Assignment 2 Report - Vehicle controllers

Created	@December 25, 2023 11:22 AM
Class	PA 046
Type	Assignment
Materials	https://github.com/andreatirelli3/VM_assignment/tree/main
Reviewed	☑

Introduzione

In questo report vengono esposti e commentati i risultati ottenuti dei modelli di sterzo:

- Pole Placement
- LQ-Regolator

In aggiunti entrambe le simulazione, hanno implementato il controllore PI per l'accelerazione, con lo scopo di raggiungere una velocità longitudinale desiderata.

Controllore di sterzo

Il modello ci serve per permettere al nostro veicolo di seguire un path desiderato. Per il path desiderato, si utilizza come modello del veicolo, il modello Road Aligned, questo perché è un modello che ci ritorna come stati gli errori del path attuale rispetto quello desiderato. Questi errori saranno usati dal nostro controllore di sterzo per far convergere il veicolo verso il path desiderato.

Pole Placement

Il modello Pole Placement, si basa nel piazzare 4 autovalori, di cui 2 immaginari che saranno i nostri autovalori dominanti.

```
function K = design_k(A, B, delta, w_n)
    % Pole computation
    p1 = -delta*w_n + 1i*w_n*sqrt(1 - delta^2); % Dominant eiger
    p2 = -delta*w_n - 1i*w_n*sqrt(1 - delta^2); % Dominant eiger
    p = [p1; p2; -50; -100];                    % eigenvalues ve

    % Design K placing the eigenvalues in the space
    K = place(A,B,p);
end
```

LQ-Regolator

Il modello LQ-Regolator, viene implementato attraverso la funzione lqr di matlab.

```
function [Q, K, S, CLP] = lq_regolator(A, B)
    Q = 0.01;
    R = 10;
    N = 0;
    [K,S,CLP] = lqr(A, B, Q, R, N);
end
```

Controllore di velocità

Il controllore PI è il nostro controllore di velocità; o meglio di accelerazione (a) per arrivare a una velocità desiderata.
Si basa nel integrare per ogni istante di tempo, la differenza la la velocità attuale e quella desiderata, attraverso i valori di
k_p e k_i ottenere una a. I valori di k_p e k_i sono regolati in base al T_s (settling time) massimo per arrivare a tale velocità.

```
function [sys, k_p, k_i, k_pinterval] = find_pic_values(tau, rat
    k_pinterval = 0:.01:.75;

    sys = tf([1 ratio], [tau 1 0 0]);
```

```
    k_p = .7;
    k_i = k_p * ratio;
end

function x_des = PI_integrator(t_i, v_x, v_des)
    % Symbolic variables
    syms x_des_sym(t)
    x_des_sym(t) = int(v_x - v_des, 0, t);

    x_des = double(x_des_sym(t_i));
end

function PI = PI(t, v_x, v_des, k_p, k_i)
    x_des = PI_integrator(t, v_x, v_des);

    PI = -k_p*(v_x - v_des) - k_i*x_des;
end
```

⚠️ Purtroppo non sono riuscito a capire come gestire il T_s nella funzione `tf()` per ottenere dei k_p e k_i corretti, quindi ho usato valori di `.7` e `.7 * ratio`, consigliati nel libro.

Risultati ottenuti

I risultati ottenuti sono tutti svolti su intervalli di velocità positivi:

- 70 to 90;
- 30 to 120.

Però per svolgere correttamente la simulazione, bisogna aggiungere un altro componente, che viene chiamato `power_train()` nella simulazione. Il suo scopo è quello di calcolare i diversi K per le varie velocità. Questo ci serve, perché il modello Road

Aligned è dipendente da v_x, quindi A, B e B_d sono diversi e il controllore K è dipendente da A e B.

```
function K = power_train(v_interval, flag)
% flag = true => Pole Placement, call design_k
%
% flag = false => LQ Regolator, call lq_regolator

d_vx = 10;

% Vehicle geomtry
[mass, i_z, l_f, l_r, C_f, C_r] = vehicle_geometry();

v_min = v_interval(1) * 1000 / 3600;
v_max = v_interval(2) * 1000 / 3600;

% Min v_x in the velocity interval
% Road Allignment system matrices A, B, B_d
[A_min, B_min, B_dmin] = road_aligned_matrices(mass, i_z, l_f, l

% Max v_x in the velocity interval
% Road Allignment system matrices A, B, B_d
[A_max, B_max, B_dmax] = road_aligned_matrices(mass, i_z, l_f, l

% Steering constraints
[Ts, overshoot_max] = steering_constraints();

% Controller constraints
[delta, S, w_n] = controller_constraints(Ts, overshoot_max);

if flag
    K_min = design_k(A_min, B_min, delta, w_n);
    K_max = design_k(A_max, B_max, delta, w_n);
else
    [Q_min, K_min, S_min, CLP_min] = lq_regolator(A_min, B_min),
```

```
    [Q_max, K_max, S_max, CLP_max] = lq_regolator(A_max, B_max);
end

% Initialize cell arrays
K = {};
K{end + 1} = K_min;

for v_x = v_interval(1) + 10:d_vx:v_interval(2) - 1
    % Normalize in m/s
    v_x = v_x * 1000 / 3600;

    % Road Allignment system matrices A, B, B_d
    [A, B, B_d] = road_aligned_matrices(mass, i_z, l_f, l_r, C_
    if flag
        K_t = design_k(A, B, delta, w_n);
    else
        [Q_t, K_t, S_t, CLP_t] = lq_regolator(A, B);
    end

    % Add the new controller K_t to K
    K{end+1} = K_t;
end



K{end + 1} = K_max;

end
```

⚠️ La fase di power training per il calcolo dei vari K lavora su step di 10 km/h. Ciò non è completamente corretto, perché andrebbe implementata l'equazione di Lyapunov per controllare se un K è stabile per una determinata v_x.

70 to 90

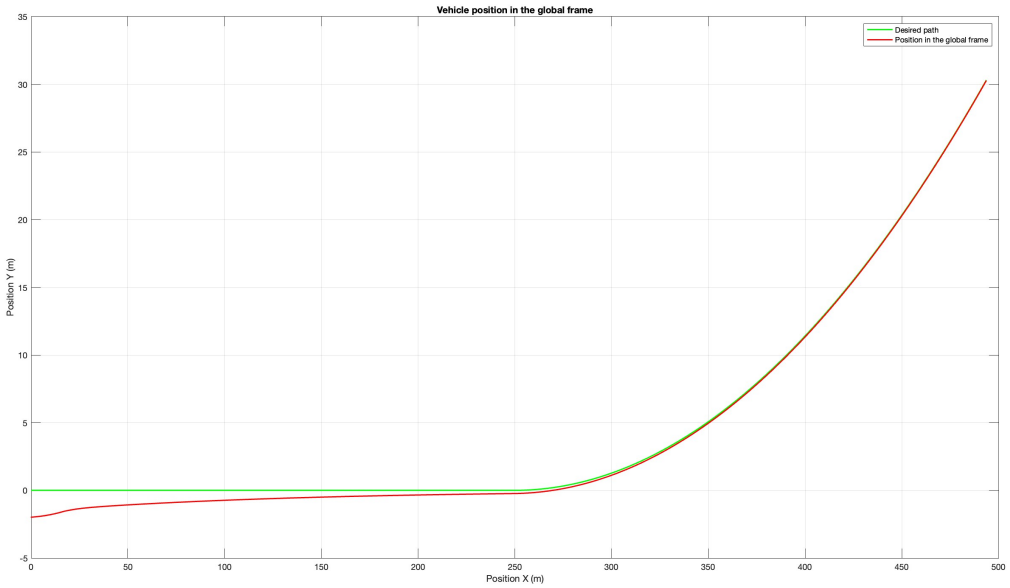


Grafico della posizione desiderata e nel frame globale usato il controllore K Pole Placement.

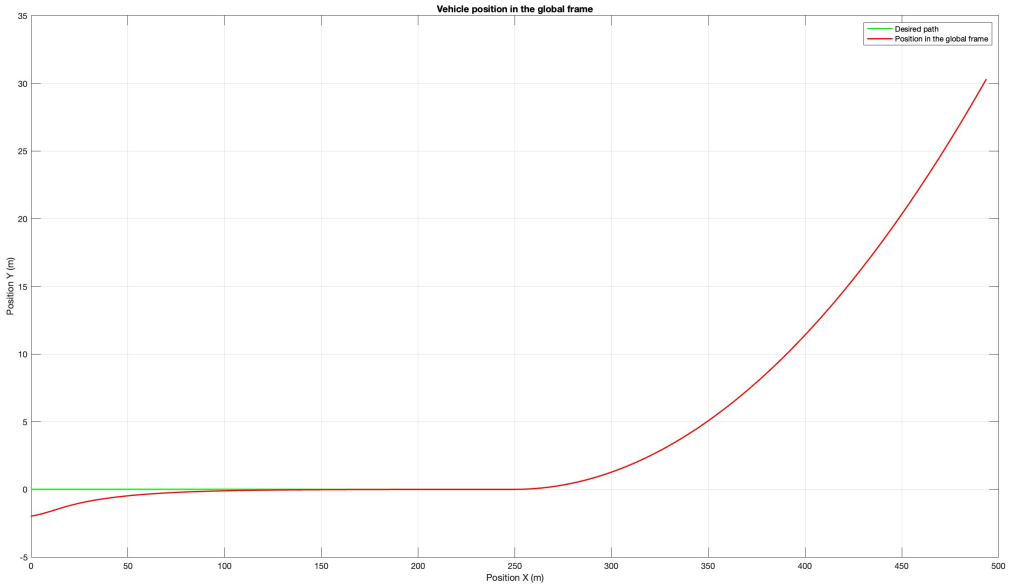
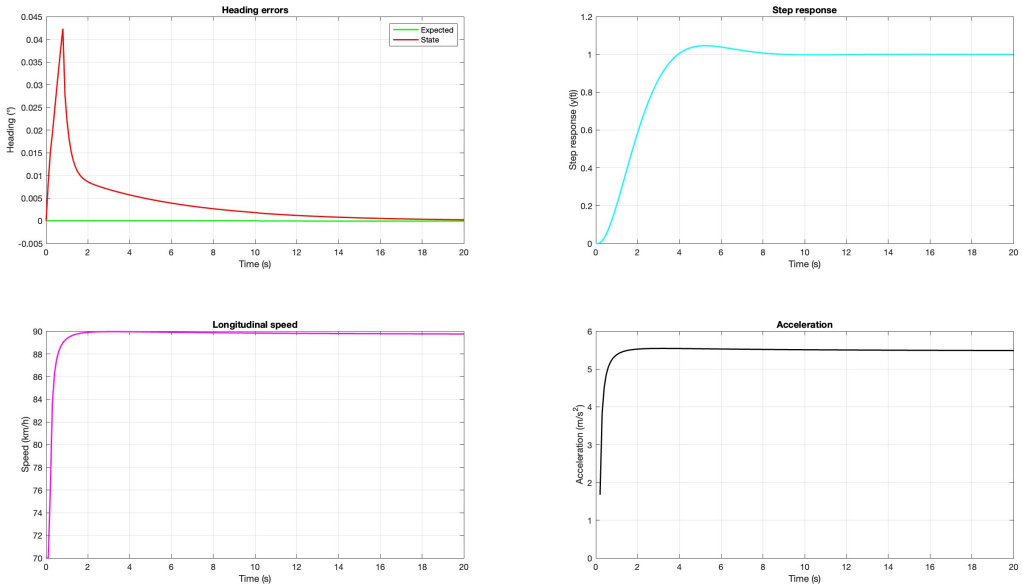


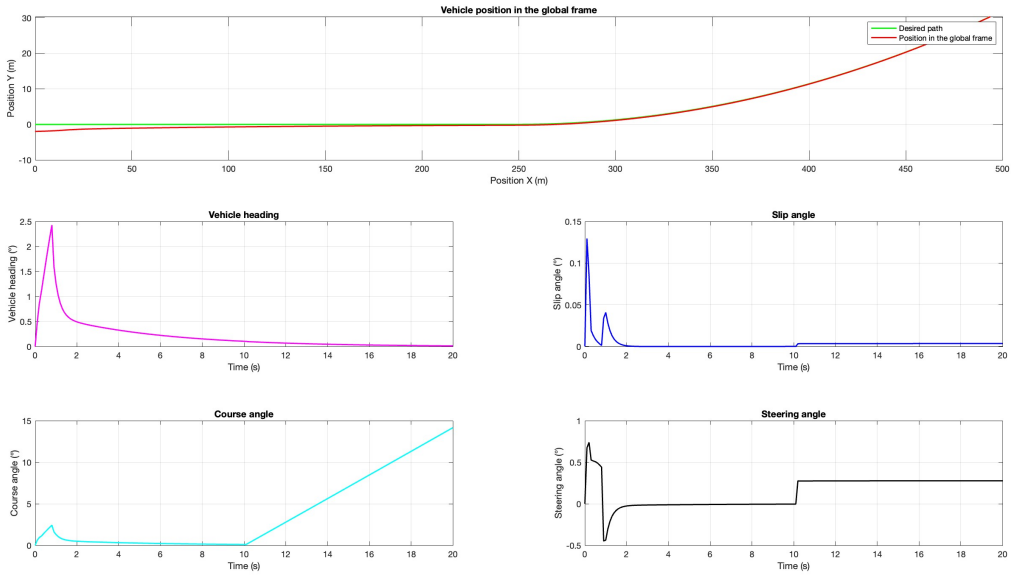
Grafico della posizione desiderata e nel frame globale usato il controllore K LQ-Regolator.

Quello che notiamo è che l’implementazione LQ-Regolator converge più velocemente alla soluzione corretta, questo comportamento è dovuto perché il controllore K è più preciso rispetto a quello generato da Pole Placement per una determinata v_x.
Il comportamento del Pol Placement può essere mitigato con un po’ più di tweak degli autovalori generati e con un settling time più preciso per il caso d’uso richiesto.

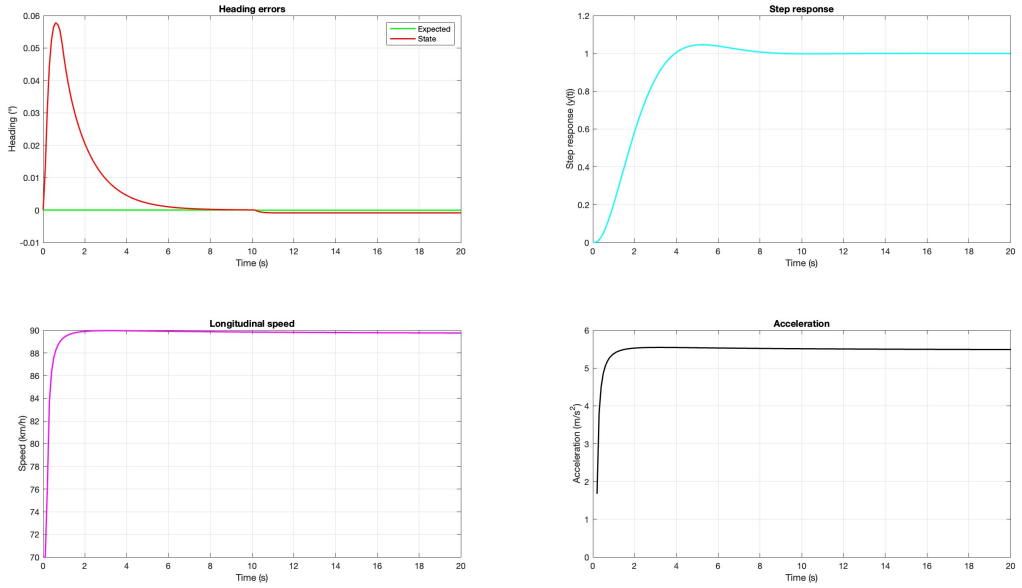
Restanti grafici



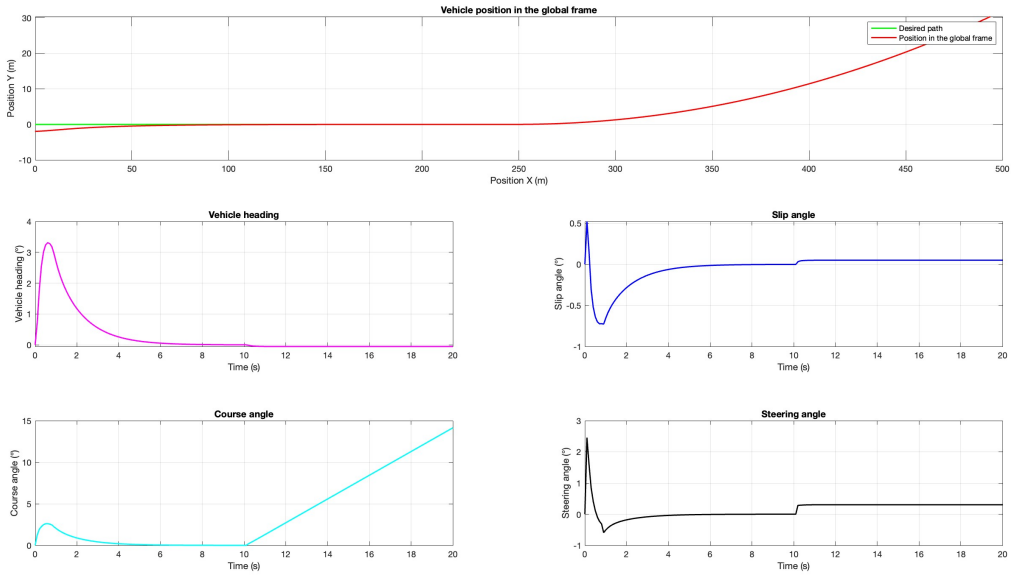
Pole Placement - Heading error, Step response, Longitudinal speed e Acceleration fornita dal controllore.



Pole Placement - Position in the global frame, Heading, Slip angle, Course angle, Steering angle.



LQ Regolator - Heading error, Step response, Longitudinal speed e Acceleration fornita dal controllore.



LQ Regolator - - Position in the global frame, Heading, Slip angle, Course angle, Steering angle.

30 to 120

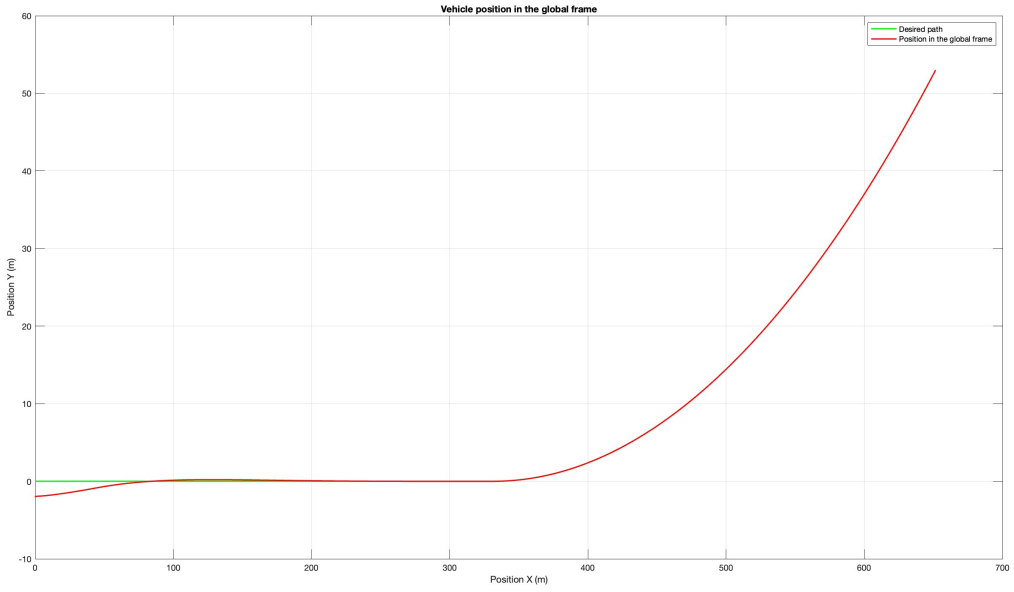


Grafico della posizione desiderata e nel frame globale usato il controllore K Pole Placement.

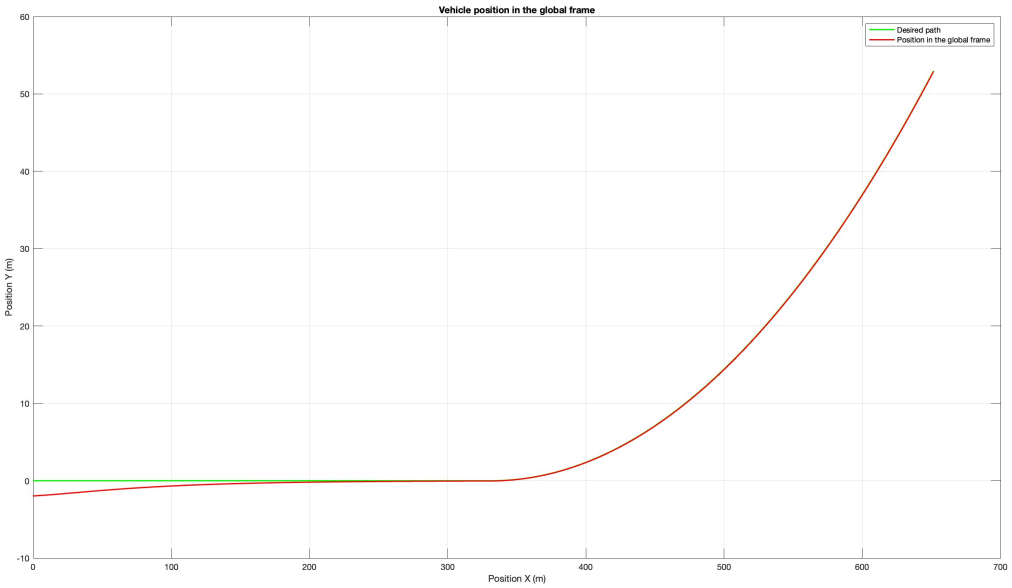
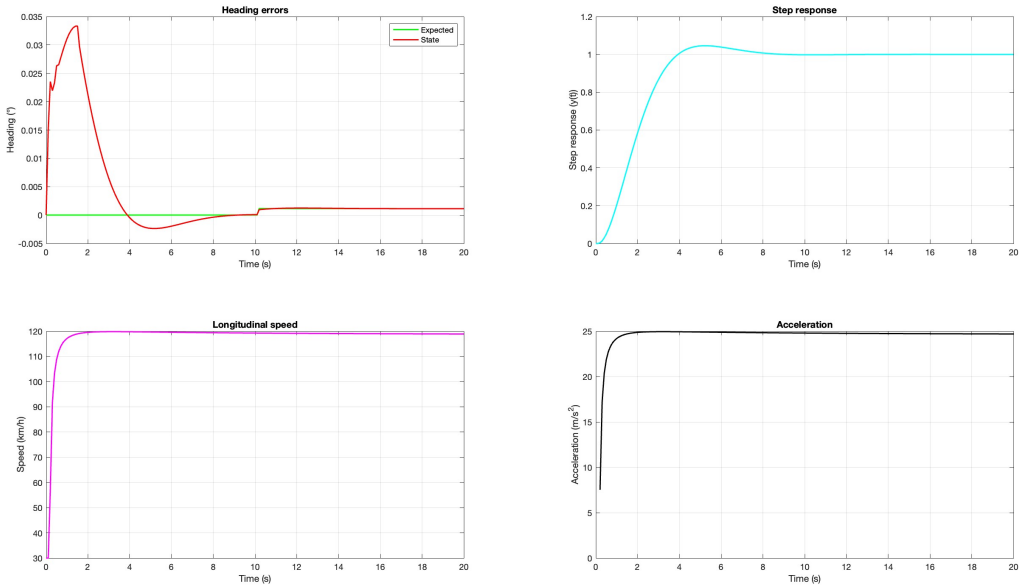


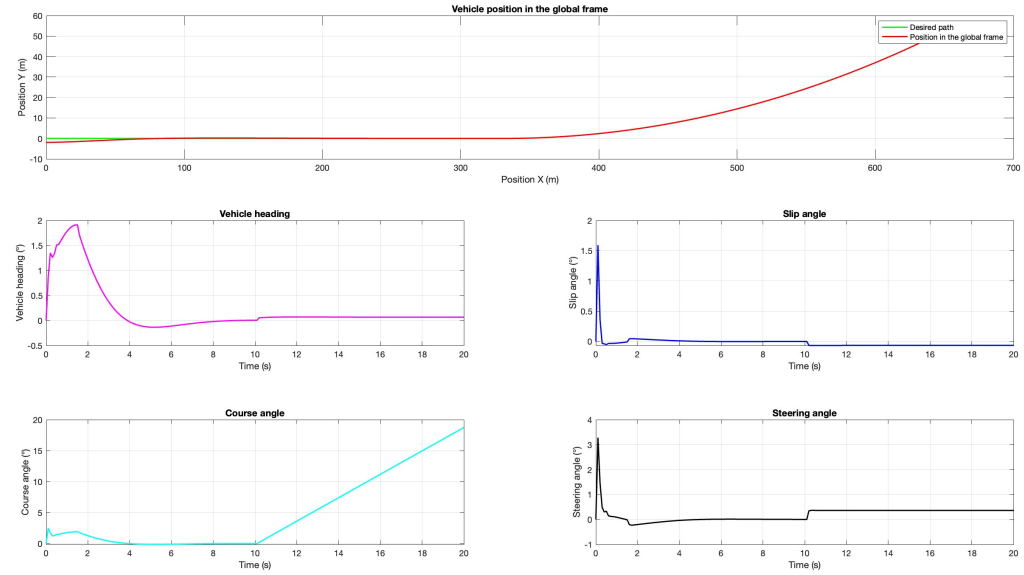
Grafico della posizione desiderata e nel frame globale usato il controllore K LQ-Regolator.

Il comportamento che otteniamo è analogo al fenomeno descritto prima. Quello che è osservabile che la differenza tra Pole Placement e LQ-Regolator non è molta, anzi Pole Place converge più rapidamente verso la soluzione, però con un overshoot lievemente più alto. Questo è dovuto al fatto che con un intervallo così ampio, Pole Placement beneficia del fatto che il controllore K supporti overshoot e sia tweakato per supportare margini di errore e velocità un pelo più ampie; cosa che LQ-Regolator soffre perché i vari controllori generati nella fase di power training sono ad hoc per gli step v_x, ma non per le varie v_{x-avg} che possono essere presenti nella simulazione dovuti dal controllore di velocità. Una possibile soluzione potrebbe essere quella di ridurre il range di step di v_x, passare da 10 a 5, oppure implementare il controllo con l'equazione di Lyapunov.

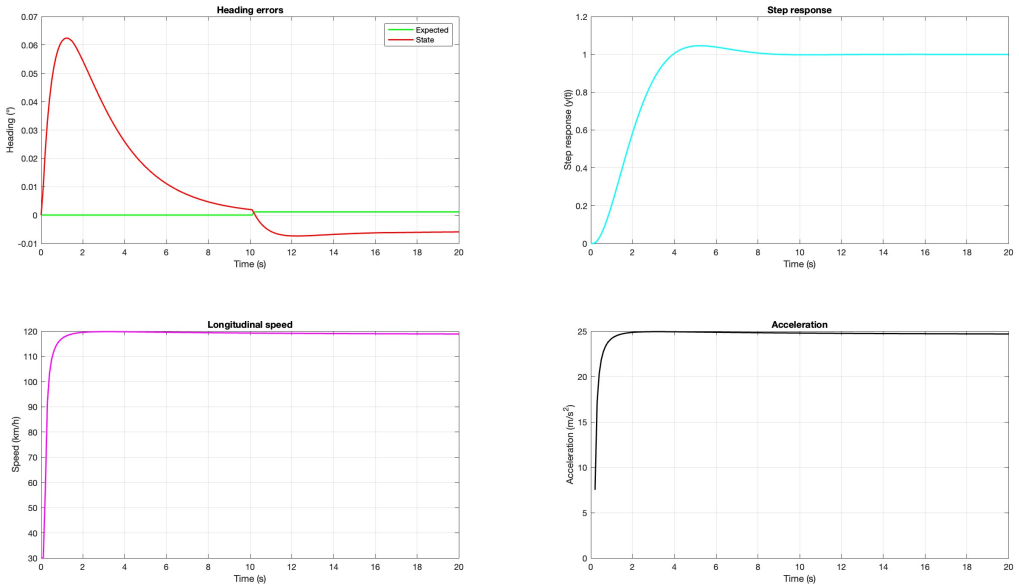
Restanti grafici



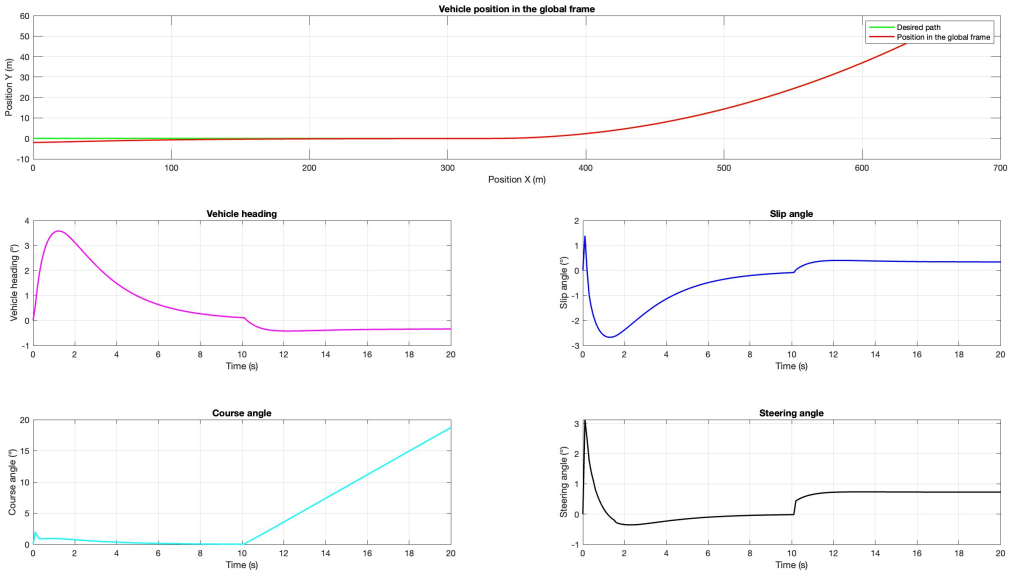
Pole Placement - Heading error, Step response, Longitudinal speed e Acceleration fornita dal controllore.



Pole Placement - Position in the global frame, Heading, Slip angle, Course angle, Steering angle.



LQ Regolator - Heading error, Step response, Longitudinal speed e Acceleration fornita dal controllore.



Guida sulle simulazioni

Per eseguire le simulazioni seguire i seguenti percorsi:

- Pole Placement: `$root/controllers/steering/sim_pole_place.m`

- LQ Regolator: `$root/controllers/steering/sim_lqr.m`

Sottostante vi sono elencati i vari import all'interno del progetto dove si possono trovare le singole implementazioni delle varie funzioni usate nella simulazione.

```
%% Envirorment imports
% project root directory
root_folder = fileparts(fileparts(pwd));
% controllers root directory
controllers_folder = fileparts(pwd);
% common directory
common_sym_path = [root_folder, '/common'];
% models directory
models_folder = [root_folder, '/models/src'];
% road aligned directory
road_aligned_folder = [models_folder, '/road_aligned'];
% controllers source directory
controllers_src_folder = [pwd, '/src'];
% pole placement directory
pole_placement_folder = [controllers_src_folder, '/pole_placemer
% velocity controllers directory
velocity_controlleres_folder = [controllers_folder, '/velocity/s
```