MATLAB scripts PDF

Gear_adjustment.m

```
%% --- Viper Gearing Efficiency Analysis (Powerband-Based, Auto Gear Ratio
Suggestion) ---
% Detects braking zones, checks RPM vs powerband, and suggests revised gear
ratios.
clear; clc; close all;

%% --- LOAD DATA ---
[file, path] = uigetfile('*.mat', 'Select your MoTeC .mat log file');
if isequal(file,0)
    error('No file selected.');
end
load(fullfile(path, file));
fprintf('✅ Loaded %s\n', file);

%% --- MAP SIGNALS (adjust if variable names differ) ---
t         = Running_Lap_Time.Time(:);
speed     = Ground_Speed.Value(:);    % [kph]
rpm       = Engine_RPM.Value(:);
throttle = Throttle_Pos.Value(:);
latG      = G_Force_Lat.Value(:);
longG     = G_Force_Long.Value(:);
damperFL = Damper_Pos_FL.Value(:);
damperFR = Damper_Pos_FR.Value(:);
damperRL = Damper_Pos_RL.Value(:);
damperRR = Damper_Pos_RR.Value(:);
tyreFL_in  = Tyre_Temp_FL_Inner.Value(:);
tyreFL_mid = Tyre_Temp_FL_Centre.Value(:);
tyreFL_out = Tyre_Temp_FL_Outer.Value(:);
tyreFR_in  = Tyre_Temp_FR_Inner.Value(:);
tyreFR_mid = Tyre_Temp_FR_Centre.Value(:);
tyreFR_out = Tyre_Temp_FR_Outer.Value(:);

%% --- PARAMETERS ---
redline = 6500;              % rpm
moderateThresh = 5150;       % moderate gearing inefficiency
severeThresh = 4800;         % severe gearing inefficiency
brakeG_threshold = -0.8;     % g threshold for braking
brake_min_separation = 0.5; % seconds (debounce)
cluster_time_s = 2.0;        % cluster braking detections within 2s
preBrakeOffset_s = 0.05;     % sample RPM/speed 50 ms before braking

gear_ratios = [8.509, 5.496, 4.395, 3.707, 3.273, 2.865];  % nominal
gear_top_kph = [100, 154, 190, 223, 251, 290];             % top speeds
[kph]

%% --- BRAKING ZONE DETECTION (with 2s clustering filter) ---
isBraking = longG <= brakeG_threshold;
fs_est = 1/median(diff(t));
min_sep_samples = max(1, round(brake_min_separation * fs_est));
edges = find(diff([0; isBraking]) == 1);

cluster_samples = round(cluster_time_s * fs_est);
filtered_edges = [];
lastEdge = -inf;
for i = 1:length(edges)
```

```matlab
    if edges(i) - lastEdge > cluster_samples
        filtered_edges(end+1) = edges(i); %#ok<AGROW>
        lastEdge = edges(i);
    end
end
brake_start_idx = filtered_edges;

%% --- SAMPLE PRE-BRAKE SPEEDS & RPMs ---
preBrakeSpeeds = [];
preBrakeRPMs = [];
preBrakeTimes = [];
for i = 1:length(brake_start_idx)
    idx = brake_start_idx(i);
    offset_samples = round(preBrakeOffset_s * fs_est);
    sample_idx = max(1, idx - offset_samples);
    preBrakeSpeeds(end+1) = speed(sample_idx); %#ok<AGROW>
    preBrakeRPMs(end+1) = rpm(sample_idx); %#ok<AGROW>
    preBrakeTimes(end+1) = t(sample_idx); %#ok<AGROW>
end
nEvents = numel(preBrakeSpeeds);

%% --- ASSIGN GEARS BASED ON SPEED RANGE ---
assignedGear = zeros(nEvents,1);
for i=1:nEvents
    v = preBrakeSpeeds(i);
    gidx = find(v <= gear_top_kph, 1, 'first');
    if isempty(gidx), gidx = numel(gear_top_kph); end
    assignedGear(i) = gidx;
end

%% --- COMPUTE SUGGESTED RATIOS (DATA-BASED) ---
suggested_ratios = gear_ratios;
gear_events = cell(6,1);
for g=1:6
    idxs = find(assignedGear == g);
    gear_events{g} = preBrakeSpeeds(idxs);
    if ~isempty(idxs)
        V_target = median(preBrakeSpeeds(idxs)); % typical pre-brake speed
        V_current_top = gear_top_kph(g);
        suggested_ratios(g) = gear_ratios(g) * (V_current_top / V_target);
    end
end

%% --- POWERBAND-BASED INEFFICIENCY DETECTION ---
moderateIdx = find(rpm < moderateThresh & throttle > 50 & speed > 20);
severeIdx   = find(rpm < severeThresh & throttle > 50 & speed > 20);

%% --- OUTPUT SUMMARY ---
fprintf('\n--- GEARING EFFICIENCY ANALYSIS ---\n');
fprintf('Detected %d braking events after clustering (threshold %.2f g)\n',
nEvents, brakeG_threshold);

% Print a few pre-brake samples
for i=1:min(10,nEvents)
    fprintf(' - t=%.2fs: %.1f kph, %.0f rpm, gear=%d\n', preBrakeTimes(i),
preBrakeSpeeds(i), preBrakeRPMs(i), assignedGear(i));
end

% Flag RPM inefficiency
```

```matlab
fprintf('\nPowerband inefficiency (throttle>50%%, speed>20kph):\n');
fprintf('⚠ Moderate (<%.0f rpm): %d samples\n', moderateThresh,
numel(moderateIdx));
fprintf('🚨 Severe (<%.0f rpm): %d samples\n', severeThresh,
numel(severeIdx));

% Gear ratio suggestions
fprintf('\nSuggested Gear Ratios (data-based):\n');
fprintf('Gear | Old Ratio | TopSpeed(kph) | Median PreBrake(kph) |
Suggested Ratio\n');
for g=1:6
    medV = median(gear_events{g});
    if isempty(medV), medV = NaN; end
    fprintf('%4d | %9.4f | %12.1f | %20.1f | %14.4f\n', g, gear_ratios(g),
gear_top_kph(g), medV, suggested_ratios(g));
end

fprintf('\n--- Analysis complete. ---\n');

%% --- PLOTS ---

figure('Name','Speed & RPM Overview','NumberTitle','off');
subplot(3,1,1);
plot(t, speed, 'b'); hold on;
scatter(preBrakeTimes, preBrakeSpeeds, 30, 'r', 'filled');
xlabel('Time [s]'); ylabel('Speed [kph]'); grid on;
title('Speed with Braking Entry Markers');

subplot(3,1,2);
plot(t, rpm, 'k'); hold on; grid on;
yline(moderateThresh,'y--','Moderate Limit');
yline(severeThresh,'r--','Severe Limit');
xlabel('Time [s]'); ylabel('Engine RPM');
title('RPM with Powerband Limits');

subplot(3,1,3);
plot(t, throttle, 'g'); hold on;
plot(t, latG, 'b'); plot(t, longG, 'r');
legend('Throttle','Lat G','Long G');
xlabel('Time [s]'); ylabel('Value');
grid on; title('Throttle & G Forces');

figure('Name','Gear Ratios: Current vs Suggested','NumberTitle','off');
barData = [gear_ratios; suggested_ratios]';
bar(barData); grid on;
xticks(1:6); xticklabels({'1','2','3','4','5','6'});
xlabel('Gear'); ylabel('Gear Ratio');
legend('Current','Suggested');
title('Current vs Suggested Gear Ratios');

%% --- SAVE PLOTS ---
outdir = fullfile(pwd,'plots','GearingAnalysis');
if ~exist(outdir,'dir'), mkdir(outdir,'s'); end
figs = findall(0,'Type','figure');
for k=1:length(figs)
    f = figs(k);
    fname = sprintf('Fig_%s.png', strrep(f.Name,' ','_'));
    saveas(f, fullfile(outdir, fname));
end
```

```
fprintf('Saved plots to %s\n', outdir);
```

revised_gearing_improvements

```
%% --- Compare Two MoTeC Logs: Long G & Speed with Color-Coded Zones ---
clear; clc; close all;

%% --- LOAD FILES ---
disp('Select baseline (before gear change) log:');
[file1, path1] = uigetfile('*.mat');
if isequal(file1,0), error('No file selected.'); end
load(fullfile(path1, file1));
log1.longG = G_Force_Long.Value(:);
log1.speed = Ground_Speed.Value(:);
log1.dist  = Lap_Distance.Value(:);

disp('Select new (after gear change) log:');
[file2, path2] = uigetfile('*.mat');
if isequal(file2,0), error('No file selected.'); end
load(fullfile(path2, file2));
log2.longG = G_Force_Long.Value(:);
log2.speed = Ground_Speed.Value(:);
log2.dist  = Lap_Distance.Value(:);

%% --- DEFINE ANALYSIS ZONES (in meters) ---
zones = [
    584 1282;
    1488 1576;
    1690 2100;
    2264 2352;
    2455 2532;
    2636 2811;
    2936 3018;   % ✅ corrected zone 7
    3116 3520;
    3804 4057;
    4186 4269;
    4336 4806;
    4956 5396
];
nZones = size(zones,1);

%% --- INTERPOLATE BOTH FILES TO COMMON DISTANCE VECTOR ---
maxDist = min(max(log1.dist), max(log2.dist));
dist_common = linspace(0, maxDist, 20000);

longG1 = interp1(log1.dist, log1.longG, dist_common, 'linear', 'extrap');
longG2 = interp1(log2.dist, log2.longG, dist_common, 'linear', 'extrap');
speed1 = interp1(log1.dist, log1.speed, dist_common, 'linear', 'extrap');
speed2 = interp1(log2.dist, log2.speed, dist_common, 'linear', 'extrap');

%% --- COMPUTE ZONE DELTAS ---
zoneDelta = zeros(nZones,2); % [ΔLongG, ΔSpeed]
for i = 1:nZones
    d1 = zones(i,1); d2 = zones(i,2);
    idx = dist_common >= d1 & dist_common <= d2;
    if sum(idx) < 10, continue; end
    zoneDelta(i,1) = mean(longG2(idx)) - mean(longG1(idx));
    zoneDelta(i,2) = mean(speed2(idx)) - mean(speed1(idx));
end
```

```
%% --- CREATE COLOR FUNCTION ---
getColor = @(delta) (delta >= 0) * [0.2 0.8 0.2] + (delta < 0) * [0.9 0.3
0.3];

%% --- CREATE PLOTS ---
figure('Name','Gear Ratio Comparison (Color-Coded
Zones)','NumberTitle','off','Position',[100 100 1200 800]);

% --- SPEED OVERLAY ---
subplot(2,1,1);
hold on; grid on;
for i = 1:nZones
    d1 = zones(i,1); d2 = zones(i,2);
    c = getColor(zoneDelta(i,2)); % green/red by Δspeed
    xpatch = [d1, d2, d2, d1];
    ypatch = [0, 0, max([speed1 speed2]), max([speed1 speed2])];
    patch(xpatch, ypatch, c, 'FaceAlpha', 0.25, 'EdgeColor', 'none');
end
plot(dist_common, speed1, 'r', 'LineWidth', 1.2);
plot(dist_common, speed2, 'b', 'LineWidth', 1.2);
xlabel('Distance [m]');
ylabel('Corrected Speed [km/h]');
legend('Before Gearing', 'After Gearing', 'Location', 'best');
title('Speed Comparison (Green = Gain, Red = Loss)');

% --- LONG G OVERLAY ---
subplot(2,1,2);
hold on; grid on;
for i = 1:nZones
    d1 = zones(i,1); d2 = zones(i,2);
    c = getColor(zoneDelta(i,2)); % same color as speed improvement
    xpatch = [d1, d2, d2, d1];
    ypatch = [min([longG1 longG2]), min([longG1 longG2]), max([longG1
longG2]), max([longG1 longG2])];
    patch(xpatch, ypatch, c, 'FaceAlpha', 0.25, 'EdgeColor', 'none');
end
plot(dist_common, longG1, 'r', 'LineWidth', 1.2);
plot(dist_common, longG2, 'b', 'LineWidth', 1.2);
xlabel('Distance [m]');
ylabel('Longitudinal G [g]');
legend('Before Gearing', 'After Gearing', 'Location', 'best');
title('Longitudinal G Comparison (Green = Gain, Red = Loss)');

%% --- PRINT DELTAS ---
fprintf('\n--- Mean Deltas per Zone ---\n');
fprintf('%5s | %12s | %12s\n', 'Zone', 'Δ LongG (mean)', 'Δ Speed (mean)');
fprintf('----------------------------------------\n');
for i = 1:nZones
    if zoneDelta(i,2) == 0, continue; end
    fprintf('%5d | %+12.4f | %+12.3f\n', i, zoneDelta(i,1),
zoneDelta(i,2));
end
fprintf('----------------------------------------\n');
fprintf('⍰ Positive = improved accel/speed | ⍰ Negative = performance
loss\n');
```

revised_diff.m

```matlab
%% --- Compare Apex Speeds Between Two MoTeC Logs ---
clear; clc; close all;

%% --- LOAD FIRST LOG ---
[file1, path1] = uigetfile('*.mat', 'Select FIRST MoTeC log (.mat)');
if isequal(file1,0)
    error('No file selected.');
end
load(fullfile(path1, file1));
fprintf('✅ Loaded first log: %s\n', file1);

% Extract key channels
lapDist1 = Lap_Distance.Value(:);
speed1   = Ground_Speed.Value(:);
longG1   = G_Force_Long.Value(:);

%% --- LOAD SECOND LOG ---
[file2, path2] = uigetfile('*.mat', 'Select SECOND MoTeC log (.mat)');
if isequal(file2,0)
    error('No file selected.');
end
load(fullfile(path2, file2));
fprintf('✅ Loaded second log: %s\n', file2);

lapDist2 = Lap_Distance.Value(:);
speed2   = Ground_Speed.Value(:);
longG2   = G_Force_Long.Value(:);

%% --- FIND APEX POINTS ---
% Apex ≈ where longitudinal G crosses zero (transition from braking to
accel)
gSign1 = sign(longG1);
apexIdx1 = find(diff(gSign1) > 0);  % braking (neg G) → accel (pos G)
apexSpeeds1 = speed1(apexIdx1);

gSign2 = sign(longG2);
apexIdx2 = find(diff(gSign2) > 0);
apexSpeeds2 = speed2(apexIdx2);

% Match by position (trim to same number of corners)
numCorners = min(length(apexSpeeds1), length(apexSpeeds2));
apexSpeeds1 = apexSpeeds1(1:numCorners);
apexSpeeds2 = apexSpeeds2(1:numCorners);

%% --- CALCULATE DELTAS ---
deltaApex = apexSpeeds2 - apexSpeeds1;  % + means second log faster at apex

%% --- DISPLAY RESULTS ---
fprintf('\n--- Apex Speed Comparison ---\n');
fprintf('Corner | Apex1 [km/h] | Apex2 [km/h] | Δ Speed (2-1)\n');
fprintf('-----------------------------------------------\n');
for i = 1:numCorners
    fprintf(' %2d     |    %7.2f     |    %7.2f     |   %+6.2f\n', ...
        i, apexSpeeds1(i), apexSpeeds2(i), deltaApex(i));
end
fprintf('-----------------------------------------------\n');
fprintf('Mean Δ Apex Speed: %+6.2f km/h\n', mean(deltaApex));
```

```
%% --- PLOTS ---
figure('Name','Apex Speed Comparison','NumberTitle','off','Position',[100
100 900 600]);
hold on; grid on;
bar(1:numCorners, deltaApex, 'FaceColor', [0.2 0.6 1]);
yline(0, 'k--');
xlabel('Corner #');
ylabel('Δ Apex Speed [km/h]');
title('Apex Speed Delta (Second Run – First Run)');
for i = 1:numCorners
    if deltaApex(i) < 0
        bar(i, deltaApex(i), 'FaceColor', [1 0.3 0.3]); % red if slower
    else
        bar(i, deltaApex(i), 'FaceColor', [0.3 0.9 0.3]); % green if faster
    end
end
legend('Δ Apex Speed', 'Location', 'best');
hold off;

fprintf('\n✅ Comparison complete.\n');
```

apex_speed.m

```
%% --- Apex Speed Comparison (Dual Bar Plot) ---
clear; clc; close all;

%% --- USER INPUT (for labeling only) ---
baselineName = input('Enter BASELINE run name : ', 's');
improvedName = input('Enter IMPROVED run name : ', 's');

%% --- DATA (km/h) ---
% Improved vs Baseline apex speeds
improved = [125.6, 100.8, 116.8, 107.4, 116.5,  87.4, 101.0, 101.1, 107.2,
94.4, 118.7, 113.1, 144.7];
baseline = [115.8,  96.5, 112.1, 104.7, 108.2,  97.6, 101.2,  98.6, 111.0,
96.0, 129.8, 102.5, 128.2];

apexCount = numel(improved);
delta = improved - baseline;

%% --- PRINT TABLE ---
fprintf('\n--- Apex Speed Comparison (%s vs %s) ---\n', improvedName,
baselineName);
fprintf('Apex |  %s [km/h]  |  %s [km/h]  |  Δ (Imp - Base)\n',
improvedName, baselineName);
fprintf('-------------------------------------------------\n');
for i = 1:apexCount
    fprintf(' %2d   |     %6.1f     |     %6.1f     |    %+6.2f\n', ...
        i, improved(i), baseline(i), delta(i));
end
fprintf('-------------------------------------------------\n');
fprintf('Mean Δ Apex Speed: %+6.2f km/h\n', mean(delta));

%% --- PLOT ---
figure('Name','Apex Speed Comparison','NumberTitle','off','Position',[100
100 1000 600]);
hold on; grid on;

barData = [baseline(:), improved(:)];
```

```
b = bar(1:apexCount, barData, 'grouped');
b(1).FaceColor = [0.85, 0.33, 0.10];  % orange for baseline
b(2).FaceColor = [0.20, 0.80, 0.20];  % green for improved

xlabel('Apex #');
ylabel('Apex Speed [km/h]');
title(sprintf('Apex Speeds: %s vs %s', improvedName, baselineName));
xticks(1:apexCount);
legend({baselineName, improvedName}, 'Location', 'northoutside',
'Orientation', 'horizontal');

% Add delta text annotations
for i = 1:apexCount
    yMax = max(barData(i,:));
    deltaStr = sprintf('%+.1f', delta(i));
    text(i, yMax + 1.5, deltaStr, 'HorizontalAlignment','center', ...
        'FontWeight','bold', 'Color', delta(i)>=0*[0 0.5 0]-[0.7 0 0]);
end

ylim([min([baseline, improved]) - 5, max([baseline, improved]) + 10]);
hold off;

fprintf('\n✅ Dual-bar apex speed comparison complete.\n');
```

top_speed.m

```
%% --- Top-End Speed Comparison (10 zones, manual input) ---
clear; clc; close all;

%% --- USER INPUT ---
baselineName = input('Enter BASELINE run name : ', 's');
improvedName = input('Enter IMPROVED run name : ', 's');

fprintf('\nEnter the 10 TOP-END speeds [km/h] for each run.\n');

fprintf('\n--- %s ---\n', baselineName);
baseline = zeros(1,10);
for i = 1:10
    baseline(i) = input(sprintf('  Top speed %2d: ', i));
end

fprintf('\n--- %s ---\n', improvedName);
improved = zeros(1,10);
for i = 1:10
    improved(i) = input(sprintf('  Top speed %2d: ', i));
end

%% --- CALCULATIONS ---
nZones = numel(improved);
delta = improved - baseline;
percentDelta = (delta ./ baseline) * 100;

%% --- PRINT TABLE ---
fprintf('\n--- Top-End Speed Comparison (%s vs %s) ---\n', improvedName,
baselineName);
fprintf('Zone |  %s [km/h]  |  %s [km/h]  |  Δ (km/h)  |  Δ (%%)\n',
improvedName, baselineName);
fprintf('------------------------------------------------------------\n');
for i = 1:nZones
```

```
    fprintf(' %2d   |     %6.1f     |     %6.1f     |   %+6.2f   |
%+6.2f%%\n', ...
        i, improved(i), baseline(i), delta(i), percentDelta(i));
end
fprintf('-------------------------------------------------------------\n');
fprintf('Mean Δ Top Speed: %+6.2f km/h (%+.2f%%)\n', mean(delta),
mean(percentDelta));

%% --- PLOT ---
figure('Name','Top-End Speed
Comparison','NumberTitle','off','Position',[100 100 1000 600]);
hold on; grid on;

barData = [baseline(:), improved(:)];
b = bar(1:nZones, barData, 'grouped');
b(1).FaceColor = [0.85, 0.33, 0.10];  % orange for baseline
b(2).FaceColor = [0.20, 0.80, 0.20];  % green for improved

xlabel('Zone #');
ylabel('Top-End Speed [km/h]');
title(sprintf('Top-End Speeds: %s vs %s', improvedName, baselineName));
xticks(1:nZones);
legend({baselineName, improvedName}, 'Location', 'northoutside',
'Orientation', 'horizontal');

% --- Add delta annotations ---
for i = 1:nZones
    yMax = max(barData(i,:));
    deltaStr = sprintf('%+.1f km/h (%+.1f%%)', delta(i), percentDelta(i));
    if delta(i) >= 0
        text(i, yMax + 1.5, deltaStr, 'HorizontalAlignment','center', ...
            'FontWeight','bold', 'Color',[0 0.6 0]);
    else
        text(i, yMax + 1.5, deltaStr, 'HorizontalAlignment','center', ...
            'FontWeight','bold', 'Color',[0.8 0 0]);
    end
end

ylim([min([baseline, improved]) - 5, max([baseline, improved]) + 12]);
hold off;

fprintf('\n✅ 10-zone top-end speed comparison complete.\n');
```

wheel_speed_comparison.m

```
% ============================================================
% Compare Differential Lock Tendency Between Two MoTeC Logs
% ============================================================

% Ask user for the two .mat files
[file1, path1] = uigetfile('*.mat', 'Select baseline run');
filePath1 = fullfile(path1, file1);
[file2, path2] = uigetfile('*.mat', 'Select comparison run');
filePath2 = fullfile(path2, file2);

% Load the data
data1 = load(filePath1);
data2 = load(filePath2);

% --- Adjust these field names if different in your file ---
```

```
RL1 = data1.Wheel_Speed_RL.Value;   % Rear Left wheel speed
RR1 = data1.Wheel_Speed_RR.Value;   % Rear Right wheel speed
SA1 = data1.Steering_Wheel_Angle.Value; % Steering angle (deg)

RL2 = data2.Wheel_Speed_RL.Value;
RR2 = data2.Wheel_Speed_RR.Value;
SA2 = data2.Steering_Wheel_Angle.Value;

% Define thresholds
angleThreshold = 10;   % degrees
speedTolerance = 0.3;  % same units as your data (usually km/h)

% Function to compute diff lock percentage
computeLockPct = @(RL, RR, SA) ...
    sum(abs(RR - RL) < speedTolerance & (abs(SA) > angleThreshold)) ...
    / sum(abs(SA) > angleThreshold) * 100;

% Compute for both runs
lockPct1 = computeLockPct(RL1, RR1, SA1);
lockPct2 = computeLockPct(RL2, RR2, SA2);

% Display results
fprintf('--- Differential Lock Tendency ---\n');
fprintf('Baseline Run   : %.2f%% of cornering time wheels nearly locked\n',
lockPct1);
fprintf('Comparison Run : %.2f%% of cornering time wheels nearly locked\n',
lockPct2);

if lockPct2 > lockPct1
    fprintf('→ The differential seems to lock MORE in the comparison
run.\n');
elseif lockPct2 < lockPct1
    fprintf('→ The differential seems to lock LESS in the comparison
run.\n');
else
    fprintf('→ Both runs show equal locking tendency.\n');
end
```

body_roll.m

```
%% --- Compare Ride-Height Difference Between Two Logs (Sample-by-Sample) -
--
clear; clc; close all;

%% --- Load File 1 ---
[file1, path1] = uigetfile('*.mat', 'Select FIRST MoTeC .mat file');
if isequal(file1,0), error('No first file selected.'); end
data1 = load(fullfile(path1, file1));

%% --- Load File 2 ---
[file2, path2] = uigetfile('*.mat', 'Select SECOND MoTeC .mat file');
if isequal(file2,0), error('No second file selected.'); end
data2 = load(fullfile(path2, file2));

%% --- CHANGE THESE IF YOUR VARIABLE NAMES DIFFER ---
getSignals = @(D) struct( ...
    'latG', D.G_Force_Lat.Value(:), ...
    'FL',   D.Ride_Height_FL.Value(:), ...
    'FR',   D.Ride_Height_FR.Value(:), ...
    'RL',   D.Ride_Height_RL.Value(:), ...
    'RR',   D.Ride_Height_RR.Value(:) );

S1 = getSignals(data1);
S2 = getSignals(data2);

%% --- Parameters ---
latG_threshold = 1.0;   % Only samples where |latG| > 1g

%% --- Compute sample-by-sample differences ---
function [frontDiff, rearDiff] = calcDiff(S, latG_threshold)

    idx = find(abs(S.latG) > latG_threshold);  % indices of interest

    % Absolute L - R differences
    frontDiff = abs(S.FL(idx) - S.FR(idx));
    rearDiff  = abs(S.RL(idx) - S.RR(idx));
end

[F1, R1] = calcDiff(S1, latG_threshold);
[F2, R2] = calcDiff(S2, latG_threshold);

%% --- PLOTS ---

% FRONT RIDE HEIGHT DIFF
figure('Name','Front Ride Height Difference');
hold on; grid on;
plot(F1, 'b.-', 'LineWidth', 1.2, 'MarkerSize', 12);
plot(F2, 'r.-', 'LineWidth', 1.2, 'MarkerSize', 12);
xlabel('Sample Count (only |latG| > 1)');
ylabel('|FL - FR| (mm)');
title('Front Ride Height Difference (High Lateral G Samples)');
legend(file1, file2);

% REAR RIDE HEIGHT DIFF
figure('Name','Rear Ride Height Difference');
hold on; grid on;
plot(R1, 'b.-', 'LineWidth', 1.2, 'MarkerSize', 12);
plot(R2, 'r.-', 'LineWidth', 1.2, 'MarkerSize', 12);
xlabel('Sample Count (only |latG| > 1)');
ylabel('|RL - RR| (mm)');
title('Rear Ride Height Difference (High Lateral G Samples)');
legend(file1, file2);
```

camber_test.m

```
%% Simple tyre-temperature comparison between two MoTeC .mat logs
clear; clc;

%% --- Load files ---
[file1, path1] = uigetfile('*.mat', 'Select FIRST log');
if isequal(file1,0), error('No file selected'); end
D1 = load(fullfile(path1,file1));

[file2, path2] = uigetfile('*.mat', 'Select SECOND log');
if isequal(file2,0), error('No file selected'); end
D2 = load(fullfile(path2,file2));

fprintf('Loaded:\n 1) %s\n 2) %s\n', file1, file2);

%% --- Helper to extract .Value array ---
```

```matlab
val = @(S) double(S.Value(:));

%% --- REQUIRED: you must edit these two lines to match your log names ---
lat1  = val(D1.G_Force_Lat);     long1 = val(D1.G_Force_Long);
lat2  = val(D2.G_Force_Lat);     long2 = val(D2.G_Force_Long);

%% --- High-load mask (|lat|>1 OR |long|>1) ---
mask1 = (abs(lat1)>1) | (abs(long1)>1);
mask2 = (abs(lat2)>1) | (abs(long2)>1);

%% --- Extract tyre temps for each log ---
extractTemps = @(D) struct( ...
    'FL_in',  val(D.Tyre_Temp_FL_Inner),  'FL_mid',
val(D.Tyre_Temp_FL_Centre), 'FL_out', val(D.Tyre_Temp_FL_Outer), ...
    'FR_in',  val(D.Tyre_Temp_FR_Inner),  'FR_mid',
val(D.Tyre_Temp_FR_Centre), 'FR_out', val(D.Tyre_Temp_FR_Outer), ...
    'RL_in',  val(D.Tyre_Temp_RL_Inner),  'RL_mid',
val(D.Tyre_Temp_RL_Centre), 'RL_out', val(D.Tyre_Temp_RL_Outer), ...
    'RR_in',  val(D.Tyre_Temp_RR_Inner),  'RR_mid',
val(D.Tyre_Temp_RR_Centre), 'RR_out', val(D.Tyre_Temp_RR_Outer));

T1 = extractTemps(D1);
T2 = extractTemps(D2);

%% --- Compute averages ---
computeAvg = @(T,mask) struct( ...
    'FL_in_mid',  mean(T.FL_in(mask) - T.FL_mid(mask)), ...
    'FL_out_mid', mean(T.FL_out(mask) - T.FL_mid(mask)), ...
    'FR_in_mid',  mean(T.FR_in(mask) - T.FR_mid(mask)), ...
    'FR_out_mid', mean(T.FR_out(mask) - T.FR_mid(mask)), ...
    'RL_in_mid',  mean(T.RL_in(mask) - T.RL_mid(mask)), ...
    'RL_out_mid', mean(T.RL_out(mask) - T.RL_mid(mask)), ...
    'RR_in_mid',  mean(T.RR_in(mask) - T.RR_mid(mask)), ...
    'RR_out_mid', mean(T.RR_out(mask) - T.RR_mid(mask)) );

A1_high = computeAvg(T1, mask1);
A1_low  = computeAvg(T1, ~mask1);
A2_high = computeAvg(T2, mask2);
A2_low  = computeAvg(T2, ~mask2);

%% --- Display results ---
corners = {'FL','FR','RL','RR'};
fields  = {'in\_mid','out\_mid'};

fprintf('\n=== Comparison of inner-mid and outer-mid averages ===\n');

for c = corners
    C = c{1};
    fprintf('\n--- %s ---\n', C);

    fprintf(' High load | Log1: inner-mid = %.2f, outer-mid = %.2f | Log2:
inner-mid = %.2f, outer-mid = %.2f\n', ...
        A1_high.([C '_in_mid']), A1_high.([C '_out_mid']), ...
        A2_high.([C '_in_mid']), A2_high.([C '_out_mid']));

    fprintf(' Low load  | Log1: inner-mid = %.2f, outer-mid = %.2f | Log2:
inner-mid = %.2f, outer-mid = %.2f\n', ...
        A1_low.([C '_in_mid']), A1_low.([C '_out_mid']), ...
        A2_low.([C '_in_mid']), A2_low.([C '_out_mid']));
```

```
end

fprintf('\nDone.\n');
```