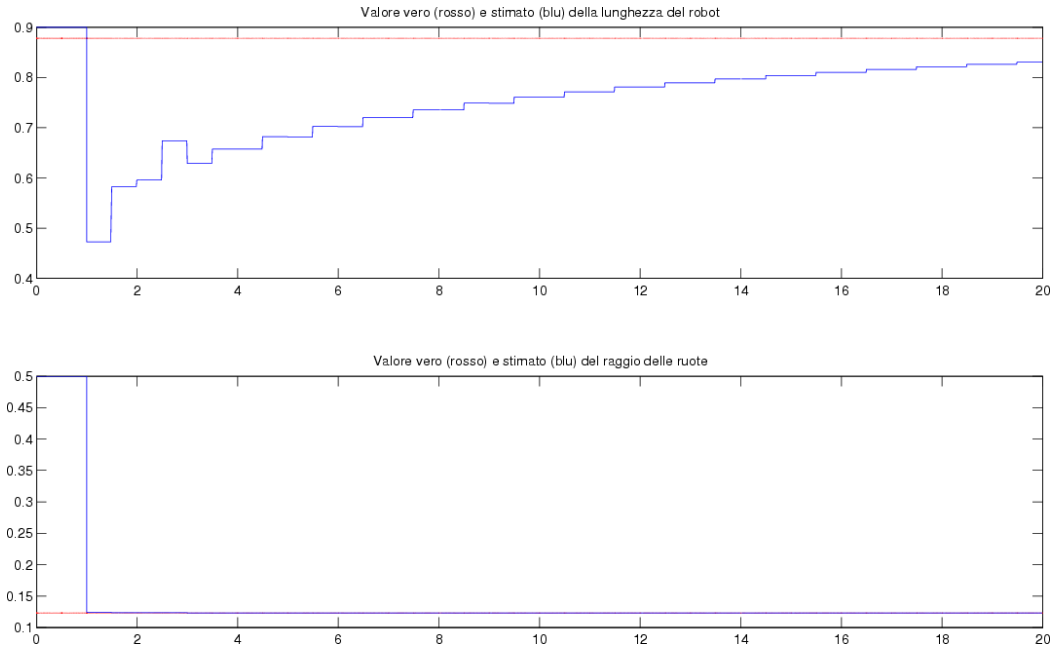

Figura 6.3: Primo esperimento - adattamento parametri

se la traiettoria desiderata fosse persistentemente eccitante[2].

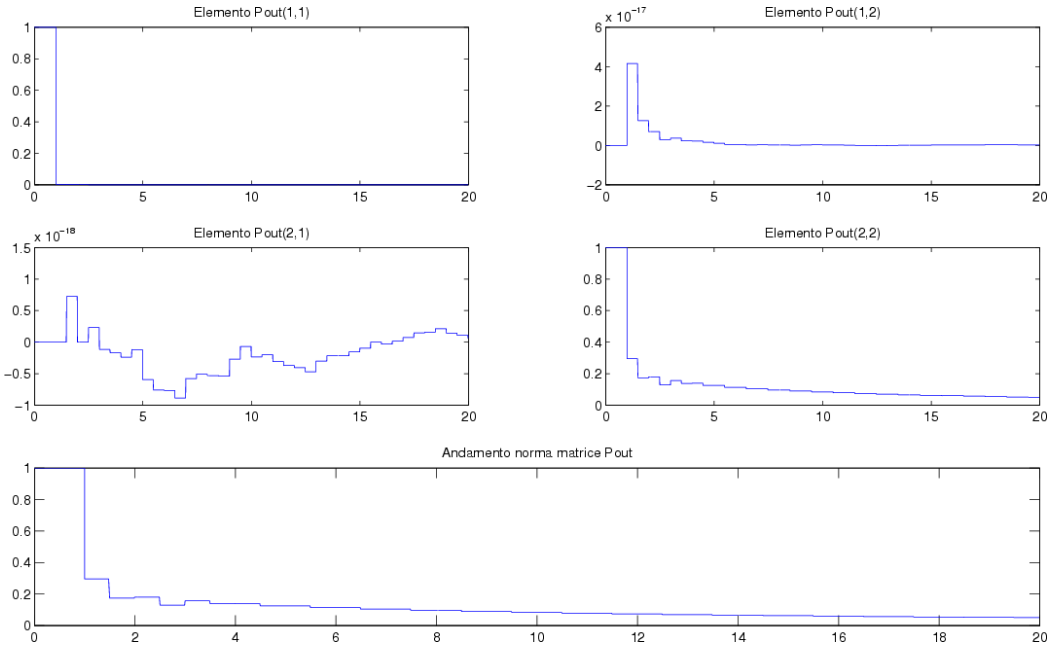

Figura 6.4: Primo esperimento - matrice $\mathbf{P}$

La figura 6.4, infine, mostra l'andamento della matrice $\mathbf{P}$ durante l'esperimento. Il valore iniziale è posto uguale a $\mathbf{I}$ e si nota che, mentre l'elemento p_{11}, riguardante la stima di ρ, tende a decrescere molto velocemente[3], non altrettanto si può dire per l'elemento p_{22}, che dipende in parte anche dal parametro ℓ, la cui stima è più difficoltosa.

[2] in un sistema lineare si ha una traiettoria *persistentemente eccitante* se il numero di componenti frequenziali è pari ad almeno il doppio del numero dei coefficienti dinamici incogniti; nel caso non-lineare, la verifica si può effettuare soltanto a posteriori, su un certo integrale temporale.

[3] questo significa che il sistema di identificazione sta procedendo ad un'accurata stima di ρ.

6.2 Secondo esperimento

Nel secondo esperimento al robot viene imposta una traiettoria ad otto, 5 metri di estensione lungo x e $2,5$ metri lungo y. La figura 6.5 mostra i risultati del test.

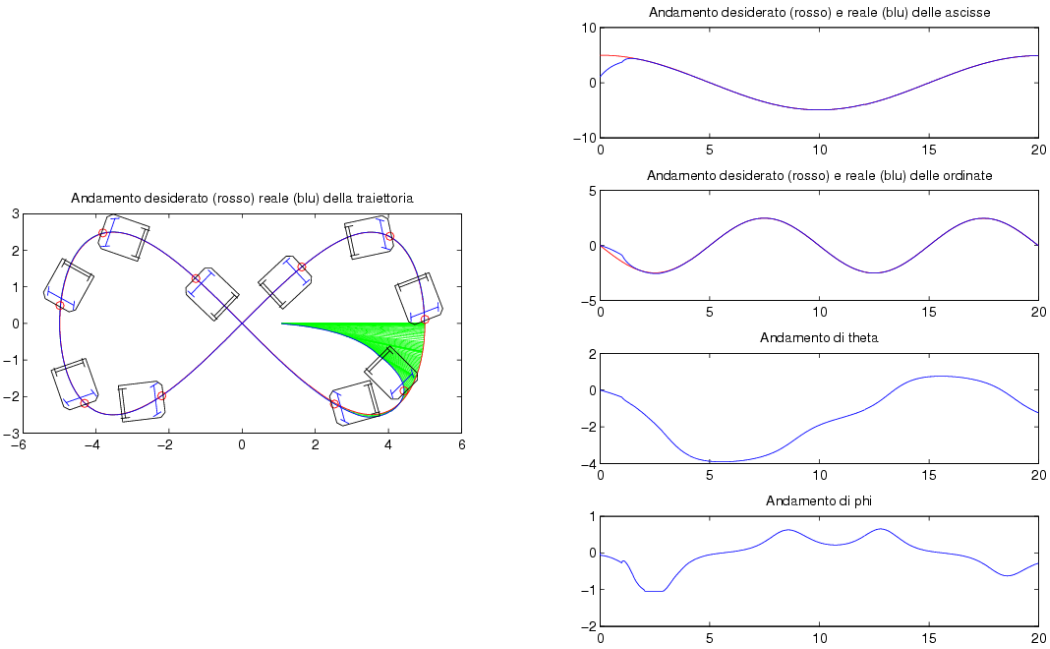

Figura 6.5: Secondo esperimento - traiettoria e stato

Il robot riesce ad agganciare la traiettoria ma nel finale della curva si scontra con la saturazione dello sterzo (in tutti gli esperimenti $\phi_{max} = \pi/3$). Per qualche istante l'errore aumenta, tuttavia nel tratto rettilineo della traiettoria il controllore riesce a recuperare la giusta direzione e l'aggancio riesce.

Il fenomeno della saturazione dello sterzo si nota con più facilità in figura 6.6: per un breve periodo di tempo il comando di velocità dell'attuatore dello sterzo si discosta dall'andamento reale. Quando lo sterzo satura, il controllore non se ne accorge e, misurando l'aumento dell'errore di traiettoria, tenta di aumentare ulteriormente la sterzata.

In figura 6.7 si nota come questa volta l'adattamento di ℓ sia più efficace. L'esperimento si interrompe quando il valore non è ancora assestato. Lo si vede da figura 6.8, in cui, nonostante il valore dell'elemento p_{22} decresca molto più velocemente rispetto al precedente test, si mantiene sempre ad un valore tale da permettere una certa variazione della stima di ℓ.

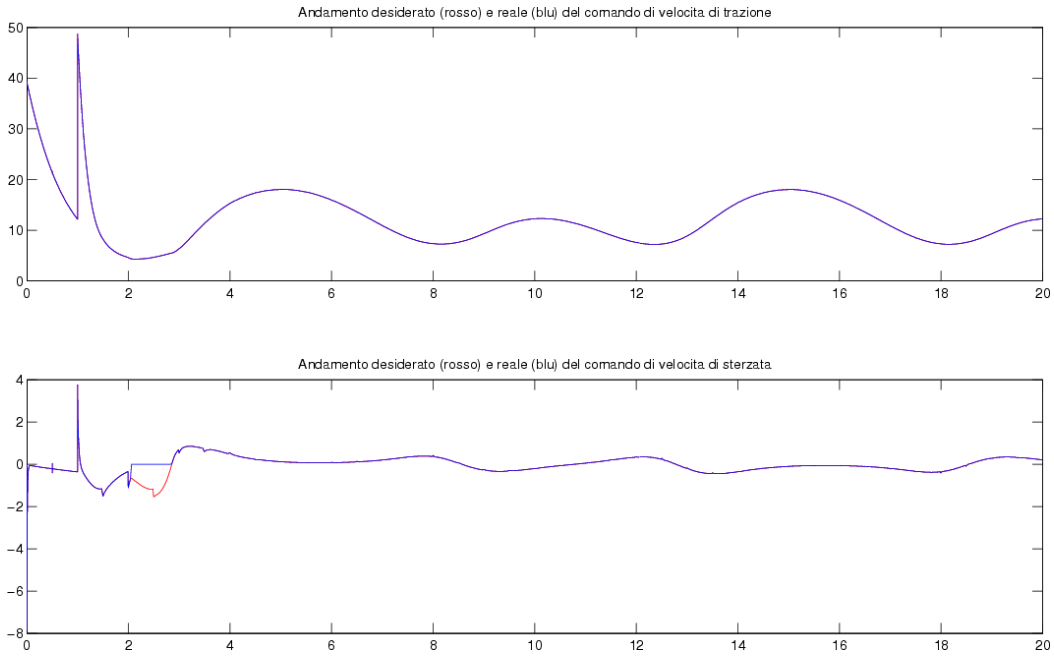

Figura 6.6: Secondo esperimento - comandi di trazione e sterzo

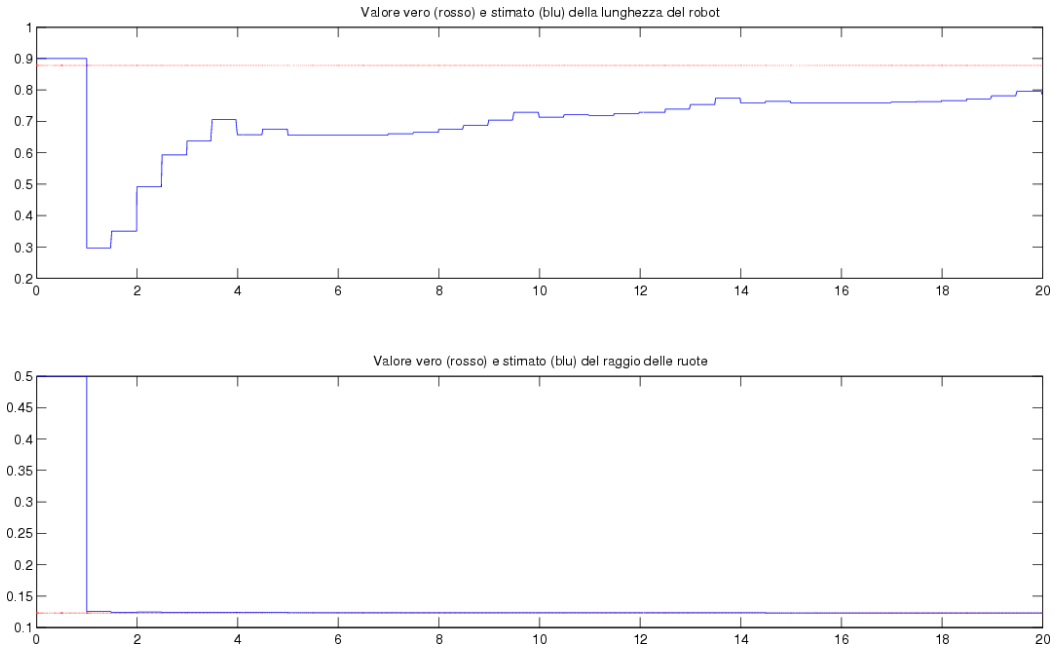

Figura 6.7: Secondo esperimento - adattamento parametri

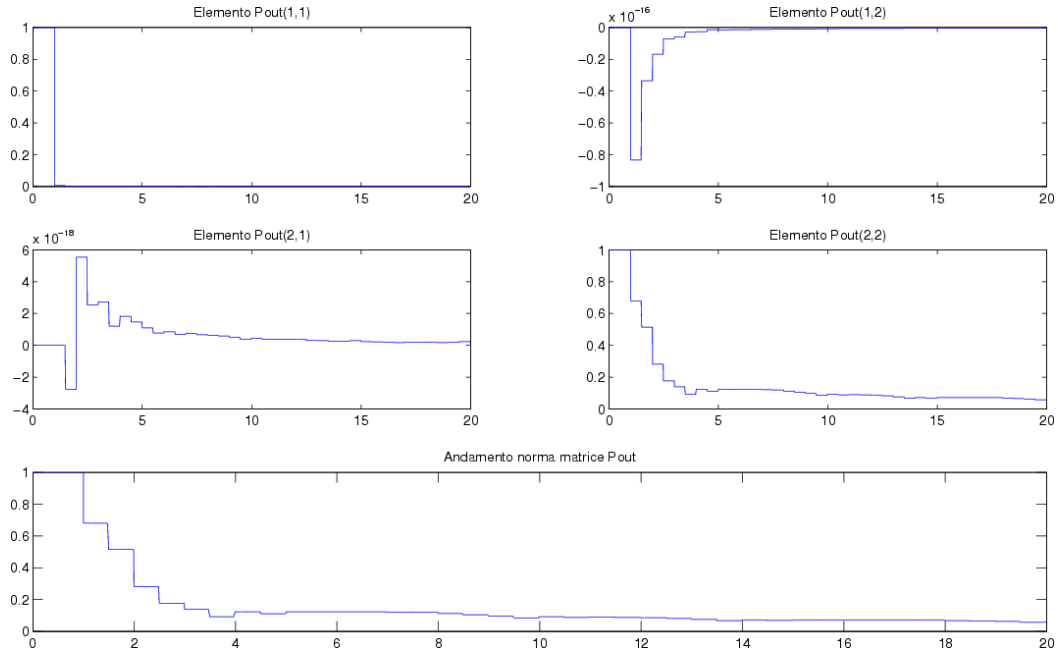

Figura 6.8: Secondo esperimento - matrice **P**

6.3 Terzo esperimento

Nel terzo esperimento si dimostra quanto la non-linearità dello sterzo venga trattata in modo efficace anche nel caso in cui la sua entità non sia trascurabile. La traiettoria imposta al robot presenta una curva molto stretta e successivamente un tratto a curvatura larga per permettere una buona visualizzazione del recupero (figura 6.9).

Il recupero riesce in poco spazio: il robot riesce subito a ri-agganciare la traiettoria desiderata.

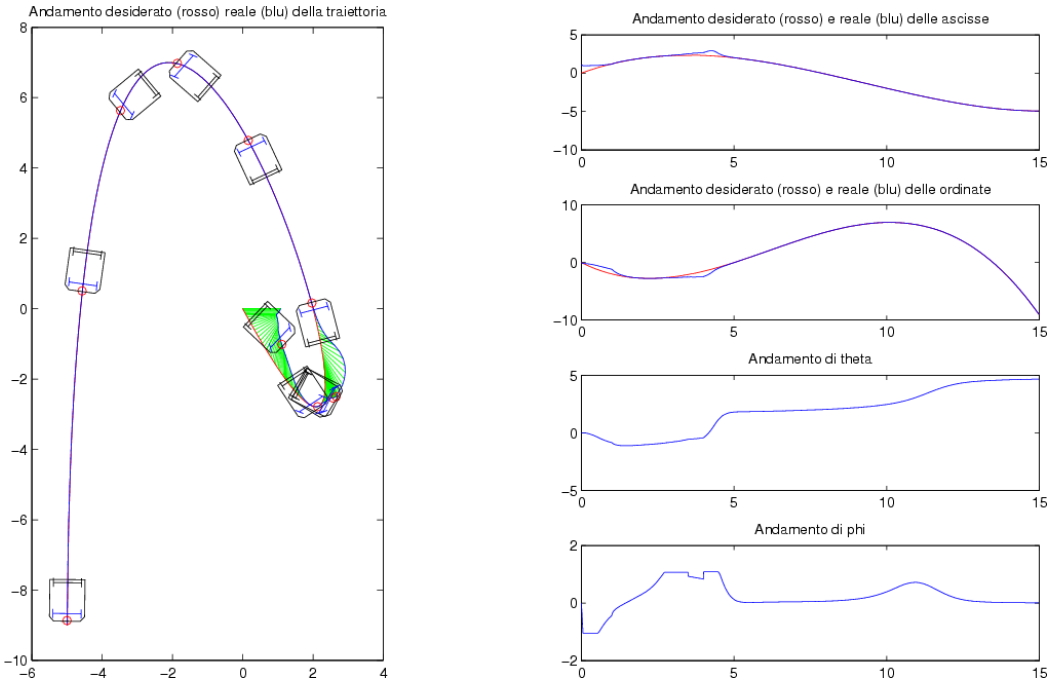

Figura 6.9: Terzo esperimento - traiettoria e stato

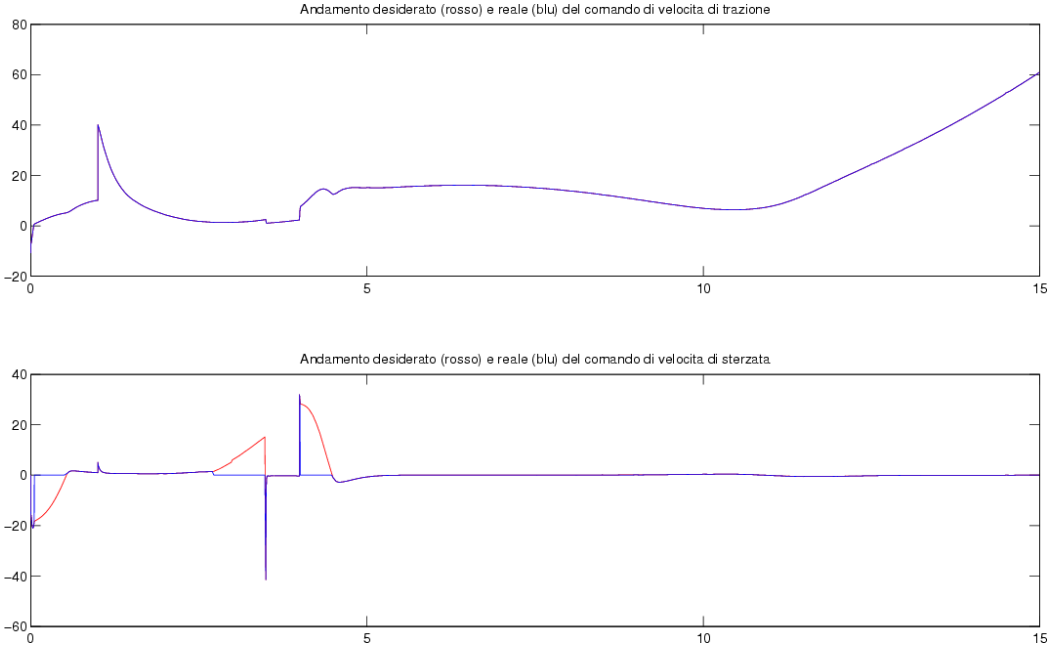

Figura 6.10: Terzo esperimento - comandi di trazione e sterzo

6.4 Quarto esperimento

Il quarto esperimento prevede l'attivazione delle dinamiche non modellate degli attuatori[4]. È stata scelta una traiettoria curva da $(5, 5)$ a $(0, 0)$.

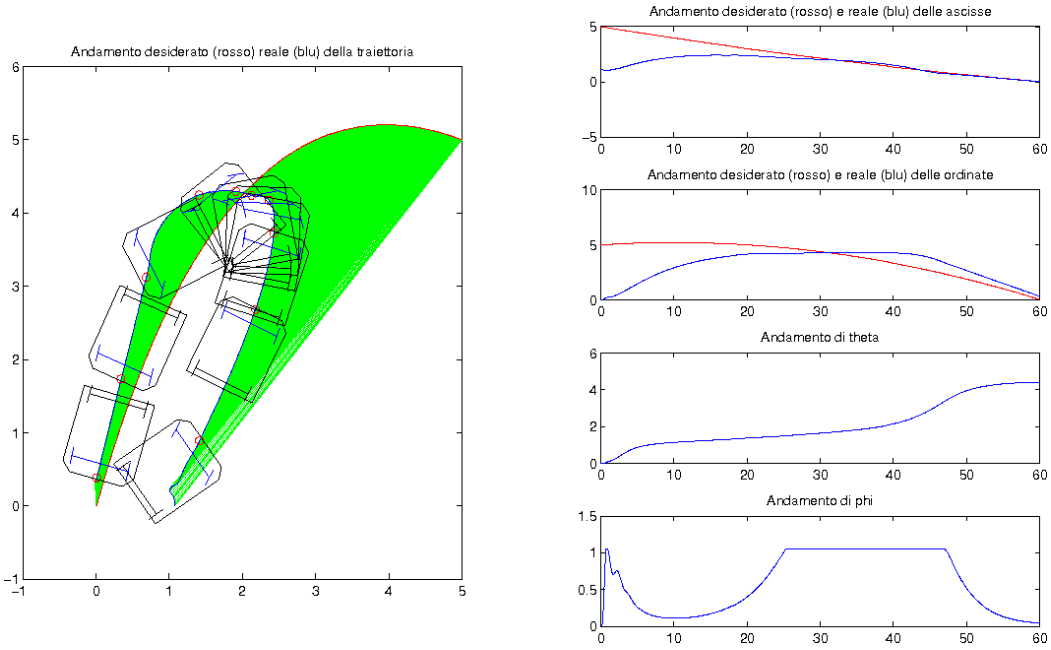

Figura 6.11: Quarto esperimento - traiettoria e stato

L'andamento della traiettoria è scelto appositamente per evidenziare il comportamento in aggancio. La traiettoria imposta al robot non è ammissibile, secondo la definizione data all'inizio del capitolo. Tuttavia il comportamento del robot è accettabile e l'aggancio avviene in circa 60 secondi.

In figura 6.12 si nota che la presenza della dinamica non modellata degli attuatori si rivela nell'eliminazione delle discontinuità e delle componenti ad alta frequenza del segnale di comando, come anche in un ritardo dovuto all'inerzia dei motori. Per quanto riguarda l'attuatore dello sterzo, anche in questo caso lo scostamento fra la velocità attesa (in rosso) e quella reale (in blu) diventa notevole non appena l'angolo di sterzata raggiunge il massimo valore ammissibile: il controllore forza il robot a sterzare ulteriormente, ma la velocità di rotazione che si ottiene è nulla. L'andamento reale torna a seguire quello di comando non appena l'angolo di sterzata imposto rientra nei limiti meccanici.

[4] nello specifico della simulazione, sono stati adottati i seguenti parametri:

```
wmax = 50; % velocità massima attuatore dello sterzo (in rad/sec)
wrmax = 50; % velocità massima attuatore di trazione (in rad/sec)
motorbw = 15.7; % banda passante attuatori (in rad/sec)
```

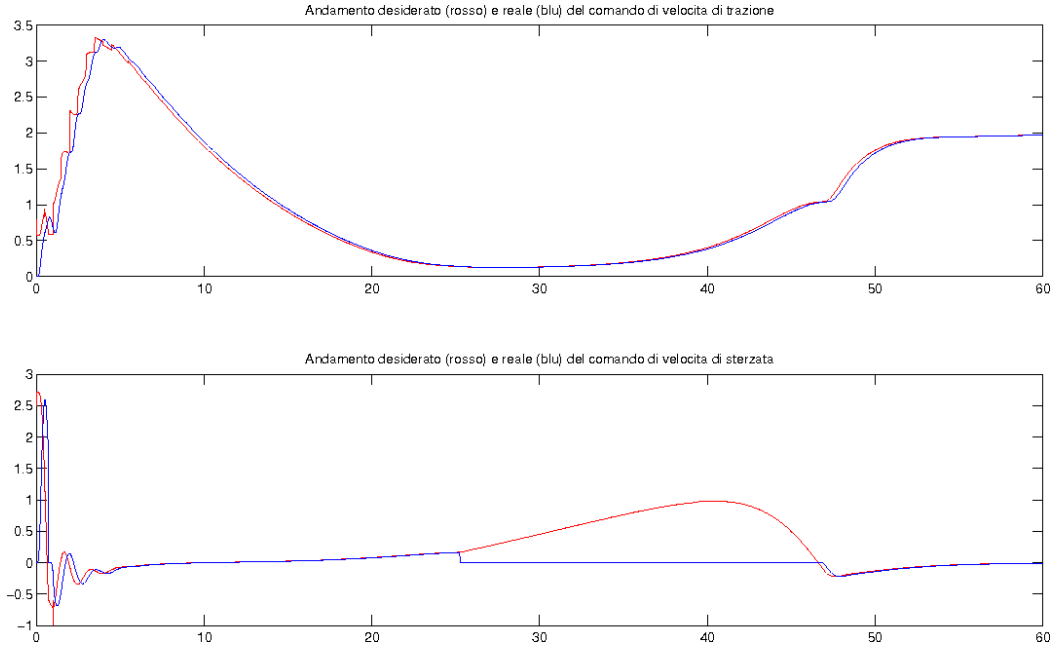

Figura 6.12: Quarto esperimento - comandi di trazione e sterzo

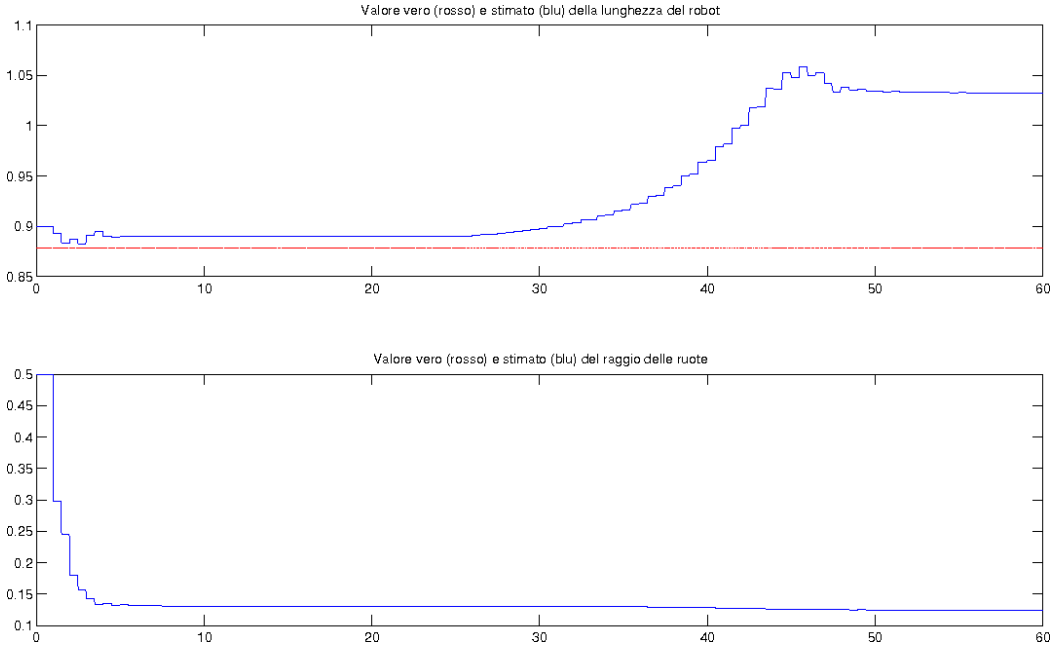

Figura 6.13: Quarto esperimento - adattamento parametri

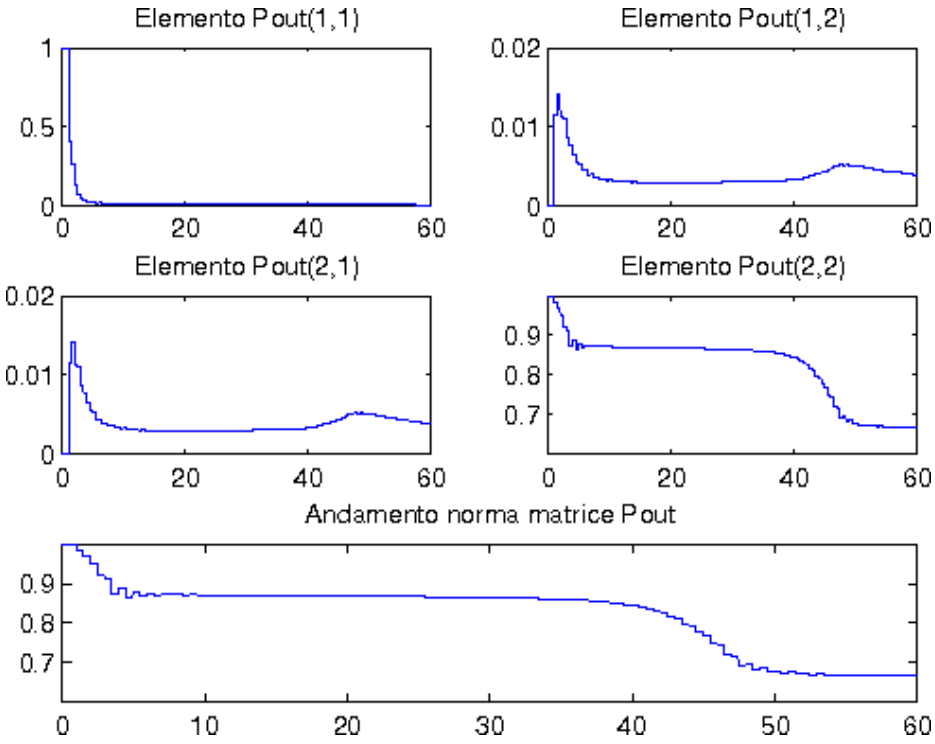

Figura 6.14: Quarto esperimento - matrice **P**

6.5 Quinto esperimento

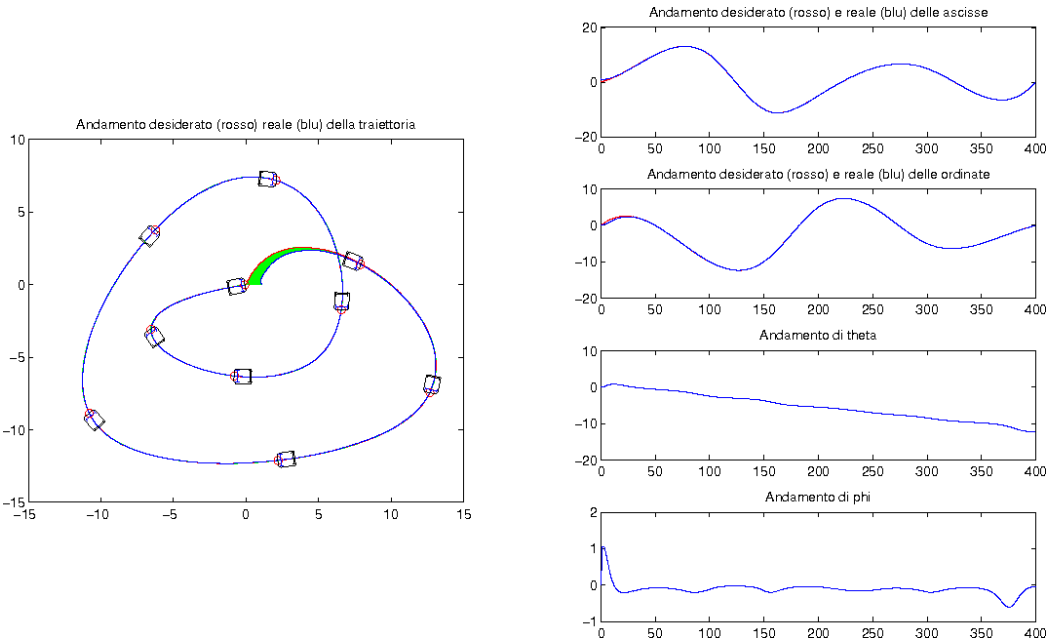

Figura 6.15: Quinto esperimento - traiettoria e stato

Nel quinto esperimento al robot è stata imposta una traiettoria estremamente lenta, applicabile ad esempio in compiti di sorveglianza. Il lieve errore iniziale, come mostrato in figura 6.15, viene prontamente recuperato e la "buona ammissibilità" della traiettoria — corroborata da vincoli temporali molto laschi — fa sì che il comportamento del robot sia ottimo. Si noti che in questo esperimento non subentra quasi mai la saturazione dello sterzo.

Anche per quanto riguarda il comportamento degli attuatori, a parte una sollecitazione iniziale in quello che governa lo sterzo, la figura 6.16 mostra che il profilo di velocità imposto dal controllore viene seguito dalla meccanica senza particolari difficoltà.

Come avviene negli esperimenti precedenti, anche in questo la stima della lunghezza del robot non è precisa (figura 6.17). L'andamento nel tempo della stima nonché il valore di finale dipendono, oltre che dalla traiettoria percorsa, anche dal valore iniziale attribuito al parametro.

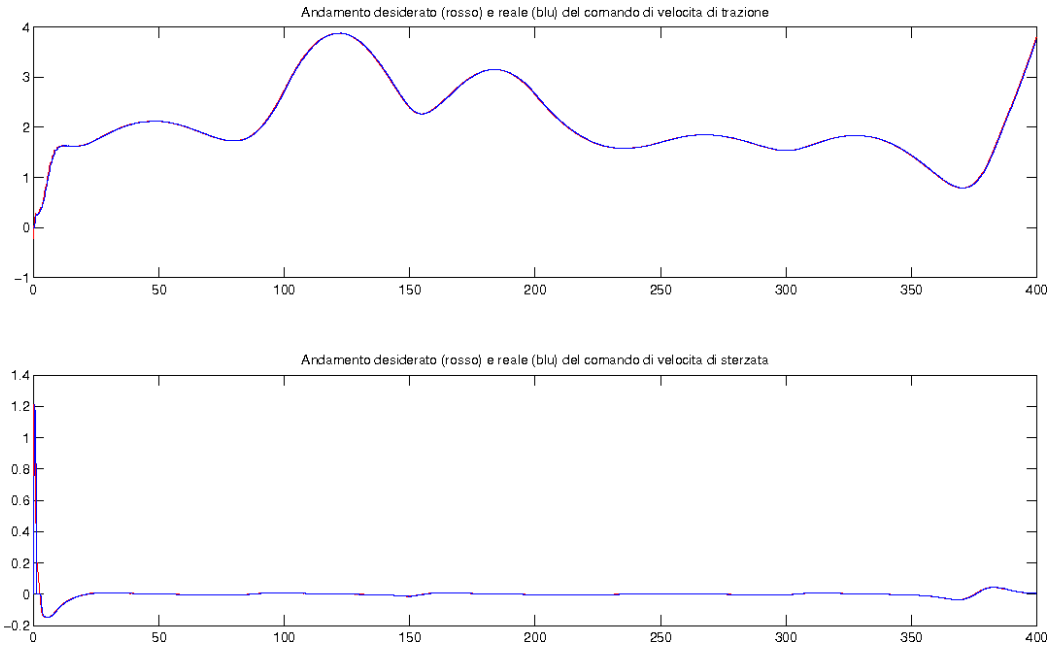

Figura 6.16: Quinto esperimento - comandi di trazione e sterzo

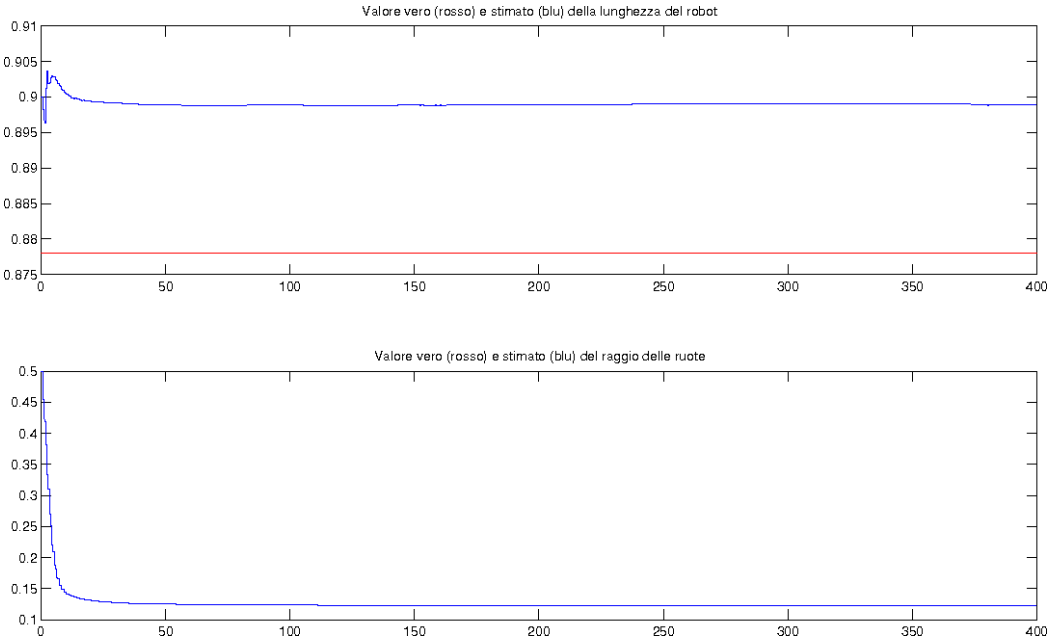

Figura 6.17: Quinto esperimento - adattamento parametri

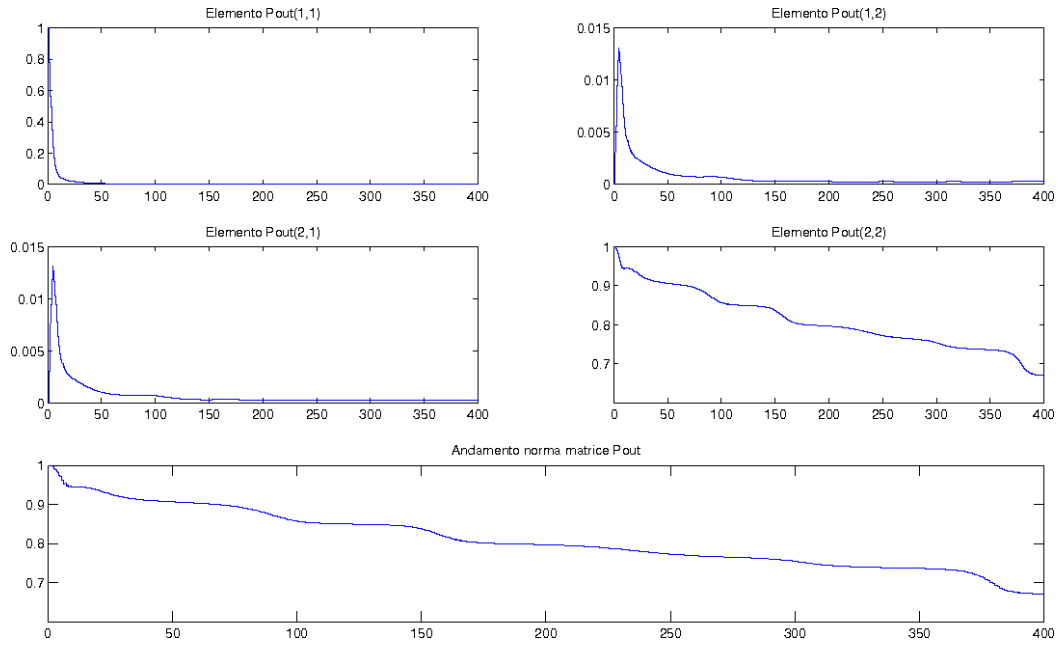

Figura 6.18: Quinto esperimento - matrice **P**

6.6 Sesto esperimento

Il sesto esperimento differisce dal quinto solo per quanto riguarda l'orizzonte temporale: la traiettoria richiesta deve essere percorsa in quaranta secondi contro i tre minuti e quaranta secondi del precedente esperimento.

I risultati, come dimostra la figura 6.19, sono del tutto diversi: il robot non solo fallisce nell'agganciare e seguire la traiettoria, ma entra in instabilità ed inizia a ruotare su se stesso in modo aperiodico. L'andamento di ϕ mostra chiaramente la crisi del controllore che si scontra con le non-linearità e le dinamiche non modellate. L'ammissibilità della traiettoria imposta è pesantemente violata.

Anche per quanto riguarda i comandi di trazione e sterzo (figura 6.20), sono palesi le difficoltà del controllo. Il comportamento peggiore si registra con l'attuatore dello sterzo, la cui velocità viene continuamente annullata dal limite nell'angolo massimo di sterzata. Il controllore, non "vedendo" la saturazione, aumenta ulteriormente l'entità della sterzata piuttosto che provare a compensare e tentare un'altra via per ripristinare la stabilità.

L'adattamento di ℓ (figura 6.21) fallisce e solo grazie alla *proiezione*[5] si riesce a mantenere tale valore compreso fra 0.001 e 5. Questa volta neppure la stima di ρ funziona, ed anche in questo caso entra in gioco la proiezione.

Come mostrato in figura 6.22, a dire il vero poco indicativa, la matrice **P** vie-

[5] si ricorda che per proiezione in un intervallo (x_{min}, x_{max}) di una variabile x si intende un controllo sulla variabile che pone $x = x_{min}$ se $x < x_{min}$ e $x = x_{max}$ nel caso in cui $x > x_{max}$.

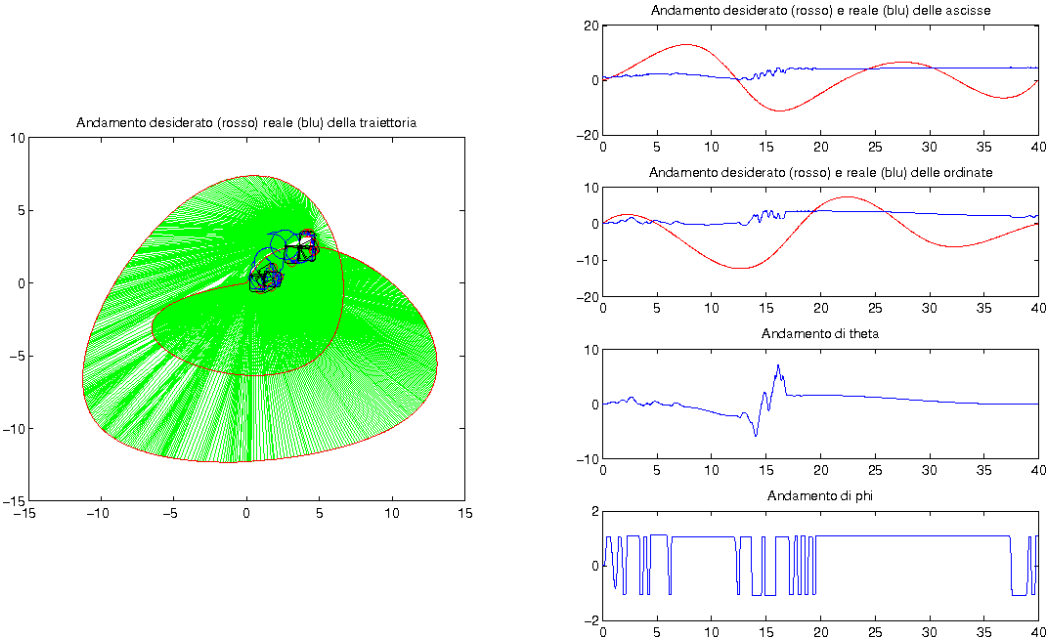

Figura 6.19: Sesto esperimento - traiettoria e stato

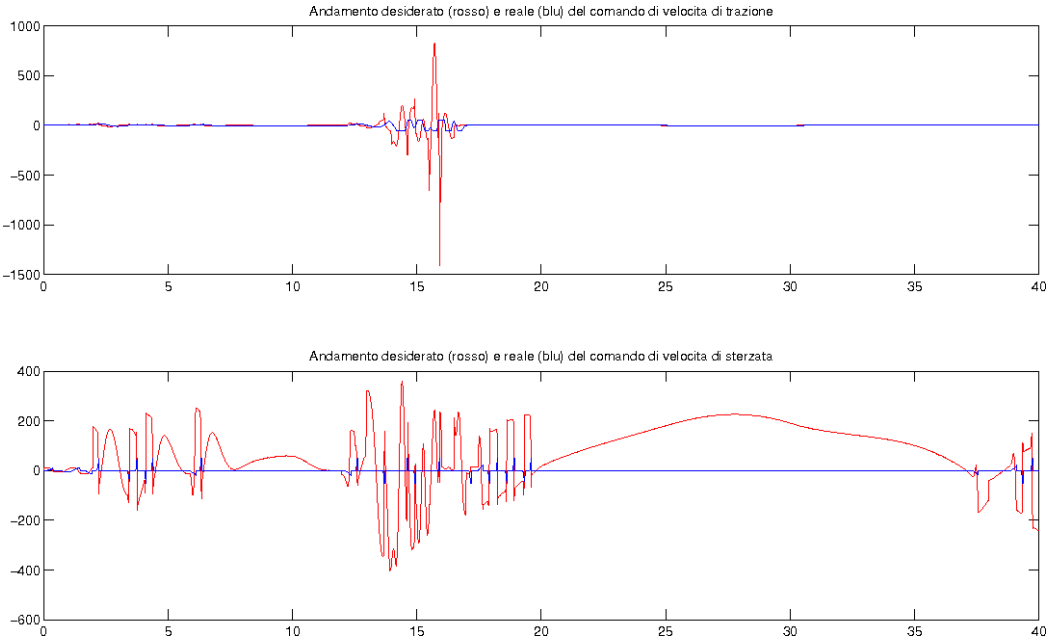

Figura 6.20: Sesto esperimento - comandi di trazione e sterzo

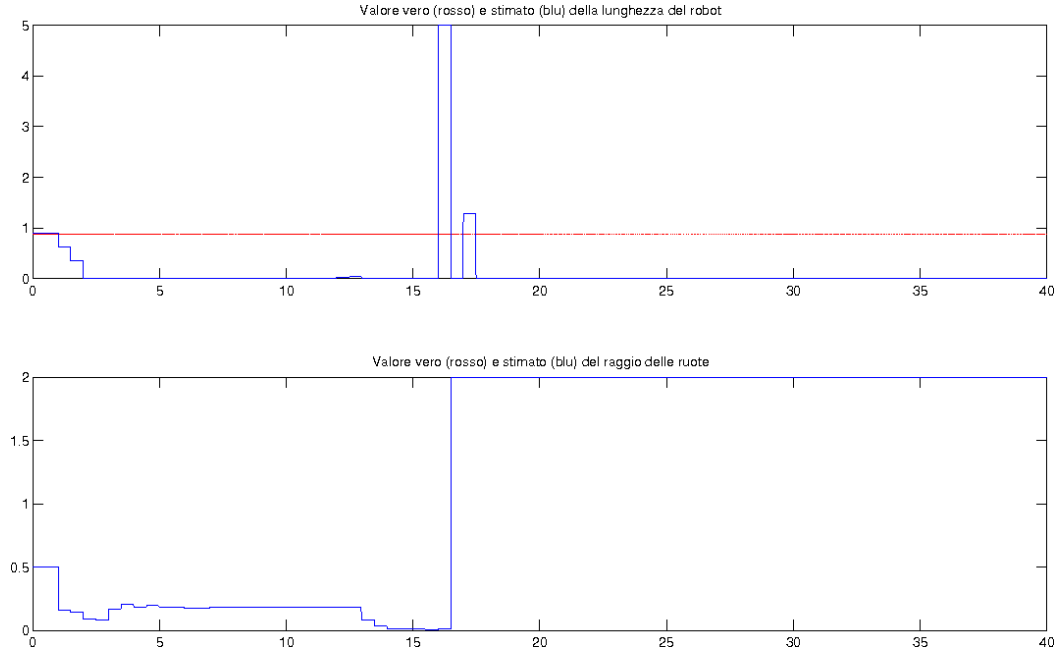

Figura 6.21: Sesto esperimento - adattamento parametri

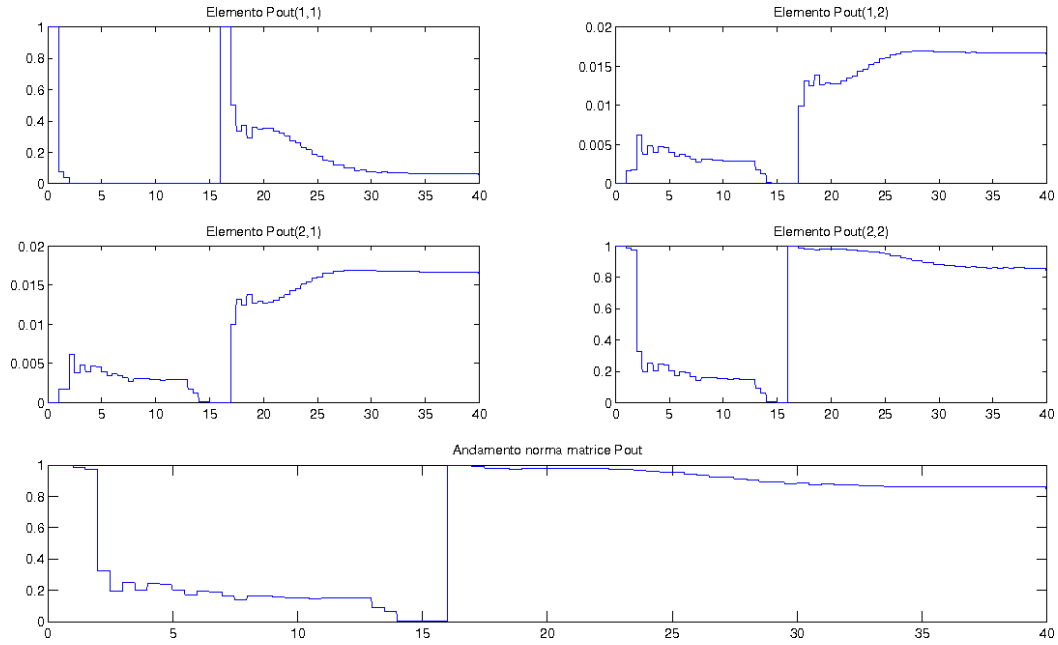

Figura 6.22: Sesto esperimento - matrice $\mathbf{P}$

ne reinizializzata al valore **I** ogniqualvolta la sua norma scende sotto una soglia (impostata a 10^{-4}).

6.7 Settimo esperimento

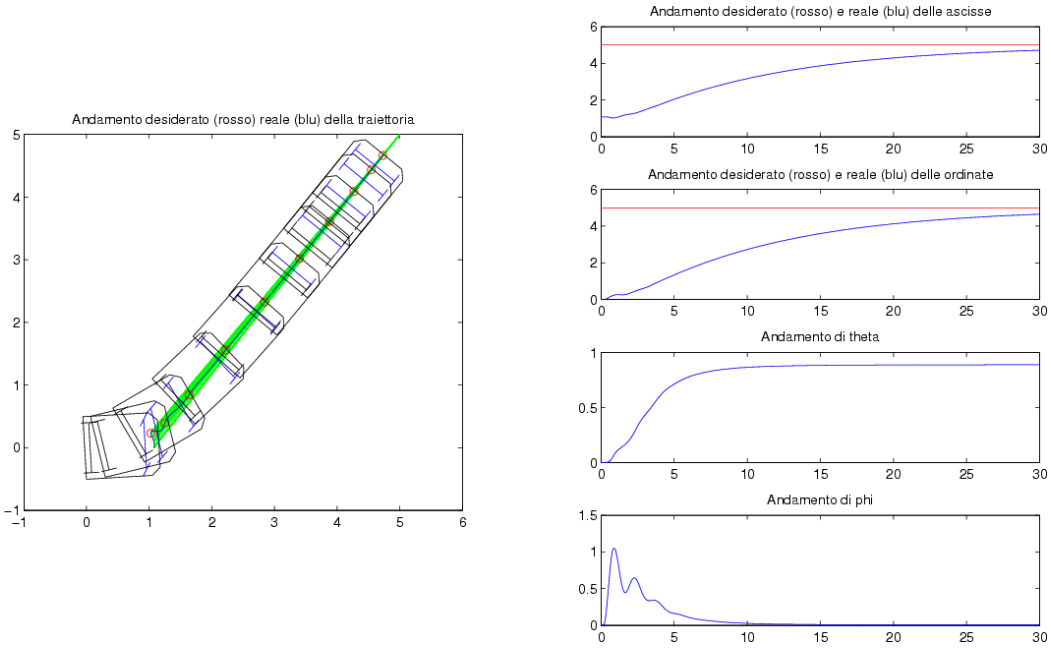

Figura 6.23: Settimo esperimento - traiettoria e stato

Il settimo esperimento consiste in un compito di posizionamento "mascherato" da *tracking*. La traiettoria da inseguire è degenerata in un punto, $(5, 5)$, che il robot deve raggiungere in trenta secondi. L'esperimento riesce ed il robot non dà segni di instabilità.

Questa volta l'adattamento funziona molto bene, anche se la convergenza ai valori "veri" non è perfetta.

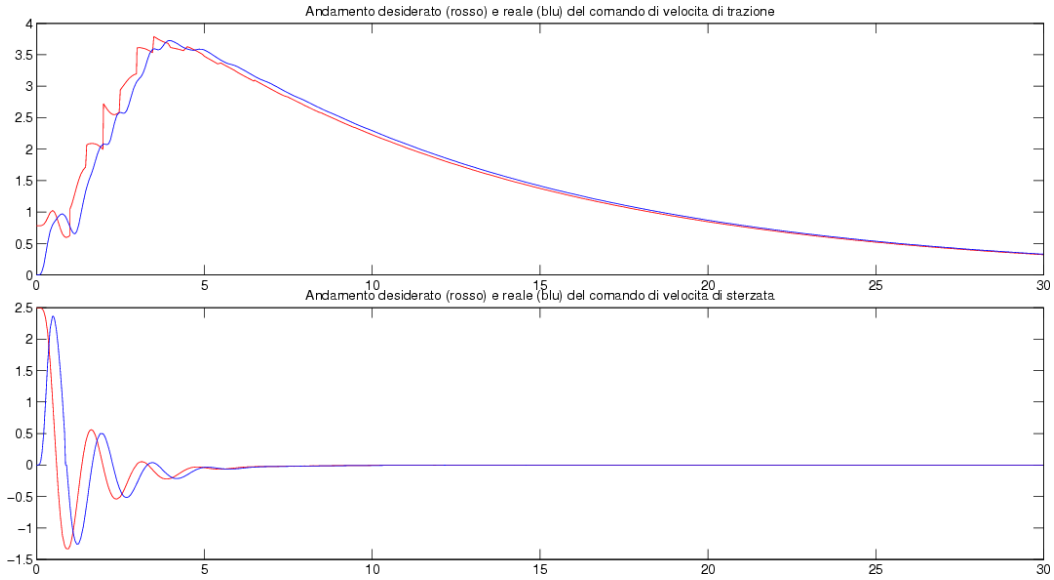

Figura 6.24: Settimo esperimento - comandi di trazione e sterzo

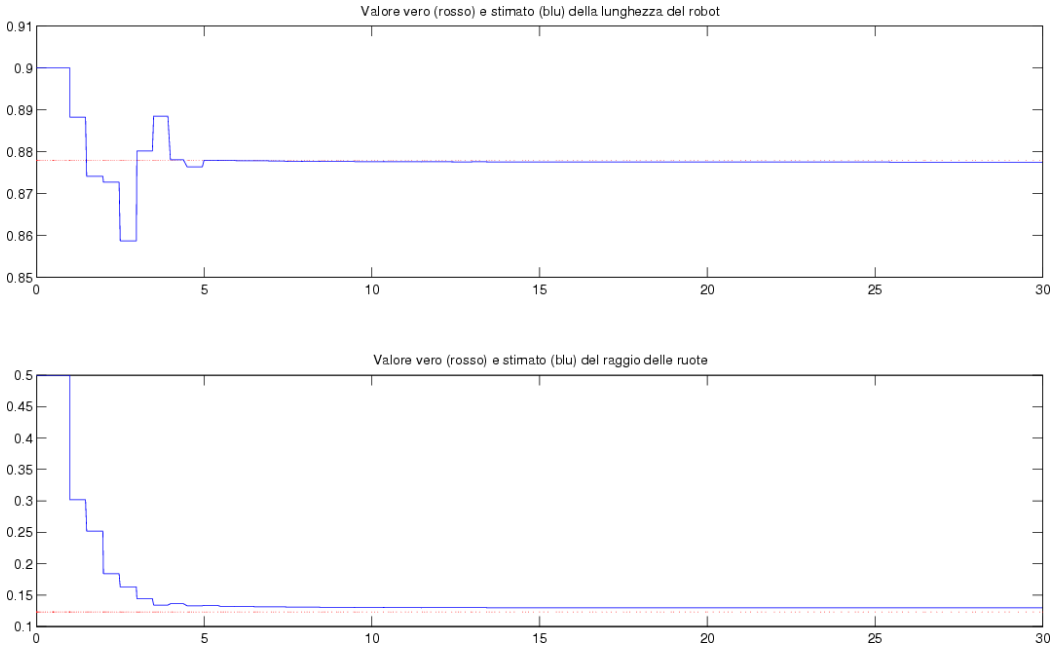

Figura 6.25: Settimo esperimento - adattamento parametri

6.8 Ottavo esperimento

L'ottavo ed ultimo esperimento mostra come i buoni risultati per un compito di posizionamento non sono assicurati. E' sufficiente infatti cambiare le coordinate di arrivo in $(10, 10)$, utilizzando lo stesso orizzonte temporale, per rendere il controllo infruttuoso.

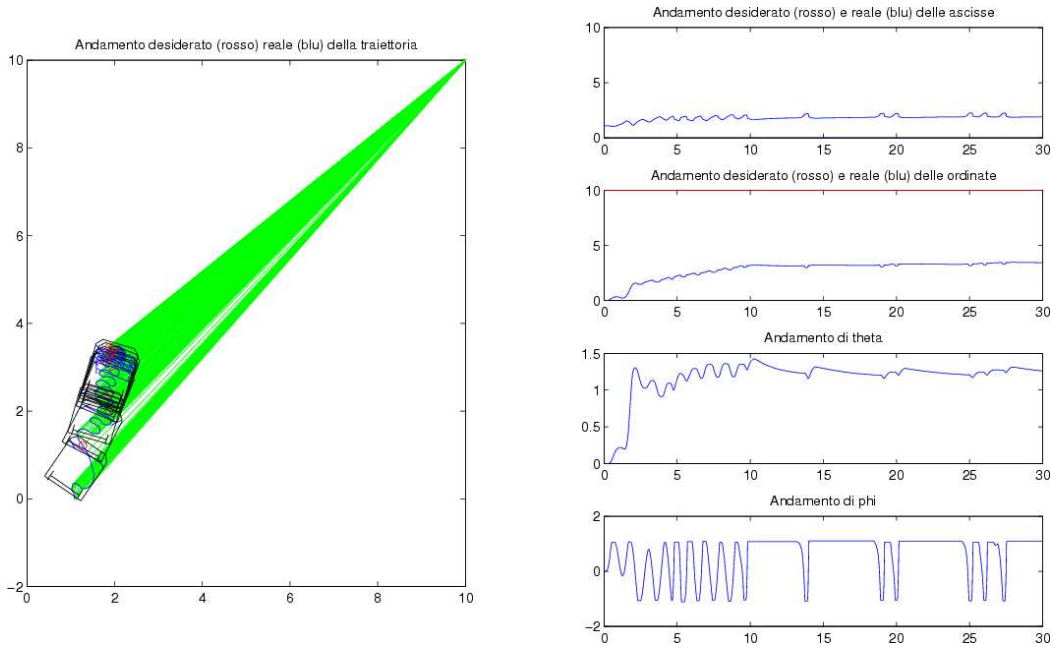

Figura 6.26: Ottavo esperimento - traiettoria e stato

Il robot, in questo caso, non riesce a dirigersi verso la posizione finale, ma inizia ad oscillare e, dopo trenta secondi, si trova ancora vicino al punto da cui è partito. La pessima prestazione descritta in figura 6.26 è da imputarsi al fatto che, a differenza dell'esperimento precedente, l'errore iniziale è maggiore e i comandi teorici sugli attuatori si scontrano con i comportamenti reali degli stessi.

In figura 6.27 tale situazione si rende ancor più chiara: il ritardo introdotto dalla non-idealità degli attuatori è alla base dell'inefficacia del controllo. Le prestazioni possono essere migliorate aumentando la banda passante dei filtri con cui gli attuatori sono stati modellati, riducendo il ritardo di risposta e rendendo conseguentemente l'attuatore più fedele.

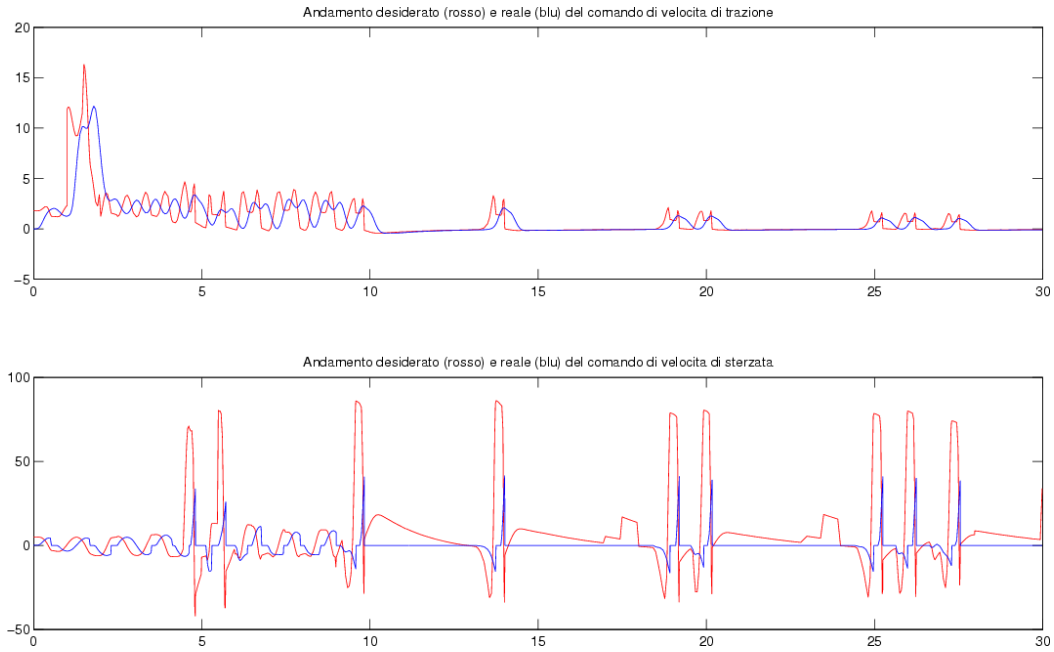

Figura 6.27: Ottavo esperimento - comandi di trazione e sterzo

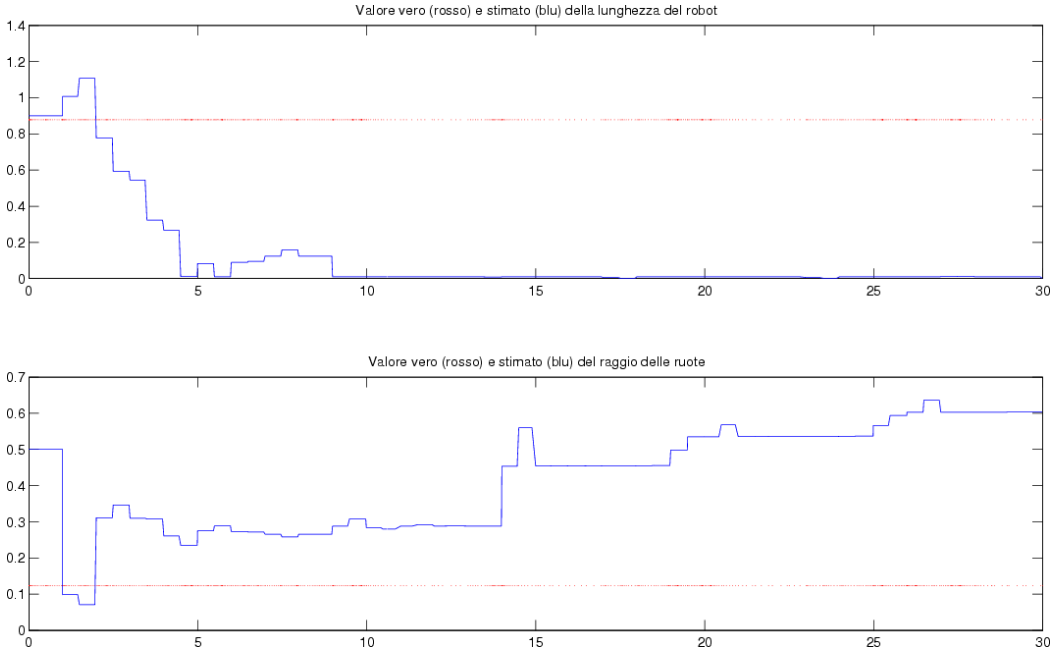

Figura 6.28: Ottavo esperimento - adattamento parametri

Capitolo 7

Conclusioni

Il problema del controllo di robot *Car-Like* si incontra in numerosi campi di applicazione: dalla domotica alle applicazioni di tipo spaziale, da compiti di *search&rescue* a compiti di sorveglianza. Mediante la teoria della *feedback-linearizzazione* descritta nella sezione 3.1 è possibile controllare un robot descritto da un modello non-lineare mediante l'uso di un semplice PD (figura 7.1).

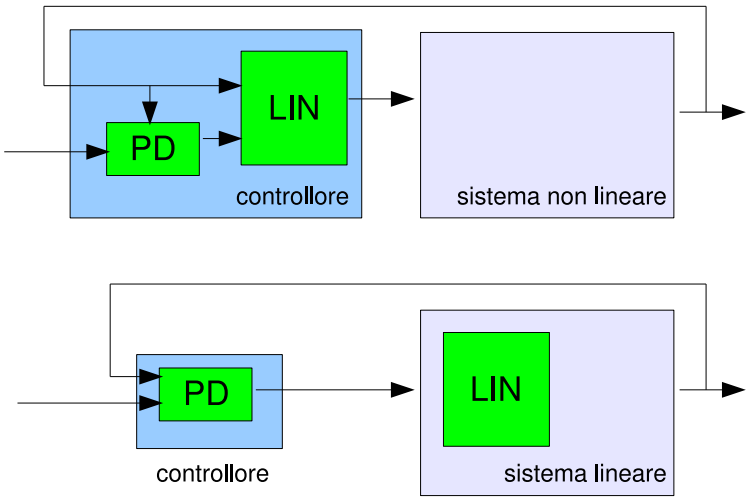

Figura 7.1: Modelli equivalenti *feedback-linearizzazione*

La trasformazione da non-lineare a lineare effettuata dal blocco LIN è tanto più robusta quanto più il linearizzatore riesce a cancellare le non-linearità — strutturate — dipendenti da parametri incerti. Alla stima dei parametri pensa in modo adattativo l'identificatore di sistema che, ad ogni passo di aggiornamento, legge la posizione del robot (proveniente da un efficiente sistema di localizzazione eventualmente *on-board*) e fornisce al linearizzatore le stime aggiornate dei parametri (figura 4.1). La presenza di questi parametri nelle equazioni differenziali (4.2) e (4.3) che descrivono il modello non è lineare e si può aggirare il problema:

- linearizzando le equazioni differenziali mediante uno sviluppo bidimensionale in serie di Taylor arrestato al primo ordine (con i risultati mostrati nella sottosezione 4.2.1). In questo caso avviene l'adattamento esplicito di ρ ed ℓ
- applicando una trasformazione di parametri che eviti la linearizzazione del punto precedente (sottosezione 4.2.2).

L'adattamento di tali parametri mediante l'algoritmo descritto nella sottosezione 4.3.3 permette al controllore di gestire al meglio il robot e di annullare, per quanto possibile, l'errore di traiettoria. Non è assicurato che la stima dei parametri converga ai valori veri: la convergenza dipende dalle caratteristiche della traiettoria imposta al robot.

La presenza di un sistema di adattamento permette di:

- non preoccuparsi della precisione della misura dei parametri. Per un algoritmo adattativo è sufficiente che il valore iniziale si avvicini "abbastanza" al valore vero
- poter riutilizzare lo stesso controllore su robot di dimensioni diverse, posto che abbiano le stesse caratteristiche di trazione[1]

La robustezza del controllo viene messa a dura prova da non-linearità e dinamiche non modellate in quanto:

- il controllore, attraverso il blocco LIN, tiene conto soltanto delle equazioni (4.2) e (4.3), nelle quali non viene modellata la saturazione dello sterzo, la quale pone un vincolo sul comando ω_s
- il controllore non tiene conto della massima velocità raggiungibile dai motori $\omega_{s,max}, \omega_{r,max}$ (nel caso in cui il vincolo sia attivo)
- il controllore non tiene conto della dinamica degli attuatori, modellata come un filtro di Bessel del sesto ordine che, oltre ad introdurre un ritardo fra il segnale di comando e il forzamento[2], pone dei vincoli sull'accelerazione massima che possono raggiungere i motori (nel caso in cui il vincolo sia attivo)

Nelle sezioni 6.5 e 6.6 si dimostra mediante esperimenti che, nel caso in cui le non-linearità e dinamiche non modellate diventino non trascurabili (ad esempio diminuendo l'orizzonte temporale a parità di cammino), il controllo non riesce più a mantenere il sistema stabile e i risultati sono quelli di figura 6.20.

[1] il controllore progettato si basa sulla modellizzazione di un robot a trazione posteriore

[2] ovvero fra l'istante in cui il controllore impone all'attuatore una determinata velocità e l'instante nel quale tale velocità viene effettivamente raggiunta

Appendice A

Codice Matlab

In questo capitolo si trovano le parti di codice Matlab cui si fa riferimento nel resto del documento.

A.1 Saturazione dello sterzo

```
function wsat = wsaturation(wflt, phi, phimax)
if ((phi >= phimax) && (wflt > 0))
    wsat = 0;
    phi = phimax;
elseif ((phi <= -phimax) && (wflt < 0))
    wsat = 0;
    phi = -phimax;
else
    wsat = wflt;
end
```

A.2 Saturazione degli attuatori

```
function wflt = wfilter(wpf, wf, wmax, sat\_en)
if ((wf >= wmax) && (sat\_en == 1))
    wflt = wmax;
elseif ((wf <= -wmax) && (sat\_en == 1))
    wflt = -wmax;
elseif (sat_en == 1)
    wflt = wf;
else
    wflt = wpf;
end
```

A.3 Generazione traiettoria tramite spline

```
function [x_s, y_s] = traiettoria(punti)
```

```
% Per sicurezza, ordiniamo i punti secondo il tempo
(...)
punti = sortrows(punti, 1);
% "Smontiamo" la matrice per lavorare più agevolmente con le singole
% coordinate
x = punti(:, 2);
y = punti(:, 3);
t = punti(:, 1);
% Interpolazione della traiettoria con spline
% Vengono generati x_s e y_s in funzione del tempo
x_s = spline(t, x);
y_s = spline(t, y);
(...)
```

A.4 Inizializzazione parametri

```
%%% Inizializzazione parametri stimatore
norm_reset = 1e-4; % Condizione per il reset della matrice P
lambda_r = 1; % Fattore di dimenticanza per la stima di rho
lambda_l = 1; % Fattore di dimenticanza per la stima di l
rhoo = .5; % Valore iniziale della stima di rho
lo = .9; % Valore iniziale per la stima di l
upd_rate = 1; % Periodo di aggiornamento parametri
upd_time = upd_rate/2;
rho_max = 2; % Valore massimo di rho (per la proiezione)
l_max = 5; % Valore massimo di l (per la proiezione)
% Variabile 1 - animazione 3D/2D / 0 - simulazione 2D statica
en_3D = 0;
en_2Dan = 0;
sat_en = 1; % Variabile inserimento_1/esclusione_0 saturation, limite
            % sull'angolo di sterzo sempre operativo
kp = [5, 0; 0, 5]; % Guadagno proporzionale del controllore
delta = .2; % Offset del punto rappresentativo [m]
rho = .123; % Raggio delle ruote [m]
l = .878; % Lunghezza del robot [m]
d = .4; % Semiasse delle ruote [m]
phimax = pi / 3; % Angolo di sterzo massimo
wsmax = 50; % Velocità angolare massima del carlike [rad/sec]
wrmax = 50; % Velocità angolare massima delle ruote [rad/sec]
motorbw = 15.7; % Banda passante motori [rad/sec]
(...)
%%% Condizioni iniziali del robot
xo = 0;
yo = 0;
z1o = l + delta;
z2o = 0;
thetao = 0;
phio = 0;
```

```
(...)
if (en_3D == 0)
    [t, state, z1, z2, theta, phi, z1d, z2d, wr, ws, wrout, wout, ...
     l_c, rho_c] = sim('allxxxx06', tmax);
(...)
```

A.5 Stimatore

```
function [rho_cout, l_cout, Pout11, Pout12, Pout21, Pout22, normP, ...
    condP] = adattamento(rho_cin, l_cin, Pin11, Pin12, Pin21, Pin22, ...
    zp1, zp2, theta, phi, wr, ws, lambda_l, lambda_r, delta, rhoo, lo, ...
    rho_max, l_max, norm_reset, tipo_adattamento)
% Questo blocco si occupa della stima dei parametri del robot

% Valori assegnabili a tipo_adattamento:
%   1: utilizza la formula linearizzata, quindi stima rho e l
%   0: utilizza la formula non linearizzata, quindi stima rho e rho/l
% In ogni caso, dal blocco usciranno rho ed l

% Matrice identità 2x2
I = [1, 0; 0, 1];
% Fattori di dimenticanza
lambda = [lambda_r, 0; 0, lambda_l];
% Matrice P risultata dall'iterazione precedente
Pin = [Pin11, Pin12; Pin21, Pin22];
% Vettore dei parametri: rho e rho/l
if (tipo_adattamento == 1)
    beta_cin = [rho_cin; l_cin];
else
    beta_cin = [rho_cin; rho_cin / l_cin];
end
% Valore a cui reinizializzare P quando necessario
Po = [1, 0; 0, 1];

% ... calcola la matrice Phi (dipendente dagli ingressi e tipica dello
% stimatore che abbiamo scelto...
if tipo_adattamento == 1
    Phi = [wr * (cos(theta) - tan(phi) * sin(theta) - ...
           delta * sin(theta + phi) * tan(phi) / lo), ...
           wr * delta * sin(theta + phi) * tan(phi) * rhoo / lo ^ 2;
           wr * (sin(theta) + tan(phi) * cos(theta) + ...
           delta * cos(theta+phi) * tan(phi) / lo), ...
           -wr * delta * cos(theta + phi) * tan(phi) * rhoo / lo ^ 2];
else
    Phi = [wr * (cos(theta) - tan(phi) * sin(theta)), ...
           -wr * delta * sin(theta + phi) * tan(phi);
           wr * (sin(theta) + tan(phi) * cos(theta)), ...
           wr * delta * cos(theta + phi) * tan(phi)];
```

```
end

%... calcola il vettore y (dipendente dagli ingressi)...
if tipo_adattamento == 1
    y = [zp1 + delta * sin(theta + phi) * (ws + wr * tan(phi) * rhoo / lo);
         zp2 - delta * cos(theta + phi) * (ws + wr * tan(phi) * rhoo / lo)];
else
    y = [zp1 + ws * delta * sin(theta + phi);
         zp2 - ws * delta * cos(theta + phi)];
end

%...aggiorna le matrici di stima...
Gamma = inv(Phi * Pin * Phi' + I);
Pout = Pin - Pin * Phi' * Gamma * Phi * Pin;

%Reset della matrice P in caso la sua norma sia inferiore di norm_reset
if(norm(Pout) < norm_reset)
   Pout = Po;
end;

%...calcola la norma e il condizionamento della matrice P...
normP = norm(Pout,2);


%...aggiorna il vettore dei parametri stimati...
beta_cout = beta_cin + Pout * Phi' * (y - Phi * theta_cin);

rho_cout = beta_cout(1, 1);
if tipo_adattamento == 1
    l_cout = beta_cout(2, 1);
else
    l_cout = rho_cout / beta_cout(2, 1);
end

%Proiezione parametri
if(rho_cout > rho_max)
    rho_cout = rho_max;
end;
if(rho_cout <= 0)
    rho_cout = 0.01;
end;

if(l_cout > l_max)
    l_cout = l_max;
end;
if(l_cout <= 0)
    l_cout = 0.01;
end;
```

```
Pout11 = Pout(1, 1);
Pout12 = Pout(1, 2);
Pout21 = Pout(2, 1);
Pout22 = Pout(2, 2);
```

Bibliografia

[1] A. De Luca, G. Oriolo, C. Samson, *Feedback Control of a Nonholonomic Car-like Robot*, in J.-P. Laumond, *Robot Motion Planning and Control*

[2] K. M. Passino, *Biomimicry for Optimization, Control, and Automation*, Springer-Verlag

[3] L. Sciavicco, B. Siciliano, *Robotica industriale — Modellistica e controllo di manipolatori, seconda edizione*, McGraw-Hill

[4] G. Ciccarella, P. Marietti, A. Trifiletti: *Strumentazione e misure elettroniche, seconda edizione*, Casa Editrice Ambrosiana