E-TEAM SQUADRA CORSE

Settore

POWERTRAIN

Fausto Antonelli 2024/25

Contents

Chapter 1

Capitolo 1

Fausto Antonelli 2024/25

1.1 Stato dell'arte

1.2 Relazione tra coppia e corrente del motore

Quello che segue è una sintesi di quanto per ora fatto da altri settori, che commenterò in base agli attuali svluppi del veicolo.
La relazione tra coppia T (in Nm) e corrente I (in A) è data da: $T = k \cdot I$, dove k è la costante di coppia del motore. Anche se il valore esatto di k non è noto con precisione (datasheet e misure discordanti), si può considerare un valore iniziale di riferimento: $k = 0,479$.

1.2.1 Distribuzione della corrente nei due motori

Entrambi i differenziali spartiscono la corrente (e quindi la coppia) tra i due motori secondo una variabile detta ΔI. La relazione tra corrente richiesta e posizione del pedale è **lineare**:

- 0% pedale $\rightarrow$ 0 A (0 Nm)
- 100% pedale $\rightarrow$ corrente massima (in *Autocross*: 230 A)

Differenziale 1: Skidpad

Ingressi

- Velocità angolari ruote anteriori: ω_L, ω_R
- Corrente massima richiesta $I_{\max}$ (da posizione pedale)

Velocità veicolo stimata

$$v = \frac{\omega_L + \omega_R}{2} \cdot r \cdot 3.6 \quad r := \text{raggio della ruota in metri.} \tag{1.1}$$

Legge per la spartizione della corrente

$$\Delta I = k \cdot 3.6 \cdot r \cdot v \quad k := \text{costante di regolazione nel range } [5,\ 8]. \tag{1.2}$$

Corrente inviata ai motori

$$I_L = I_{\max} \cdot \left(1 - \frac{\Delta I}{2}\right), \qquad I_R = I_{\max} \cdot \left(1 + \frac{\Delta I}{2}\right). \tag{1.3}$$

Commento: L'idea di fondo è corretta e questo schema di ripartizione è lo stesso impiegato nell'attuale versione di differenziale.
Purtroppo non trovo accurata la "Legge per la spartizione" in quanto non tiene di conto della possibilità che le ruote anteriori subiscano bloccaggi o che queste non copino perfettamente l'asfalto a causa di disconnessioni o asperità.
Quindi concettualmente l'idea è ottima ma viste le condizioni dei tracciati che ci si presentano questa soluzione non performerebbe. Discorso analogo si può fare per il prossimo modello:

Differenziale 2: EVO

Ingressi

- Velocità angolari ruote: ω_L, ω_R
- Corrente massima richiesta $I_{\max}$

Legge per la spartizione della corrente

$$\Delta I = k \cdot (\omega_R - \omega_L) \qquad k \in [0.007,\ 0.01]. \tag{1.4}$$

Nota: il valore corretto di k potrebbe essere circa un ordine di grandezza inferiore.

Saturazione per sicurezza

$$\Delta I \in [-1.5, +1.5]. \tag{1.5}$$

Corrente inviata ai motori

$$I_L = I_{\max} \cdot \left(1 + \frac{\Delta I}{2},\right) \qquad I_R = I_{\max} \cdot \left(1 - \frac{\Delta I}{2}\right). \tag{1.6}$$

1.3 Differenziale elettronico

La progettazione di questo differenziale elettronico e quindi la struttura finale totalmente sono totalmente condizionate dalla disponibilità di in put utili che i sensori montati a bordo possono darci.
Nella forma finale il differenziale si basa, oltre che sui dati fisici fissi del veicolo, sui dati dei 4 potenziometri affiancati al sistema sospensivo e ai dati di velocità delle singole gomme

L'idea alla base del funzionamento del differenziale è calcolare l'angolo di sterzo del veicolo ad un tempo t sapendo quanto sono caricati o scaricati gli ammortizzatori in quel momento.
Inoltre il differenziale svolge altre due attività:

- **SlipControll** : confrontando la velocità del veicolo (ottenuta indirettamente come media della velocità delle ruote anteriori) con la velocità delle singole ruote posteriori ed in caso di discrepanze tagliando in maniera proporzionale la corrente al motore designato.
- **RearBaias** : sempre leggendo i dati dei 4 ammortizzatori possiamo capire l'assetto assunto dal veicolo ad ogni istante perciò questa funzione limita la coppia massima disponibile a seconda se l'asse posteriore è più o meno caricato.

1.4 Differenziale con potenziometri

Il nocciolo teorico del differenziale è la seguente formula, che si occupa di tradurre la forza esercitata dai 4 ammortizzatori nell'angolo di sterzo che il veicolo sta seguendo:

$$steeringAngle = \frac{lf * 2 * k_{spring} * delta_{lat} * cos(\theta)}{mass * vehicleSpeed^2},$$

dove:

- lf : passo del veicolo,
- k_{spring} : costante elastica delle molle montate a bordo
- $delta_{lat}$: differenza di forza esercitata dalle molle tra lato destro e sinistro
- $mass$: massa del veicolo,
- $vehicleSpeed$: velocità del veicolo.

Considerando che l'auto stia percorrendo una curva a sinistra allora lo schema delle forze a cui è soggetta è il seguente:

Di conseguenza avremo: $steeringAngle = \frac{l_f}{R}$ con R angolo di sterzo. Bilanciando quindi le forze otteniamo:

$$F_{centripeta} = \frac{m * v^2}{R} = 2 * k_{spring} * delta_{lat} * cos(\theta) = F_{sospensioni}.$$

$$\frac{1}{R} = \frac{2 * k_{spring} * delta_{lat} * cos(\theta)}{m * v^2}.$$

$$steeringAngle = \frac{l_f * 2 * k_{spring} * delta_{lat} * cos(\theta)}{m * v^2}.$$

```
function [perc_RL, perc_RR] = DiffBaseV3(sRL, sRR, sFL, sFR, wRL, wRR, wFL, wFR)

% === Parametri base ===
k_spring = 30000;                           % rigidezza molla [N/m]
theta = deg2rad(10);                        % inclinazione ammortizzatore
mass = 320;                                 % massa veicolo [kg]
g = 9.81;                                   % gravit  [m/s^2]
lf = 2.5;                                   % Passo totale del veicolo [m]
lr = 1.6;                                   % Carreggiata posteriore [m]
wheelRadius = 0.225;                        % raggio ruota [m]
hCG = 0.3;                                  % altezza baricentro

% === Conversione rpm     m/s ===
wRL = wRL * wheelRadius * 2 * pi / 60;
wRR = wRR * wheelRadius * 2 * pi / 60;
wFL = wFL * wheelRadius * 2 * pi / 60;
wFR = wFR * wheelRadius * 2 * pi / 60;

% === Velocit  stimata veicolo (da anteriori) ===
vehicleSpeed = (wFL + wFR) / 2;
vehicleSpeed = max(1, vehicleSpeed);      % minimo per evitare div. per 0

% === Forze verticali dalle sospensioni ===
d_FL =  sFL * cos(theta);
d_FR =  sFR * cos(theta);
d_RL =  sRL * cos(theta);
d_RR =  sRR * cos(theta);

avg_Rear = (d_RR + d_RL)/2;
avg_Front = (d_FL + d_FR)/2;
delta_long = avg_Rear - avg_Front;        % sbilanciamento asse longitudinale
avg_Left = (d_FL + d_RL)/2;
avg_Right  = (d_FR + d_RR)/2;
delta_lat = avg_Left - avg_Right;         % sbilanciamento asse laterale

rearBias = (delta_long)/(avg_Rear/2 + avg_Front/2);      % distribuzione dinamica del
    carico
steeringAngle = (lf * 2 * k_spring * delta_lat)/(mass * vehicleSpeed^2);
% steeringAngle = (lf * delta_d * lr * k_spring) / (mass * vehicleSpeed^2 * hCG);
steeringAngle = max(min(steeringAngle, pi), -pi);        % +/-180

slip_RL = (wRL - vehicleSpeed) / max(vehicleSpeed, 0.1); % perc. slittamento post. Sx
slip_RR = (wRR - vehicleSpeed) / max(vehicleSpeed, 0.1); % perc. slittamento post. Dx

torque_bias_factor = 0.2 + 0.1 * rearBias;
mario = min(torque_bias_factor * steeringAngle, 0.5);    % mario

if delta_lat < 0
    perc_RL = (1 - mario) * (1 - abs(slip_RL));           % curva a sinistra
    perc_RR = (1 + mario) * (1 - abs(slip_RR));
else
    perc_RL = (1 + mario) * (1 - abs(slip_RL));           % curva a destra
    perc_RR = (1 - mario) * (1 - abs(slip_RR));
end
```

1.5 Differenziale con sensori avanzati

Supponendo di avere una gamma più vasta di input potremmo implementare questa versione:

```
function [torqueLeft, torqueRight] = controlTorqueDistributionNessy3(...
    wheelSpeedLeft, wheelSpeedRight, wheelSpeedLeftFront, wheelSpeedRightFront, ...
    steeringAngle, accelerometerXYZ, brakeForce)

%% 1. Parametri veicolo e costanti
maxTorque = 300;                      % Coppia massima in Nm
vp.t2 = 1.6;                          % Carreggiata posteriore in metri
vp.l = 2.5;                           % Passo totale del veicolo in metri
tireFriction = 1.0;                  % Coefficiente di attrito del pneumatico
wheelRadius = 0.225;                 % Raggio della ruota in metri
massVehicle = 320;                   % Massa del veicolo in kg
rearBias = 0.2;                      % Sbilanciamento virtuale verso il retrotreno

%% 2. Conversione velocit  ruote in m/s
wheelSpeedLeft = rpmToSpeed(wheelSpeedLeft, wheelRadius);
wheelSpeedRight = rpmToSpeed(wheelSpeedRight, wheelRadius);
wheelSpeedLeftFront = rpmToSpeed(wheelSpeedLeftFront, wheelRadius);
wheelSpeedRightFront = rpmToSpeed(wheelSpeedRightFront, wheelRadius);

%% 3. Calcolo velocit  veicolo (pesata sul posteriore)
frontWeight = 1 - rearBias;
vehicleSpeed = frontWeight * (wheelSpeedLeftFront + wheelSpeedRightFront) / 2;

%% 4. Calcolo dello slip relativo (approssimato linearmente)
tireSlipLeft = computeSlip(wheelSpeedLeft, vehicleSpeed);
tireSlipRight = computeSlip(wheelSpeedRight, vehicleSpeed);

%% 5. Calcolo del raggio di curvatura
radiusOfCurvature = computeCurvature(steeringAngle, vp.l);

%% 6. Calcolo della forza di trazione disponibile
forceTraction = maxTorque / wheelRadius;      % Forza = coppia / raggio
brakeForce = abs(brakeForce);

%% 7. Fattori di aderenza longitudinale e laterale
lateralFactor = computeLateralFactor(vehicleSpeed, radiusOfCurvature, tireFriction,
    massVehicle);
longitudinalFactor = computeLongitudinalFactor(accelerometerXYZ(2), brakeForce,
    massVehicle, tireFriction, forceTraction);

%% 8. Trasferimento di carico in curva
accelLong = (forceTraction - brakeForce) / massVehicle;
loadTransfer = computeLoadTransfer(vehicleSpeed, radiusOfCurvature, massVehicle, 0.3,
     vp.t2, accelLong);

%% 9. Coppia massima regolata
adjustedTorque = maxTorque * lateralFactor * longitudinalFactor;

%% 10. Distribuzione della coppia in curva
if steeringAngle > 0                    % Curva sinistra
    torqueLeft = adjustedTorque * (1 - loadTransfer);
    torqueRight = adjustedTorque * (1 + loadTransfer);
elseif steeringAngle < 0                % Curva destra
    torqueLeft = adjustedTorque * (1 + loadTransfer);
    torqueRight = adjustedTorque * (1 - loadTransfer);
else                                    % Rettilineo
    torqueLeft = adjustedTorque / 2;
    torqueRight = adjustedTorque / 2;
end

%% 11. Limitazione della coppia
% torqueLeft = max(0, min(torqueLeft, maxTorque));
```

```
% torqueRight = max(0, min(torqueRight, maxTorque));

%% 12. Correzione della coppia in base allo slip
torqueLeft = torqueLeft * (1 - tireSlipLeft);
torqueRight = torqueRight * (1 - tireSlipRight);
end

%% Funzioni di supporto

function speed = rpmToSpeed(rpm, wheelRadius)
    speed = rpm * 2 * pi * wheelRadius / 60;
end

function slip = computeSlip(wheelSpeed, vehicleSpeed)
    slip = abs(wheelSpeed - vehicleSpeed) / max(vehicleSpeed, 1);
    slip = min(max(slip, 0), 1);                    % Limite tra 0 e 1
end

function radius = computeCurvature(steeringAngle, wheelbase)
    if abs(steeringAngle) > 1e-2
        curvature = tan(steeringAngle) / wheelbase;
        radius = 1 / curvature;
    else
        radius = 1000000000;
    end
end

function lateralFactor = computeLateralFactor(vehicleSpeed, radiusOfCurvature,
    tireFriction, massVehicle)
    if isfinite(radiusOfCurvature)
        Fc = massVehicle * vehicleSpeed^2 / radiusOfCurvature;
        maxLateralForce = tireFriction * massVehicle * 9.81;
        lateralFactor = min(Fc / maxLateralForce, 1);
    else
        lateralFactor = 0;
    end
end

function longitudinalFactor = computeLongitudinalFactor(accelY, brakeForce,
    massVehicle, tireFriction, forceTraction)
    accelBrake = brakeForce / massVehicle;
    accelLat = abs(accelY);
    maxAccel = tireFriction * 9.81;
    if (accelLat + accelBrake) >= maxAccel
        longitudinalFactor = 0;
    else
        longitudinalFactor = forceTraction / (massVehicle * (maxAccel - accelLat -
            accelBrake));
        longitudinalFactor = min(max(longitudinalFactor, 0), 1);
    end
end

function loadTransfer = computeLoadTransfer(vehicleSpeed, radiusOfCurvature,
    massVehicle, cgHeight, vp_t2, accelLong)
    if isfinite(radiusOfCurvature)
        loadTransferLat = (massVehicle * vehicleSpeed^2 * cgHeight) / (
            radiusOfCurvature * vp_t2 * 9.81);
        loadTransferLong = (massVehicle * accelLong * cgHeight) / vp_t2;
        loadTransfer = loadTransferLat + loadTransferLong;
    else
        loadTransfer = (massVehicle * accelLong * cgHeight) / vp_t2;
    end
end
```

Listing 1.1: Distribuzione della coppia con aderenza e curvatura

Questo differenziale non è mai stato montato nella macchina per mancanza dei sensori richiesti. Ho quindi fatto uno script che emula degli input verosimili per vere come si sarebbe comportato il differenziale.

waring per test futuri:

- Non è stato facile fare una decente funzione per acquisire l'angolo di sterzo da un file *.mat*, quindi ne lascio 2, una nello script e una commentata sotto.
- I grafici che riportano i risultati nel test sono da leggere nella loro totalità come andamento qualitativo, i dati in input non erano pesati e quindi i valori di output possono essere molto elevati.

```
    %% Carica il tracciato
track_file = 'Hockenheim.mat';
track_data = load(track_file);
track_fields = fieldnames(track_data);
track_points = track_data.(track_fields{1});
x_interp = track_points(:, 1);
y_interp = track_points(:, 2);

%% Parametri veicolo
maxTorque = 600;
vp.t2 = 1.6;
vp.l = 2.5;
tireFriction = 1.0;
wheelRadius = 0.225;
massVehicle = 320;
slip_ratio = 0.05;
dt = 0.01;
mu = 1.0;
g = 9.81;
v_max = 30;
tau = 0.5;

num_points = length(x_interp);

%% Inizializzazione variabili
vehicleSpeed = zeros(1, num_points);
torqueLeft = zeros(1, num_points);

torqueRight = zeros(1, num_points);
wheelSpeedLeft = zeros(1, num_points);
wheelSpeedRight = zeros(1, num_points);
wheelSpeedLeftFront = zeros(1, num_points);
wheelSpeedRightFront = zeros(1, num_points);
brakeForce = zeros(1, num_points);
accelerometerXYZ = zeros(num_points, 3);
steeringAngles = zeros(1, num_points);
curvature = zeros(1, num_points);
radiusCurvature = zeros(1, num_points);

%% Simulazione del tracciato
for i = 2:num_points-1
    dx = x_interp(i) - x_interp(i-1);
    dy = y_interp(i) - y_interp(i-1);
    ddx = x_interp(i+1) - 2*x_interp(i) + x_interp(i-1);
    ddy = y_interp(i+1) - 2*y_interp(i) + y_interp(i-1);

    denom = max((dx^2 + dy^2)^(3/2), 1e-6); % Evita divisioni per zero
    if abs(denom) < 1e-6
        curvature(i) = 0; % Curvatura nulla se il denominatore    troppo piccolo
    else
        curvature(i) = 2 * (dx * ddy - dy * ddx) / denom;
    end

    radiusCurvature(i) = max(1e-3, 1 / max(abs(curvature(i)), 1e-6)); % Limita il
       raggio di curvatura
```

```
55
56      steeringAngles(i) = atan(vp.l / (radiusCurvature(i) + 0.5 * vp.t2 * tan(
            steeringAngles(i)))); % Angolo di sterzata
57      steeringAngles(i) = steeringAngles(i) * sign(curvature(i));
58
59      % steeringAngles(i) = atan(vp.l / radiusCurvature(i)) * sign(curvature(i));
60
61
62      % Calcolo della velocit  massima consentita dalla curvatura
63      V_lim = min(v_max, sqrt(mu * g * radiusCurvature(i))); % Velocit  massima
            consentita
64      V_lim = min(V_lim, sqrt(maxTorque / (massVehicle * wheelRadius))); % Limite
            basato sulla potenza del motore
65      vehicleSpeed(i) = vehicleSpeed(i-1) + (V_lim - vehicleSpeed(i-1)) * (1 - exp(-dt/
            tau)); % Velocit  del veicolo
66
67
68      % Calcolo della forza frenante
69      brakeForce(i) = abs(sin(steeringAngles(i))); % Forza frenante
70
71      % Calcolo della velocit  angolare del veicolo
72      omega_vehicle = vehicleSpeed(i) / radiusCurvature(i);
73
74      % Velocit  delle ruote posteriori
75      wheelSpeedLeft(i)  = (vehicleSpeed(i) - 0.5 * omega_vehicle * vp.t2) / (2 * pi *
            wheelRadius) * 60;
76      wheelSpeedRight(i) = (vehicleSpeed(i) + 0.5 * omega_vehicle * vp.t2) / (2 * pi *
            wheelRadius) * 60;
77
78      % Velocit  delle ruote anteriori (considerando lo slip)
79      wheelSpeedLeftFront(i)  = wheelSpeedLeft(i)  * (1 - slip_ratio);
80      wheelSpeedRightFront(i) = wheelSpeedRight(i) * (1 - slip_ratio);
81
82      % Calcolo dell'accelerazione longitudinale e laterale
83      accelLong = (vehicleSpeed(i) - vehicleSpeed(i-1)) / dt; % Accelerazione
            longitudinale
84      accelLong = max(min(accelLong, mu * g), -mu * g); % Limita l'accelerazione
            longitudinale
85
86      accelLat = min((vehicleSpeed(i)^2 / radiusCurvature(i)) * sign(steeringAngles(i))
            , mu * g); % Accelerazione laterale
87      accelLat = max(min(accelLat, mu * g), -mu * g); % Limita l'accelerazione laterale
88
89      accelerometerXYZ(i, :) = [accelLong, accelLat, 0]; % Accelerometro
90  end
91
92  %% Calcolo distribuzione coppia
93  for i = 1:num_points
94      [torqueLeft(i), torqueRight(i)] = controlTorqueDistributionNessy3(...
95          wheelSpeedLeft(i), wheelSpeedRight(i), wheelSpeedLeftFront(i),
                wheelSpeedRightFront(i), ...
96          steeringAngles(i), accelerometerXYZ(i, :), brakeForce(i));
97
98      % Controllo per evitare coppie nulle o non valide
99      if torqueLeft(i) == 0 || torqueRight(i) == 0
100         fprintf('Zero torque at i=%d: Curv=%f, Steer=%f, SpeedL=%f, SpeedR=%f\n', ...
101                 i, curvature(i), steeringAngles(i), wheelSpeedLeft(i),
                        wheelSpeedRight(i));
102     end
103 end
104
105 %% Visualizzazione
106 figure('Name', 'Analisi del veicolo', 'NumberTitle', 'off', 'Position', [100 100 1200
        800]);
107
108 subplot(3, 2, 1);
```

```
plot(steeringAngles, 'g-', 'LineWidth', 2);
title('Angolo di sterzata'); xlabel('Punti'); ylabel('[rad]'); grid on;

subplot(3, 2, 2);
plot(wheelSpeedLeft, 'm-', 'LineWidth', 2); hold on;
plot(wheelSpeedRight, 'c-', 'LineWidth', 2);
title('Velocit  delle ruote'); xlabel('Punti'); ylabel('[rpm]');
legend('Ruota sinistra', 'Ruota destra'); grid on;

subplot(3, 2, 3);
plot(accelerometerXYZ(:, 2), 'k--', 'LineWidth', 2);
title('Accelerazione laterale'); xlabel('Punti'); ylabel('[m/s^2]'); grid on;

subplot(3, 2, 4);
plot(brakeForce, 'r-', 'LineWidth', 2);
title('Forza frenante'); xlabel('Punti'); ylabel('[N]'); grid on;

subplot(3, 2, [5, 6]);
plot(torqueLeft, 'r-', 'LineWidth', 2); hold on;
plot(torqueRight, 'g-', 'LineWidth', 2);
title('Distribuzione della coppia'); xlabel('Punti'); ylabel('[Nm]');
legend('Coppia sinistra', 'Coppia destra'); grid on;

figure;
plot(x_interp, y_interp, 'k-', 'LineWidth', 2); hold on;
scatter(x_interp, y_interp, 20, steeringAngles, 'filled');
colorbar; title('Percorso con angoli di sterzata');
axis equal;
```

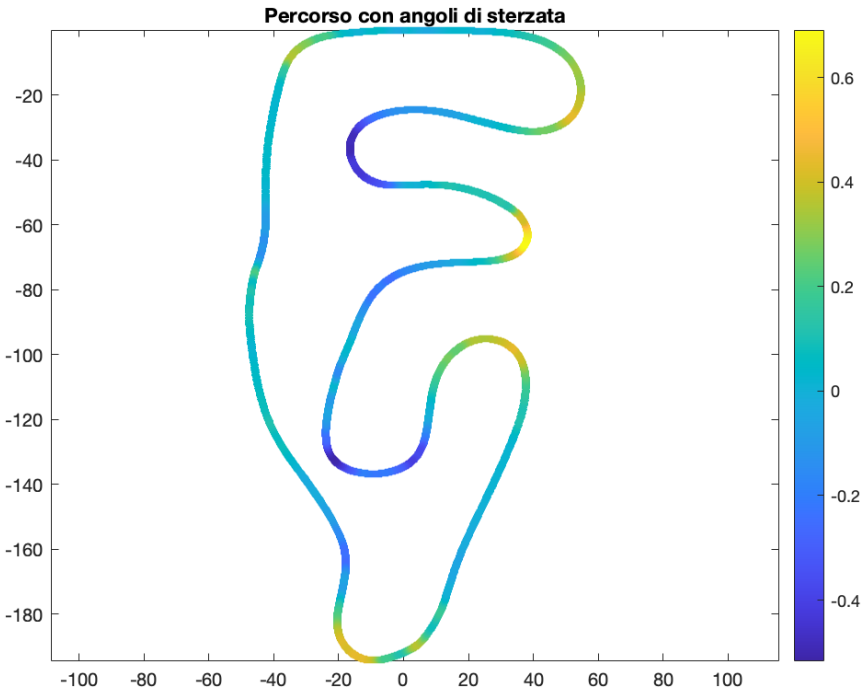
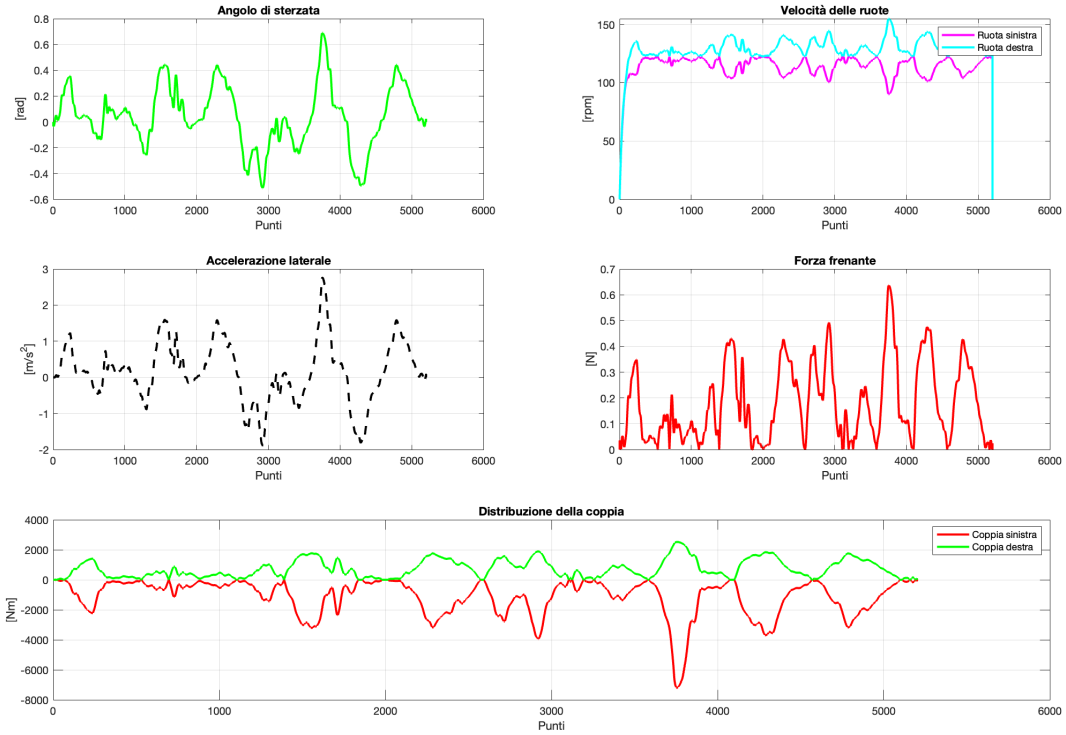


Figure 1.1: Tracciato di Hockenheim, scalato cromaticamente rispetto agli angoli di curvatura tenuti nel test



1.6 Obiettivo raggiunto?

1.6.1 si, a che livello?

1.6.2 no, perché ?

Chapter 2

Capitolo 2

2.1 Analisi telemetria test

In questa sezione saranno analizzati i dati delle telemetrie rilevati in diversi contesti. Considerazioni iniziali:

- In condizioni ideali l'andamento del calore dovrebbe essere una curva del tipo

  $$\gamma(t) = ke^{\frac{1}{t^2}}, \quad k = \textit{temperatura di regime alla quale il motore si dovrebbe stabilizzare.}$$

  Questo profilo però, viene assunto in un quantitativo di tempo molto maggiore di quello impiegato nei test.

- di conseguenza l'unica parte da noi visibile e computabile è quella di ascesa iniziale che può essere interpretata come una retta.
  Ovvero "linearizzando" possiamo associare ad ogni test una retta e confrontare queste.
  In questo modo dovremmo creare 2 gruppi ben distinti, uno per il raffreddamento acceso con rette a pendenza minore e uno a raffreddamento spento con rette più ripide.

2.1.1 Buti 24/11/2024

I test sono stati svolti nel parcheggio del "PENNY" di Cascine di Buti in data 24/11/2024. La temperatura media esterna era $T_{amb} \sim 9°\text{C}$.

In questa giornata di test doveva essere quello di studiare il riscaldamento dei motori a raffreddamento acceso e spento per capire l'effettiva necessità di quest'ultimo.
Purtroppo per problemi tecnici l'auto non ha potuto viaggiare a raffreddamento spento per un tempo prolungato e questo ha limitato l'efficacia del test.

In questo test abbiamo preso come riferimento il motore destro (il sinistro rimane costantemente circa 2/3 gradi sotto) riportando:

- $T_{max} \sim 39.9°\text{C}$
- $\Delta T_{alminuto,Cooling:ON} = 2,24°\text{C}/min$
- $\Delta T_{alminuto,Cooling:OFF} = 4.65°\text{C}/min$
- $m_{Motortr,\ Cooling:ON} \sim 0,051$
- $m_{Motortr,\ Cooling:OFF} \sim 0,068$

Possiamo quindi concludere che il riscaldamento del motore avviene più veloce del 33% a raffreddamento spento rispetto che a raffreddamento acceso.

2.1.2 Buti 20/11/2024

I test sono stati svolti nel parcheggio del "PENNY" di Cascine di Buti in data 20/11/2024. La temperatura media esterna era $T_{amb} \sim 11°\text{C}$.

In questa giornata di test l'obiettivo è stato quello di studiare il comportamento della macchina andando a cambiare diverse mappe motore.
In alcuni casi il raffreddamento è stato spento e possiamo riportare le seguenti considerazioni:

In questo test abbiamo preso come riferimento il motore destro (il sinistro rimane costantemente circa 2/3 gradi sotto) che ha riportato:

- $T_{max} \sim 44.6°\text{C}$
- $\Delta T_{media,Cooling:ON} = 3,01°\text{C}/min$
- $\Delta T_{media,Cooling:OFF} = 4.01°\text{C}/min$
- $m_{Motortr,\ Cooling:ON} \sim 0,037$
- $m_{Motortr,\ Cooling:OFF} \sim 0,077$

Possiamo quindi concludere che il riscaldamento del motore avviene il doppio più veloce a raffreddamento spento rispetto che a raffreddamento acceso.

2.1.3 Dati Competizioni 2023/2024

I test/gare sono stati svolte in condizioni esterne variabili nell'arco dei mesi di Luglio, Agosto e Settembre, quindi possiamo ipotizzare una temperatura esterna media di circa 30°C.

Notiamo che il test con la migliore resa è quello dell'Endurance di Varano con alla guida il pilota Lecci dove la macchina ha avuto modo di riprodurre il grafico completo in un arco temprale di circa $1025s$ raggiungendo la temperatura di regime di 51°C.
In questo test come considerato nel relativo file il motore ha descritto un grafico riconducibile a:

$$T(t) = T_{amb} + (T_{max} - T_{amb})e^{\frac{k}{t}},$$

con k coefficiente di "dispersione" termica.

IDEA: 51°C di regime con 30°C fuori vuol dire un ΔT di 20°C che se fatto a Buti porta la temperatura di regime a circa $20 + 10$°C gradi.
Consideriamo quindi il caso di Cooling:OFF il ΔT sarà incrementato del 75% e utilizzando

$$T_{regime} = T_{amb} + \Delta T_{Cooling:OFF}$$

la temperatura di regime passerebbe da 51°C a $30 + 20 * (1 + 0.75)$°C = 65°C.

Un altro dato utile che possiamo estrarre dalle telemetrie delle gare è la temperatura massima $T_{max} = 54.9$°C. Ottenuta a Buti dal pilota Battiostini.

Stima al volo: considerando $T_{max} = T_{amb} + \Delta T$ si dipanano 4 casi:

- Il test è stato svolto in inverno ($T_{amb} \sim 10$°C):
  - Cooling ON: MESSI MALE, se cosi fosse il ΔT sarebbe di circa 45°C quindi in estate sarebbe possibile toccare i $30 + 45$°C.
    Questa temperatura non è critica ma è comunque considerevole.
  - Cooling OFF: Situazione migliore che a raffreddamento acceso ma comunque da controllare in quanto se quel valore si ripresentasse il estate con gli aumenti le variazioni di temperatura visti a raffreddamento spento, si potrebbe raggiungere valori di temperatura critici.
- Il test è stato svolto in estate ($T_{amb} \sim 30$°C):
  - Cooling ON: I valori sono coerenti con quanto già visto e sono minori di quanto registrato nel Endurandce di Varano.
  - Cooling OFF: I valori sono coerenti con quanto già visto e sono minori di quanto registrato nel Endurandce di Varano.

Chapter 3

Capitolo 3

3.1 Miglioramenti per il prossimo anno

3.1.1 consigli che sono venuti in mente dopo

Fuggire!!

3.1.2 errore di base da non ripetere

ChatGPT non capisce un cazzo!