Metodi e Tecniche di Simulazione

Simulazione e controllo di un sistema *ballbot*

Grasselli Alessio
Luciani Irene
Terramani Simone
Nkou Luc Dieudonne

Anno accademico
2017 / 2018

Indice

Premessa

In questo progetto relativo al corso di metodi e tecniche di simulazione si vuole controllare e gestire un sistema *ballbot*.

Un ballbot è un robot situato al di sopra di un corpo sferiforme, come ad esempio una palla da basket, in grado di auto bilanciarsi ed evitare quindi la naturale caduta verso il terreno. Esso è decomponibile nei due componenti principali: la palla sottostante (sistema denominato *ball*) che si trova a contatto con il suolo ed una struttura meccanica (sistema *pendulum*) sul quale sono posti degli attuatori con lo scopo di mantenere la situazione di equilibrio.

La palla ha due soli gradi di libertà dovuti alla sua movimentazione sul piano longitudinale (assumendo si possa spostare lungo l'asse x e lungo l'asse y).
Il pendolo che la controlla, invece, è un sistema dotato di tre gradi di libertà traslazionali e di tre gradi di libertà rotazionali che possono essere definiti dai tre angoli:

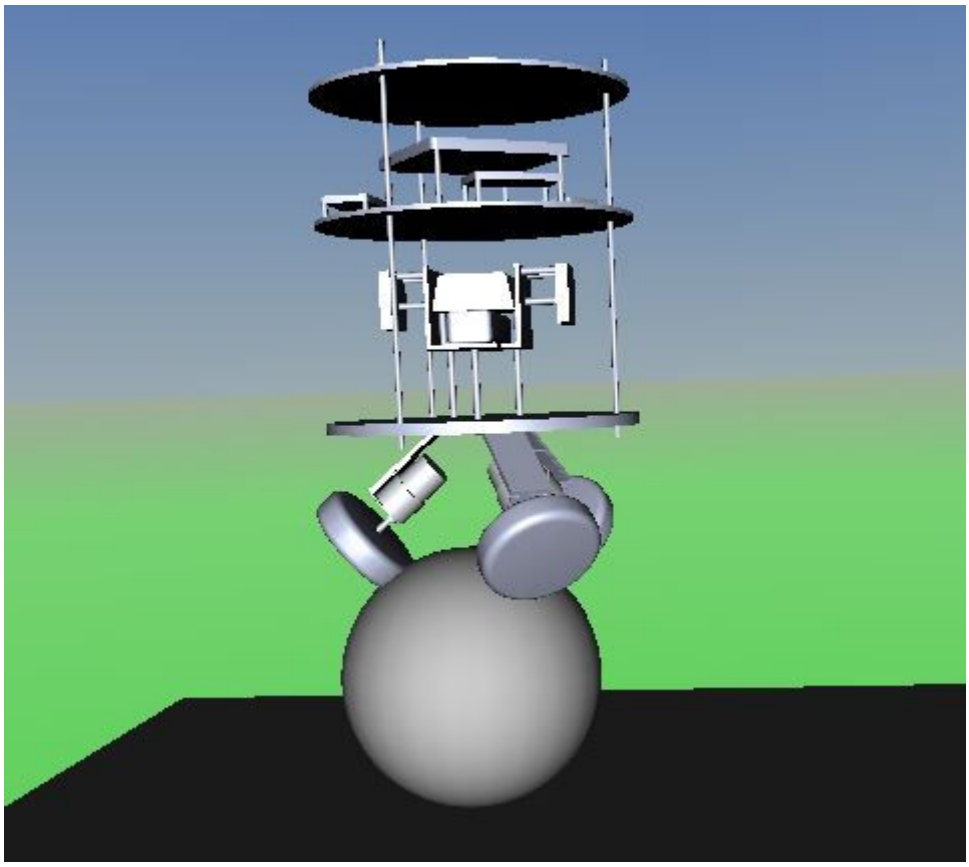
- ϕ è l'angolo di rollio (*roll*);
- θ è l'angolo di beccheggio (*pitch*);
- Ψ è l'angolo di imbardata (*yaw*).

Lo scopo del progetto è quello di ottenere un modello dinamico del sistema tramite le equazioni di Newton Eulero che risulterà quindi essere un sistema di equazioni non lineari dipendenti dai numerosi parametri che caratterizzano e descrivono il sistema e la sua dinamica.

In secondo luogo, vista la non linearità del sistema così definito, ci si propone di ottenere un secondo modello tramite un processo di linearizzazione intorno al punto di equilibrio.
In questo modo si perderà sicuramente di accuratezza in quanto il modello non lineare di un sistema fisico reale sarà sempre più fedele nel descrivere il comportamento effettivo dell'oggetto.
Si vuole linearizzare allo scopo di ottenere una rappresentazione semplificata del modello che descrive il ballbot, ottenendo dal sistema di partenza un nuovo sistema di equazioni semplificato.
Si osserverà, infatti, che il modello non lineare è estremamente complesso e poco leggibile.

Lo scopo finale del progetto verterà nella definizione di un sistema di controllo per il ballbot linearizzato, e di una simulazione grafica con Simulink, come mostrato in figura.

Visto che questo approccio per lo studio del sistema fa riferimento ad un robot realmente esistente in laboratorio, di seguito vengono evidenziate le principali costanti che caratterizzano il modello.



Modello 3D ballbot

Valori numerici costanti per tutto il progetto:

Parametro	Valore
Raggio della palla	0,12 cm
raggio della omni-wheel	0,053 cm
massa del pendolo	5,045 g
accelerazione gravitazionale	9,81 m/s^2
massa della palla	0,65 g
inerzia del pendolo (asse x)	0,47 kg m^2
inerzia del pendolo (asse y)	0,47 kg m^2
inerzia del pendolo (asse z)	0,049 kg m^2
inerzia della palla	0,005 kg m^2
distanza dal centro di massa della palla	0,410 cm
raggio del pendolo	0,14 cm
angolo di inclinazione delle omniwheels	$2\pi/9$

Struttura del *ballbot* e sistemi di riferimento (matrici di rotazione)

Come accennato nella premessa il *ballbot* non è altro che un robot mobile che è in grado di auto stabilizzarsi su di una palla, la quale ha un unico punto di contatto con il terreno.

È possibile affermare allora che il ballbot è un sistema omnidirezionale, ovvero ha la capacità di spostarsi fisicamente in tutte le direzioni, come un vero e proprio veicolo.

Il funzionamento del ballbot è analogo a quello di un pendolo inverso: si punta a controllare il pendolo agendo tramite impulsi e forze esterne al fine di mantenerlo nella posizione denotata dall'angolo $\theta=0$ che rappresenta un punto di equilibrio instabile, a causa delle forze che agiscono sul sistema.

Ad esempio, quando si andranno a descrivere le equazioni della dinamica del sistema nel complesso, in esse compariranno parametri relativi alle forze come quella di attrito col terreno e quelle di reazione.

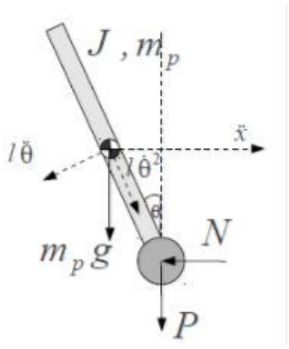


Figura 1 - Pendolo invertito e Forze

I parametri fondamentali per descrivere un ballbot sono i seguenti: massa, peso, centro di gravità e le forze che agiscono sul sistema.

La componente *ball* è quella principale, che trasmette e riceve tutte le forze entranti al sistema ed è preferibile abbia un coefficiente di attrito elevato, per avere una maggiore aderenza con il terreno sottostante.

Per quanto riguarda il sistema di attuazione, ci sono svariati metodi per implementarlo. Nel caso in esame il pendolo verrà controllato con tre ruote omnidirezionali (*omni-wheels*) azionate da tre motori che forniscono la potenza.

Si possono costruire ballbot più o meno complessi, ad esempio caratterizzati da un pendolo azionato da più di tre ruote.
I motori fanno muovere le ruote in modo tale da non entrare in contrasto tra loro nel comandare la direzione ed in generale il movimento.
Sono questi attuatori a ripristinare la stabilità del sistema nel complesso anche, ad esempio, quando si sottopone volontariamente l'azione di una forza esterna.

I ballbot più sofisticati (e complessi) possono essere dotati anche di un apparato di sensori per monitorare l'orientamento della palla e la posizione ad un dato istante, in modo tale da avere più punti di vista per l'osservazione dello stato del sistema.

Nella figura sottostante si mostra lo schema generale di un ballbot, ed i relativi sistemi di riferimento.

- E: sistema di riferimento *earth* (del terreno, fisso).
- P: sistema di riferimento *pendulum* (del pendolo).
- B: sistema di riferimento *ball* (della palla).

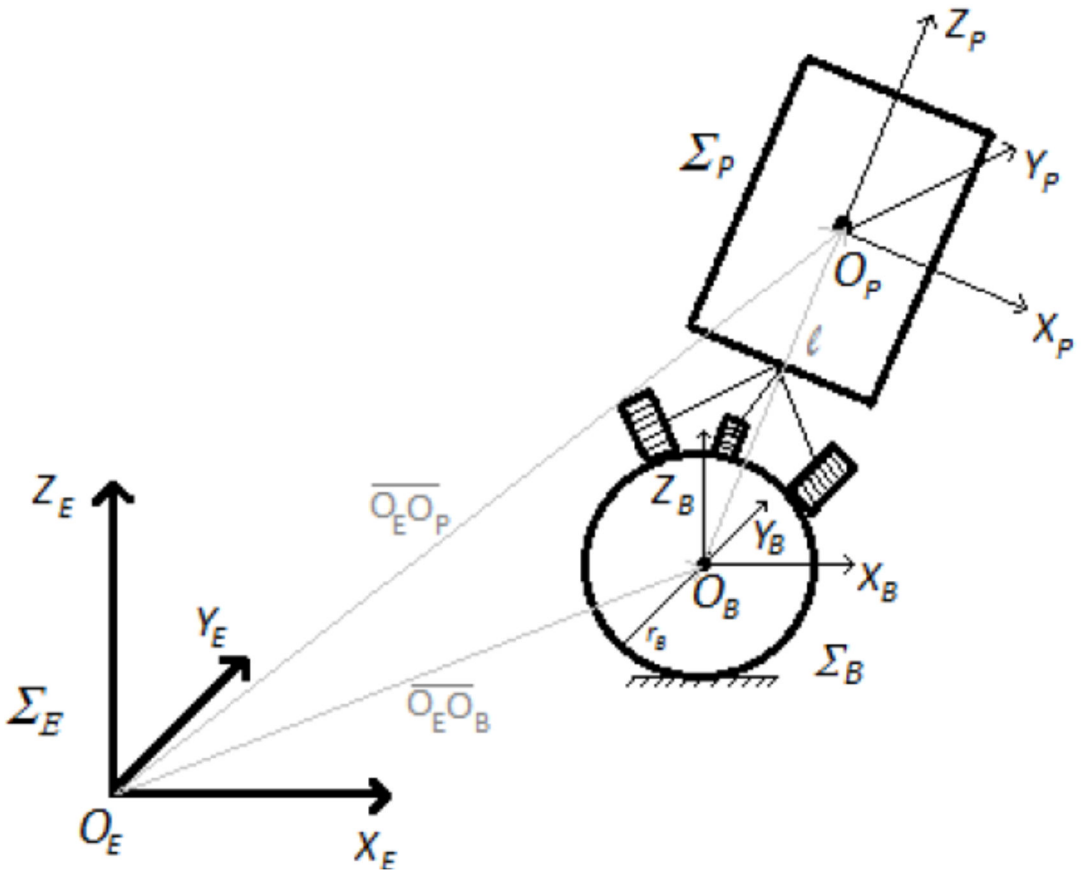


Figura 2 - Schema ballbot - Sistemi di riferimento

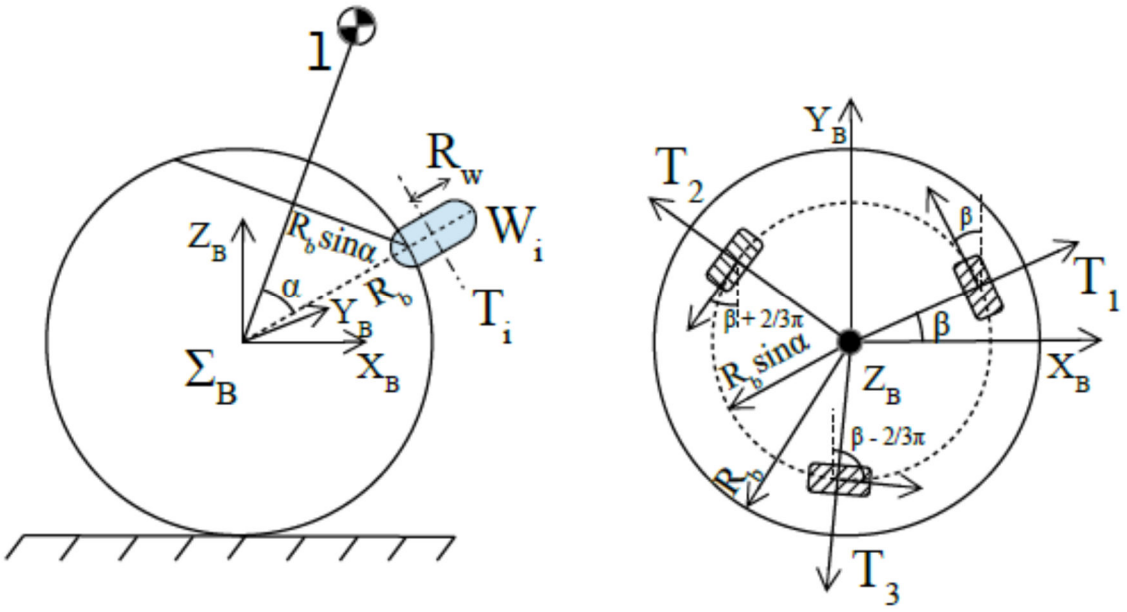


Figura 3 - Struttura ballbot e attuatori

Visto che il ballbot risulta essere l'unione di due corpi rigidi, se ne necessita un piccolo approfondimento in modo tale da chiarire quali sono le principali caratteristiche.
Un corpo rigido gode della proprietà che presi due punti generici, essi sono vincolati a mantenere costante la loro distanza nel tempo, nello studio di un corpo rigido è essenziale definire i sistemi di riferimento rispetto ai quali consideriamo le varie grandezze, in modo tale da tenerle sempre in relazione.

Il sistema di riferimento *ball*, per come è stato definito, non risulta mai ruotato rispetto al sistema *earth*. Questo discorso non è però valido se rivolto al sistema di riferimento *pendulum*.
In questo ultimo caso occorre definire dei parametri che permettono di effettuare una trasformazione di coordinate, che permetta il passaggio da un sistema di riferimento di partenza ad un diverso sistema di riferimento finale.

A questo scopo vanno definite le tre matrici di rotazione.
I tre angoli di rotazione descritti RPY (*roll, pitch* e *yaw*) vengono assunti come descrizione della configurazione dell'orientamento del corpo rigido rispetto ad un sistema fisso di orientamento.
Si assume, inoltre, che la rotazione in senso antiorario sia quella positiva.
Tali matrici godono della proprietà di essere antisimmetriche, e se ne definisce una per ogni angolo di rotazione (rollio, imbardata, beccheggio).

Imponendo che la rotazione sia in senso antiorario, la sequenza di rotazioni elementari per gli angoli RPY sono descritti dalle matrici sottostanti.

Prima matrice di rotazione (ϕ, angolo di *rollio*):

$$R_{\phi}^{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

Seconda matrice di rotazione (θ, angolo di *beccheggio*):

$$R_{\theta}^{T} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\phi) \end{bmatrix}$$

Terza matrice di rotazione (Ψ, angolo di *imbardata*):

$$R_{\Psi}^{T} = \begin{bmatrix} \cos(\Psi) & \sin(\Psi) & 0 \\ -\sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La proprietà di R matrice antisimmetrica si definisce come segue.

$$R^{-1} = R^{T} = R_{\phi}^{T} \cdot R_{\theta}^{T} \cdot R_{\Psi}^{T}$$

Da questa equazione si ricava l'espressione R^{-1}.

$$R^{-1} = \begin{bmatrix} \cos(\theta) \cdot \cos(\Psi) & \cos(\theta) \cdot \sin(\Psi) & -\sin(\theta) \\ \sin(\phi) \cdot \sin(\theta) \cdot \cos(\Psi) - \cos(\phi) \cdot \sin(\Psi) & \cos(\phi) \cdot \cos(\Psi) + \sin(\phi) \cdot \sin(\theta) \cdot \sin(\Psi) & \sin(\phi) \cdot \cos(\theta) \\ cos(\phi) \cdot \sin(\theta) \cdot \cos(\Psi) + \sin(\phi) \cdot \sin(\Psi) & \sin(\theta) \cdot cos(\phi) \cdot \sin(\Psi) - \sin(\phi) \cdot \cos(\Psi) & \cos(\theta) \cdot \cos\phi) \end{bmatrix}$$

In questo modo si potranno trasformare grandezze da un sistema di riferimento ad un altro, se necessario, per comporre adeguatamente le equazioni della dinamica che descrivono il modello non lineare del ballbot che si dovrà poi linearizzare.

Ipotesi per la costruzione del modello del ballbot

Lo scopo del progetto è la costruzione di un modello che riprenda il comportamento del ballbot, modello che sarà non lineare e solamente in una fase successiva ci si propone di linearizzarlo e di imporre la legge di controllo che lo stabilizzi (dare in ingresso una coppia generata da un motore in corrente continua).
In questo capitolo si tratteranno tutte le equazioni fisiche e le assunzioni necessarie per lo sviluppo del sistema del modello.

A partire dai due sistemi sopra definiti, si avranno anche due centri (origini): *OB* ed *OP*.
Queste due origini sono coincidenti con i centri di inerzia dei corrispondenti corpi rigidi (palla e pendolo).

Va prestata particolare attenzione alle equazioni del moto, che saranno quelle che andranno a costituire il nucleo del modello, tali equazioni sono formulate rispetto al sistema di riferimento relativo (non quello fisso).
Esse vanno implementate in questo modo poiché per prima cosa la matrice di inerzia è invariante rispetto al fattore tempo. Inoltre, si considera il vantaggio dell'avere un corpo rigido simmetrico a semplificare le equazioni del moto del modello non lineare.
Le misure sono facilmente convertibili rispetto al sistema di riferimento fisso, a partire da uno dei sistemi relativi dei corpi rigidi.
Le forze di controllo, generalmente, si ricavano in termini di sistema di riferimento fisso.

Si pone come ipotesi, inoltre, che non vi sia scivolamento tra la palla ed il terreno e tra la palla e le ruote. La palla non può muoversi verticalmente, dunque ha solamente due gradi di libertà traslazionali, il che comporta movimenti solamente orizzontali.

Equazioni non lineari della dinamica

Il primo passo del progetto sta nel ricavare il sistema di equazioni non lineari della dinamica del ballbot che descrivano formalmente il sistema.
Dunque non si fa altro che ricavarne un modello matematico complesso.

Il metodo di Newton Eulero permette di estrapolare le equazioni della dinamica di un sistema attraverso precise equazioni differenziali che descrivono il moto al fine di modellare il comportamento di un sistema formato da più componenti.
Il metodo di Eulero per la definizione di sistemi fisici si basa sulla riduzione dei gradi di libertà.
Nel caso specifico del ballbot i gradi di libertà complessivi sono otto: due gradi di libertà per la palla e sei gradi di libertà per il pendolo (3 rotazionali + 3 traslazionali).

Dalle equazioni si vogliono ricavare: le velocità traslazionali dei vari sottosistemi, le velocità angolari, la dinamica sia rotazionale che traslazionale (tramite il metodo di Newton) e si introducono le forze esterne e le coppie. Il ballbot è un sistema dinamico e la parte cinematica del sistema è relativa proprio a tali velocità traslazionali e rotazionali.

La sezione sulla cinematica è suddivisa in due macrocategorie, ognuna relativa ad uno dei corpi rigidi che compongono il ballbot: cinematica del corpo principale (ovvero il *body*), cinematica della palla.

La parte relativa alla dinamica dei due corpi rigidi (palla e pendolo) ricava le equazioni dinamiche impiegando la seconda legge di Newton.
La dinamica si suddivide in quattro categorie;
Dinamica rotazionale della palla, dinamica rotazionale del corpo.
Dinamica traslazionale della palla, dinamica traslazionale del corpo.

Ricavare un modello non lineare: Metodo di Newton – Eulero

In questo capitolo saranno mostrate le leggi fisiche e le varie relazioni dalle quali si sono poi ricavate le equazioni non lineari del modello del ballbot in questione, sotto le specifiche ipotesi, ricorrendo al metodo di *Newton – Eulero*.

Ricapitolando, le funzioni principali di questo metodo applicate alla necessità di modellare un sistema dinamico seguono i seguenti passi.

Come prima cosa ci si propone di scrivere separatamente le equazioni dinamiche per i diversi elementi del sistema.
Le equazioni saranno poi valutate numericamente soltanto alla fine della costruzione del modello, lasciando espressi tutti i parametri fino alla costruzione di un sistema finale di equazioni completo.
Il modello finale sarà adeguato per sintetizzare uno schema di controllo che agirà sul modello stesso, in questo caso tramite un ingresso forzante proveniente da un sistema esterno (dei motori).

Sostituendo ricorsivamente le espressioni ottenute si ottiene un sistema di equazioni dinamiche in forma chiusa, successivamente da linearizzare.
La sostituzione ricorsiva delle equazioni è dovuta alla scelta di adottare variabile simboliche (utilizzando il programma di calcolo *Matlab*) anche se non hanno una particolare convenienza, in quanto fino all'ultimo calcolo e sostituzione si tiene conto di tutti i risultati precedenti e si ha occupazione di un ingente spazio di memoria e poca leggibilità delle espressioni stesse.

Nei capitoli successivi si presenta le modalità con cui si ricava un modello del sistema non lineare che descriva la dinamica completa, partendo prima dalla costruzione delle varie equazioni passo per passo partendo da quelle necessarie alla modellazione della parte cinematica.

Modellazione:
Equazioni cinematiche delle velocità

Si è deciso di partire dalle equazioni cinematiche delle velocità traslazionali e delle velocità angolari che caratterizzano il sistema complessivo, si sviluppano tali equazioni per ciascuno dei due corpi rigidi, palla e pendolo.
Le velocità da analizzare sono le velocità traslazionali e le velocità angolari di entrambi i corpi.

Considerando *P* il corpo principale e le sue velocità in relazione al sistema di riferimento ΣP, come prima cosa è necessario effettuare la conversione in velocità relative al sistema ΣE;
Occorre dunque spostarci al sistema di riferimento relativo, a partire dal sistema fisso.
Per effettuare questi passaggi tra sistemi di riferimento differenti occorre sempre disporre di una matrice di trasformazione che leghi le grandezze dal sistema ΣP al sistema ΣE, o viceversa, in modo tale da utilizzarla per ottenere la stessa grandezza nelle coordinate di destinazione.

La matrice di trasformazione relativa alle velocità angolari del corpo *P* dal sistema fisso a quello relativo è quella sottostante.

$$T_{\theta^{-1}} = \begin{bmatrix} 1 & \sin(\Phi)\cdot T_\theta & \cos(\Phi)\cdot T_\theta \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \dfrac{\sin(\Phi)}{\cos(\theta)} & \dfrac{\cos(\Phi)}{\cos(\theta)} \end{bmatrix}$$

Identicamente a quanto fatto per il corpo principale del sistema si effettuano le trasformazioni delle velocità (traslazionali ed angolari) della palla *B* tramite la matrice di trasformazione, portandole dal sistema di riferimento fisso a quello relativo.

Si ottengono le equazioni cinematiche rispettivamente del corpo *P* e della palla *B* come segue.

$$\begin{cases} \dot{\Gamma}_P^{\ E} = R_P^E \cdot V_P \\ \dot{\Theta}_P^E = T_\theta^{-1} \cdot w_P \end{cases}$$

1) Equazioni cinematiche del corpo principale P

$$\begin{cases} \dot{\Gamma}_B^{\ E} = R_Z(\Psi) \cdot V_B \\ \dot{\Theta}_B^E = R_Z(\Psi) \cdot w_B \end{cases}$$

2) Equazioni cinematiche della palla *B*

Modellazione:
Equazioni dinamiche traslazionali e rotazionali

Le equazioni cinematiche delle velocità traslazionali ed angolari dei due corpi rigidi sono descritte dalle espressioni ottenute alla fine del capitolo precedente.
Il secondo passo per la costruzione della dinamica del modello non lineare del ballbot è quello di ottenere le equazioni dinamiche, ragionando anche in questo caso in termini di dinamica traslazionale e rotazionale.
Come già detto, le equazioni vanno ricavate distintamente per ciascuno dei due corpi rigidi del sistema.

Analogamente alle equazioni cinematiche a partire da espressioni in relazione al sistema fisso di riferimento ΣE, anche in questo caso si applicano le trasformazioni opportune per il cambio di coordinate che portano ad equazioni nel sistema di riferimento relativo ΣP e tale procedimento va ripetuto per ciascun corpo.

In questo caso, identicamente a quanto fatto per velocità traslazionali e velocità angolari, si differenzia la dinamica del sistema anch'essa in traslazionale e rotazionale formulando distintamente le espressioni per la dinamica del corpo *P* e quelle della palla *S*.

Vogliamo studiare la dinamica dei corpi rigidi al fine di studiarne il moto e le sue cause, oltre che conoscere come esso varia nel tempo.
In particolare per la dinamica rotazionale si necessiterà di grandezze specifiche, quali momento d'inerzia dei corpi, momento delle forze esterne, momento angolare ed anche il vettore delle coppie che agiscono sul corpo. Le velocità traslazionali e rotazionali ottenute nel paragrafo precedenti saranno utili per ricavare queste nuove espressioni.

Partendo dal sistema corpo *P* si vuole ottenere la legge della dinamica traslazionale.

A partire dal primo assioma della seconda legge di *Newton* si dispone dell'espressione che lega le componenti lineari del moto del corpo.

$$m_p \cdot \frac{d}{dt} \cdot (R_P^E \cdot V_P) = F_{ext}{}_P^E \qquad 3)\ \text{Seconda legge di Newton – primo assioma}$$

Si osserva che il vettore delle forze esterne deve essere trasformato dal sistema di riferimento fisso a quello relativo, come svolto nei passaggi precedenti.

A partire da questo primo assioma si ricava la legge della dinamica traslazionale del corpo *P*, formulata come segue.
Si osserva che le forze esterne nella legge ottenuta sono tutte definite rispetto al sistema relativo e rappresentano la somma di tutte le forze che provengono dall'esterno del sistema ed agiscono sul corpo principale del ballbot.

$$m_P \cdot \left(\omega_P \times V_P + \dot{V}_P\right) = F_{ext}{}_P^P \qquad 4)\ \text{Equazioni della dinamica traslazionale del corpo principale P}$$

Si vuole ora ottenere la dinamica rotazionale del corpo.
Considerando il secondo assioma della seconda legge di *Newton* si ottiene l'espressione che lega momento di inerzia e momento angolare del corpo principale *P* e si ricava, inoltre, la formula matematica che esprime tutte le coppie esterne che agiscono sul corpo espresse rispetto al sistema di riferimento relativo e definite sul sistema fisso (dunque la relativa trasformazione).

$$\frac{d}{dt} \cdot (R_P^E \cdot M_P) = \frac{d}{dt} \cdot (R_P^E \cdot I_P \cdot \omega_P) = \tau_{ext}{}_P^E \qquad 5)\ \text{Seconda legge di Newton – secondo assioma}$$

L'equazione della dinamica rotazionale del corpo principale finale è la seguente.
Si osserva che, come previsto, il vettore delle coppie è espresso ora nel sistema Σ_P.

$$(\omega_P \times I_P \cdot \omega_P + I_P \cdot \dot{\omega}_P) = \tau_{ext}{}_P^P \qquad 6)\ \text{Equazioni della dinamica rotazionale del corpo principale P}$$

Per ricavare le leggi della dinamica, traslazionale e rotazionale, del sistema palla *B* ci si riconduce sempre come primo passo al assioma della seconda legge di *Newton* che fornisce l'espressione che lega le componenti lineari del moto della palla.
Tali componenti sono caratterizzate dall'equazione sottostante.

$$\frac{d}{dt} \cdot \left(m_B \cdot \dot{I}_B^E\right) = m_B \cdot \frac{d}{dt} \cdot (R_P^E \cdot R_B^E \cdot V_B) = F_{ext}{}_B^E \qquad 7)\ \text{Seconda legge di Newton – primo assioma}$$

Il primo assioma permette, perciò, di ottenere un'espressione matematica del vettore delle forze esterne che agiscono sul sistema palla espresse nel sistema di riferimento fisso Σ_E.
A partire da ciò è possibile ricavare la legge finale della dinamica traslazionale della palla *B* come segue.

$$m_B \cdot R_P^E \cdot \left[\omega_B \times (R_B^P \cdot V_B) + R_B^P \cdot \left(\omega_B^* \times V_B + \dot{V}_B\right)\right] = R_P^E \cdot {F_{ext}}_B^P$$

8) Equazioni della dinamica traslazionale della palla B

Nella legge finale ottenuta le forze esterne sono, però, trattate come somma di tutte le forze agenti sul sistema palla *B* in relazione al sistema di riferimento relativo Σ_P.
Il vettore ω_B^* è nella forma sottostante.

$$\omega_B^* = \begin{bmatrix} -p \\ -q \\ 0 \end{bmatrix}$$

Va ripetuto esattamente quello che è stato fatto per il sistema corpo principale *P* per ottenere la dinamica rotazionale del corpo, ovvero, si considera il secondo assioma della prima legge di *Newton* per ottenere il vettore delle coppie esterne agenti sul sistema palla *B* espresso secondo il sistema di riferimento fisso Σ_E.
Esattamente come prima, sviluppando l'equazione, si otterrà la legge della dinamica rotazionale della palla che esprime il vettore delle coppie esterne questa volta espresso in relazione al sistema di riferimento relativo Σ_P al fine di seguire le ipotesi per la costruzione del modello del ballbot.

$$\frac{d}{dt} \cdot (R_B^E \cdot M_B) = \frac{d}{dt} \cdot (R_B^E \cdot I_B \cdot \omega_B) = {\tau_{ext}}_B^E$$ 9) Seconda legge di Newton – secondo assioma

Dallo sviluppo di tale equazione, si ricava la legge desiderata.

$$\omega_P \times R_B^P \cdot I_B \cdot \omega_B + R_B^P \cdot (\omega_B^* \times I_B \cdot \omega_B) + R_B^P \cdot I_B \cdot \dot{\omega}_B = {\tau_{ext}}_B^P$$

10) Equazioni della dinamica rotazionale della palla B

Modellazione:
Equazioni per imporre i vincoli di sistema

Un sistema reale è complesso e bisogna considerare che non è totalmente libero di muoversi ed evolversi, soprattutto nel caso del ballbot, essendo decomponibile in due sottosistemi (la palla ed il pendolo) tra loro strettamente vincolati.
Proprio per considerare questa relazione di collegamento nel sistema di equazioni vanno introdotte altre due espressioni con lo scopo di sottoporre al sistema i vincoli necessari per garantire il corretto funzionamento secondo le regole e sotto determinate restrizioni.
Si hanno dunque due leggi vincolanti: una di tipo geometrico ed una che gestisce il puro rotolamento della palla sul suolo.

Nello specifico si studia il primo dei due vincoli, quello geometrico, che lega il sistema palla *B* al corpo principale *P*.
Tale restrizione si basa sul fatto che i due moti dei due corpi rigidi non sono affatto indipendenti, ma sono in stretta connessione tra loro: in primis la distanza tra i centri di massa è costante dunque tempo – invariante.
Il vincolo da aggiungere deve imporre che il parametro della lunghezza che rappresenta tale distanza non deve assolutamente subire variazioni nel tempo.

Il sistema di riferimento che si andrà ad utilizzare deve sempre essere conforme alle ipotesi, ovvero se alcuni parametri sono esposti in riferimento al sistema fisso devono come sempre essere trasformati nelle coordinate relative al sistema relativo ΣP.

Il vincolo geometrico è il seguente.
Il parametro l è la distanza tra i due centri di massa.

$$\Gamma_B^E = \Gamma_P^E + R_P^E \cdot l \qquad 11)\ \text{Primo vincolo: geometrico}$$

Il vincolo di partenza va scisso in altri due, in modo tale che si possano esprimere distintamente il vincolo che lega le velocità delle componenti del ballbot ed il vincolo che lega le accelerazioni.

Per calcolare l'espressione che vincola le velocità del sistema a partire dal vincolo geometrico, con opportune sostituzioni si ottiene quanto segue.

$$R_B^P \cdot V_B = V_P + \omega_P \times l$$ 12) Vincolo geometrico: velocità

Si noti che effettivamente in tale espressione compaiono i parametri VB che è la velocità traslazionale della palla, VP che è la velocità traslazionale del corpo principale ed infine ωP la velocità rotazionale dello stesso corpo.

A partire dal vincolo di velocità è possibile calcolare il secondo vincolo geometrico relativo alle relazioni tra le accelerazioni delle componenti di sistema.
È un passaggio immediato, in quanto a partire da una legge espressa in termini di velocità, non occorre far altro che derivarla per ottenere un'espressione in termini di accelerazioni.
Il vincolo finale ottenuto è quello sottostante.

$$R_B^P \cdot \left(\omega_B^* \times V_B + \dot{V}_B\right) = \left(\dot{V}_P + \dot{\omega}_P \times l\right)$$ 13) Vincolo geometrico: accelerazioni

La parte relativa ai vincoli legati alla geometria dei corpi rigidi è terminata.
È necessario passare alla definizione del secondo vincolo fondamentale che regolamenta il rotolamento del sistema palla B del ballbot per completare questa parte.

Il secondo vincolo ha come scopo, quindi, quello di imporre al sistema che la palla rotoli a contatto con il suolo senza avere slittamenti.
Come detto in fase di definizione delle ipotesi sul modello, la palla si muove solamente orizzontalmente e non verticalmente, perciò dispone di due soli gradi di libertà traslazionali.
Essa può avere esclusivamente un moto di rotolamento sul suolo.

Comparando con il vincolo geometrico ottenuto anche questa volta occorrerà ottenere due equazioni che formeranno nell'insieme il vincolo di puro rotolamento;
Non si distingue, però, il vincolo relativo alle velocità dal vincolo relativo alle accelerazioni.
I ragionamenti vengono svolti solamente sulla base delle accelerazioni delle componenti di sistema, giustapponendo da una parte l'accelerazione tangenziale della palla e dall'altra l'accelerazione angolare della stessa.
Si ha, dunque, un'accelerazione tangenziale a cui essa è soggetta durante il moto lungo una traiettoria curva con velocità di movimento variabile in modulo ed un'accelerazione angolare che è sempre una grandezza vettoriale che rappresenta la variazione temporale della velocità angolare.

Per calcolare le due espressioni che formano il vincolo di puro rotolamento si deriva, come detto in per il primo caso, le equazioni ottenute dai vincoli precedenti e si ottiene quanto segue.

Il vincolo del rotolamento è quello sottostante.

$$V_B = \omega_B \times R_B$$

Si deriva tale espressione.

$$\dot{V}_B = \dot{\omega}_B \times R_B$$

Da questa equazione si estraggono i vincoli relativi alle accelerazioni tangenziale ed angolare con opportune sostituzioni, dove comparirà il parametro R_b che è il raggio della palla B ed ha la struttura vettoriale che segue.

$$R_B = \begin{bmatrix} 0 \\ 0 \\ -R_b \end{bmatrix}$$

A questo punto si ottengono le leggi finali dal calcolo della derivata $\dot{V}_B$.

$\dot{p}_B = -\frac{\dot{v}_B}{R_B}$ 14) Primo vincolo di rotolamento $\dot{q}_B = \frac{\dot{u}_B}{R_B}$ 15) Secondo vincolo di rotolamento

Modellazione:
Equazioni delle coppie e delle forze esterne

Il modello non lineare è quasi completo.
Nella descrizione del sistema ballbot è stato detto che sicuramente il sistema nel complesso è soggetto a forze esterne provenienti da varie fonti dall'ambiente circostante.
Lo scopo del progetto sta proprio nel sintetizzare un controllore che sia capace di stabilizzare il ballbot in qualsiasi stato, quando non si danno impulsi al sistema forzanti (perciò il ballbot è soggetto alle sole forze di gravità e di reazione vincolare), ma anche quando si dà in ingresso una coppia che lo destabilizzi.
In questa sezione vengono introdotti tutti i vincoli che rappresentano i legami tra le varie forze e coppie esterne agenti sul sistema e le relative limitazioni.
Va prestata particolare attenzione al fatto che il controllore che si vuole sintetizzare è un motore in corrente continua che introdurrà proprio una coppia forzante nel sistema, necessaria per la stabilizzazione del ballbot.

Quando si parla di forze esterne per il sistema, è bene differenziare lo studio relativo alle forze potenziali ed alle forze non potenziali.
Inoltre, come forza totale va considerata che una stessa forza può agire tanto sul sistema palla tanto sul sistema pendolo, dunque si sommano i due contributi.
Nello specifico le forze potenziali agiscono su un campo conservativo e saranno indicate con GPE quando si considerano solamente quelle agenti sul pendolo relativamente al sistema di riferimento fisso ΣE;
Saranno indicate, invece, con GBE quando si considerano le forze di carattere potenziale che agiscono esclusivamente sulla palla, rispetto al sistema di riferimento fisso ΣE.
Le forze non potenziali inglobano l'insieme di forze agenti sul sistema e l'insieme di forze che il sistema di per sé produce.
Una forza non potenziale è la reazione vincolare che collega strettamente i due corpi rigidi;
Il parametro impiegato per rappresentare questa forza è Rv.
Si ha reazione vincolare quando un corpo non subisce accelerazioni nonostante ci siano forze con una risultante non nulla ad agire su di esso.
In questo caso specifico la reazione vincolare agisce esattamente sul centro di massa della palla che sta a contatto con il suolo.
Per finire, il parametro Rt rappresenta la forza di reazione applicata tra la palla ed il suolo.
Anche questa volta tutte le forze, le coppie, ed i parametri andranno espressi in termini di sistema di riferimento relativo in base al corpo cui si riferiscono (sistema ΣP).

Nel sistema non lineare che modellizza il ballbot si vuole ridurre il grado di complicazione, dunque si evita la distinzione tra forze potenziali e non potenziali semplicemente introducendo un'equazione che descriva le coppie e le forze totali esterne.

Il valore totale è la somma dei singoli contributi dei vari tipi di forze.
Come imposto dalle ipotesi sulla costruzione del modello, come è stato fatto per le espressioni relative ai vincoli definiti nei capitoli precedenti, si definiranno le equazioni delle forze e coppie esterne per il sistema palla ed il sistema pendolo distintamente.

Per prima cosa si introducono le equazioni di tali forze e coppie partendo dal pendolo.
Il parametro L è da non confondere con l, che rappresenta la distanza tra i centri di massa dei due corpi rigidi introdotto per definire i vincoli geometrici.
La nuova grandezza introdotta per questi vincoli, cioè L, è il vettore che definisce il centro di gravità della palla rispetto al sistema relativo (come già detto la forza di reazione vincolare ha azione proprio sul centro di gravità del sistema B).

$$L = \begin{bmatrix} 0 \\ 0 \\ -l \end{bmatrix}$$

Ora si dispone di tutte le definizioni dei parametri necessari per comporre le equazioni dei vincoli di forze e coppie, per il sistema pendolo, rispetto al sistema relativo Σ_P.

$$F_{ext_P}^{\ P} = (R_P^E)^T \cdot G_P^E + R_v$$ 16) Forze totali esterne sul sistema pendolo

$$\tau_{ext_P}^{\ P} = L \times R_v - U$$ 17) Coppie totali esterne sul sistema pendolo

Da ultimo si definiscono le equazioni dei vincoli di forze e coppie per il sistema palla, rispetto al sistema relativo Σ_P.

$$F_{ext_B}^{\ P} = (R_P^E)^T \cdot (G_P^E + R_t) - R_v$$ 18) Forze totali esterne sul sistema palla

$$\tau_{ext_B}^{\ P} = (R_P^E)^T \cdot (-R_B \times R_t) + U$$ 19) Coppie totali esterne sul sistema palla

Il parametro U rappresenta l'ingresso fornito al sistema.

Modello finale del ballbot: Costruzione del sistema non lineare

Dopo aver calcolato tutte le equazioni che rappresentano i legami tra le grandezze che compongono il sistema ed i vincoli agenti sul sistema reale, il sistema non lineare è completato e si compone dell'unione di tutte le espressioni ricavate nei passaggi precedenti.

In questo modo si tiene conto, per entrambi i corpi rigidi, della cinematica, della dinamica rotazionale e traslazionale e di tutti i vincoli quali quello geometrico tra la palla ed il corpo principale, quello di solo rotolamento della palla sul suolo e delle forze e delle coppie provenienti dall'esterno.
Il modello è ora completo.

Si utilizza il programma di calcolo *Matlab* per dichiarare tutti i parametri che saranno introdotti con le varie espressioni, tenendo conto di quanto detto in precedenza: i parametri vanno mantenuti simbolici fino alla fine senza mai introdurre valori numerici nei passaggi intermedi.
Le dichiarazioni dei parametri vengono fatte tramite il comando *syms* e si preferisce specificare *real* per evitare che nei calcoli si portino avanti numeri complessi.

Una volta che tutte le dichiarazioni sono state inserite nello script, si procede con la scrittura delle equazioni ottenute nella fase di modellazione.
Il modello finale della dinamica completa del ballbot, ovvero il sistema non lineare di equazioni è quello sottostante.

$$
\begin{cases}
m_p \cdot \left(\omega_P \times V_P + \dot{V}_P\right) = (R_P^E)^T \cdot G_P^E + R_v \\
(\omega_P \times I_P \cdot \omega_P + I_P \cdot \dot{\omega}_P) = L \times R_v - U \\
m_b \cdot \left[\omega_B \times (R_B^P \cdot V_B) + R_B^P \cdot \left(\omega_B^* \times V_B + \dot{V}_B\right)\right] = (R_P^E)^T \cdot (G_P^E + R_t) - R_v \\
\omega_P \times R_B^P \cdot I_B \cdot \omega_B + R_B^P \cdot (\omega_B^* \times I_B \cdot \omega_B) + R_B^P \cdot I_B \cdot \dot{\omega}_B = (R_P^E)^T \cdot (-R_B \times R_t) + U \\
R_B^P \cdot V_B = V_P + \omega_P \times l \\
R_B^P \cdot \left(\omega_B^* \times V_B + \dot{V}_B\right) = \left(\dot{V}_P + \dot{\omega}_P \times l\right) \\
\dot{p}_B = -\dfrac{\dot{v}_B}{R_B} \\
\dot{q}_B = \dfrac{\dot{u}_B}{R_B} \\
\dot{\omega} = 0
\end{cases}
$$

Occorre tener conto che il sistema non lineare finale può essere risolto (in *Matlab* tramite la chiamata alla funzione *solve*) rispetto ad un vettore di incognite, il seguente:

$$x_{solve} = \begin{bmatrix} \dot{V}_P \\ \dot{\omega}_P \\ \dot{V}_B \\ \dot{\omega}_B \\ R_t \\ R_v \\ V_P \end{bmatrix}$$

Il modello ottenuto è definito da un sistema di equazioni differenziali ordinarie non lineari.
La risoluzione rispetto a tali incognite definite dal vettore *x* genera una soluzione poco chiara ed utilizzabile: questo è dovuto al fatto che il sistema è altamente complesso ed alla scelta dell'utilizzo delle variabili *symbolic*.
Tale soluzione ricavata, inoltre, non è adottabile per scopi di controllo ma potrebbe essere utile per effettuare delle simulazioni di sistema.

A questo punto, è alquanto difficile sintetizzare un controllore per un modello così complesso. Questo è il motivo per cui per procedere con la parte di sintesi si passa prima per la linearizzazione del modello non lineare ricavato che permette una semplificazione notevole.

Modello finale del ballbot: Linearizzazione del sistema con vettore di stato e legge di controllo

Lo scopo finale è quello di arrivare ad una linearizzazione del sistema non lineare ottenuto e mostrato nel capitolo precedente.
Si vuole dunque ottenere la rappresentazione in spazio di stato del sistema ballbot nella forma

$$\dot{x} = A \cdot x + B \cdot u$$

che è un modello lineare.

Dalla rappresentazione in spazio di stato osserviamo che occorre introdurre due vettori per calcolare le matrici della forma lineare.
Il vettore *x* di stato ed il vettore *u* della legge di controllo, in ingresso al sistema.
Il vettore di stato è impostato come segue, poiché solamente le seguenti variabili hanno un significato fisico interessante allo scopo di controllo:

$$x = \begin{bmatrix} \phi \\ p \\ \theta \\ q \\ \psi \\ r \\ \phi_b \\ p_b \\ \theta_b \\ q_b \\ x_b \\ u_b \\ y_b \\ v_b \end{bmatrix}$$

Si utilizza *Matlab* per effettuare tutti i calcoli necessari.
Si vuole ricavare la forma lineare introducendo un vettore *u* di ingressi, ovvero la legge di controllo che forza il ballbot a porsi nella posizione con tutte le componenti del vettore di stato a zero che corrisponde al punto di equilibrio instabile per il sistema.

$$u = [u1 \;\; u2 \;\; u3]$$

In corrispondenza di tale posizione (tutti gli stati a zero) il ballbot ha la libertà di inclinarsi in qualsiasi direzione.
Il vettore u presenta tre componenti e nella fase successiva si vuole passare ad un controllo in tensione, in modo tale da avere in ingresso le tre tensioni V1, V2 e V3.

Una volta introdotti i due vettori x ed u è possibile ricavare con *Matlab* le due matrici dello spazio di stato lineare A e B.
Per la linearizzazione si può pensare ad impiegare lo sviluppo di *Taylor* arrestato al prim'ordine della matrice *A'* non lineare del sistema, calcolandone lo *jacobiano* e considerando trascurabile il residuo.

Nei capitoli seguenti si mostra lo sviluppo dello script in linguaggio *Matlab* per la definizione dei parametri e la stesura delle equazioni del sistema, fino all'implementazione del sistema non lineare finale e dei comandi adottati per la risoluzione.
I valori delle incognite calcolati tramite la funzione *solve* che risolve il sistema non vengono riportati in questa relazione, essendo espressioni particolarmente lunghe composte da una gran quantità di operazioni tra numerosi parametri.

Implementazione tramite Matlab: modello non lineare del sistema

Prima di tutto sullo *script* si dichiarano tutte le variabili ed i parametri che saranno utili per la costruzione del modello non lineare del sistema ballbot;
Tali parametri sono ricavati dalle equazioni ottenute nei capitoli precedenti che descrivono il sistema nei vari aspetti.
Si tiene in considerazione che si andrà a lavorare con grandezze simboliche.

Si dispone di alcune variabili numeriche che sono dichiarate in testa allo *script*.

```
syms phi p theta q psi r phib pb thetab qb xb ub yb vb real;
Rb_n=0.12;      % raggio della palla
rw_n=0.053;     % raggio delle ruote
mp_n=5.045;     % massa del pendolo (body)
g_n=9.81;       % accelerazione gravitazionale
mb_n=0.65;      % massa della palla
Ipx_n=0.47;     % Inerzia del corpo del pendolo (asse x)
Ipy_n=0.47;     % Inerzia del corpo del pendolo (asse y)
Ipz_n=0.049;    % Inerzia del corpo del pendolo (asse z)
Ibk_n=0.005;    % Inerzia della palla
l_n=0.410;      % Distanza tra il centro di massa del pendolo ed il centro di
massa della palla (centro della palla)
rp_n=0.14;      % Raggio del corpo del pendolo
alpha_n=2*pi/9; % Angolo di inclinazione dei motori
```

Occorre, successivamente, dichiarare le variabili simboliche del sistema con cui si andrà a lavorare per costruire il modello non lineare.

```
syms phi theta psi real;
```

Le variabili sopra indicate sono relative all'angolo di *roll* (*phi*), all'angolo di *pitch* (*theta*) ed all'angolo di *yaw* (*psi*).

```
syms Rb Rw mp g mb Ipx Ipy Ipz Ibk l alpha real
```

Con l'introduzione di tali variabili simboliche è possibile iniziare a costruire le matrici di rotazione ed introdurre il vettore di inerzia della palla *ball* (*Ib*).

```
Rz=[cos(psi) -sin(psi) 0;
    sin(psi) cos(psi)  0;
    0        0         1];
Ry=[cos(theta)  0  sin(theta);
    0           1  0;
    -sin(theta) 0  cos(theta)];

Rx=[1 0        0;
    0 cos(phi) -sin(phi);
    0 sin(phi) cos(phi)];

Rpe=Rz*Ry*Rx;

Rbp=Rx'*Ry';

Ib=[Ibk_n 0 0; 0 Ibk_n 0 ; 0 0 Ibk_n  ];
```

Si introducono i vettori *beta* associati alle tre ruote che indicano gli angoli di orientazione di ciascuna ruota rispetto al piano *x y* del sistema di riferimento relativo alla palla.

```
syms beta

beta1=beta;
beta2=beta+pi*2/3;
beta3=beta-pi*2/3;
```

Occorrono altre variabili (vettori) per introdurre posizioni ed angoli delle varie componenti di sistema.

```
syms x y z xb yb zb real
syms phib thetab psib u v w p q r ub vb wb pb qb rb real

z=0;

GAMMApe=[x y z]';

THpe=[phi theta psi]';

GAMMAbe=[xb yb zb]';

THbe=[phib thetab psib]';
```

Si dispone ora dei parametri necessari per scrivere le equazioni della dinamica che compongono il modello non lineare del ballbot.
Per far ciò si dichiarano nello *script* le grandezze che compaiono in tali equazioni definite nei capitoli precedenti.

```
syms dp dq dr dpb dqb drb  ub vb wb real

wp=[p;q;r];

Vp=[u;v;w];

Vb=[ub; vb; wb];

syms Ipx Ipy Ipz real

Ip=[Ipx 0 0; 0 Ipy 0;0 0 Ipz];
```

Il vettore *Ip* rappresenta l'inerzia del pendolo (sistema *p: pendulum*).

```
dwp=[dp; dq; dr];

wb=[pb; qb; rb];

dwb=[dpb; dqb;drb];

syms du dv dw dpvb dpub dpwb real
syms rtx rty rtz rvx rvy rvz real

Rt=[rtx rty rtz]';

Rv=[rvx rvy rvz]';

dVp=[du dv dw]';

dVb=[dpub dpvb dpwb]';
```

Si suppone la terza componente sia nulla: $dpwb = 0$ (la palla non ha accelerazione lungo l'asse z).

```
Gpe=[0 0 -mp*g]';

L=[0 0 -l_n]';

syms ux uy uz real

Usegn=[ux uy uz]';

k= Rb/Rw;
```

```
syms T1 T2 T3 real
syms u1 u2 u3

u1=-k*Usegn(1);

u2=-k*Usegn(2);

u3=-k*Usegn(3);

U=[u1;u2;u3];
```

I parametri *Ti* descrivono la coppia del motore relativo alla *i*-esima ruota.
u1, *u2*, *u3* sono le tre componenti del vettore u: la legge di controllo in ingresso al sistema.

```
Gbe=[0 0 -mb*g]';

RB=[0 0 -Rb]';

Fpp=mp*(cross(wp,Vp)+dVp);

taupp=cross(wp,(Ip*wp))+ Ip*dwp;

wb1=[-p -q 0]';
```

Per il sistema pendolo nello specifico si dichiarano i parametri F_{pp} vettore delle forze esterne e τ_{pp} vettore dei momenti torcenti.

```
Fbp=mb*((cross(wp,Rbp*Vb))+Rbp*((cross(wb1,Vb))+dVb));

taubp= (cross(wp,Rbp*Ib*wb))+Rbp*(cross(wb1,Ib*wb))+Rbp*Ib*dwb;
```

Si passa successivamente al sistema palla definendo il vettore F_{bp} delle forze esterne rispetto a tale sistema ed il vettore τ_{bp} dei momenti torcenti.

A questo punto sono disponibili nello *script* di *Matlab* tutti i parametri, le variabili simboliche e numeriche necessarie per descrivere il ballbot attraverso un sistema di equazioni non lineari.

A partire dal modello non lineare esposto nel capitolo *'Modello finale del ballbot: Costruzione del sistema non lineare'* a pagina (21) si costruiscono le varie equazioni del moto come segue.

```
S1= Fpp-(Rpe')*Gpe-Rv;

S2= taupp-(cross(L,Rv))+U;
```

Prima e seconda equazione non lineare del moto.
Il vettore U è la legge di controllo, G_{pe} è l'energia potenziale a cui è soggetto il sistema pendolo ed R_v è la reazione vincolare tra palla e pendolo.

```
S3= Fbp-(Rpe')*(Gpe+Rt)+Rv;
```

Il vettore R_t esprime la reazione vincolare tra suolo e palla.
La terza espressione del moto descrive la dinamica del sistema palla dove *cr* è un fattore di proporzionalità.

```
syms cr
S4= taubp-(Rpe')*(cross(RB,Rt))-U-[0;0;cr];
```

Il parametro simbolico *cr* è un fattore di proporzionalità.

```
S5=Rbp*(cross(wb1,Vb)+dVb)-dVp-(cross(dwp,L));
```

La quinta equazione del moto descrive il vincolo geometrico palla – pendolo in relazione all'accelerazione del sistema.

La sesta e la settima equazione non sono inserite nello *script* insieme alle altre ma per comodità, essendo di semplice formulazione, saranno inserite direttamente nel sistema non lineare.
Tali espressioni hanno la struttura seguente:

```
dpb=-dpvb/Rb;          (Equazione S6)

dqb=dpub/Rb;           (Equazione S7)
```

Non è presente l'equazione relativa a *dpwb* che, come è stato detto, assume valore nullo.

```
S8=Vp+cross(wp,L)-Rbp*Vb;
```

L'ottava equazione del moto descrive il vincolo geometrico sull'accelerazione tra palla e pendolo, indotto dal fatto che la distanza tra i rispettivi centri di massa è costante.

È possibile costruire il sistema finale non lineare.

```
SIS=[S1; S2; S3; S4; S5; dpb-dpvb/Rb; dqb+dpub/Rb; S8];
```

Attraverso *SIS* vengono incluse tutte le equazioni del moto sopra definite e si modella quindi la dinamica del ballbot.
Per risolvere il sistema si utilizza il comando *solve* rispetto al vettore di variabili:

Vettore delle incognite in forma espansa.

$$x_{solve} = \begin{bmatrix} \dot{V}_P \\ \dot{\omega}_P \\ \dot{V}_B \\ \dot{\omega}_B \\ R_t \\ R_v \\ V_P \end{bmatrix} = \begin{bmatrix} du \\ dv \\ dw \\ dp \\ dq \\ dr \\ dpub \\ dpvb \\ dpb \\ dqb \\ cr \\ rtx \\ rty \\ rtz \\ rvx \\ rvy \\ rvz \\ u \\ v \\ w \end{bmatrix}$$

Il parametro *drb* non compare nel vettore soluzione in quanto viene posto uguale a zero, non essendo rilevante per il modello in questione.

```
Sol=solve(SIS==0,du,dv,dw,dp,dq,dr,dpub,dpvb,dpb,dqb,cr,rtx,rty,rtz,rvx,rvy,rvz,
u,v,w);
```

```
du=Sol.du;
dv=Sol.dv;
dw=Sol.dw;

dp=Sol.dp;
dq=Sol.dq;
dr=Sol.dr;

dpub=Sol.dpub;
dpvb=Sol.dpvb;

dpb=Sol.dpb;
dqb=Sol.dqb;

cr=Sol.cr;

rtx=Sol.rtx;
rty=Sol.rty;
rtz=Sol.rtz;

rvx=Sol.rvx;
rvy=Sol.rvy;
rvz=Sol.rvz;

u=Sol.u;
v=Sol.v;
w=Sol.w;
```

In questo modo ad ognuna delle componenti del vettore soluzione ricercato verrà assegnato il valore calcolato dal programma *Matlab* al termine dell'esecuzione.

Come detto nei paragrafi precedenti non è possibile mostrare la soluzione ricavata del sistema non lineare, in quanto ha poco senso il calcolo della stessa;
A causa dell'utilizzo di variabili simboliche la soluzione ricavata è estremamente complessa, di difficile leggibilità e di scarso valore a scopi di controllo.

Si procede ora con il calcolo del modello linearizzato utilizzando il vettore di stato x appena ricavato ed il vettore u di ingresso, per arrivare alla forma linearizzata:

$$\dot{x} = A \cdot x + B \cdot u$$

Implementazione tramite Matlab: modello lineare del sistema

In questa sezione della relazione, per i motivi sopracitati, si ricava il modello del sistema ballbot linearizzato.

Il primo passo è sostituire nelle equazioni che compongono il sistema non lineare il valore *0* ai parametri che si è verificato essere nulli.
L'operazione di sostituzione in *Matlab* è effettuata tramite il comando *subs* richiamato esclusivamente su espressioni e valori simbolici.
La sostituzione è applicata separatamente per ciascuna equazione.

```
QuartaEQ=simplify(subs(Sol.dp,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

QuintaEQ=simplify(subs(Sol.dq,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

SestaEQ=simplify(subs(Sol.dr,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

SettimaEQ=simplify(subs(Sol.dpub,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

OttavaEQ=simplify(subs(Sol.dpvb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

NonaEQ=simplify(subs(Sol.dpb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));

DecimaEQ=simplify(subs(Sol.dqb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
```

Una volta modificati tutti i valori occorre aggiornare anche il sistema memorizzato nel *workspace*, sovrascrivendo le equazioni contenenti i valori precedenti la sostituzione.
La variabile *Sistema* sarà aggiornata come segue.

```
Sistema =

[p,QuartaEQ,q,QuintaEQ,r,SestaEQ,pb,NonaEQ,qb,DecimaEQ,ub,SettimaEQ,vb,OttavaEQ]
';
```

Allo scopo di ottenere un'approssimazione lineare del sistema per ricavarne la forma in spazio di stato nella forma $A \cdot x + B \cdot u$, occorre effettuare due operazioni.

Comando jacobian

Il primo comando da utilizzare è *jacobian*.
Sono richiesti in ingresso due parametri: una matrice *M* ed un vettore *v*.
Il comando calcola la matrice jacobiana di *M* rispetto a *v*, perciò l'elemento in posizione (i, j) di tale matrice risultante sarà:

$$m_{i,j} = \frac{\partial f(i)}{\partial v(j)}$$

Per effettuare la linearizzazione del sistema occorre calcolare inoltre l'espansione di Taylor arrestata al prim'ordine sia della matrice jacobiana ottenuta derivando rispetto al vettore di stato *x* di cui si dispone (per calcolare la matrice della dinamica lineare A), sia derivando rispetto alle tensioni v_i date in ingresso dai motori (per calcolare la matrice lineare B relativa agli ingressi del sistema).
Per ottenere la matrice B occorreranno una serie di passaggi da aggiungere allo *script* per porre il sistema in relazione alle tensioni in ingresso ricavandole dalle equazioni delle coppie dei motori.
Allo scopo di linearizzare tramite Taylor occorre, dunque, avere a disposizione la matrice jacobiana del sistema ovvero sia la derivata rispetto alle variabili di stato, sia la derivata rispetto alle variabili di ingresso (rispettivamente per le matrici lineari A e B desiderate).

Si parte dalla matrice della dinamica A e si applica il comando *jacobian* su *Sistema* rispetto al vettore di stato *x= [phi p theta q psi r phib pb thetab qb xb ub yb vb]*.

```
Matrice = jacobian(Sistema,[phi p theta q psi r phib pb thetab qb xb ub yb vb]);
```

La variabile *Matrice* contiene appunto la matrice jacobiana del sistema rispetto al vettore *x*.

Si potrà effettuare lo stesso calcolo anche su B una volta però definito il vettore rispetto al quale si vuole derivare, ovvero il vettore riga T=[T1;T2;T3] che rappresenta le coppie in ingresso al ballbot generate dai motori che lo controllano.

In ogni caso, su tale *Matrice* si calcola, come detto in precedenza, lo sviluppo di Taylor arrestato al prim'ordine per ottenere il sistema nella forma linearizzata.

Il linguaggio *Matlab* permette di calcolare tale espansione in serie mediante l'apposito comando *taylor*: tale funzione calcola i primi termini della serie di Taylor data in input rispetto alle variabili del vettore fornito attorno al punto di lavoro desiderato.

Comando taylor

Si dispone della matrice jacobiana di sistema relativamente al vettore *x* di stato.
Si aggiunge allo *script*:

```
MatriceLinearizzata=taylor(Matrice,[phi p theta q psi r phib pb thetab qb xb ub
yb vb],'ExpansionPoint',0,'Order',1);

simplify(MatriceLinearizzata);
```

La variabile *MatriceLinearizzata* è la matrice della dinamica A lineare ricercata.
I termini della serie da calcolare sono forniti dalla variabile *Matrice* (ricordando che è la matrice jacobiana del sistema rispetto ad *x*) rispetto alle variabili del vettore di stato *x* attorno al punto di lavoro che si sceglie in $x0 = 0$ (definito dall'opzione *ExpansionPoint*).

Volendo arrestare l'espansione al primo ordine l'opzione *Order* permette di scegliere tale ordine di arresto, che sarà dunque impostato ad *1*.

Matrice A lineare (senza motore)

$$A(2,1) = \frac{g\ l\ m_p(I_b + R_b^2 m_b + R_b^2 m_p)}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = A(4,3)$$

$$A(8,1) = \frac{R_b g l^2 m_p^2}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = A(10,3)$$

$$A(14,1) = \frac{R_b^2 g l^2 m_p^2}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = -A(12,3)$$

La matrice che si ottiene in questo modo è la seguente:

$$A = \begin{vmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
33.2666 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 33.2666 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
94.9019 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 94.9019 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -11.3882 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
11.3882 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}$$

Calcolo della matrice B lineare rispetto alle coppie T in ingresso al sistema

Nel paragrafo precedente si è parlato di coppie e tensioni fornite dai motori che controllano il ballbot tramite gli ingressi.

Per ricavare la matrice B linearizzazione del sistema non lineare rispetto a tali ingressi, occorre prima definirli nello script.
Come primo passo si associa al vettore di ingresso $u = [u_x\ u_y\ u_z]$ le coppie fornite dai motori in ingresso, in modo che il sistema diventi con la linearizzazione:

$\dot{x} = A \cdot x + B \cdot T$ Coppie T in ingresso

Dopodiché, calcolando l'espressione che lega le coppie alle tensioni erogate dai motori, si può ottenere la forma finale dello spazio di stato con dei cambi di variabile che avrà le tensioni come ingresso:

$\dot{x} = A \cdot x + B \cdot V$ Tensioni V in ingresso (forma finale)

Occorre assicurarsi, prima di procedere, che nello *script* siano già state dichiarate tutte le variabili ed eventualmente i parametri numerici che serviranno per definire le coppie e le tensioni.
Per far ciò, per sicurezza, si dichiarano le variabili tramite il comando *syms* che compaiono nelle espressioni di T e V.

```
syms phi p theta q psi r phib pb thetab qb xb ub yb vb real

syms phi theta psi real

syms Rb Rw mp g mb Ipx Ipy Ipz Ibk l alpha real

syms beta real

syms x y z xb yb zb real

syms phib thetab psib u v w p q r ub vb wb pb qb rb real

syms dp dq dr dpb dqb drb  ub vb wb real

syms Ipx Ipy Ipz real

syms du dv dw dpvb dpub dpwb real

syms rtx rty rtz rvx rvy rvz real

syms ux uy uz real

syms T1 T2 T3 real

syms u1 u2 u3 real

syms val1 val2 val3 real
```

Le variabili *val1*, *val2*, *val3* sono introdotte per definire le coppie.
Si parte dando in ingresso al sistema la coppia vettoriale *T* fornita alle omniwheels ed erogata dai motori che controllano il ballbot.
Dunque il vettore *u* degli ingressi dovrà avere le componenti legate a tali coppie, secondo la relazione seguente:

Correzione rispetto al documento n° 1:

mancano i coefficienti T2.

$$u = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \cdot [T_1 \cdot \cos(\beta) + T_2 \cdot \cos\left(\beta + \frac{2}{3\pi}\right) + T_3 \cdot \cos\left(\beta - \frac{2}{3\pi}\right)] \\ \cos(\alpha) \cdot [(T_1 \cdot \sin(\beta) + T_2 \cdot \sin\left(\beta + \frac{2}{3\pi}\right) + T_3 \cdot \sin\left(\beta - \frac{2}{3\pi}\right)] \\ -\sin(\alpha) \cdot (T_1 + T_2 + T_3) \end{bmatrix}$$

21) Equazione ingresso u

val_i è la variabile relativa alla componente *i*-esima del vettore *u* sopra definito.
Per cui si inseriscono le espressioni delle coppie nello *script*.

```
val1= cos(alpha)*(T1*cos(beta)+T2*cos(beta2)+T3*cos(beta3));

val2= cos(alpha)*(T1*sin(beta)+T2*sin(beta2)+T3*sin(beta3));

val3= -sin(alpha)*(T1+T2+T3);
```

Le coppie T1, T2, T3 sono state definite.
Attualmente, oltre ai parametri con valore noto nullo, si conoscono anche i valori dei parametri u_x, u_y, u_z e vanno aggiornati come fatto in precedenza richiamando la funzione *subs* sulle equazioni del sistema che contengono le variabili da aggiornare.

```
BQuartaEQ=simplify(subs(Sol.dp,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,0,
0,0}));

BQuintaEQ=simplify(subs(Sol.dq,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,0,
0,0}));

BSestaEQ=simplify(subs(Sol.dr,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,0,0
,0}));

BSettimaEQ=simplify(subs(Sol.dpub,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0
,0,0,0}));

BOttavaEQ=simplify(subs(Sol.dpvb,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,
0,0,0}));

BNonaEQ=simplify(subs(Sol.dpb,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,0,0
,0}));

BDecimaEQ=simplify(subs(Sol.dqb,{ux,uy,uz,dpwb,drb,rb,Vb(3)},{val1,val2,val3,0,0
,0,0}));
```

Le coppie sono state introdotte quindi nel modello: prestare attenzione al fatto che i valori da aggiornare sono quelli delle equazioni del sistema non lineare di partenza.
La linearizzazione verrà eseguita su tale sistema che ora dipende dai parametri delle coppie (compaiono nelle equazioni sovrastanti).

```
BSistema =
[p,BQuartaEQ,q,BQuintaEQ,r,BSestaEQ,pb,BNonaEQ,qb,BDecimaEQ,ub,BSettimaEQ,vb,BOt
tavaEQ]';
```

La variabile *BSistema* è, dunque, il sistema finale aggiornato (non lineare).

Questa volta è possibile calcolare la prima linearizzazione della matrice B rispetto agli ingressi, ovvero le coppie che ora compaiono nel modello.
Analogamente a quanto fatto per A rispetto al vettore di stato *x* si dovrà prima calcolare la matrice jacobiana di B rispetto al vettore degli ingressi (coppie):

$$T = [T1|T2\ |T3]$$

I passaggi sono analoghi.
Per prima cosa, si richiama il comando *jacobian* sul nuovo sistema non lineare (variabile *BSistema*) derivando rispetto al vettore *T* delle coppie, ovvero degli ingressi.

```
MatriceB = jacobian(BSistema,[T1 T2 T3]);
```

Dopodiché si può calcolare l'espansione di Taylor arrestata al prim'ordine della variabile *MatriceB* appena ottenuta, con il comando *taylor* definito come segue:

```
MatriceLinearizzataB=taylor(MatriceB,[beta phi p theta q psi r phib pb thetab qb
xb ub yb vb T1 T2 T3],'ExpansionPoint',0,'Order',1);
```

Il risultato è la variabile *MatriceLinearizzataB* che proprio la matrice B del sistema lineare in spazio di stato con in ingresso le coppie generate dai motori.

$$\dot{x} = A \cdot x + B \cdot T$$

Matrice B lineare (senza motore)

I valori parametrici ricavati per la matrice B sono i seguenti.

$$B(2,1) = \frac{R_b \cos(\alpha)(I_b + R_b^2 m_b + R_b^2 m_p + R_b l m_p)}{R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)}$$

$$B(2,2) = \frac{-R_b \cos(\alpha)(I_b + R_b^2 m_b + R_b^2 m_p + R_b l m_p)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = -\frac{1}{2} B(2,1) = B(2,3)$$

$$B(4,2) = \frac{\sqrt{3}\, R_b \cos(\alpha)(I_b + R_b^2 m_b + R_b^2 m_p + R_b l m_p)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = -\frac{\sqrt{3}}{2} B(2,1) = -B(4,3)$$

$$B(6,1) = \frac{-R_b \sin(\alpha)}{I_p R_w} = B(6,2) = B(6,3)$$

$$B(8,1) = \frac{-R_b \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)}$$

$$B(8,2) = \frac{R_b \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = -\frac{1}{2} B(8,1) = B(8,3)$$

$$B(10,2) = \frac{-\sqrt{3}\, R_b \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = \frac{\sqrt{3}}{2} B(8,1) = -B(10,3)$$

$$B(12,2) = \frac{-\sqrt{3}\, R_b^2 \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = R_b\, B(10,2) = -B(12,3)$$

$$B(14,1) = \frac{R_b^2 \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = -R_b\, B(8,1)$$

$$B(14,2) = \frac{-R_b^2 \cos(\alpha)(I_p + R_b l m_p + m_p l^2)}{2R_w(I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b)} = -\frac{1}{2} B(14,1) = B(14,3)$$

Anche in questo caso ne vengono sostituiti i valori numerici tramite una operazione di subs, così da ottenere la seguente matrice degli ingressi:

B =

0	0	0
10.9554	-5.4777	-5.4777
0	0	0
0	9.4876	-9.4876
0	0	0
-29.7014	-29.7014	-29.7014
0	0	0
-51.1874	25.5937	25.5937
0	0	0
0	-44.3296	44.3296
0	0	0
0	-5.3196	5.3196
0	0	0
6.1425	-3.0712	-3.0712

Inserimento delle equazioni del motore. Modello non lineare e linearizzazione

Il secondo ed ultimo passaggio è quello di ottenere il sistema in spazio di stato in forma lineare ed in relazione alle tensioni di armatura fornite in ingresso dai motori:

$$\dot{x} = A \cdot x + B \cdot V$$

Non si dispone ancora dei parametri relativi alle tensioni: occorre quindi dichiarare eventuali variabili da introdurre nello *script* ed inserire le espressioni che legano matematicamente coppie e tensioni dei motori.

Conoscendo il modello del motore *DC* (motore elettrico in corrente continua) e considerando il rapporto di trasmissione del *gear-box* (riduttore) sull'equazione di coppia di ogni motore si prosegue come segue.
Combinando l'equazione meccanica in uscita dal *gear-box* con l'equazione elettrica di ciascun motore, si ricava un'espressione finale che lega la coppia utile sul carico T alla velocità angolare di rotazione delle ruote (azionate dai motori) ed alla tensione V_a dei motori.

$$T = -\left(\frac{k_m \cdot k_e \cdot n^2 \cdot \eta_m \cdot \eta_G - K_f \cdot R}{R}\right) \cdot \dot{\theta} + \left(\frac{k_m \cdot n \cdot \eta_m \cdot \eta_G}{R}\right) \cdot V_a$$

22) Equazione della coppia utile sul carico (motore)

Le costanti che compaiono nell'espressione sono:
R resistenza del motore.
$k_m = k_e = k$ costanti del motore (si suppongono quella elettrica e quella meccanica coincidenti).
η_m, η_G, K_f, n parametri derivanti dall'equazione dinamica del riduttore.

Le ipotesi fatte per costruire l'equazione si basano sulla semplificazione del modello del motore in corrente continua: si considera che la dinamica elettrica del motore sia trascurabile per cui il valore del parametro *L* (induttanza) sia circa nullo;
Inoltre, per ottenere una semplificazione in più, si pone K_f anch'esso circa uguale a zero.

Il modello appena descritto è valido se si considera che il vettore degli ingressi sia una terna di coppie motrici, le quali vengono generate dai tre motori.
Quando si necessita di modellare in maniera specifica l'ingresso di controllo, come nel caso in esame, vanno sostituiti i valori della terna di coppie generiche T1 T2 e T3 con dei valori che corrispondano alla specifica coppia generata dagli specifici motori utilizzati.

Prendendo l'equazione del motore con T si intende la coppia utile sul carico, Km=Ke=K costanti relative al motore, Kf può essere considerata circa pari a zero per avere un modello più semplificato.
Questa equazione deriva dal modello semplificato del motore DC in cui si fa l'ipotesi che L sia circa pari a zero.
Va risulta essere la tensione di armatura del motore, e sarà quindi la nuova variabile di ingresso del sistema.

L'inserimento del motore provoca una differenza nel modello, visto che i nuovi ingressi del sistema dovranno essere le tre alimentazioni del motore e non più le tre coppie.
Per legare le equazioni del motore al resto del sistema si andranno a sostituire i valori generici dell'ingresso nel nostro sistema, ovvero *ux uy uz* utilizzando le formule appropriate della coppia.
L'inserimento del motore si tradurrà nell'aggiunta di tre equazioni al sistema le quali hanno lo scopo di legare la variabile $\ddot{\theta}$ al sistema discusso sopra.

In altre parole, la struttura del sistema iniziale (ovvero senza l'aggiunta delle equazioni del motore), si può sintetizzare come un sistema Σ_0 nel quale compaiono i generici ingressi ux uy ed uz.

$$\left\{ \quad \Sigma_0 \qquad \text{dipendente da} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \right.$$

Quello che va fatto è sostituire questi valori con la formula (21.
In questo modo il sistema Σ_0 viene trasformato in un sistema Σ_0' dipendente dalle coppie T1 T2 e T3. A questo punto le coppie possono essere scritte considerando che la generica coppia Ti si può riscrivere come: $T_i = I_{wi}\ddot{\theta}_i$ dove I_{wi} è il momento di inerzia della ruota (uguale per tutte e tre) mentre $\ddot{\theta}$ è la sua accelerazione angolare. Il sistema avrà ora la seguente forma:

$$\left\{ \quad \Sigma_0' \qquad \text{dipendente da} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} \right.$$

A questo punto possono essere inserite le equazioni che avranno lo scopo di definire $\ddot{\theta}$ in relazione alle altre variabili del sistema e a quelle di ingresso (V1 V2 e V3), che hanno la forma dell'equazione (21).
Il sistema finale sarà quindi della forma:

$$\begin{cases} \Sigma_0' \qquad \text{dipendente da} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} \\ \Sigma_1 \quad = \begin{bmatrix} I_w \ddot{\theta}_1 \\ I_w \ddot{\theta}_2 \\ I_w \ddot{\theta}_3 \end{bmatrix} = \begin{bmatrix} -k\dot{\theta}_1 + qV_1 \\ -k\dot{\theta}_1 + qV_2 \\ -k\dot{\theta}_1 + qV_3 \end{bmatrix} \end{cases}$$

Il sistema risultante è di 17 equazioni anziché 14, ma può essere comunque riscritto nella forma $\dot{x} = Ax + Bu$ considerando le ultime tre equazioni come quelle che definiscono $\ddot{\theta}$ in relazione a $\dot{\theta}$.

Anche in questo caso viene considerata la linearizzazione del sistema appena mostrato, mediante il calcolo della matrice Jacobiana, stavolta rispetto al vettore di stato di 17 variabili, ovvero tutte quelle che erano presenti nel caso precedente in aggiunta alle tre velocità angolari dei motori $(\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3)$.

Anche in questo caso viene considerato lo sviluppo in serie di Taylor arrestato al primo ordine in modo tale da considerare la sola componente lineare.

Analogamente a quanto fatto nel capitolo precedente dove la linearizzazione, ovvero l'espansione in serie di Taylor arrestata al prim'ordine, avveniva dando in ingresso il vettore *T* della coppia utile sul carico, anche in questo caso vanno dichiarati i parametri e le variabili mancanti da aggiungere l'equazione allo *script*.

```
syms wm1 wm2 wm3 real  % velocità angolari dei motori

Rwp=Rz'*Ry';

syms EQ1 EQ2 EQ3 real

syms Iwx Iwy Iwz real  % Inerzia delle omni-wheels

syms dwm1 dwm2 dwm3 real  % accelerazioni angolari dei motori

Iw= [ Iwx 0 0
        0 Iwy 0         % matrice di inerzia delle omni-wheels
        0 0 Iwz ];

syms km ke kf R n etam etag real % parametri del motore

syms V1 V2 V3 real % tensioni dei motori

T(1)=-((km*ke*n^2*etam*etag-kf*R)/R)*wm1 + ((km*n*etam*etag)/R)*V1;
T(2)=-((km*ke*n^2*etam*etag-kf*R)/R)*wm2 + ((km*n*etam*etag)/R)*V2;
T(3)=-((km*ke*n^2*etam*etag-kf*R)/R)*wm3 + ((km*n*etam*etag)/R)*V3;
% T: vettore delle coppie utili (3 componenti)
```

Si reintroducono le equazioni delle coppie *T* inserendo i parametri *wm1, wm2, wm3* come variabili che rappresentano le tre velocità angolari dei motori, contenuti rispettivamente in *T1, T2, T3* (una in ogni componente del vettore delle coppie).

Si osserva che la coppia è legata da tali velocità di rotazione dei motori e dalle tensioni di armatura fornite dagli stessi.
Le velocità saranno un parametro in più che compone il sistema non lineare che modella il ballbot. Nel vettore della dinamica di stato $\dot{x}_s$, dunque, si aggiungeranno tre variabili che sono le tre accelerazioni angolari dei motori *dwm1, dwm2, dwm3* (derivate delle velocità angolari).

A questo punto il nuovo vettore delle variabili di stato diventa:

```
Xs=[phi,p,theta,q,psi,r,phib,pb,thetab,qb,xb,ub,yb,vb,wm1,wm2,wm3];
```

Modello con motore non lineare

Occorre apportare delle modifiche al modello precedente, senza inserimento del motore, al fine di ottenere il sistema lineare in spazio di stato avente in ingresso le tensioni di armatura V dei motori (controllo in tensione).
Si calcola nuovamente la linearizzazione delle matrici A e B, dato che molti parametri sono stati aggiornati e vanno inserite nuove espressioni che li legano tra loro, tra cui i parametri dei motori di cui il modello precedente non tiene conto.
Essi vengono introdotti nel sistema e compariranno nelle nuove matrici lineari dello spazio di stato.
Il sistema potrà essere controllato dai motori tramite le tensioni ed avrà la seguente forma finale.

$$\dot{x} = A \cdot x + B \cdot V$$

```
%%% EQUAZIONI DEL SISTEMA

% Dinamica del pendolo soggetta agli ingressi U, all'energia potenziale Gpe
% e alla forza di reazione palla-pendolo, Rv

S1= Fpp-(Rpe')*Gpe-Rv;

S2= taupp-(cross(L,Rv))+U;

% Dinamica della palla che include l'effetto della forza di reazione
% palla-suolo,Rt

S3= Fbp-(Rpe')*(Gpe+Rt)+Rv;

%Inserisco il fattore di proporzionalità cr
syms cr

S4= taubp-(Rpe')*(cross(RB,Rt))-U-[0;0;cr];

% Vincolo geometrico tra palla e pendolo per l'accelerazione

S5= Rbp*(cross(wb1,Vb)+dVb)-dVp-(cross(dwp,L));

% Vincolo geometrico tra palla e pendolo per l'accelerazione

S8= Vp+cross(wp,L)-Rbp*Vb;

% Equazioni del motore

S9= Iwx*dwm1+((km*ke*n^2*etam*etag-kf*R)/R)*wm1-((km*n*etam*etag)/R)*V1;

S10= Iwy*dwm2+((km*ke*n^2*etam*etag-kf*R)/R)*wm2-((km*n*etam*etag)/R)*V2;

S11= Iwz*dwm3+((km*ke*n^2*etam*etag-kf*R)/R)*wm3-((km*n*etam*etag)/R)*V3;
```

Per introdurre la dinamica dei tre motori saranno inserite tre equazioni in più: *S9*, *S10*, *S11*.
Esse sono ricavate dall'espressione mostrata precedentemente che lega la coppia *T* applicata al carico, alle velocità angolari *wm* ed alle tensoni *V* erogate dai motori.
A tale espressione si applica una sostituzione, come segue.

$$T = I_{wz} \cdot \ddot{\theta}$$

I_{wz} : è il momento d'inerzia delle omni-wheels del ballbot.

$\ddot{\theta}_i$: è l'accelerazione angolare della ruota i-esima.

Si procede calcolando il valore dell'inerzia delle omni-wheels considerando la formula del momento d'inerzia di una corona circolare, conoscendo i seguenti dati:

$$massa\ della\ ruota\ (m_w) = 107g$$

$$raggio\ esterno\ della\ ruota\ (R_w) = 53mm$$

$$raggio\ interno\ della\ ruota\ (R_{interno}) = 38mm$$

In questo modo si dispone del valore numerico da sostituire a *Iwz* per ricavare le matrici A della dinamica e B senza parametri simbolici.

$$I_z = m_w \cdot \frac{(R_w^2 + R_{interno}^2)}{2}$$

Il sistema al di fuori delle tre equazioni aggiuntive del motore assume la stessa struttura del sistema non lineare ricavato nei capitoli precedenti, con una differenza.

```
SIS= [S1; S2; S3; S4; S5; dpb-dpvb/Rb; dqb+dpub/Rb; S8];
```

Nelle equazioni del sistema non devono comparire i parametri *ux*, *uy*, *uz*;
Per la sostituzione di tali parametri, si utilizza l'equazione che li lega alle coppie *T*.

Dopo questa sostituzione nel sistema compariranno solamente le coppie T1, T2 e T3.
La relazione che li lega è quella sottostante.

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \cdot [T1 \cdot \cos(\beta_1) + T2 \cdot \cos(\beta_2) + T3 \cdot \cos(\beta_3)] \\ \cos(\alpha) \cdot [T1 \cdot \sin(\beta_1) + T2 \cdot \sin(\beta_2) + T3 \cdot \sin(\beta_3) \\ -\sin(\alpha) \cdot (T1 + T2 + T3) \end{bmatrix}$$

Per legare l'accelerazione e la velocità angolare di ogni motore, come è stato detto, si effettua una seconda sostituzione secondo la legge:

$$T = I_{wz} \cdot \ddot{\theta}$$

Dato che il sistema di riferimento si pone con l'asse *z* passante per il centro delle ruote, la coppia generata dalle stesse è semplicemente legata al momento d'inerzia *Iwz* (che sarà dunque una costante), ovvero:

Una volta effettuata questa sostituzione nel sistema sopra definito compariranno solamente le variabili $\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3$ al posto delle coppie.

```
% SOSTITUZIONE ux, uy, uz

SIS=
subs(SIS,{ux,uy,uz},{cos(alpha)*(T1*cos(beta)+T2*cos(beta2)+T3*cos(beta3)),cos(a
lpha)*(T1*sin(beta)+T2*sin(beta2)+T3*sin(beta3)),-sin(alpha)*(T1+T2+T3)});
```

Con tali sostituzioni si effettua il passaggio da un sistema controllato mediante l'azione delle coppie, ad un sistema controllato in tensione.

```
% SOSTITUZIONE T1,T2,T3

SIS= subs(SIS,{T1,T2,T3},{dwm1*Iwx,dwm2*Iwy,dwm3*Iwz});
```

È possibile ricavare il sistema finale che contiene le equazioni ed i parametri del motore, aggiungendo le tre espressioni *S9*, *S10*, *S11* sopra dichiarate.

```
SIS=[SIS;S9;S10;S11];
```

La risoluzione di un sistema caratterizzato da tutti parametri simbolici è dispendioso in termini di tempo e risorse.
È preferibile, dunque, sostituire prima i valori numerici relativi a ciascun parametro e poi proseguire con la risoluzione tramite il comando *solve* rispetto alle variabili di sistema.

```
%sostituzione delle costanti

SISn=subs(SIS,{Ibk,Ipx,Ipy,Ipz,Rb,Rw,dpwb,drb,l,mb,rb,Vb(3),mp,g,alpha},{Ibk_n,I
px_n,Ipy_n,Ipz_n,Rb_n,rw_n,0,0,l_n,mb_n,0,0,mp_n,g_n,alpha_n});


Sol=solve(SISn==0,du,dv,dw,dp,dq,dr,dpub,dpvb,dpb,dqb,cr,rtx,rty,rtz,rvx,rvy,rvz
,u,v,w,dwm1,dwm2,dwm3);


du=Sol.du;
dv=Sol.dv;
dw=Sol.dw;
dp=Sol.dp;
dq=Sol.dq;
dr=Sol.dr;
dpub=Sol.dpub;
dpvb=Sol.dpvb;

dpb=Sol.dpb;
dqb=Sol.dqb;
cr=Sol.cr;
rtx=Sol.rtx;
rty=Sol.rty;
rtz=Sol.rtz;
rvx=Sol.rvx;
rvy=Sol.rvy;
rvz=Sol.rvz;

u=Sol.u;
v=Sol.v;
w=Sol.w;

dwm1=Sol.dwm1;
dwm2=Sol.dwm2;
dwm3=Sol.dwm3;
```

Tra le variabili rispetto alle quali risolvere il sistema non compaiono le velocità angolari, ma le accelerazioni dei motori.
Questo perché l'oggetto che si sta calcolando è la dinamica del modello, ovvero il vettore $\dot{x}$ delle derivate delle componenti del vettore X_s delle variabili di stato (nel quale, invece, le ultime componenti sono le velocità dei motori).

```
QuartaEQ=(subs(Sol.dp,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
QuintaEQ=(subs(Sol.dq,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
SestaEQ=(subs(Sol.dr,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
SettimaEQ=(subs(Sol.dpub,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
```

```
OttavaEQ=(subs(Sol.dpvb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
NonaEQ=(subs(Sol.dpb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));
DecimaEQ=(subs(Sol.dqb,{dpwb,drb,rb,Vb(3)},{0,0,0,0}));


%%% SISTEMA CON MOTORE

nuovosis =
[p,QuartaEQ,q,QuintaEQ,r,SestaEQ,pb,NonaEQ,qb,DecimaEQ,ub,SettimaEQ,vb,OttavaEQ,
dwm1,dwm2,dwm3]';
```

La variabile *nuovosis* è il modello ricavato da linearizzare e dal quale ricavare le matrici dello spazio di stato A e B.
I parametri numerici del motore vanno dichiarati nello *script* per poter essere sostituiti alle variabili simboliche del modello.
Essi sono frutto di una identificazione parametrica effettuata sul motore a spazzole *Pololu1446*.

```
% sostituzione dei valori (motore)

etam_n=1;   % rendimenti adimensionali
etag_n=1;
n_n=1;      % costante di proporzione adimensionale
km_n=5.88;  % sostituzione
ke_n=0.07;
kf_n=0;
R_n=4.8;    % resistenza del motore (Ohm)

% Inerzia delle omni-wheels

Iwz_n= 2.2754e-04;      % (kg m^2)
```

Si effettua la sostituzione di tutti i valori numerici, per eliminare i parametri simbolici sul sistema appena calcolato *nuovosis*.

```
%%% nuovosis: SISTEMA SU CUI APPLICARE jacobian E taylor.

nuovosis=simplify(subs(nuovosis,{alpha,Ibk,Ipx,Ipy,Ipz,Rb,Rw,dpwb,drb,l,mb,rb,Vb
(3),mp,g,T1,T2,T3,etam,etag,n,km,ke,kf,R,Iwx,Iwy,Iwz},{alpha_n,Ibk_n,Ipx_n,Ipy_n
,Ipz_n,Rb_n,rw_n,0,0,l_n,mb_n,0,0,mp_n,g_n,T(1),T(2),T(3),etam_n,etag_n,n_n,km_n
,ke_n,kf_n,R_n,Iwx_n,Iwy_n,Iwz_n}));
```

Linearizzazione del modello con motore: calcolo della matrice A lineare

Analogamente a quanto fatto per il modello senza motore si procede con il sistema contenente i parametri e le equazioni del motore con il calcolo delle due matrici A e B dello spazio di stato in forma lineare.
Partendo dalla matrice A, si calcola matrice jacobiana del modello non lineare richiamando il comando *taylor* su *nuovosis* rispetto alle variabili di sistema.

```
%%% MATRICE A LINEARIZZATA (con motore)
% aggiunta variabili di stato: wm1, wm2, wm3

A = jacobian(nuovosis,[phi p theta q psi r phib pb thetab qb xb ub yb vb wm1 wm2
wm3]);
```

Il passo immediatamente successivo è richiamare il comando *taylor* sulla jacobiana ottenuta considerando l'espansione in serie arrestata al prim'ordine con tutte le variabili di stato tendenti al valore zero.

```
matA=taylor(A,[beta phi p theta q psi r phib pb thetab qb xb ub yb vb wm1 wm2
wm3],'ExpansionPoint',0,'Order',1);

matA=double(matA)
```

Matrice A lineare con valori simbolici

Il calcolo della matrice A con valori simbolici deve avvenire senza effettuare la sostituzione dei valori numerici di ciascuna variabile che compare nella matrice.
Omettendo il comando:

```
% %%% sostituzione delle costanti
SISn=subs(SIS,{Ibk,Ipx,Ipy,Ipz,Rb,Rw,dpwb,drb,l,mb,rb,Vb(3),mp,g,alpha},{Ibk_n,I
px_n,Ipy_n,Ipz_n,Rb_n,rw_n,0,0,l_n,mb_n,0,0,mp_n,g_n,alpha_n});
```

Il comando *taylor* non riesce a supportare il calcolo con una tale quantità di variabili simboliche per ciascuna espressione.
Si sostituisce all'espansione di Taylor al prim'ordine l'istruzione sottostante, che considera pari a zero le variabili rispetto alle quali si linearizza.

```
%%% VARIABILI SIMBOLICHE
matAs= subs(A,{beta phi p theta q psi r phib pb thetab qb xb ub yb vb wm1 wm2
wm3 V1 V2 V3},{0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0});
matAs= simplify(matAs);
```

Si ricava la matrice con i coefficienti sottostanti.

$$A(2,1) = \frac{g\ l\ m_p(R_b^2 m_b + R_b^2 m_p + I_b)}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = A(4,3)$$

$$A(8,1) = \frac{-(R_b g l^2 m_p^2)}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = A(10,3)$$

$$A(14,1) = \frac{R_b^2 g l^2 m_p^2}{I_b I_p + I_p R_b^2 m_b + I_p R_b^2 m_p + I_b l^2 m_p + R_b^2 l^2 m_p m_b} = -A(12,3)$$

In più nelle ultime tre colonne si ottengono i coefficienti dovuti all'introduzione delle equazioni del motore.

$$A(2,15) = -\frac{Rb\ ke\ km\ cos(alpha)(Rb^2\ mb + Ibk\ +\ Rb^2\ mp -\ Rb\ l\ mp)}{R\ Rw\ (Ipx\ Rb^2\ mb +\ Ibk\ Ipx\ +\ Ipx\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -2 \cdot A(2,16)$$
$$= -2 \cdot A(2,17)$$

$$A(4,16) = -\frac{3^{\frac{1}{2}}\ Rb\ ke\ km\ cos(alpha)\ (Rb^2\ mb +\ Ibk\ +\ Rb^2\ mp -\ Rb\ l\ mp)}{2\ R\ Rw\ (Ipy\ Rb^2\ mb +\ Ibk\ Ipy\ +\ Ipy\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -A(4,17)$$

$$A(6,15) = \frac{Rb\ ke\ km\ sin(alpha)}{Ipz\ R\ Rw} = A(6,16) = A(6,17)$$

$$A(8,15) = \frac{Rb\ ke\ km\ cos(alpha)(mp\ l^2 - Rb\ mp\ l\ +\ Ipx)}{R\ Rw\ (Ipx\ Rb^2\ mb +\ Ibk\ Ipx\ +\ Ipx\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -2 \cdot A(8,16)$$
$$= -2 \cdot A(8,17)$$

$$A(10,16) = \frac{3^{\frac{1}{2}}\ Rb\ ke\ km\ cos(alpha)(mp\ l^2 -\ Rb\ mp\ l\ +\ Ipy)}{2\ R\ Rw\ (Ipy\ Rb^2\ mb +\ Ibk\ Ipy\ +\ Ipy\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -A(10,17)$$

$$A(12,16) = \frac{3^{\frac{1}{2}}\ Rb^2\ ke\ km\ cos(alpha)(mp\ l^2 -\ Rb\ mp\ l\ +\ Ipy)}{2\ R\ Rw\ (Ipy\ Rb^2\ mb +\ Ibk\ Ipy\ +\ Ipy\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -A(12,17)$$

$$A(14,15) = -\frac{Rb^2\ ke\ km\ cos(alpha)(mp\ l^2 -\ Rb\ mp\ l\ +\ Ipx)}{R\ Rw\ (Ipx\ Rb^2\ mb +\ Ibk\ Ipx\ +\ Ipx\ Rb^2\ mp +\ Ibk\ l^2\ mp\ +\ Rb^2\ l^2\ mb\ mp)} = -2 \cdot A(14,16)$$
$$= -2 \cdot A(14,17)$$

$$A(15,15) = -\frac{ke\ km}{Iwz\ R}$$

$$A(16,16) = -\frac{ke\ km}{Iwz\ R}$$

$$A(17,17) = -\frac{ke\ km}{Iwz\ R}$$

Ovvero:

$$A(15,15) = A(16,16) = A(17,17)$$

Linearizzazione del modello con motore:
calcolo della matrice B lineare e matrici C,D

Si ripete la stessa procedura per la matrice B e si dichiarano le matrici C e D (con dimensioni adeguate, C matrice identità, D matrice nulla).

```
%%% MATRICE B LINEARIZZATA (con motore)

B = jacobian(nuovosis,[V1 V2 V3]);

matB=taylor(B,[beta phi p theta q psi r phib pb thetab qb xb ub yb vb wm1 wm2
wm3],'ExpansionPoint',0,'Order',1);

matB=double(matB)


%%% MATRICI C, D

matriceC= eye(17);
matriceD= zeros(17,3);
```

Matrice B lineare con valori simbolici

Anche il calcolo della matrice B con valori simbolici deve avvenire senza effettuare la sostituzione dei valori numerici di ciascuna variabile.
Analogamente a prima, omettendo il comando:

```
% %%% sostituzione delle costanti
SISn=subs(SIS,{Ibk,Ipx,Ipy,Ipz,Rb,Rw,dpwb,drb,l,mb,rb,Vb(3),mp,g,alpha},{Ibk_n,I
px_n,Ipy_n,Ipz_n,Rb_n,rw_n,0,0,l_n,mb_n,0,0,mp_n,g_n,alpha_n});
```

Va ripetuto il procedimento di porre a zero le variabili rispetto alle quali si effettua la linearizzazione.
Questa volta le variabili *V* relative alle tensioni non vanno fatte tendere a zero.

```
%%% VARIABILI SIMBOLICHE
matBs= subs(B,{beta phi p theta q psi r phib pb thetab qb xb ub yb vb wm1 wm2
wm3},{0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0});
matBs= simplify(matBs);
```

Si ricava la matrice B con i seguenti coefficienti.

$$B(2,1) = \frac{Rb\ km\ cos(alpha)(Rb^2\ mb + Ibk + Rb^2\ mp - Rb\ l\ mp)}{R\ Rw\ (Ipx\ Rb^2\ mb + Ibk\ Ipx + Ipx\ Rb^2\ mp + Ibk\ l^2\ mp + Rb^2\ l^2\ mb\ mp)} = -2 \cdot B(2,2)$$
$$= -2 \cdot B(2,3)$$

$$B(4,2) = \frac{3^{\frac{1}{2}}\ Rb\ km\ cos(alpha)\ (Rb^2\ mb + Ibk + Rb^2\ mp - Rb\ l\ mp)}{2\ R\ Rw\ (Ipy\ Rb^2\ mb + Ibk\ Ipy + Ipy\ Rb^2\ mp + Ibk\ l^2\ mp + Rb^2\ l^2\ mb\ mp)} = -\frac{\sqrt{3}}{2}\ B(2,1)$$
$$= -B(4,3)$$

$$B(6,1) = -\frac{Rb\ km\ sin(alpha)}{Ipz\ R\ Rw} = B(6,2) = B(6,3)$$

$$B(8,1) = -\frac{Rb\ km\ cos(alpha)\ (mp\ l^2 - Rb\ mp\ l + Ipx)}{R\ Rw\ (Ipx\ Rb^2\ mb + Ibk\ Ipx + Ipx\ Rb^2\ mp + Ibk\ l^2\ mp + Rb^2\ l^2\ mb\ mp)} = -2 \cdot B(8,2)$$
$$= -2 \cdot B(8,3)$$

$$B(10,2) = -\frac{3^{\frac{1}{2}}\, Rb\, km\, cos(alpha)\, (mp\, l^2 - \, Rb\, mp\, l \, + \, Ipy)}{2\, R\, Rw\, (Ipy\, Rb^2\, mb + \, Ibk\, Ipy \, + \, Ipy\, Rb^2\, mp + Ibk\, l^2\, mp \, + \, Rb^2\, l^2\, mb\, mp)} = \, -B(10,3)$$

$$B(12,2) = -\frac{3^{\frac{1}{2}}\, Rb^2\, km\, cos(alpha)\, (mp\, l^2 - \, Rb\, mp\, l \, + \, Ipy)}{2\, R\, Rw\, (Ipy\, Rb^2\, mb + \, Ibk\, Ipy \, + \, Ipy\, Rb^2\, mp + \, Ibk\, l^2\, mp \, + \, Rb^2\, l^2\, mb\, mp)} = -B(12,3)$$

$$B(14,1) = \frac{Rb^2\, km\, cos(alpha)\, (mp\, l^2 - \, Rb\, mp\, l \, + \, Ipx)}{R\, Rw\, (Ipx\, Rb^2\, mb + \, Ibk\, Ipx \, + \, Ipx\, Rb^2\, mp + \, Ibk\, l^2\, mp \, + \, Rb^2\, l^2\, mb\, mp)} = -2 \cdot B(14,2)$$
$$= -2 \cdot B(14,3)$$

$$B(15,1) = \frac{km}{Iwz\, R}$$

$$B(16,2) = \frac{km}{Iwz\, R}$$

$$B(17,3) = \frac{km}{Iwz\, R}$$

Ovvero:

$$B(15,1) = B(16,2) = B(17,3)$$

Matrici A e B con valori numerici

A =

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33.2666	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4518	-0.2259	-0.2259
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	33.2666	0	0	0	0	0	0	0	0	0	0	0	0	0.3912	-0.3912
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.5469	2.5469	2.5469
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
94.9019	0	0	0	0	0	0	0	0	0	0	0	0	0	2.9981	-1.4991	-1.4991
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	94.9019	0	0	0	0	0	0	0	0	0	0	0	0	2.5965	-2.5965
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	-11.3882	0	0	0	0	0	0	0	0	0	0	0	0	-0.3116	0.3116
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
11.3882	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3598	-0.1799	-0.1799
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-376.8568	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-376.8568	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-376.8568

B =

0	0	0
-6.4537	3.2269	3.2269
0	0	0
0	-5.5891	5.5891
0	0	0
-36.384	-36.384	-36.384
0	0	0
-42.831	21.415	21.415
0	0	0
0	-37.092	37.092
0	0	0
0	4.4511	-4.4511
0	0	0
-5.1397	2.5698	2.5698
5383.7	0	0
0	5383.7	0
0	0	5383.7

Fase di controllo: introduzione dei PID

A questo punto si dispone del sistema in spazio di stato lineare contenente la dinamica dei motori. Quello che si vuole imporre a tale sistema è che, dal momento in cui lo si controlla in tensione, lo stato segua il comportamento richiesto (ovvero i riferimenti dati in ingresso).
Lo scopo che ci si prepone è quello di introdurre dei controllori che stabilizzino il ballbot e lo mantengano sempre nella posizione di equilibrio desiderata:

$$x_{eq} = \begin{bmatrix} phi_{eq} \\ theta_{eq} \\ psi_{eq} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

È stato dimostrato che un sistema di questo tipo, linearizzato, con tre ingressi, può essere controllato mediante l'inserimento di tre PID uno per ciascuna variabile da stabilizzare (*phi*, *theta*, *psi*);
Il controllo Proporzionale-Integrale-Derivativo, comunemente abbreviato come PID, è un sistema in retroazione negativa ampiamente impiegato nei sistemi di controllo.
Ognuno di tali controllori va posto in catena diretta con il processo (definito dalla forma in spazio di stato) e riceve come ingresso il valore *errore* che è dato dallo scostamento dallo stato dal valore desiderato (il riferimento).
In uscita ciascun PID fornirà la legge di controllo per la correzione del valore delle variabili.

I PID andranno inseriti come blocchi nel diagramma in *Simulink*, ma per far ciò occorre calcolare i parametri *Kp* (azione proporzionale), *Ki* (azione integrale) e *Kd* (azione derivativa) di ciascuno di essi.
L'azione proporzionale agisce sul processo per stabilizzare e tentare di far tendere il segnale di errore a zero, ovvero cercare di far convergere il più possibile il valore dello stato da stabilizzare al segnale di riferimento.
L'azione integrale fa sì che in qualche modo si memorizzi l'informazione relativa ai valori passati assunti dal segnale errore, anche se non è detto che il valore di tale azione sia esattamente nullo in corrispondenza di segnale errore nullo.
L'azione derivativa ha lo scopo di migliorare il controllore in termini di prestazioni, cercando di compensare piuttosto velocemente le variazioni del segnale errore.

Per calcolare i coefficienti *Kp*, *Kd* e *Ki* occorre prima di tutto ricavare la funzione di trasferimento del sistema finale in spazio di stato lineare.

Il calcolo della forma in spazio di stato in *Matlab* avviene tramite il comando *ss*.

```
sis_motore= ss(matA,matB,matriceC,matriceD);

fdt_motore= zpk(sis_motore);
```

Applicando il commando *zpk* sul sistema ottenuto si ottiene la funzione di trasferimento in formato zeri (*z*), poli (*p*), guadagno (*k*), in tempo continuo.
Essa sarà memorizzata come variabile *fdt_motore*.

Essendo un sistema MIMO (*multi-input, multi-output*) si ottiene la matrice delle funzioni di trasferimento di dimensione *17 x 3*.
Osservando tale matrice si è giunti al risultato che essa è diagonale dominante, ovvero risulta verificata la formula seguente, per ogni elemento i-esimo della diagonale principale:

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|$$

Da questa conclusione si può, quindi, realizzare un controllo disaccoppiato separatamente sulle tre variabili *phi*, *theta*, *psi*.

Potendo effettuare un controllo indipendente, sono di interesse solamente le funzioni di trasferimento relative a tre delle diciassette variabili di stato in output, quelle da controllare, ossia *phi* (prima componente nel vettore delle variabili di stato), *theta* (terza componente) e *psi* (quinta componente).

Si selezionano le funzioni di trasferimento lungo la diagonale per ciascuno dei tre angoli come conseguenza del principio di diagonale dominanza.

```
% Funzioni di trasferimento
% relative alle variabili di stato phi, theta, psi


fdtphi_mot= fdt_motore(1,1)  % phi

%              -6.4537 s
%   -----------------------------
%   (s-5.768) (s+5.768) (s+376.9)


fdttheta_mot= fdt_motore(3,2) % theta

%              -5.5891 s
%   -----------------------------
%   (s-5.768) (s+5.768) (s+376.9)


fdtpsi_mot= fdt_motore(5,3)  % psi

%     -36.384 s
%   -------------
%   s^2 (s+376.9)
```

Ora che si dispone delle funzioni di trasferimento che legano le variabili da controllare ai rispettivi ingressi, si possono calcolare i coefficienti delle azioni proporzionali, derivative, integrali dei tre PID da introdurre per il controllo.

Il comando adottato è *pidtune*, ma si può scegliere di adottare anche *pidTuner* anche se più lento e laborioso.
I parametri richiesti in ingresso sono due: la funzione di trasferimento su cui si vuole svolgere la sintesi ed il tipo di PID che si desidera ottenere (*'P'* fornisce un controllo solamente proporzionale, *'PI'* un controllo basato su azione proporzionale ed integrale e così via).
Il tipo scelto è *'PID'* in modo da ricavare un controllo che applichi azione proporzionale, integrale e derivativa.

```
%% PID + MOTORE

% phi, rispetto ad u1

pidphi_mot = pidtune(fdtphi_mot,'PID');

%               1
%   Kp + Ki * --- + Kd * s
%               s
%
%   with Kp = -3790, Ki = -112000, Kd = -31.1

% theta, rispetto ad u2


pidtheta_mot = pidtune(fdttheta_mot,'PID');

%               1
%   Kp + Ki * --- + Kd * s
%               s
%
%   with Kp = -4390, Ki = -130000, Kd = -35.9

% psi, rispetto ad u3


pidpsi_mot = pidtune(fdtpsi_mot,'PID');

%               1
%   Kp + Ki * --- + Kd * s
%               s
%
%   with Kp = -8230, Ki = -793000, Kd = -19
```

I valori ricavati *Kp*, *Ki*, *Kd* per ciascuno dei tre controllori andranno aggiunti nei blocchi PID sul modello *Simulink* in relazione alla variabile da stabilizzare.

Simulazione del sistema ballbot: Simulink

Il modello *Simulink* adottato per simulare il sistema *ballbot* ricavato tramite *Matlab* è mostrato dalla figura sottostante.

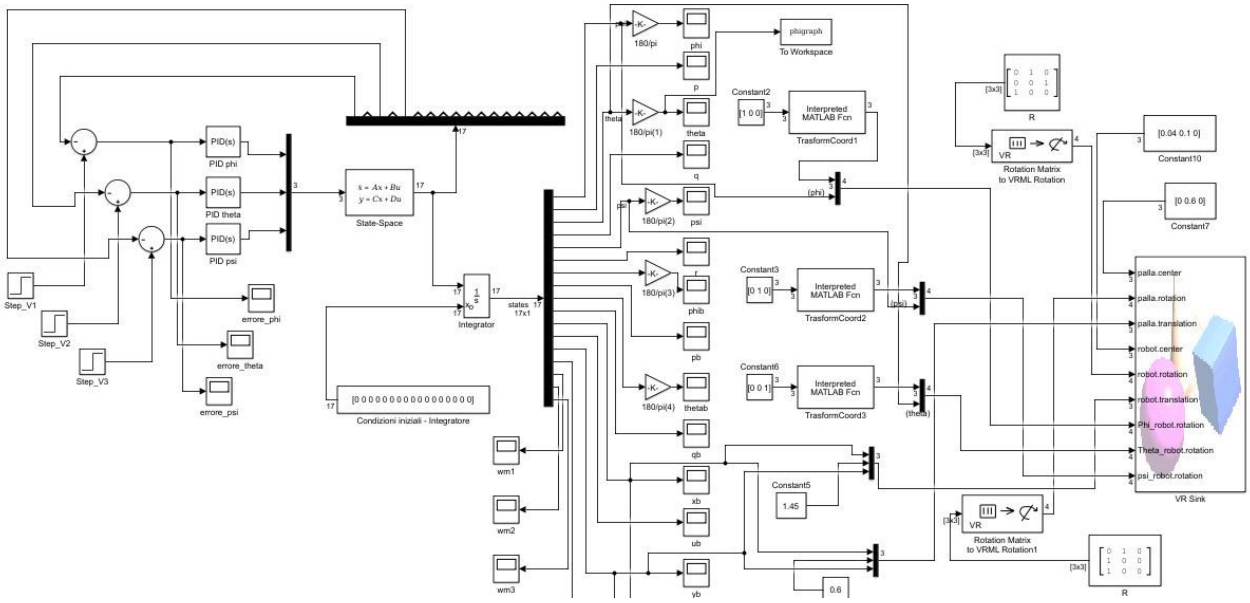


Figura 4 - Modello Simulink del ballbot

Le matrici dello spazio di stato ricavate A, B, C e D lineari vanno inserite nel primo blocco in alto a sinistra che è in entrata al blocco integratore dal quale esce lo stato *x* del sistema (di diciassette componenti).

Al blocco *State-Space* del modello in spazio di stato si pone in ingresso il blocco *Input* che si sostanzia in un vettore di tre elementi, i valori degli ingressi al sistema (dove in ingresso vengono fornite, come stabilito, le tensioni *V1, V2, V3* erogate dai tre motori in quanto il modello di stato contiene le equazioni del motore).

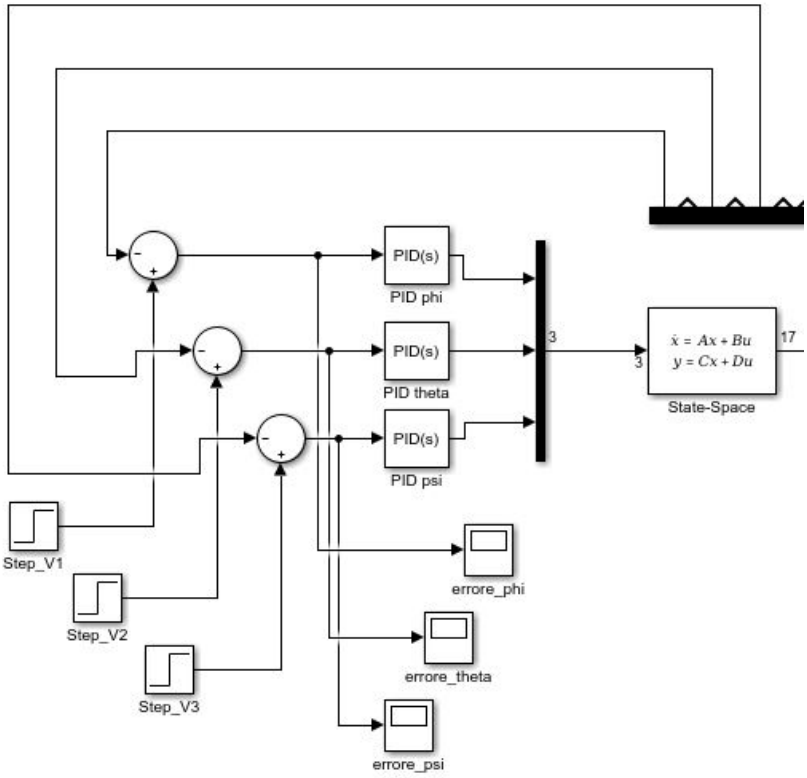


Figura 5 - Blocchi State-Space ed Input

Il blocco *State-Space* preleva dallo *script* di *Matlab* le quattro matrici A, B, C, D per ricavare il modello da simulare e definisce le condizioni iniziali dello stato.

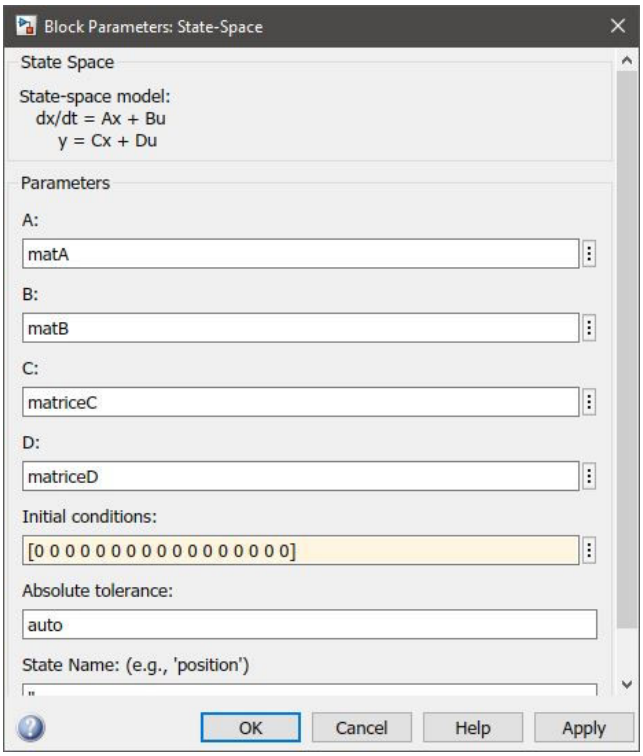


Figura 6 - State-Space

Si sceglie di impostare le condizioni iniziali definite da *Initial Conditions* a zero, lasciando la tolleranza assoluta al valore di default.
Prestare particolare attenzione nell'inserimento delle matrici nella sezione *Parameters*;
I nomi delle matrici devono essere esattamente gli stessi di quelli con i quali sono state memorizzate nel *workspace* di *Matlab*, altrimenti all'avvio della simulazione si riscontrerà un messaggio di errore per il quale *Simulink* non riesce a trovare le matrici inserite.

I blocchi relativi agli ingressi di sistema *step_Vi* generano dei gradini di ampiezza da definire ed assumono la struttura seguente.

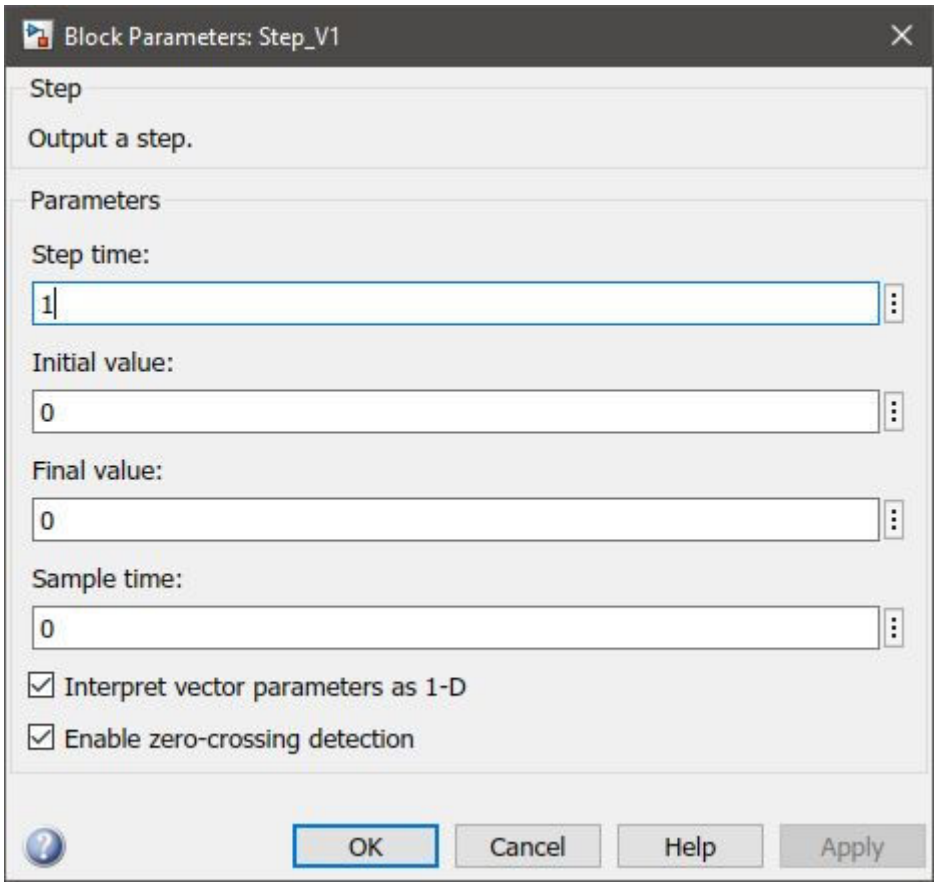


Figura 7 – Blocco ingresso i-esimo Step_Vi

Vanno impostati i parametri seguenti: S*tep time* definisce l'istante a partire dal quale il gradino assumerà il valore costante definito in *Final value*.
Si decide di impostare tale istante di inizio ad *1* e di lasciare *Initial value* a valore nullo.
Il valore di *Sample time* può essere lasciato al valore nullo di default.
Cambiando il valore di *Final value* è possibile effettuare una serie di prove di diverse simulazioni impostando diversi valori di tensione (si cambiano i riferimenti in input al sistema).

Nel modello di simulazione sono presenti tre blocchi per la trasformazione delle coordinate in relazione agli angoli *phi*, *psi*, *theta* di rotazione.
Occorre definire nello *script di Matlab* la funzione da inserire come parametro per far funzionare questi tre blocchi.
Ciascuno dei blocchi di trasformazione è del tipo sottostante.

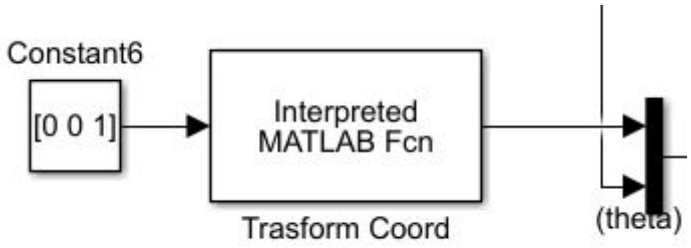


Figura 8 - Blocco trasformazione coordinate

Per prima cosa si crea un file *Matlab* che conterrà la funzione che adopera la trasformazione.
Il nome del file (della funzione) è *BB03_Transf_Coord_VRML*.
Lo si inserisce come parametro del blocco.

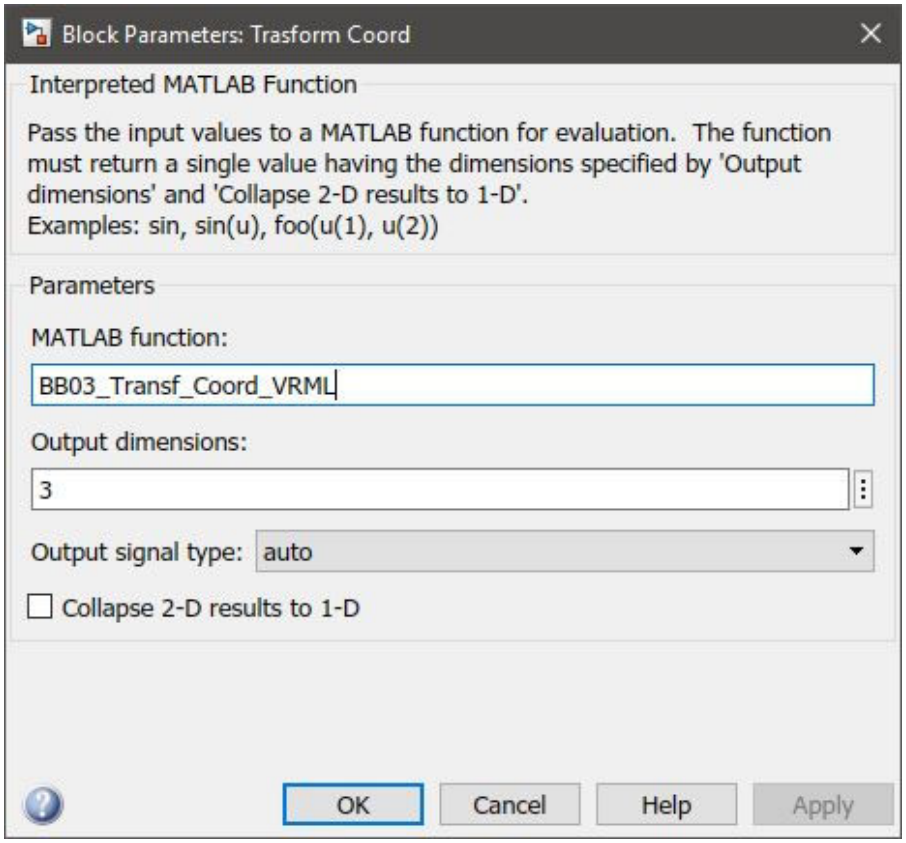


Figura 9 - Parametri blocco trasformazione coordinate

Lo *script* di *Matlab* contenente la funzione è un file *.m* contenente il seguente codice.

```
function y = BB03_Transf_Coord_VRM(u)
% Calcolo delle coordinate del sistema reale nel sistema 3D.
% Se deve eseguire una rotazione attorno a z di 90°
% successivamente attorno ad x

Rx= [1,0,0; 0,0,-1; 0,+1,0];
Rz= [0,-1,0; 1,0,0; 0,0,1];

r= u;

v= Rx*Rz*r;

y = v;

end
```

Ciascuno dei tre blocchi di trasformazione riceve in ingresso un vettore, con valori:

$$v_{phi} = [1|0|0]$$

$$v_{psi} = [0|1|0]$$

$$v_{theta} = [0|0|1]$$

Un altro blocco impiegato nel modello è quello denominato *Rotation Matrix to VRML Rotation*. Implementa, infatti, le matrici di rotazione rispetto ai possibili angoli di inclinazione del corpo. Sono blocchi caratterizzati dalla struttura seguente.

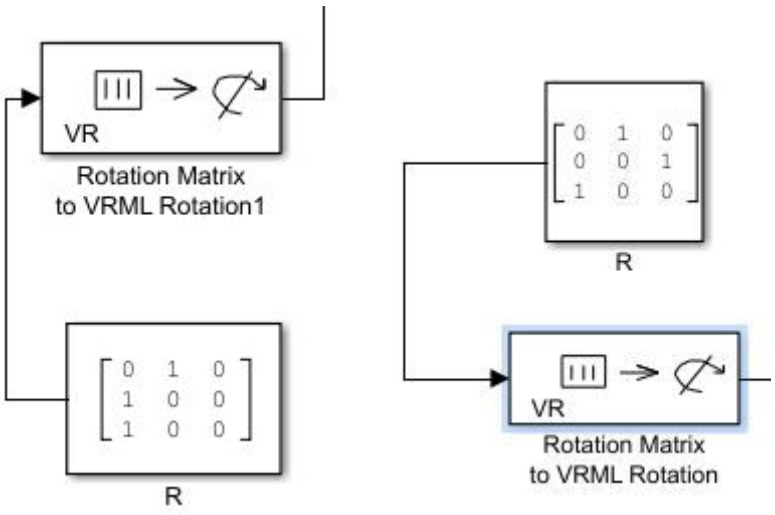


Figura 10 – Blocchi per le matrici di rotazione

Entrambi i blocchi sono pre-moltiplicati ciascuno per una matrice *R* composta da soli elementi pari ad uno e zero, come si osserva dalla figura *10*.
I parametri dei *Rotation Matrix* vanno definiti nel modo seguente.

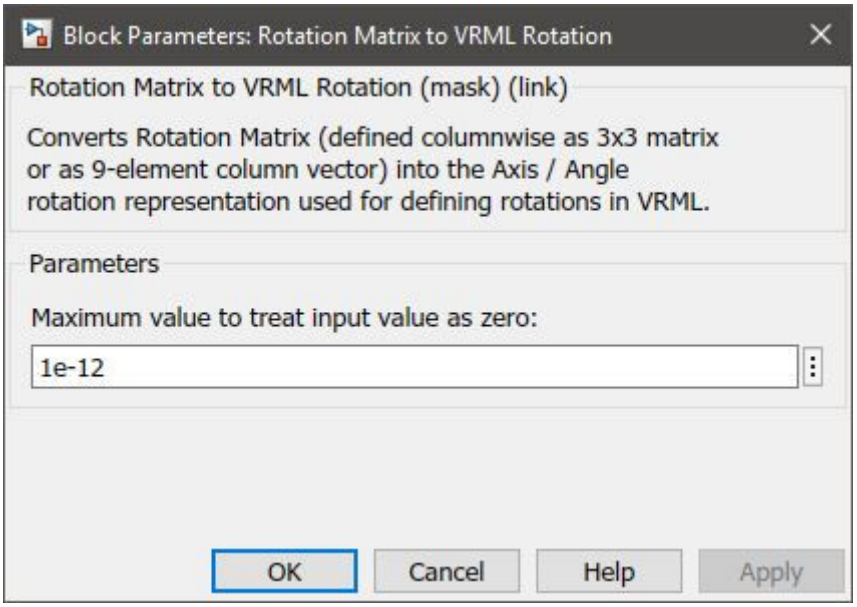


Figura 11 - Rotation Matrix to VRML Rotation (parameters)

Funzionamento modello Simulink, un esempio

Si può supporre che l'ingresso fornito al sistema sia un valore di tensione nullo per ogni motore.
Si applica, però, un valore di *phi* iniziale diverso dal valore di equilibrio ricordando che il sistema è in equilibrio per:

$$\begin{bmatrix} phi_{eq} \\ theta_{eq} \\ psi_{eq} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Scegliendo, ad esempio: $phi = \frac{\pi}{10}$

Purché l'inclinazione iniziale ragionevolmente non sia eccessiva.
Gli altri due angoli restano nello stato di equilibrio, dunque entrambi uguali a zero.
Nei tre blocchi relativi agli ingressi a gradino, perciò, impostare *Final value* a zero e nel blocco dello spazio di stato impostare *Initial condition* con un vettore del tipo seguente.

$$[pi/10\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

Avviando la simulazione, con durata settata a dieci secondi, si otterranno i risultati sottostanti.
Nonostante l'inclinazione iniziale imposta al ballbot, tramite l'azione dei PID il sistema si riassesterà alla condizione di equilibrio in meno di un secondo.
Il transitorio è perciò di assai breve durata, e ciascuna delle tre variabili *phi*, *theta* e *psi* convergono al valore zero a regime quasi istantaneamente (secondo le condizioni di tale simulazione).
Dai grafici è possibile osservare la traiettoria dei tre angoli.

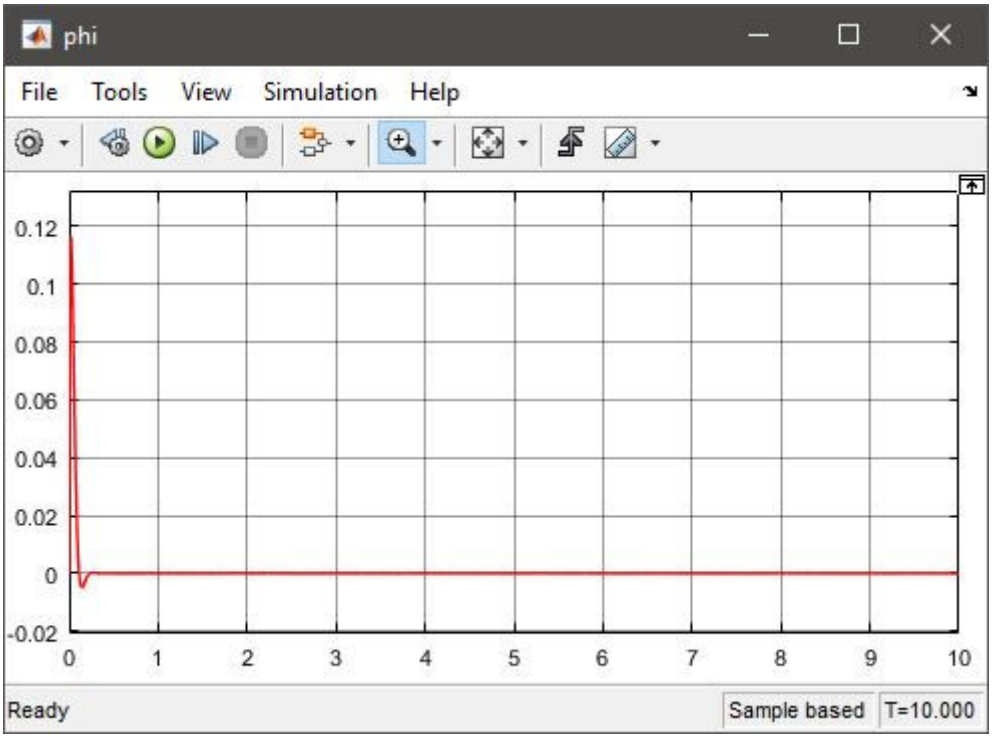


Figura 12 - simulazione: angolo phi

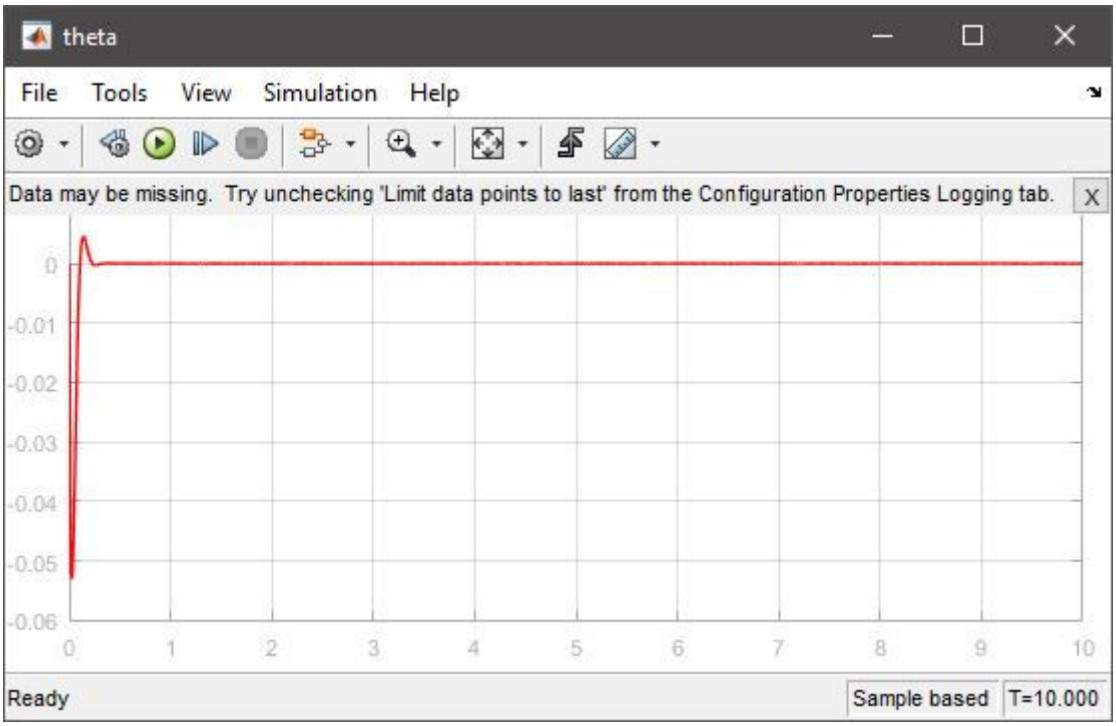


Figura 12 - simulazione: angolo theta

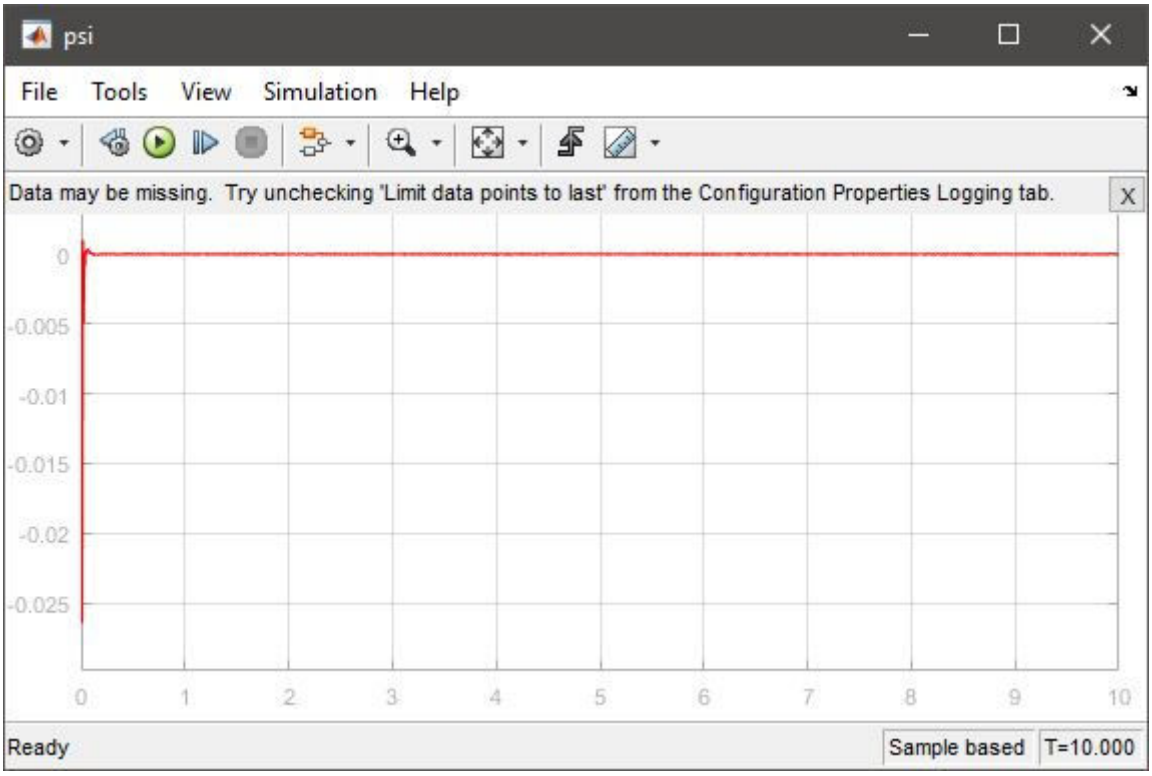


Figura 13 - simulazione: angolo psi

Il grafico sottostante *XY plot* descrive, invece, l'andamento ovvero la traiettoria descritta dal ballbot sul piano $x - y$ orizzontale. Il risultato dalla simulazione di esempio corrente è dato dalla figura *14*.

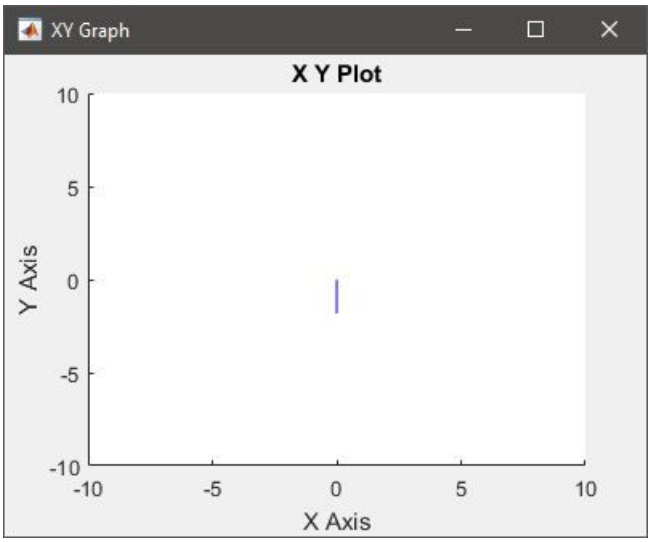


Figura 14 - traiettoria ballbot sul piano x,y

Si verifica, inoltre, che dopo aver raggiunto il regime permanente ovvero dopo che il sistema si è stabilizzato alla condizione di equilibrio del robot le omni-wheels mantengono una velocità costante di rotazione che non varia per tutto il tempo della simulazione.
Nella figura *15* si mostra, ad esempio, l'andamento assunto dalla curva che descrive la velocità angolare della ruota *1*.

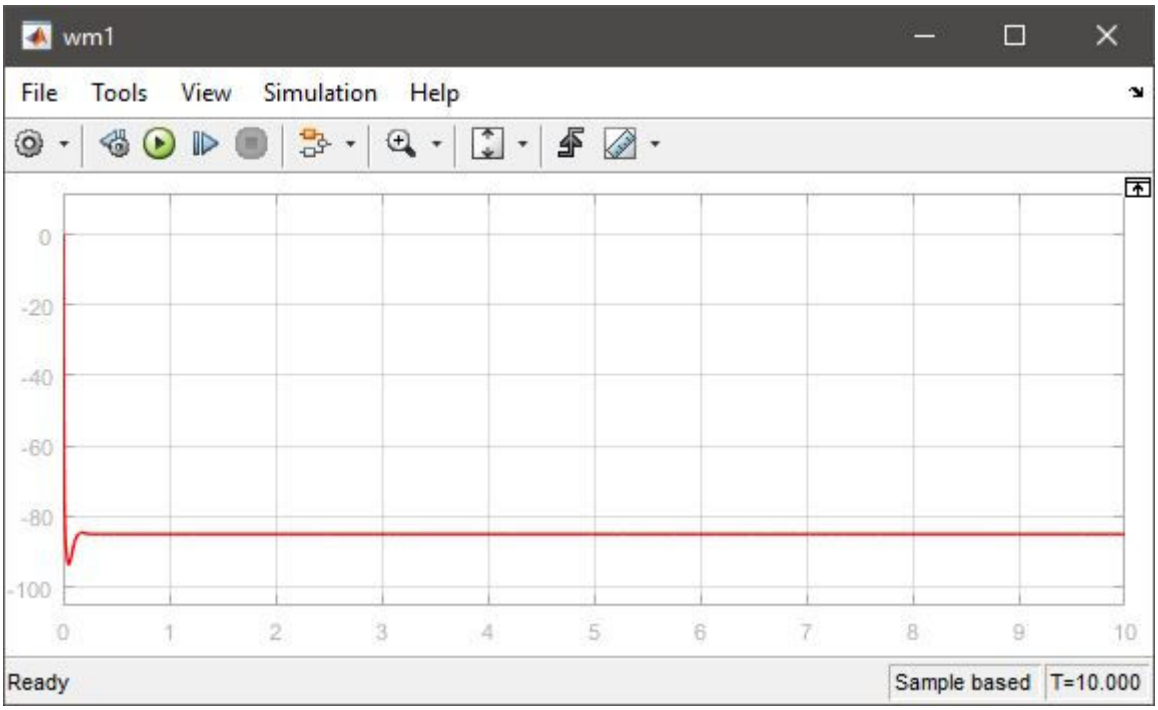
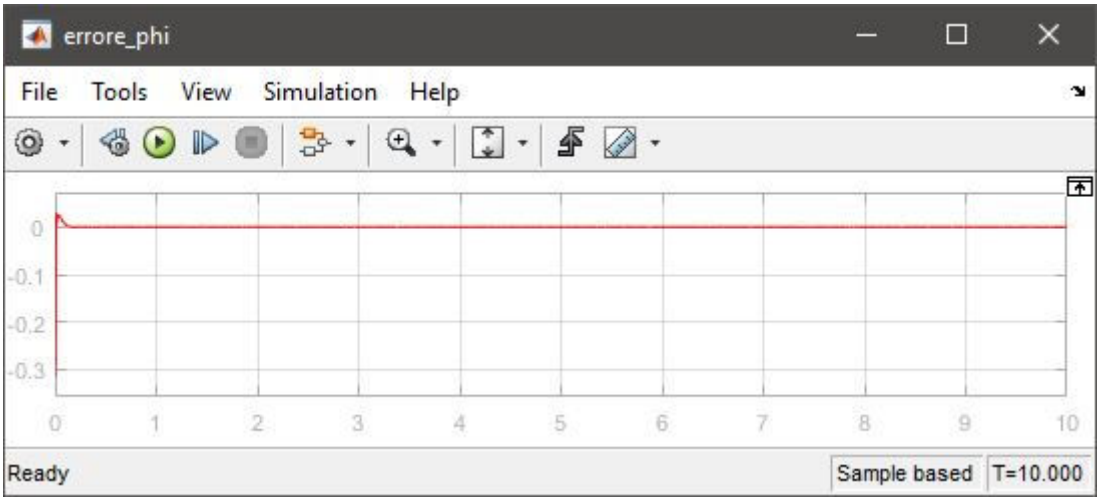
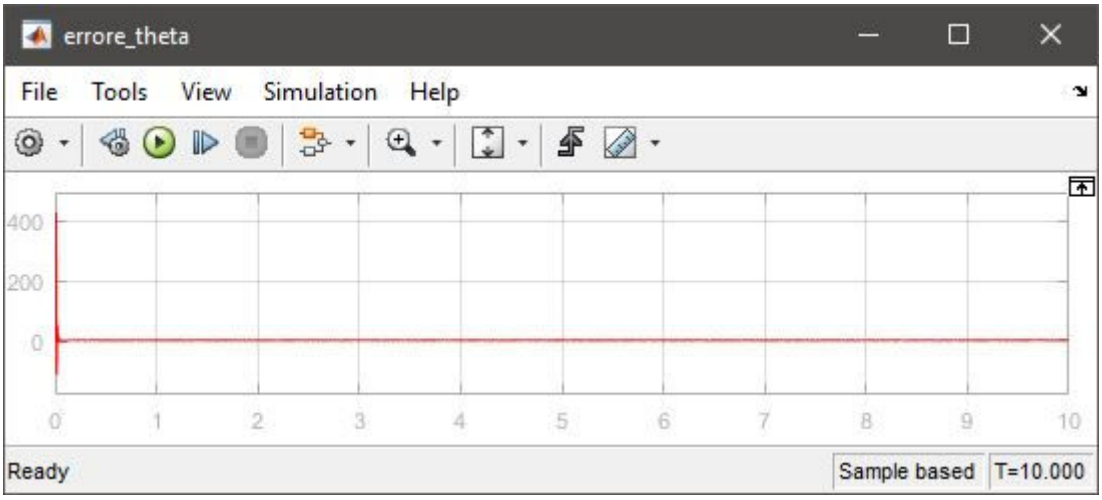
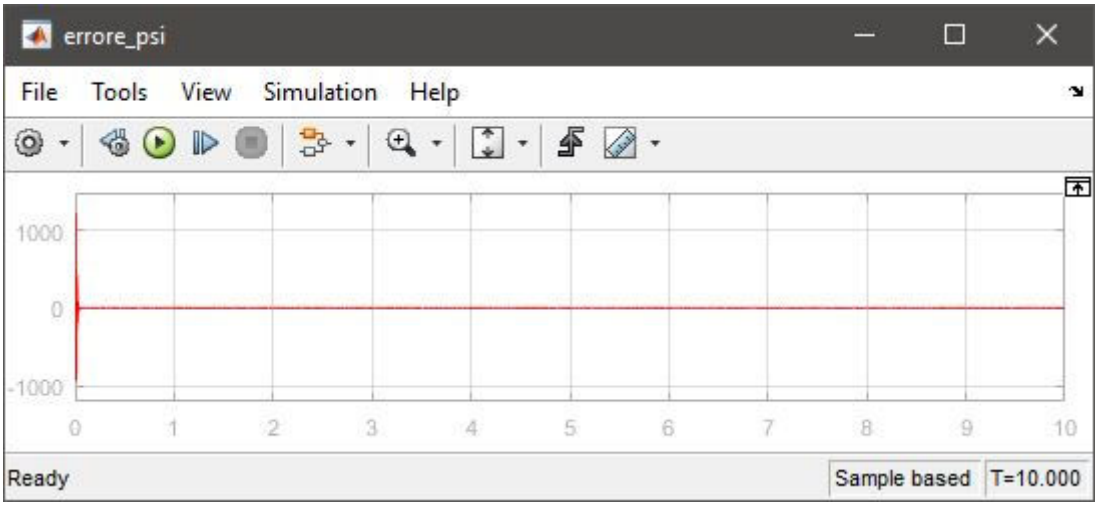


Figura 15 - velocità wm1 della omni-wheel 1

I grafi *errore_phi*, *errore_theta*, *errore_psi* sono posti in corrispondenza del ramo che collega il blocco sommatore tra riferimento e retroazione negativa della variabile di stato da stabilizzare al blocco PID corrispondente. In uscita mostrano esattamente l'andamento della curva errore dato, istante per istante, dal valore del segnale di riferimento meno la derivata della variabile da controllare (che nella pratica non è altro che la velocità di variazione della stessa). Ovviamente, si ha un funzionamento corretto del modello, solamente se si certifica che per ciascuna delle tre variabili da portare all'equilibrio a regime l'errore tende a zero.







Inserimento dei disturbi in ingresso al sistema

In aggiunta al modello *Simulink* che simula il comportamento del ballbot tramite un controllo in tensione e tramite l'azione dei PID per la stabilizzazione dei tre angoli del sistema, si può inserire l'azione di un disturbo costante su tali variabili.
La presenza dei tre controllori garantisce che le variabili vengano stabilizzate a regime, nonostante il sistema riceva in ingresso un impulso ad un istante successivo a quello di inizio della simulazione, istante nel quale il sistema si trovava in equilibrio.
Tramite la simulazione si osserva cosa accade al modello e come variano gli angoli.

I blocchi dei tre disturbi saranno aggiunti in uscita dallo *State-space* che rappresenta il processo, a simulare uno sbilanciamento imprevisto che altererà l'angolo in uscita dal sistema.

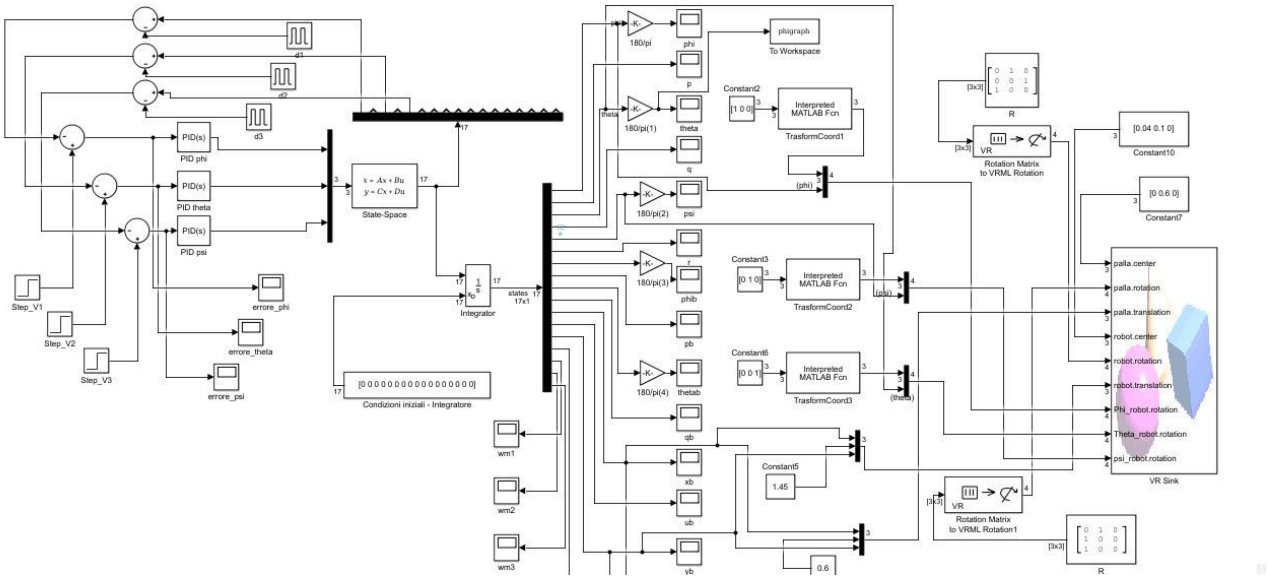
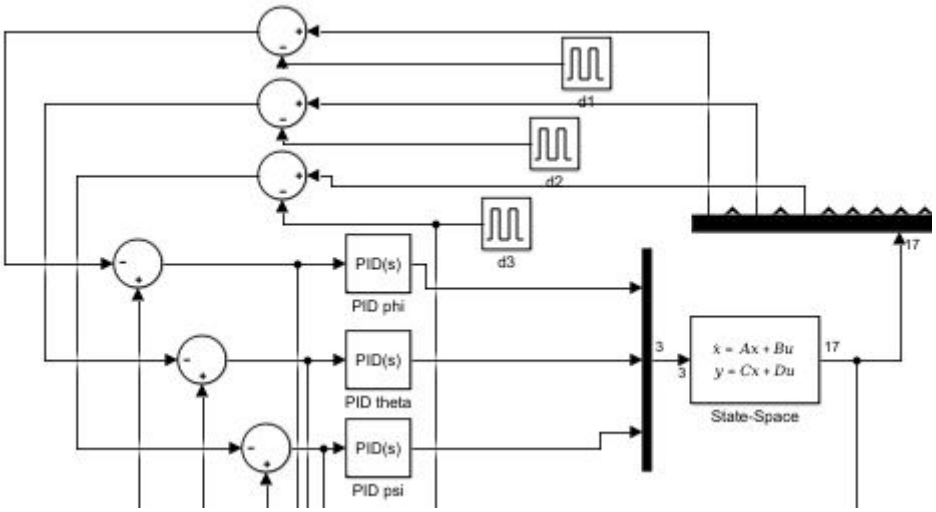


Figura 16 - Modello Simulink con disturbi d1,d2,d3



Si ipotizzi che le tensioni in ingresso siano nulle ed il sistema sia in evoluzione libera, per cui i PID agiscono in modo tale da non far cadere a terra il ballbot, mantenendolo in uno stato di equilibrio.

Sul sistema agisce un disturbo *d3* in entrata al processo sulla variabile *psi*.
Tale perturbazione ha inizio dall'istante *3s* ed è un impulso con ampiezza *0.1*.

I grafici che mostrano le curve del segnale errore in ingresso ai PID testimoniano che le variabili vengono stabilizzate durante la simulazione, in quanto è possibile verificare che l'errore tende a zero nonostante l'aggiunta del disturbo in uscita dal processo.

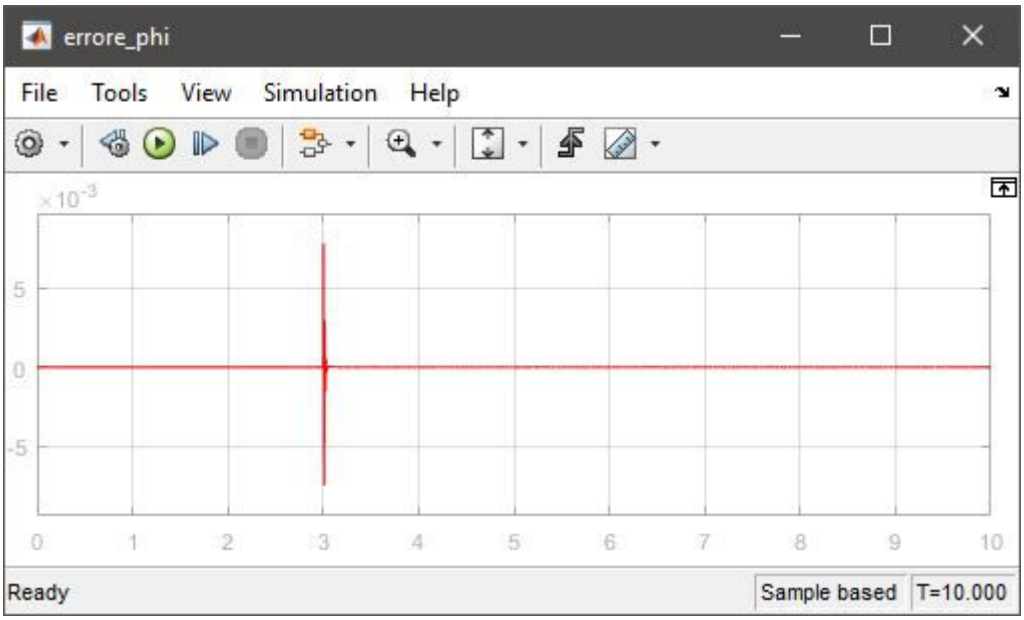


Figura 17 - Errore phi con disturbo d3

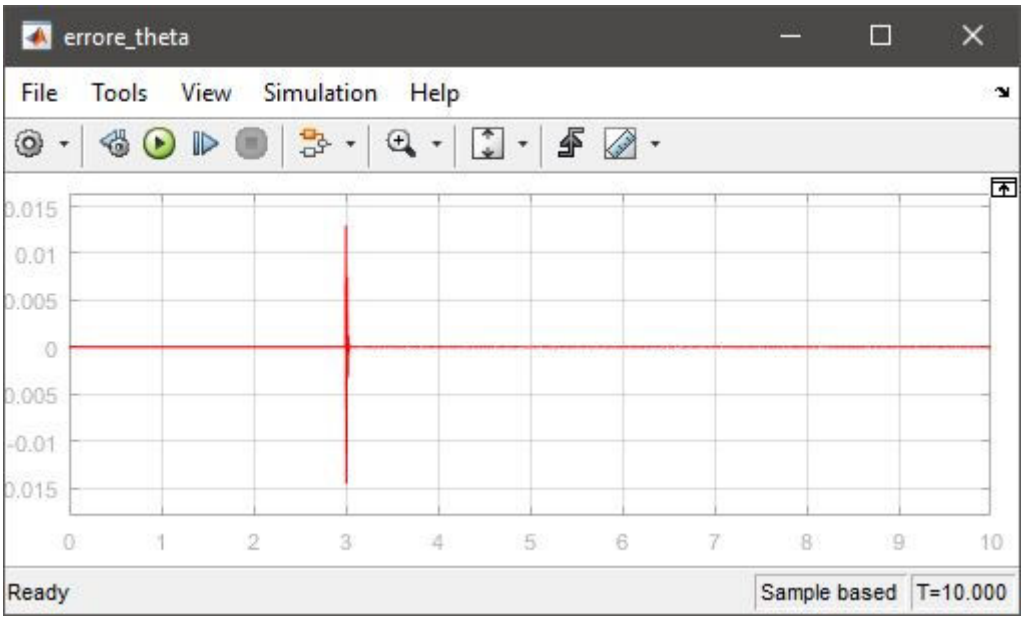


Figura 18 - Errore theta con disturbo d3

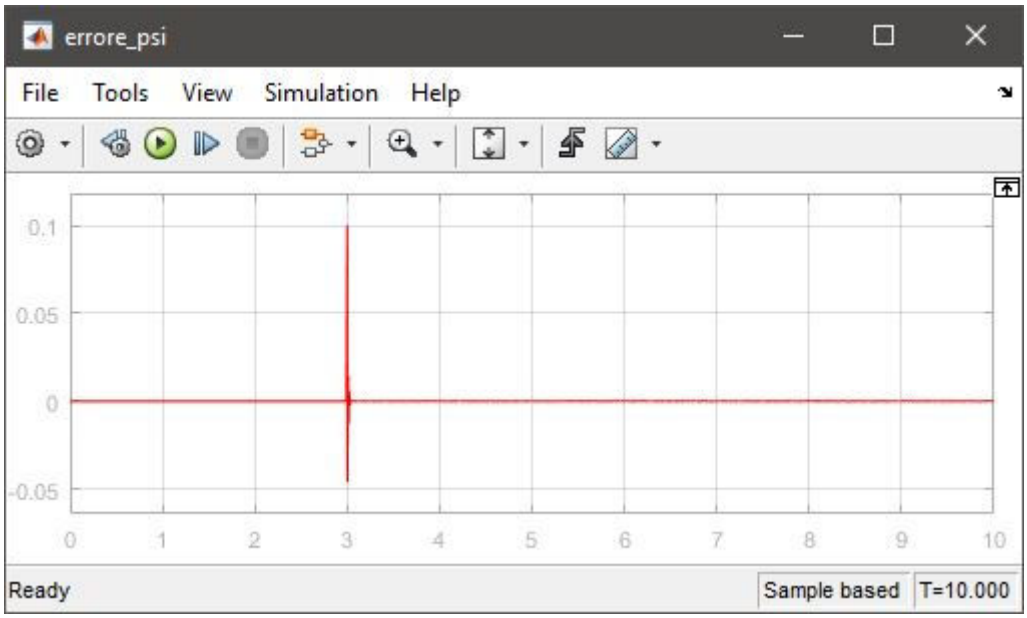


Figura 19 - Errore psi con disturbo d3

Si nota all'istante *3s* la perturbazione che altera la curva, subito corretta dall'azione dei PID.
I blocchi dei disturbi generano degli impulsi e non dei gradini, in quanto una perturbazione costante di uno degli angoli o di tutti gli angoli causerebbe una destabilizzazione fissa del sistema che non riuscirebbe più a fermarsi nel tentativo di correggerla.
Pertanto, si adottano i blocchi *Pulse generator* tarati nella maniera sottostante.

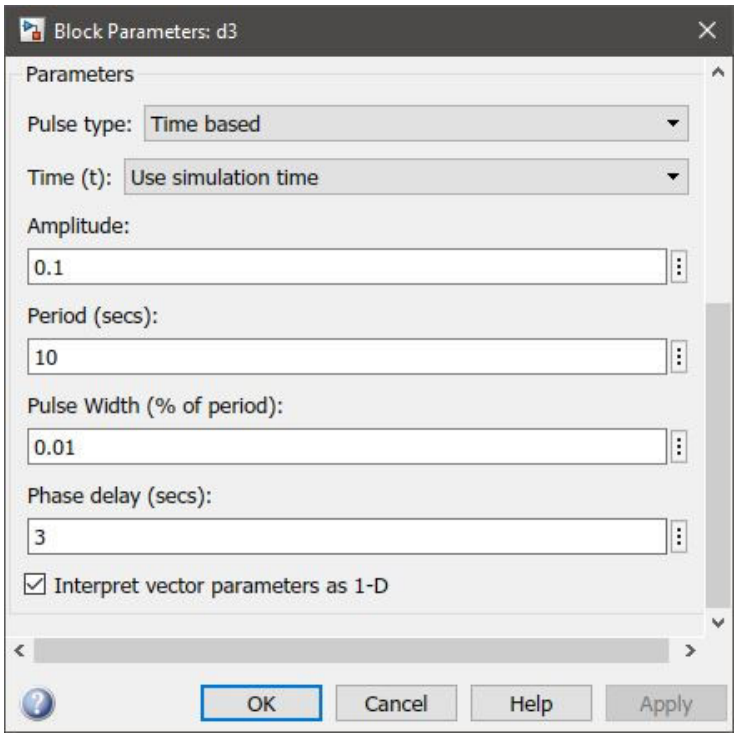


Figura 20 - Pulse generator: disturbo d3

Conclusioni e sviluppi futuri

La relazione svolta si inserisce all'interno del corso di metodi e tecniche di simulazione e si è basata sulla modellazione matematica del robot ballbot, della sua linearizzazione per scopi di controllo e della sua simulazione al calcolatore.

L'approccio (o meglio, la tecnica di modellazione matematica) attraverso le equazioni di Newton-Eulero e gli angoli di Tait-Bryan si è dimostrato didatticamente più efficace rispetto a quello di Lagrange, basato su considerazioni energetiche: infatti, durante il corso di tutta la modellazione e delle operazioni matematiche ad essa relative, si è sempre riusciti a dare un preciso significato fisico alle variabili in esame, ottenendo una migliore comprensione delle operazioni che si stavano svolgendo.
È inoltre risultata molto utile la sensibilità acquisita durante il corso riguardante aspetti peculiari delle operazioni matematiche svolte come, ad esempio, nel trattamento delle equazioni differenziali ordinarie (ODE) e nell'utilizzo del toolbox simbolico di MATLAB.

Essendo il lavoro svolto arrestatosi al controllo del sistema ballbot esclusivamente in simulazione, attraverso il software SIMULINK (tool di simulazione di MATLAB), come possibili sviluppi futuri troviamo:

- Discretizzazione del sistema e dei controllori per implementare un controllo interamente a tempo discreto (in luogo di quello a tempo continuo che è stato realizzato in questa relazione);
- Implementazione fisica del controllo sul sistema ballbot vero e proprio;
- Maggiori e più approfondite considerazioni sulla robustezza e sulla sensibilità ai disturbi del sistema (essendo stato analizzato solo il caso di disturbo di tipo impulsivo).

Bibliografia

[1] : "New dynamic model for a Ballbot system" (Andrea Bonci) ,published in: Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on 29-31 Aug. 2016;

[2] : Modeling and Control of a Ballbot - Bachelor Thesis (Fankhauser, Péter; Gwerder, Corsin Publication), 2010;

[3]: Slide del corso "Metodi e tecniche di simulazione" tenuto da: Anna Maria Perdon ed Andrea Bonci nell' anno accademico 2017/2018.