E-TEAM SQUADRA CORSE

Settore

POWERTRAIN

Fausto Antonelli 2024/25

Contents

Chapter 1

Differenziale elettronico 2023/24

1.1 Relazione tra coppia e corrente del motore

Quella che segue è una sintesi delle attività svolte da altri settori negli anni precedenti, con l'aggiunta di alcune considerazioni basate sugli sviluppi più recenti del veicolo. La relazione tra la coppia motrice T (espressa in Nm) e la corrente I (in A) erogata al motore elettrico è data dalla formula:

$$T = k \cdot I, x$$

dove k rappresenta la costante di coppia del motore. Sebbene il valore esatto di k non sia noto con precisione, a causa di discrepanze tra datasheet e misure sperimentali, è possibile assumere come valore di riferimento iniziale:

$$k = 0.479 Nm/A.$$

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

$$\Delta I = k \cdot 3.6 \cdot r \cdot \frac{\omega_L - \omega_R}{v} \qquad k \sim \frac{5}{8}. \tag{1.2}$$

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. Tuttavia, a mio avviso, la "legge di spartizione" non risulta sufficientemente accurata poiché non considera eventi di perdita di aderenza o bloccaggio parziale delle ruote anteriori.
Inoltre, la formula presuppone che le ruote anteriori copino perfettamente il terreno, ipotesi spesso non valida a causa di asperità, disconnessioni o variazioni di grip. Quindi, l'idea rimane concettualmente valida ma le condizioni dei tracciati che ci si presentano non la rendono affidabile e tanto meno performante.

Differenziale 2: EVO

Ingressi

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

Legge per la spartizione della corrente

$$\Delta I = k_{EVO} \cdot (\omega_R^2 - \omega_L^2) \quad k \sim 0.007. \tag{1.4}$$

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

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

Commento: Anche in questo caso, l'impostazione teorica e il principio di funzionamento risultano corretti, in linea con le logiche utilizzate nei differenziali elettronici attualmente in fase di sviluppo. Ma sono presenti alcune criticità pratiche. In particolare, la costante di regolazione k scelta appare eccessivamente elevata. Inoltre, la formula non tiene conto della dinamica dell'aderenza né del trasferimento di carico laterale, fenomeni che incidono sulla capacità di trazione di ciascuna ruota. Infine, la saturazione di sicurezza impostata (± 1.5) rappresenta un'interessante protezione contro tali effetti, ma si tratta più di una misura correttiva che di una soluzione alla radice del problema.

Chapter 2

Differenziale elettronico 2024/25

La progettazione di questo differenziale elettronico, e di conseguenza la sua struttura finale, è fortemente condizionata dalla disponibilità e dall'affidabilità degli input forniti dai sensori installati sul veicolo.
Nella sua configurazione definitiva, il differenziale si basa, oltre che sui dati fisici noti del veicolo, sui segnali provenienti dai quattro potenziometri collegati al sistema sospensivo e sui valori di velocità angolare delle singole ruote. L'idea principale alla base del funzionamento di questo sistema è quella di analizzare quanto risultano caricati o scaricati gli ammortizzatori nei vari momenti della guida, estrarne l'angolo di sterzo e di ripartire la coppia di conseguenza. Inoltre il differenziale svolge altre due attività secondarie ma fondamentali per la stabilità e la trazione del veicolo:

- **SlipControll** : confronta 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 taglia in maniera proporzionale la corrente al motore designato.
- **RearBias** : legge i dati dei 4 ammortizzatori e capisce l'assetto assunto dal veicolo ad ogni istante. Questa funzione limita la coppia massima disponibile a seconda se l'asse posteriore è più o meno caricato.

2.1 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:

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

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, ad esempio che il veicolo stia affrontando una curva verso sinistra, lo schema delle forze a cui esso è soggetto può essere rappresentato come segue:

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

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

Sapendo che $steeringAngle = \frac{l_f}{R}$ con R angolo di sterzo. Otteniamo:

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

Questa espressione consente di stimare in modo continuo la traiettoria seguita dal veicolo, fornendo un'informazione fondamentale per la gestione della coppia sui due motori posteriori.

Il codice MATLAB riportato di seguito rappresenta una prima implementazione funzionale del modello proposto. Esso calcola la distribuzione della coppia sui due motori posteriori in base ai segnali dei potenziometri delle sospensioni e alle velocità delle singole ruote.

```
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
```

Chapter 3

Differenziale con sensori avanzati

Supponendo di avere a disposizione una gamma più ampia di input, è possibile implementare una versione avanzata del differenziale elettronico capace di reagire in modo più preciso e dinamico alle condizioni reali di guida. Il modello che segue introduce sensori aggiuntivi, come la IMU, sensori di frenata, angolo di sterzo, e i classici sensori di velocità ruota. Questa configurazione permette di stimare con maggiore accuratezza la distribuzione di coppia.

```
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 3.1: 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 vedere come si sarebbe comportato il differenziale.

```
    %% 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

    steeringAngles(i) = atan(vp.l / (radiusCurvature(i) + 0.5 * vp.t2 * tan(
        steeringAngles(i)))); % Angolo di sterzata
    steeringAngles(i) = steeringAngles(i) * sign(curvature(i));

    % steeringAngles(i) = atan(vp.l / radiusCurvature(i)) * sign(curvature(i));


    % Calcolo della velocit  massima consentita dalla curvatura
    V_lim = min(v_max, sqrt(mu * g * radiusCurvature(i))); % Velocit  massima
        consentita
    V_lim = min(V_lim, sqrt(maxTorque / (massVehicle * wheelRadius))); % Limite
        basato sulla potenza del motore
    vehicleSpeed(i) = vehicleSpeed(i-1) + (V_lim - vehicleSpeed(i-1)) * (1 - exp(-dt/
        tau)); % Velocit  del veicolo


    % Calcolo della forza frenante
    brakeForce(i) = abs(sin(steeringAngles(i))); % Forza frenante

    % Calcolo della velocit  angolare del veicolo
    omega_vehicle = vehicleSpeed(i) / radiusCurvature(i);

    % Velocit  delle ruote posteriori
    wheelSpeedLeft(i)  = (vehicleSpeed(i) - 0.5 * omega_vehicle * vp.t2) / (2 * pi *
        wheelRadius) * 60;
    wheelSpeedRight(i) = (vehicleSpeed(i) + 0.5 * omega_vehicle * vp.t2) / (2 * pi *
        wheelRadius) * 60;

    % Velocit  delle ruote anteriori (considerando lo slip)
    wheelSpeedLeftFront(i)  = wheelSpeedLeft(i)  * (1 - slip_ratio);
    wheelSpeedRightFront(i) = wheelSpeedRight(i) * (1 - slip_ratio);

    % Calcolo dell'accelerazione longitudinale e laterale
    accelLong = (vehicleSpeed(i) - vehicleSpeed(i-1)) / dt; % Accelerazione
        longitudinale
    accelLong = max(min(accelLong, mu * g), -mu * g); % Limita l'accelerazione
        longitudinale

    accelLat = min((vehicleSpeed(i)^2 / radiusCurvature(i)) * sign(steeringAngles(i))
        , mu * g); % Accelerazione laterale
    accelLat = max(min(accelLat, mu * g), -mu * g); % Limita l'accelerazione laterale

    accelerometerXYZ(i, :) = [accelLong, accelLat, 0]; % Accelerometro
end

%% Calcolo distribuzione coppia
for i = 1:num_points
    [torqueLeft(i), torqueRight(i)] = controlTorqueDistributionNessy3(...
        wheelSpeedLeft(i), wheelSpeedRight(i), wheelSpeedLeftFront(i),
            wheelSpeedRightFront(i), ...
        steeringAngles(i), accelerometerXYZ(i, :), brakeForce(i));

    % Controllo per evitare coppie nulle o non valide
    if torqueLeft(i) == 0 || torqueRight(i) == 0
        fprintf('Zero torque at i=%d: Curv=%f, Steer=%f, SpeedL=%f, SpeedR=%f\n', ...
                i, curvature(i), steeringAngles(i), wheelSpeedLeft(i),
                    wheelSpeedRight(i));
```

```
    end
end

%% Visualizzazione
figure('Name', 'Analisi del veicolo', 'NumberTitle', 'off', 'Position', [100 100 1200 800]);

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;
```

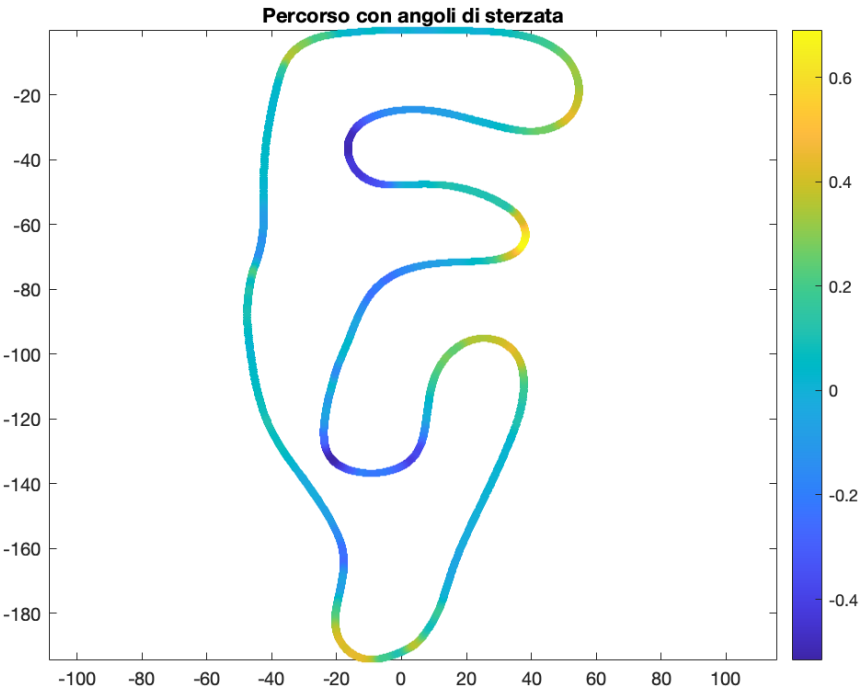
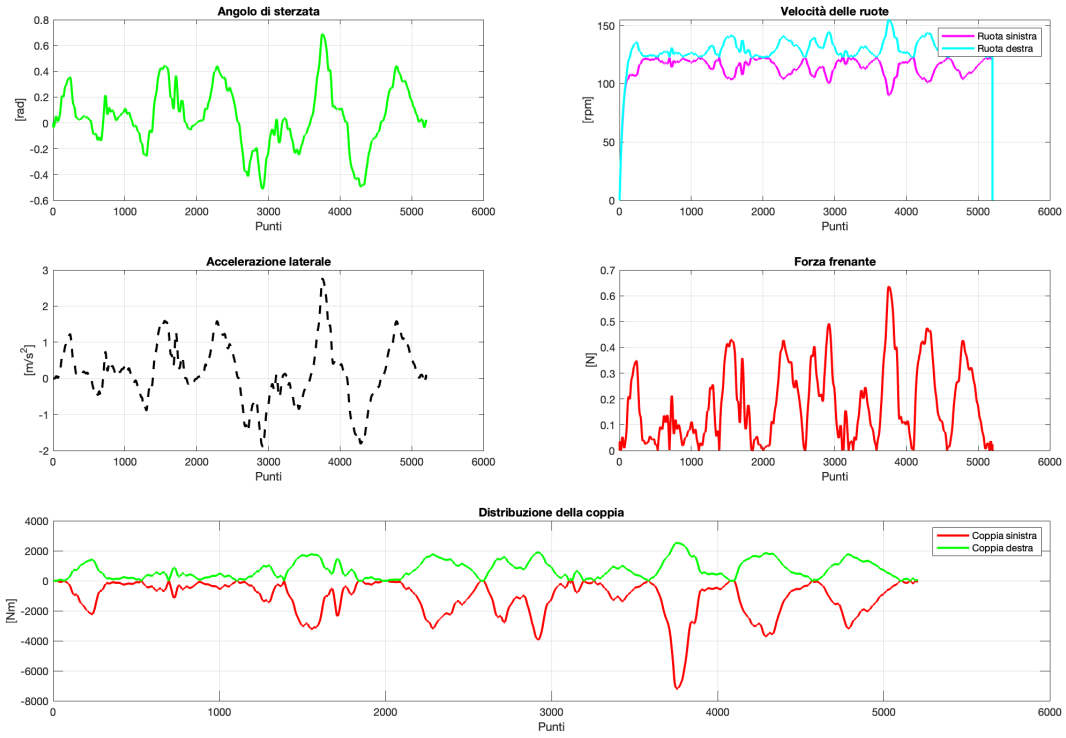


Figure 3.1: Tracciato di Hockenheim, scalato cromaticamente rispetto agli angoli di curvatura tenuti nel test



Chapter 4

Analisi delle telemetrie

In questa sezione saranno analizzati i dati provenienti dalle telemetrie raccolte in diverse sessioni di test e di gara, con l'obiettivo di valutare il comportamento termico dei motori in differenti condizioni operative.
Considerazioni iniziali:

- In condizioni ideali, l'andamento della temperatura del motore nel tempo dovrebbe seguire una curva di tipo esponenziale crescente, descritta dall'equazione:

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

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

- Di conseguenza, l'unica parte effettivamente visibile e analizzabile durante le prove è la fase iniziale di riscaldamento, che può essere approssimata in modo lineare. Linearizzando la curva di temperatura nel tempo, è possibile associare a ogni test una retta di crescita termica, confrontando le diverse pendenze ottenute.

- In questo modo dovremmo creare 2 gruppi ben distinti, uno per il raffreddamento acceso (Cooling: ON) con rette a pendenza minore e uno a raffreddamento spento (Cooling: OFF) con rette più ripide.

4.0.1 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 $T_{amb} \sim 9°\text{C}$.

L'obiettivo della giornata era studiare il riscaldamento dei motori con raffreddamento acceso e spento, per valutare l'effettiva necessità del sistema di cooling in fase di utilizzo continuativo. A causa di problemi tecnici, l'auto non ha potuto marciare per un tempo sufficiente con il raffreddamento disattivato, limitando parzialmente 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.

4.0.2 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 $T_{amb} \sim 11°C$.

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:

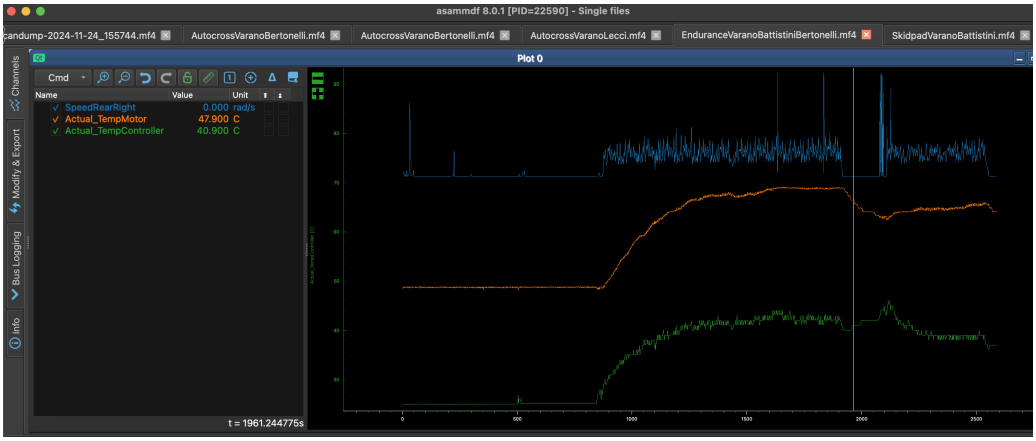
- $T_{max} \sim 44.6°C$
- $\Delta T_{media,Cooling:ON} = 3,01°C/min$
- $\Delta T_{media,Cooling:OFF} = 4.01°C/min$
- $m_{Motortr,\ Cooling:ON} \sim 0,037$
- $m_{Motortr,\ Cooling:OFF} \sim 0,077$

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

4.0.3 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 assumere che la temperatura esterna media si attesta a circa 30°C.

L'analisi dei dati ha evidenziato che la miglior prestazione termica è stata registrata durante la prova Endurance di Varano, con alla guida il pilota Lecci. In questa occasione, il veicolo ha mantenuto un funzionamento stabile per circa $1025s$, raggiungendo una temperatura di regime di circa 51°C.



Il comportamento del motore durante la prova può essere descritto dalla seguente equazione di crescita termica:

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

con k coefficiente di "dispersione" termica, che rappresenta la rapidità con cui i motori tendono verso la temperatura di regime.

Nel test di Varano, con temperatura ambiente $T_{amb} = 30°C$ e temperatura di regime $T_{regime} = 51°C$ si ottiene un ΔT di 20°C.
Traslando lo stesso comportamento alle condizioni di Buti (temperatura ambiente di circa 10°C), si otterrebbe:

$$T_{regime,Buti} = 20°C + 10°C$$

consideriamo quindi il caso di Cooling:OFF e ipotizzando un incremento di ΔT del 75% e utilizzando

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

la temperatura di regime passerebbe da 51°C a

$$T_{regime,CollingOFF} = 30°\text{C} + 20°\text{C} \cdot (1 + 0.75) = 65°\text{C}$$

Questo significa che, senza raffreddamento, la temperatura di regime stimata si innalzerebbe di circa 14°C rispetto alla configurazione standard.
Un altro dato utile che possiamo estrarre dalle telemetrie delle gare è la temperatura massima $T_{max} = 54.9°\text{C}$. Ottenuta a Buti dal pilota Battiostini.