Dinamica Veicolo Autobilanciato - gruppo Calegari

Indice

[Setup iniziale dell'ambiente](#MW_H_9E58691F)   
[Definizione delle variabili simboliche](#MW_H_ACACDDA1)   
[Definizione masse e gravità](#MW_H_09E078D6)   
[Definizione delle altezze](#MW_H_BA266E12)   
[Definizione inerzie](#MW_H_62D560EB)   
[Definizione delle costanti geometriche e rapporti di trasmissione](#MW_H_B9A86F27)   
[Calcolo componenti dinamiche e potenziali per ogni corpo rigido del sistema](#MW_H_32137170)   
 [Asta](#MW_H_8F6E8682)   
 [Utente](#MW_H_E0B2E187)   
 [Chassis (o base)](#MW_H_36F684B8)   
 [Ruota](#MW_H_4FB8AE5A)   
 [Motore](#MW_H_FF4934B3)   
 [Lagrangiana dell'intero sistema](#MW_H_190EE358)   
[Equazioni del moto](#MW_H_4F6B3CF1)   
 [Calcolo equazioni del moto rispetto a phi ()](#MW_H_7C2700E8)   
 [Calcolo equazioni del moto rispetto a theta ()](#MW_H_3726A56E)   
[Risoluzione delle equazioni del moto](#MW_H_C1ADC820)   
[Linearizzazione delle equazioni](#MW_H_CF3F1450)   
[Salvataggio workspace](#MW_H_4B52C06D)

# Setup iniziale dell'ambiente

clear

clc

digits(7)

# Definizione delle variabili simboliche

syms t; % Tempo

syms T; % Energia cinetica del sistema

syms U; % Energia potenziale del sistema

alpha = sym('0'); % Angolo di offset (inclinazione utente) che sposta in avanti il baricentro

syms q\_1 q\_1\_p q\_1\_pp;

syms q\_2 q\_2\_p q\_2\_pp; % Coordinate libere

syms phi(t); % Posizione carrello / r

vel = diff(phi, t); % Velocità carello / r

syms theta(t); % Angolo inclinazione

vel\_ang = diff(theta, t); % Vel angolare

syms C\_m C\_m\_star; % Forza esterna

syms u;%input sistema dinamico

# Definizione masse e gravità

Masse espresse tutte in [kg].

% Massa singola ruota

m\_r = sym('3');

% Massa asta

m\_a = sym('3.5');

% Massa uomo

syms m\_c;

% Massa carello(base) + batterie

m\_b = sym('34.4');

% Massa totale del sistema

M = m\_a+m\_b+m\_c+2\*m\_r;

% Costante di accelerazione di gravità

g = sym('9.80665');

# Definizione delle altezze

Dimensioni espresse tutte in [m]

% Altezza asta

h\_a = sym('1.4');

% Altezza base

h\_b = sym('0.2');

% Profondità base

w\_b = sym('0.5');

% Posizione baricentro base (valore assoluto)

z\_b = sym('0.1');

% Altezza uomo

h\_c = sym('1.77');

% Raggio ruota

r = sym('0.22');

# Definizione inerzie

Inerzie espresse in []

% Inerzia asta

J\_a = sym('0.1669');

% Inerzia della base

J\_b = sym('0.6667');

% Inerzia del corpo espressa rispetto all'asse principale

altezza\_cm = h\_c \* 100; % Altezza dell'uomo espressa in [cm]

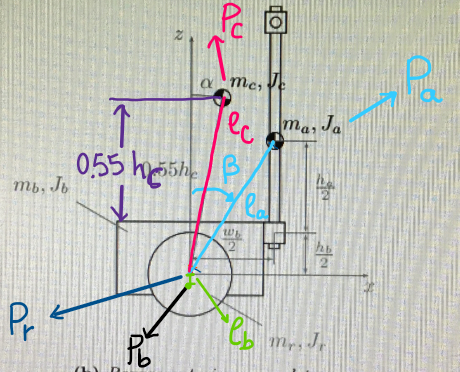
J\_c = (1732353 \* altezza\_cm + 619298\*2.20462\*m\_c - 277625773) \* (10^-7);

% Inerzia della singola ruota

J\_r = sym('0.1452');

# Definizione delle costanti geometriche e rapporti di trasmissione

Si definiscono i parametri geometrici d'interesse come nell'immagine, la quale si trova in una condizione di 



temp\_1 = (h\_a / 2) + (h\_b / 2);

temp\_2 = w\_b / 2;

% Al baricentro del corpo umano vado ad aggiungere l'altezza della base

% sulla quale esso appoggia i piedi (è una quota quindi da aggiungere)

l\_c = 0.55 \* h\_c + h\_b \* 0.5;

l\_b = 0.1;

l\_a = sqrt((temp\_1)^2 + (temp\_2)^2);

beta = atan(temp\_2/temp\_1);

Per il rapporto di trasmissione andiamo a calcolarlo come moltiplicazione tra il rapporto trasmissivo della trasmissione epicicloidale e quello della cinghia.

Nello specifico il rapporto della cinghia andiamo a calcolarlo come:

num\_denti\_driver = sym('22');

num\_denti\_driven = sym('26');

tau\_cinghia = num\_denti\_driver / num\_denti\_driven;

tau\_riduttore = sym('0.1');

tau = 0.1 \* tau\_cinghia;

J\_m = sym('3.7e-4');

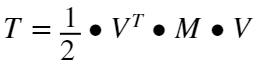
# Calcolo componenti dinamiche e potenziali per ogni corpo rigido del sistema

Per il calcolo delle equazioni dinamiche del sistema siamo andati a considerare ogni singolo corpo rigido componente il sistema, calcolandone le grandezze fisiche di posizione e velocità, con un approccio cartesiano.

Nello specifico abbiamo considerato il sistema composto da:

* Asta ()
* Utente a bordo dello chassis ()
* Chassis (o base) ()
* Ruota () che poi sarà considerata con un contributo doppio, essendo 2 le ruote del sistema

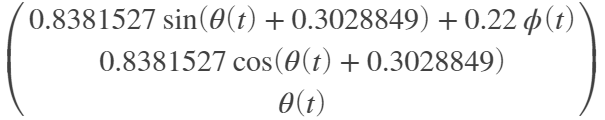
Per ognuno di questi corpi rigidi siamo andati appunto a calcolare:

* Coordinate spaziali nel sistema di riferimento XZ, con l'aggiunta delle coordinate angolari  vettore 3 x 1
* Vettore delle velocità  vettore 3 x 1
* Matrice delle masse  matrice 3 x 3
* Energia cinetica dello specifico corpo rigido  
* Energia potenziale dello specifico corpo rigido 
* Lagrangiana dello specifico corpo rigido  

## Asta

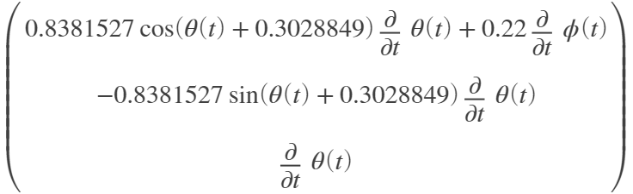
P\_a = [phi\*r+l\_a\*sin(theta+beta); l\_a\*cos(theta+beta); theta]

P\_a(t) =



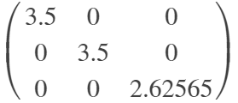
V\_a = diff(P\_a, t)

V\_a(t) =



M\_a = [m\_a,0,0; 0,m\_a,0; 0,0,J\_a + m\_a \*l\_a^2] % Matrice delle masse/inerzie

M\_a =



T\_a = collect(simplify((1/2 \* transpose(V\_a) \* M\_a \* V\_a), 'Steps', 50))

T\_a(t) =



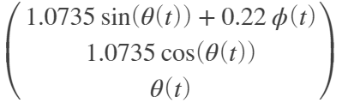
U\_a = m\_a \* g \* l\_a \* cos(theta+beta);

L\_a = collect(simplify((T\_a - U\_a), 'Steps', 50));

## Utente

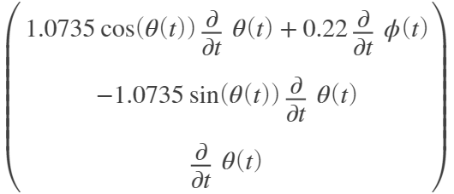
P\_c = [phi\*r+l\_c\*sin(theta+alpha); l\_c\*cos(theta+alpha); theta + alpha]

P\_c(t) =



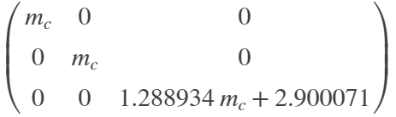
V\_c = diff(P\_c,t)

V\_c(t) =



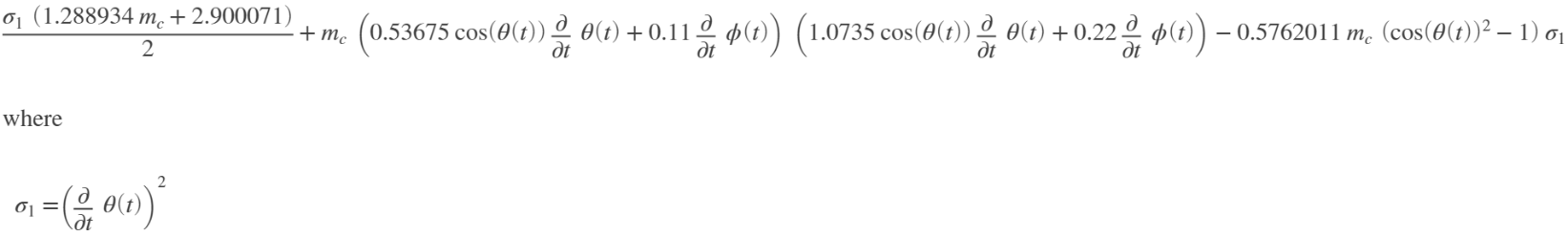
M\_c = [m\_c,0,0; 0,m\_c,0; 0,0,J\_c + m\_c \* l\_c^2]

M\_c =



T\_c = collect(simplify((1/2 \* transpose(V\_c) \* M\_c \* V\_c), 'Steps', 50))

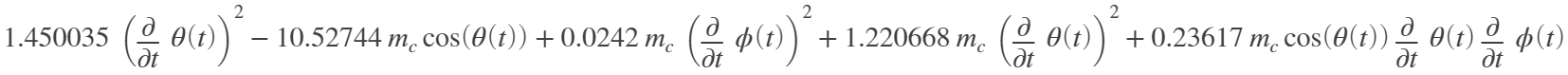
T\_c(t) =



U\_c = m\_c \* g \* l\_c \* cos(alpha+theta);

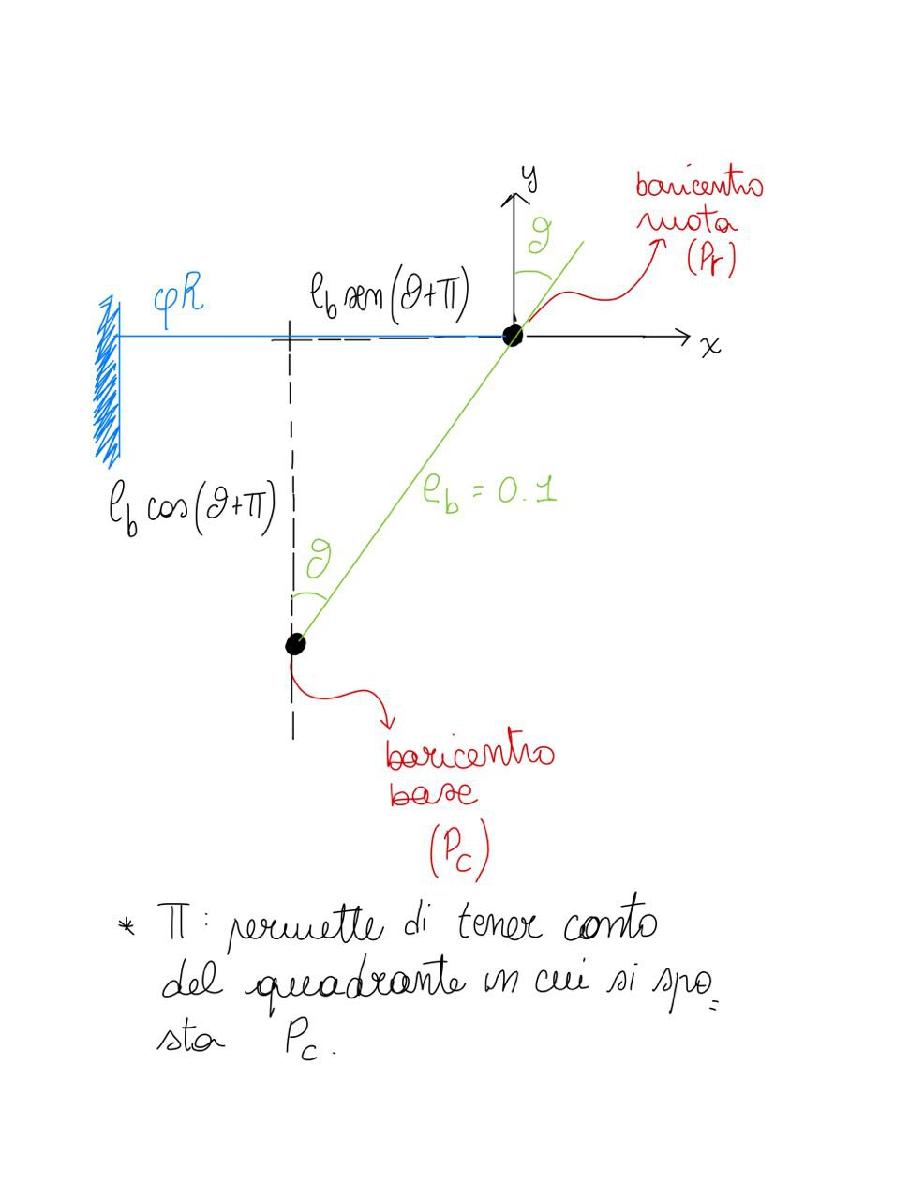
L\_c = collect(simplify((T\_c - U\_c), 'Steps', 50))

L\_c(t) =



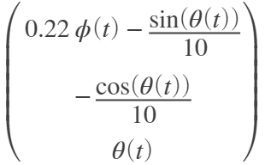
## Chassis (o base)

Come da specifiche, il baricentro della base risulta essere posizionato ad una quota differente rispetto al centro del sistema di coordinate preso come riferimento. Per questo motivo che il suo contributo cinetico e potenziale dipende da quale inclinazione presenta lo chassis stesso, ovvero dall'angolo caratteristico del sistema . Questo fatto è messo in evidenza nell'immagine seguente:



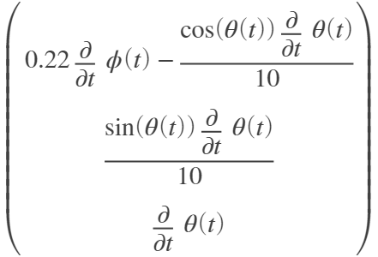
P\_b = [phi\*r + l\_b\*sin(pi+theta); l\_b\*cos(pi+theta); theta]

P\_b(t) =



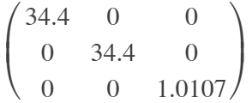
V\_b = diff(P\_b,t)

V\_b(t) =



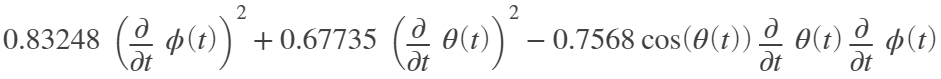
M\_b = [m\_b,0,0; 0,m\_b,0; 0,0,J\_b + m\_b\*l\_b^2]

M\_b =



T\_b = collect(simplify((1/2 \* transpose(V\_b) \* M\_b \* V\_b), 'Steps', 50))

T\_b(t) =



U\_b = m\_b \* g \* l\_b \* cos(pi+theta);

L\_b = collect(simplify((T\_b - U\_b), 'Steps', 50))

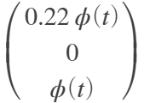
L\_b(t) =



## Ruota

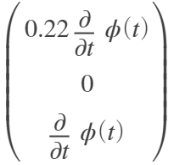
P\_r = [phi\*r; 0; phi]

P\_r(t) =



V\_r = diff(P\_r,t)

V\_r(t) =



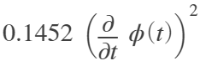
M\_r = [m\_r,0,0; 0,m\_r,0; 0,0,J\_r]

M\_r =



T\_r = collect(simplify((1/2 \* transpose(V\_r) \* M\_r \* V\_r), 'Steps', 50))

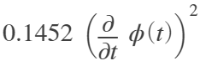
T\_r(t) =



U\_r = 0;

L\_r = collect(simplify((T\_r - U\_r), 'Steps', 50))

L\_r(t) =



## Motore

T\_m = collect(simplify(1/2 \* J\_m \* (vel/tau)^2))

T\_m(t) =



U\_m = 0;

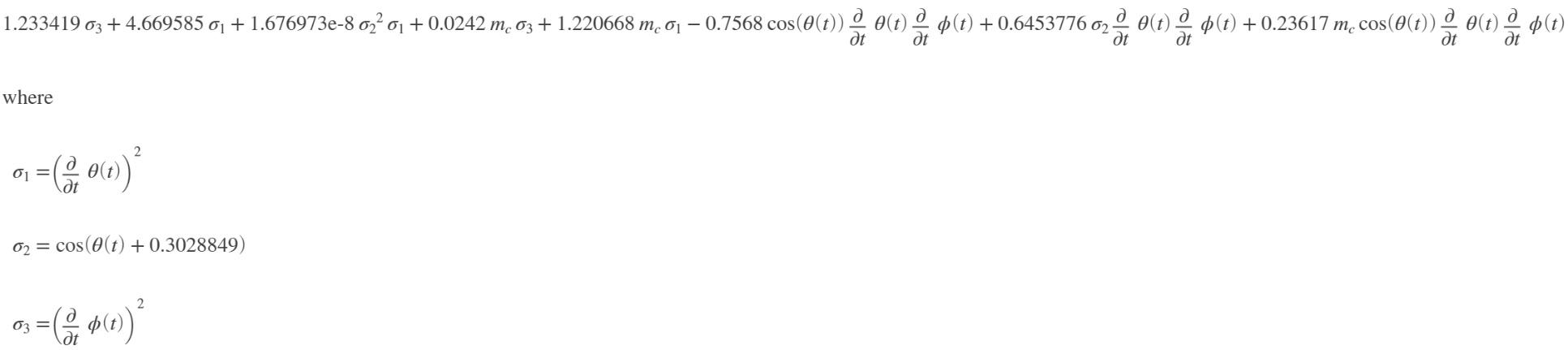
L\_m = T\_m - U\_m;

## Lagrangiana dell'intero sistema

Per confronto, andiamo a definire in prima battuata l'energia cinetica e potenziale totale dell'intero sistema:

T\_totale = collect(simplify((T\_a + T\_b + T\_c + 2 \* T\_r + T\_m), 'Steps', 50))

T\_totale(t) =



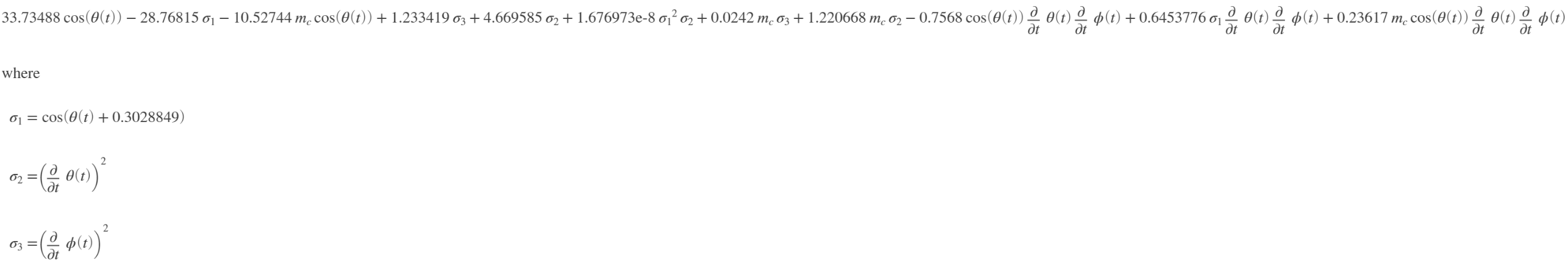
U\_totale = simplify((U\_a + U\_b + U\_c + 2 \* U\_r + U\_m), 'Steps', 50)

U\_totale(t) = 

Andiamo ora a sommare tutte le singole lagrangiane dei vari corpi rigidi (considerando doppio l'apporto delle ruote).

L = collect(simplify((2\*L\_r + L\_c+ L\_a + L\_b + L\_m), 'Steps', 50))

L(t) =



L\_q = subs(L,{phi,vel,theta,vel\_ang},{q\_1 q\_1\_p q\_2 q\_2\_p}) % Esprimo la Lagrangiana rispetto ai simboli di coordinate libere generiche (q\_1 etc)

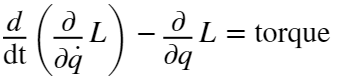
L\_q(t) = 

# Equazioni del moto

Calcoliamo ora le equazioni del moto, considerando, come coordinate libere

* 
* 

Utilizziamo questa formulazione:



## Calcolo equazioni del moto rispetto a phi ()

dL\_dq1 = diff(L\_q,q\_1);

dL\_dphi = subs(dL\_dq1,{q\_1 q\_1\_p q\_2 q\_2\_p},{phi,vel,theta,vel\_ang});

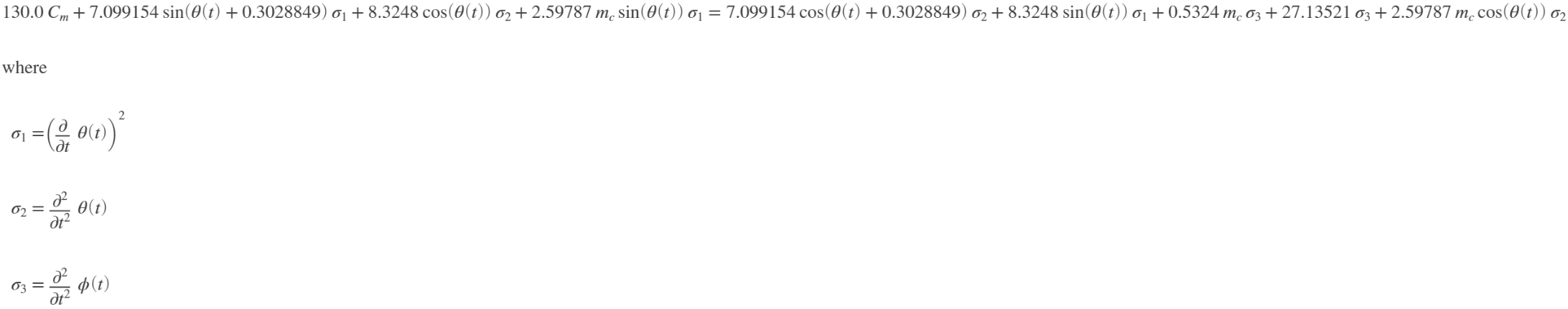
dL\_dq1p = diff(L\_q,q\_1\_p);

dL\_dvel = subs(dL\_dq1p,{q\_1 q\_1\_p q\_2 q\_2\_p},{phi,vel,theta,vel\_ang});

dL\_dvel\_dt = diff(dL\_dvel,t);

E\_L\_phi = collect(simplify((dL\_dvel\_dt - dL\_dphi == +C\_m / tau), 'Steps', 50))

E\_L\_phi(t) =



## Calcolo equazioni del moto rispetto a theta ()

dL\_dq2 = diff(L\_q,q\_2);

dL\_dtheta = subs(dL\_dq2,{q\_1 q\_1\_p q\_2 q\_2\_p},{phi,vel,theta,vel\_ang});

dL\_dq2p = diff(L\_q,q\_2\_p);

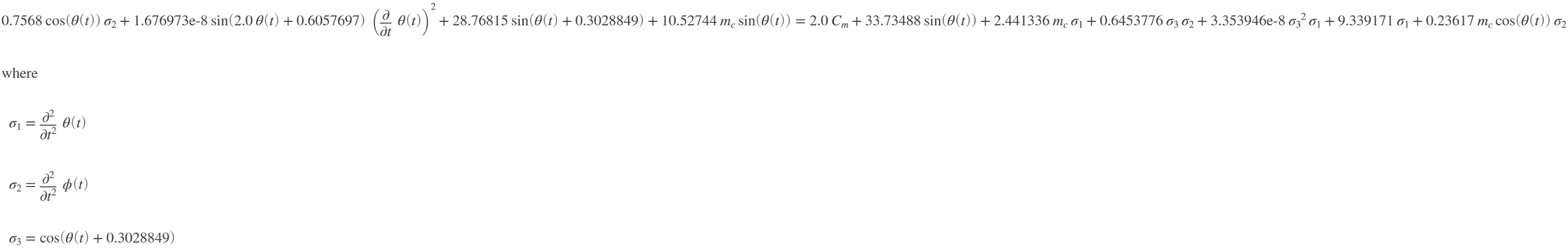
dL\_dvel\_ang = subs(dL\_dq2p,{q\_1 q\_1\_p q\_2 q\_2\_p},{phi,vel,theta,vel\_ang});

dL\_dvel\_ang\_dt = diff(dL\_dvel\_ang,t);

Nel caso dello chassis sono presenti due motori che imprimono una coppia doppia alla struttura rigida.

E\_L\_theta = collect(simplify((dL\_dvel\_ang\_dt - dL\_dtheta == -2 \* C\_m), 'Steps', 50))

E\_L\_theta(t) =



# Risoluzione delle equazioni del moto

Risolviamo ora le due equazioni del moto, ottenendo le variabili e .

acc\_ang = diff(theta, t, 2);

acc = diff(phi, t, 2);

eqns = [subs(E\_L\_theta, {acc,acc\_ang},{q\_1\_pp,q\_2\_pp}), subs(E\_L\_phi, {acc,acc\_ang},{q\_1\_pp,q\_2\_pp})];

%To ensure the order of the returned solutions, specify the variables vars.

%For example, the call [b,a] = solve(eqns,b,a) assigns the solutions for a to a and the solutions for b to b.

S = solve(eqns, q\_1\_pp, q\_2\_pp)

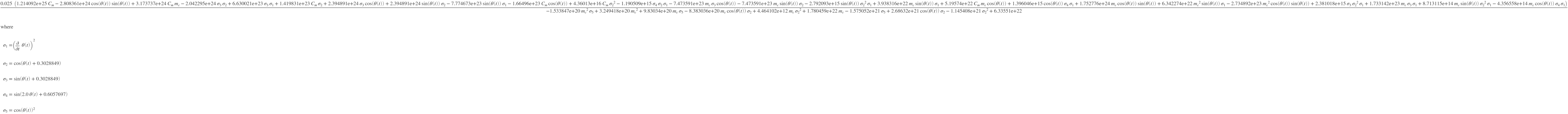
S = *struct with fields:*

q\_1\_pp: [1×1 sym]

q\_2\_pp: [1×1 sym]

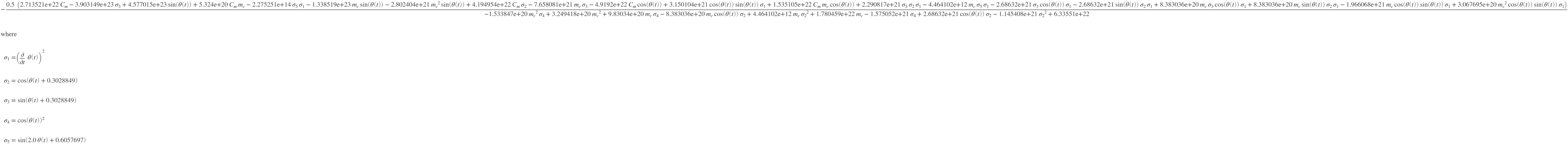
phi2 = S.q\_1\_pp

phi2 =



theta2 = S.q\_2\_pp

theta2 =

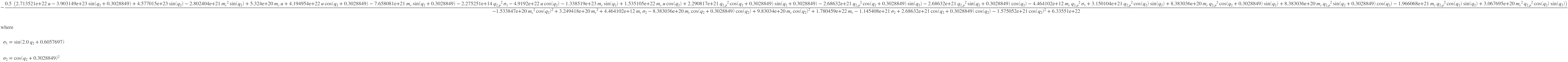


Le due seguenti equazioni simboliche rappresentano le **equazioni non lineari del modello dinamico.**

syms u\_star

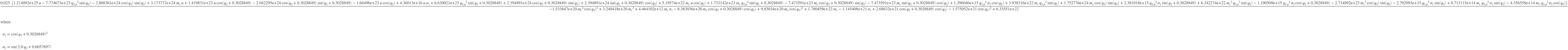
theta2\_differenziabile = subs(theta2,{phi,vel,theta,vel\_ang,C\_m},{q\_1 q\_1\_p q\_2 q\_2\_p u})

theta2\_differenziabile =



phi2\_differenziabile = subs(phi2,{phi,vel,theta,vel\_ang,C\_m},{q\_1 q\_1\_p q\_2 q\_2\_p u })

phi2\_differenziabile =



Le soluzioni così ottenute rappresentano e governano la dinamica del sistema: esse verranno quindi utilizzate per andare ad analizzare il comportamento del sistema considerato reale, utlizzando il controllore progettato sul sistema sucessivamente linearizzato.

% Files for NON linear simulation

matlabFunction(theta2\_differenziabile,'File','theta\_secondo');

matlabFunction(phi2\_differenziabile,'File','phi\_secondo');

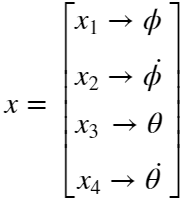
# Linearizzazione delle equazioni

Avrò 2 equilibri per l'angolo theta

* 0 --> equilibrio instabile
* k +  --> equilibrio stabile

*Question ?? : l'equilibrio instabile dell'angolo theta, essendo presente l'asta spostata di un certo valore in avanti, risulterà essere diverso da 0°. Come possiamo porci nei confronti di questo offset sull'equilibrio? Andiamo a linearizzare intorno a questo equilibrio (in cui quindi per l'angolo* *non avremmo 0° ma circa 1°?).*

Il vettore di stato con il quale andremo a lavorare è il seguente:



Inoltre, per definire il nostro sistema SIMO in forma matriciale, avremo bisogno del supporto delle matrici A-B-C-D.

Per fare questo definiamo queste funzioni di supporto:





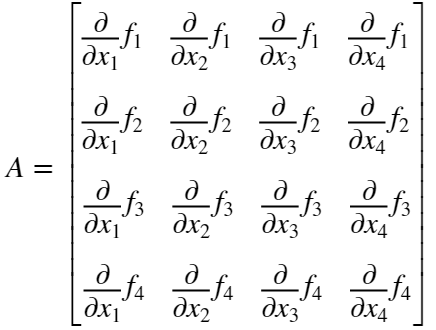
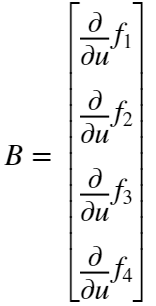
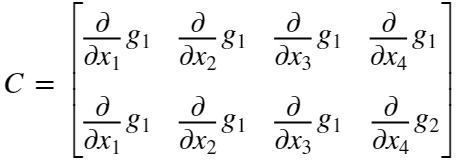
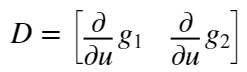




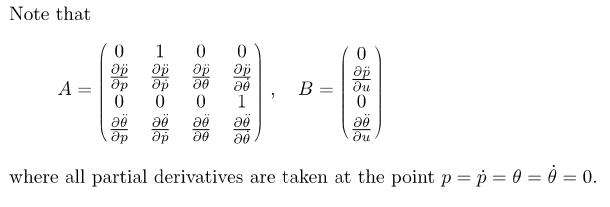




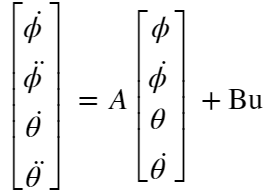
Da qui è quindi possibile poi ricavare le matrici A-B-C-D come segue:

* 
* 
* 
* 

Dove le matrici A e B hanno nello specifico contesto in cui stiamo lavorando avranno questa forma:



Queste matrici ci permettono di scrivere:



Come prima cosa andiamo ad assegnare, al simbolo  il valore 0.

phi2\_differenziabile = subs(phi2\_differenziabile,{m\_c},{70});

theta2\_differenziabile = subs(theta2\_differenziabile,{m\_c},{70});

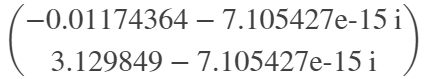
Cerchiamo poi l'angolo in cui si ha l'equilibrio, ovvero quello in cui, con lo stato tutto posto a 0 (ovvero posizioni e velocità), si ottiene accelerazione nulla: da questo ne consegue quindi che il sistema risulta essere fermo.

NB: questo angolo è differente da 0, per la presenza dell'asta che sposta in avanti l'equilibrio del segway, che non è quindi più lungo la verticale.

NB2: abbiamo anche verificato che questo angolo *theta\_equilibrio* sia uguale a 0 nel caso in cui l'asta del manubrio si trovi sopra la base (ovvero w\_b = 0).

equilibri = solve(subs(theta2\_differenziabile,{q\_1 q\_1\_p q\_2\_p u},{0,0,0,0})==0,q\_2,'IgnoreAnalyticConstraints',false)

equilibri =



La soluzione di questa equazione ritorna due posizioni di equilirbio: guardando i valori si nota che la parte complessa può essere trascurata, poichè prossima a zero.

Le due posizioni di equilibrio quindi sono

values\_rad = real(equilibri); % [rad]

theta\_equilibrio = values\_rad(1)

theta\_equilibrio = 

Andiamo a linearizzare intorno al seguente equilibrio.

equilibrio = {0,0,theta\_equilibrio,0,0,0};

%equilibrio = {0,0,0,0,0};

**Definizione della matrice A:**

A11 = 0;

A12 = 1;

A13 = 0;

A14 = 0;

A21 = subs(diff(phi2\_differenziabile,q\_1), {q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A22 = subs(diff(phi2\_differenziabile,q\_1\_p),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A23 = subs(diff(phi2\_differenziabile,q\_2), {q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A24 = subs(diff(phi2\_differenziabile,q\_2\_p),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A31 = 0;

A32 = 0;

A33 = 0;

A34 = 1;

A41 = subs(diff(theta2\_differenziabile,q\_1), {q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A42 = subs(diff(theta2\_differenziabile,q\_1\_p),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A43 = subs(diff(theta2\_differenziabile,q\_2), {q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A44 = subs(diff(theta2\_differenziabile,q\_2\_p),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

A = double([A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33 A34; A41 A42 A43 A44])

A = 4×4

0 1.0000 0 0

0 0 -15.2286 0

0 0 0 1.0000

0 0 5.4392 0

**Definizione della matrice B:**

B11 = 0;

B21 = subs(diff(phi2\_differenziabile,u),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

B31 = 0;

B41 = subs(diff(theta2\_differenziabile,u),{q\_1 q\_1\_p q\_2 q\_2\_p u u\_star},equilibrio);

B = double([B11;B21;B31;B41])

B = 4×1

0

2.7498

0

-0.2612

**Definizione della matrice C:**

%C = [0,0,1,0] % Riportiamo in uscita solamente l'angolo

C = eye(4) % Riportiamo in uscita tutte le variabili di stato

C = 4×4

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

**Definizione della matrice D:**

D = zeros(4,1)

D = 4×1

0

0

0

0

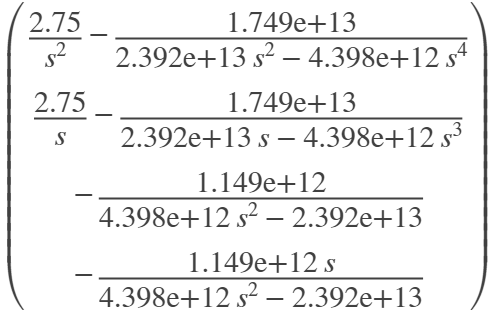
La procedura per la determinazione della funzione di trasferimento di un sistema, non strettamente proprio, è definita dalla relazione:



syms s G(s);

G = vpa((C\*(s\*eye(4)-A)^(-1)\*B+D),4)

G =



# Salvataggio workspace

Andiamo a salvare il workspace per poterlo poi facilmente riutilizzare in fase di simulazione.

save('WS\_VAB\_wb\_0\_5.mat');