CHALMERS
UNIVERSITY OF TECHNOLOGY

MMS210

Group Project

Group 15

Theresa Calisir - theresac@chalmers.se
Anton Johannes Forsler - antonfor@chalmers.se
Isuru Samuddha Lanka Ranasinghege - samuddha@student.chalmers.se
Elin Sunisara Karlsson - kael@chalmers.se
Matias Viinikainen Soares Lema - matiasl@chalmers.se
Yongren Lin - yongren@chalmers.se
Mohammed Ali El Masri - alimas@student.chalmers.se
Aditya Nalawade - nalawade@chalmers.se
Sebastian Tommy Peder Öhman - ohmanse@chalmers.se

Examiner:
Ola Benderius
2025

June 1, 2025

Contents

1 Introduction

This report and project has been conducted as part of the course MMS210 Connected Fleets in Data-Driven Engineering during study period 4, year 2025.
The project investigates how rural and mixed driving conditions affect passenger car dynamics, with a particular focus on braking and acceleration patterns, lateral forces during cornering, and vibrations caused by road surface quality. The aim is to collect and analyze real-world data to improve ride comfort and optimize suspension system performance.
Data was collected using a logging device that recorded time-stamped positional data, including distance, speed, altitude, and satellite coverage. This dataset provides valuable insights into how current vehicles perform in varying environments with road surfaces of lower quality and mixed driving characteristics.

1.1 Purpose and Objectives

The primary objective of this study is to provide Original Equipment Manufacturers (OEMs) with data-driven tools and insights that support the development of future-generation vehicles optimized for rural and mixed road conditions. This work provides OEMs with valuable insights into how rural and mixed road conditions influence key aspects of vehicle dynamics, including braking, cornering, and ride quality. By analyzing real-world data from a current passenger vehicle, the project supports the design and calibration of future systems such as adaptive damping, electronic stability control, and optimized suspension setups. These insights enable OEMs to enhance vehicle performance, reduce maintenance demands, and create products better suited for challenging environments, ultimately strengthening their competitiveness and profitability in rural and emerging markets.

2 Methodology

2.1 Route Selection

The focus of the project was on rural and mixed road conditions. To capture relevant data, Marstrand was chosen as the destination due to its accesibility and the nature of the roads leading to it. Care was taken to select a route that minimized highway use in favor of smaller roads to better reflect rural driving dynamics.

The route did not include any unpaved or particularly bumpy roads. These features could have enhanced the rural driving analysis however the route does represents one of the most rural options realistically that could be driven from Gothenburg. In general, Sweden has very well-developed roads with few dirt roads and poorly maintained surfaces.

The trip began at Chalmers and proceeded through Gothenburgs urban center before transitioning to suburban and country roads (see Figure 1). While the early portion of the trip involves city driving, this reflects realistic travel patterns for commuters or day trips to recreational areas.

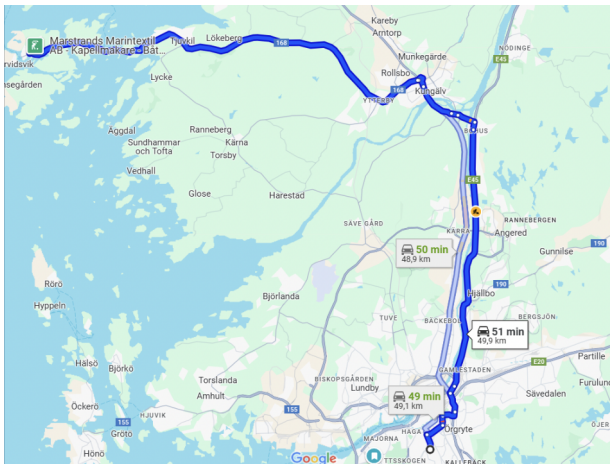

Figure 1: The chosen route from Chalmers to Marstrand

2.2 Modelling

To support OEMs optimizing vehicle performance and safety under rural and urban conditions, the modeling approach focuses on transforming the data into insights. The three key dynamic aspects are:

Braking and Acceleration Dynamics
Longitudinal vehicle dynamics were analyzed from accelerometers, including frequency and intensity of braking and acceleration. Compared with standard maintenance intervals, the analysis helps identify markers for wear on the brakes in rural driving conditions. Models such as these, may support OEM in developing adaptive braking systems and anticipating maintenance characteristics.

Lateral Dynamics and Stability Control
Lateral acceleration is calculated using GPS and inertial data, applying $a_y = \frac{v^2}{r}$. Allowing quantification of vehicle behaviour on rural and urban roads. By modeling these dynamics, the following can be siulated: tire grip, slippage, and steering correction frequency. The results may assist OEMs in calibrating electronic stability control systems.

Vertical Dynamics and Suspension Response
Vertical acceleration data collected from the sensor was used to model the vibrations. The sata collected help evaluate suspension performance, ride comfort and overall road quality. This information could help OEMs to enhance adaptive damping systems, detect early signs of suspension wear and more.

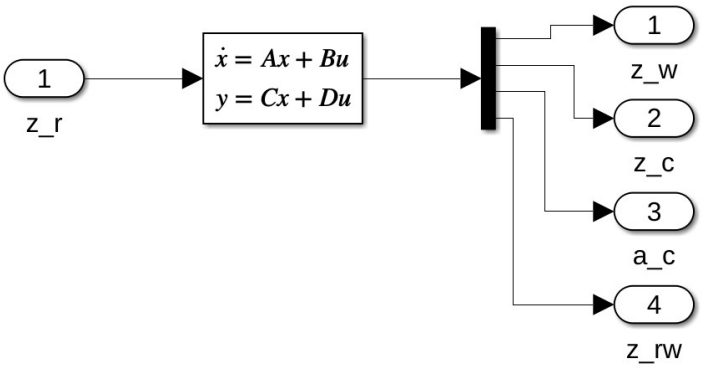

Figure 2: Quarter Car Model

2.3 Data Extraction

The raw data collected by the logger was processed and analyzed using MATLAB. The group developed scripts to import the raw data and organize it in a way that made analysis easier. Key variables such as speed, acceleration, position, altitude and satellite coverage were extracted and aligned using the timestamps of the trip. The processed data was then used to make plots. The full MATLAB code can be found in the appendix.

3 Results

As mentions in the earlier chapter, 2 Methodology, the logged data was extracted and then processed using MATLAB to help answer the proposed questions for the OEM.

3.1 General Data

The general data consists of information about the route taken directly from the logger, that is, the raw extracted data. This data consists of information that was deemed to be necessary to include when creating the codes and to confirm their validity.

3.1.1 Distance traveled

In figure 3, the cumulative distance of the trip to Marstrand and back to Chalmers can be seen. Since the trip used the same route back as it did there, the majority of graphs shown in this report will be mirrored. The trip will also show a flat line, or plateau, in the middle indicating our stay at Marstrand, between the 70 to 118 minute mark. The total data logged for this trip was 102.84 km , with a total time and time traveled of approximately 11960 and 9087 seconds, respectively. The only usable data were the data recorded while traveling.

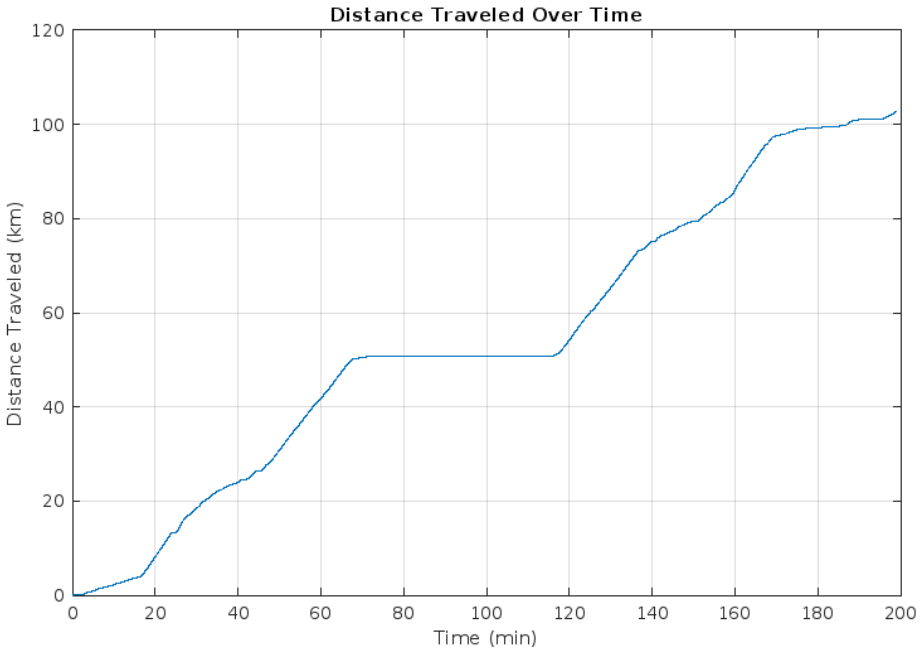

Figure 3: Cumulative Distance Traveled Over Time

3.1.2 Altitude

As for the altitude of the trip (figure 4), one will notice the gigantic dip, showing an altitude of around -140 meters, occurring approximately 170 minutes into the trip. This is where our first falsely logged data occurred. Another more minor source of error occurred during our trip in Marstrand, where the altitude changes despite the knowledge that the vehicle should be stationary.

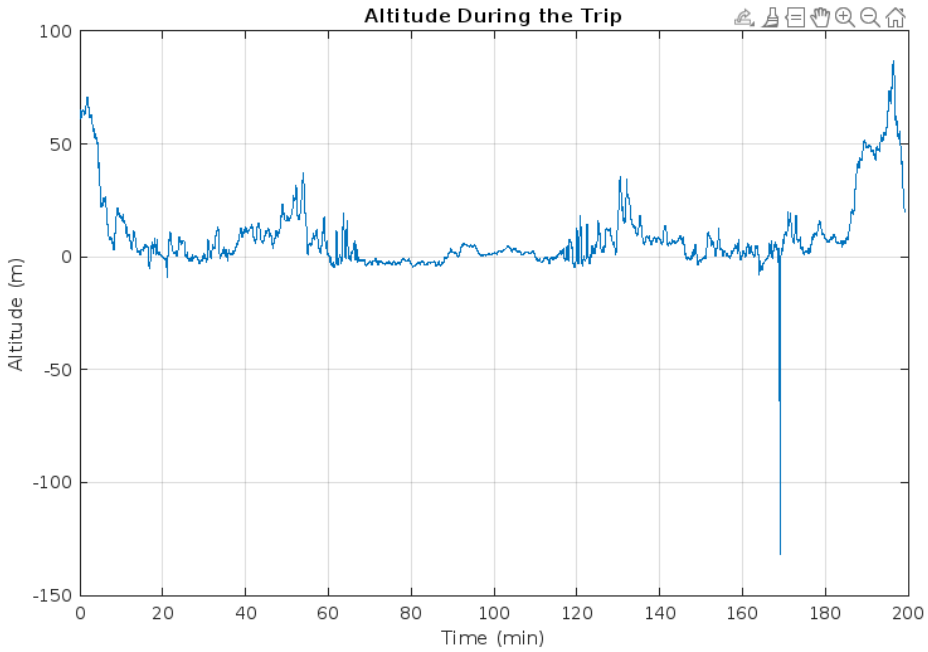

Figure 4: Altitude During the Trip

3.1.3 Speed

In figure 5 the speed kept during the trip is visualized. The speed was later used to visualize the throttle and braking pattern of the trip. Minuscule change occurred between the trip to Marstrand and back, indicating that this might be how the average speed looks like during a similar trip.

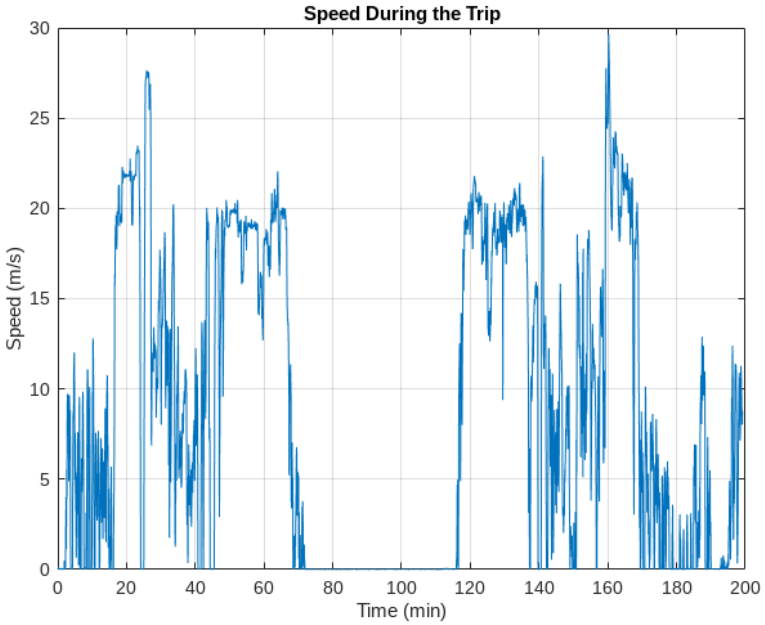

Figure 5: Speed During the Trip

3.1.4 Satellite coverage

The satellite coverage as shown in figure 6, answers some the previous questions regarding the sources of error which occurred during the trip. In the graph, four dips can be seen, with the largest of which is showing a dip down to only four satellites connected to the logger at around the 18 minute mark.

Furthermore, the other dips can be seen at the end of the trip back, going down to 8, 10, and 9 satellites, receptively. The less satellites connected, the greater the chance of incorrect logged data. When analyzing the data, this was kept in mind and the faulty data was discarded.
Nevertheless, the trip as a whole showed a great satellite coverage of 12, which should indicate the validity of the other logged data. However, this doesn't answer the reason of change in altitude during the stay in Marstrand.

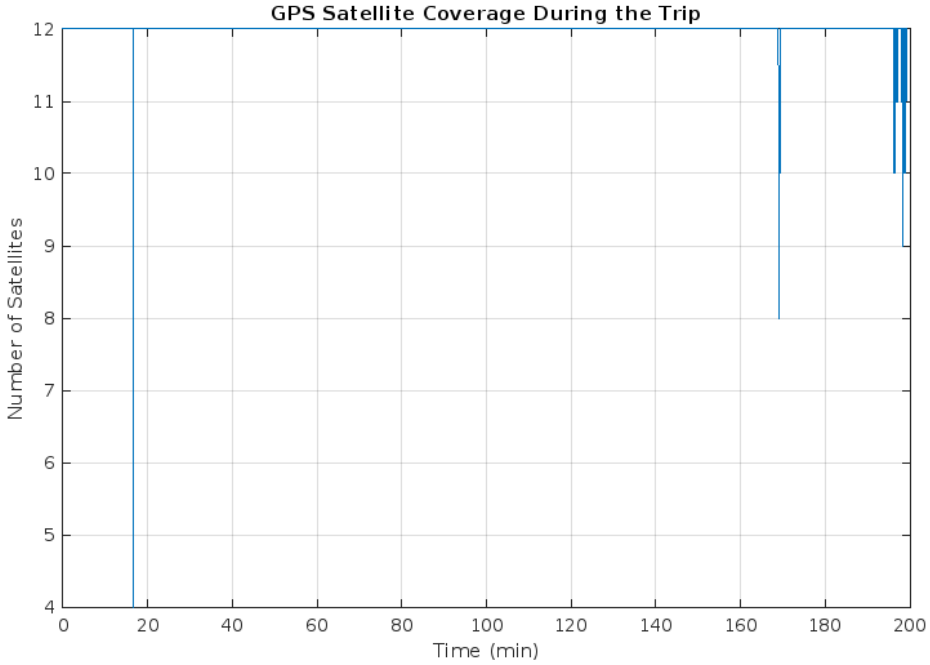

Figure 6: Satellite Coverage During the Trip

3.2 Braking and Acceleration Patterns

In figure 7, the detected braking and throttle (left), and speed kept during the trip (right) is visualized using the logged GPS data. The red triangles indicated positions where braking occurred, in this case it was when the acceleration decreased with a greater value than 1.0 m/s^2 to eliminate the risk of engine brakes being processed in the code. The green triangles show the throttle, that is when the acceleration increased, either by pressing the gas pedal or going downhill.

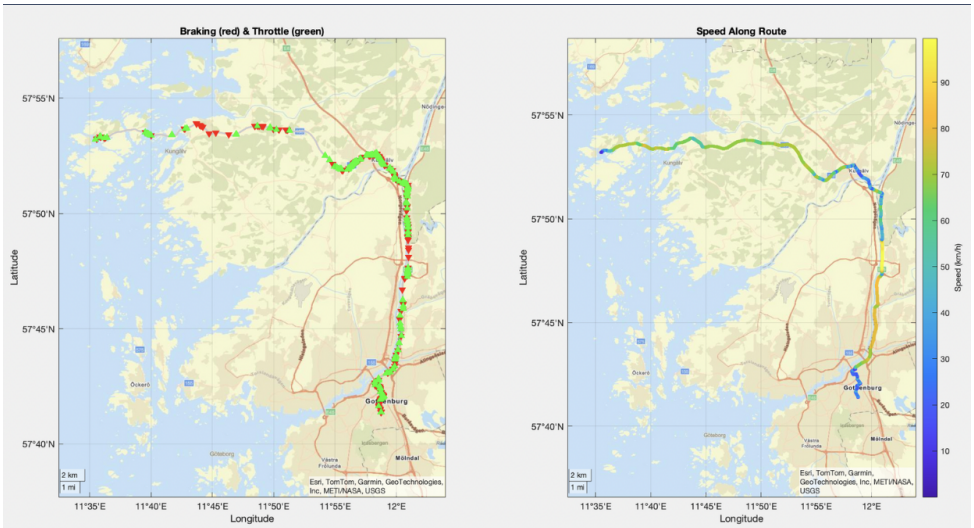

Figure 7: Braking Occurrence

The logged data was then further processed to get values of the frequency-, force distribution-, and the energy rate per timestep of braking as shown in figure 8. From the braking frequency histogram, the frequency of braking increases as the vehicle came closer to Marstrand, keeping this frequency on the way back to Chalmers. This showed that there where in fact less frequent braking while driving in an urban environment. The reason as to why the frequency of braking stayed high on the way back was because of the rush hour traffic that happened during the data logging.
Throughout the trip, the braking force peaked at 18000 N with 1172 such occurrences. The absolute force used was 60000 N showing some necessity of aggressive braking, which happened 3 times during the trip.

In figure 8 (c), the previous two histograms are merged to show the amount of force used in each braking event.

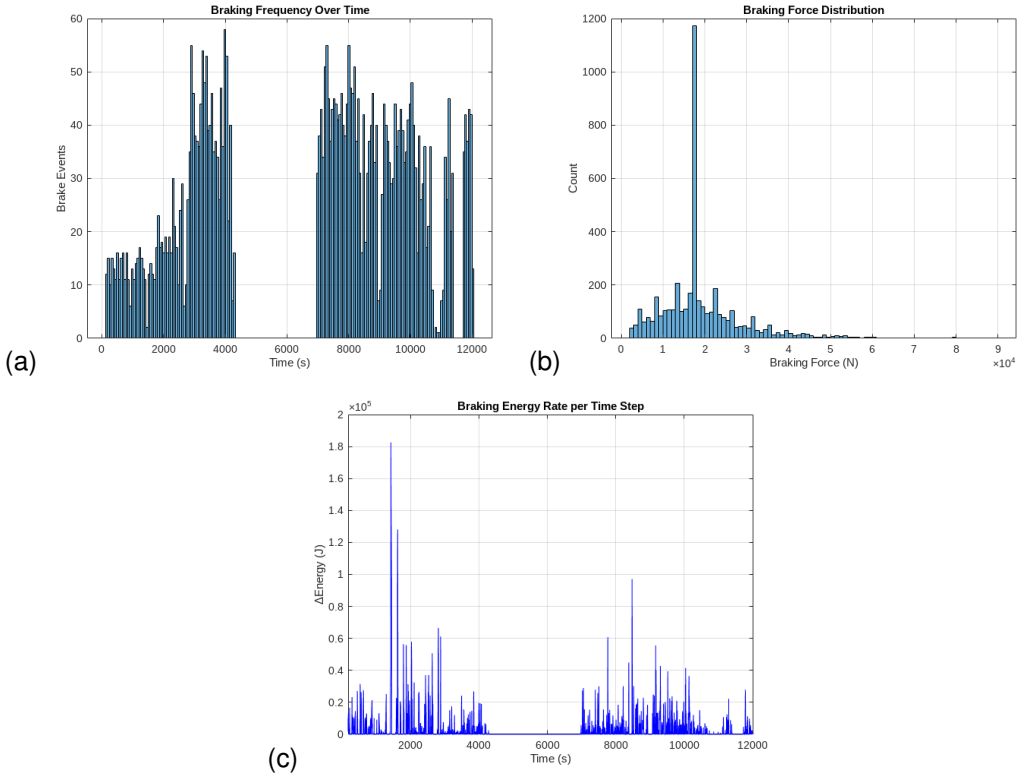

Figure 8: (a) Braking Frequency (b) Braking Force (c) Energy per Timestep

Before the cumulative braking energy could be constructed as in figure 9, parameters and assumptions had to be made. The vehicle used had a service weight of 1870 kg as given by the manufacturer. Together with the cars weight, the four passengers was also accounted for, adding 75 kg each to the total mass, as this is the average human body weight. The total mass of the vehicle was determined to be 2170 kg. The braking force was derived using Newton's second law, $F_{Brake} = m \cdot a_x$, where m is the vehicle mass, and a_x the de-acceleration. When calculation the cumulative braking energy, it was assumed to be estimated as the drop in kinetic energy.

$$\Delta E = \frac{1}{2} m (v_{Previous}^2 - v_{Current}^2) \tag{1}$$

Since no concrete values of the brake discs maximum energy absorption existed, assumptions could be made. By researching different sources on the web, a stop from 100 km/h generates around 100 kW in 5 seconds (Segeren, 2023), which is 500 kJ, assuming that the brake disc can handle a maximum of 5000 such event in its lifetime, it can absorb a total of 2.5 GJ. As to not get too optimistic values, the assumed total energy absorption of a brake disc was 1 GJ, with an recommendation of maintenance when the disc brake reaches 60% of its total capacity.

Using the previous mentioned assumptions and calculations regarding the braking system, the wear of the brake disc could be plotted as in figure 9. The total cumulative braking energy ended up being $1.66 \cdot 10^7$ J, indicating that the trip can be done 35 more times, which totals up to 3702.24 km, before it reaches the 60% mark and it is recommended to inspect the brake discs.

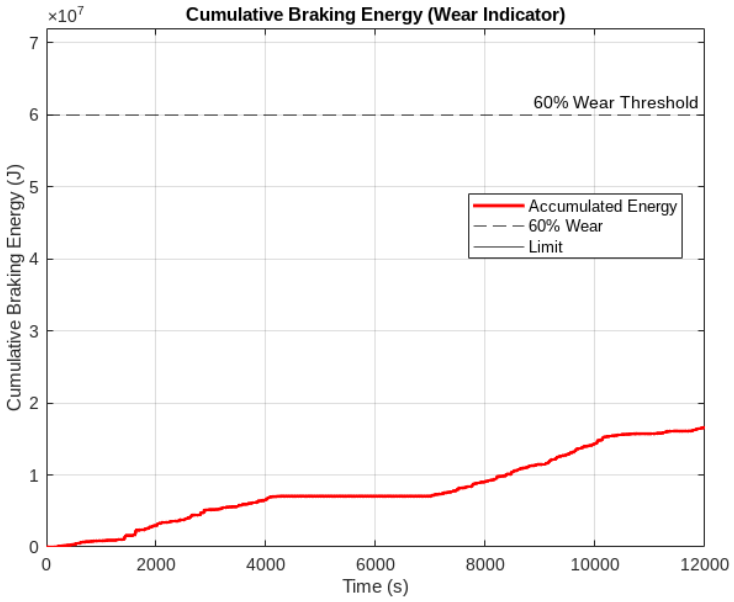

Figure 9: Cumulative Braking Energy

3.3 Lateral acceleration

The lateral acceleration during the trip can be seen in figure 10. The graph does not differentiate between acceleration in the left (driver's side) or right (passenger's side) side of the car because the data is intended to help improve both left and right suspension suysems equally. As seen in the graph, the majority of the lateral acceleration stays in the 2 to 4 m/s^2 range throughout the trip, peaking at 8 m/s^2 at 14:00 and later again rising to 6 m/s^2 at around 16:20.

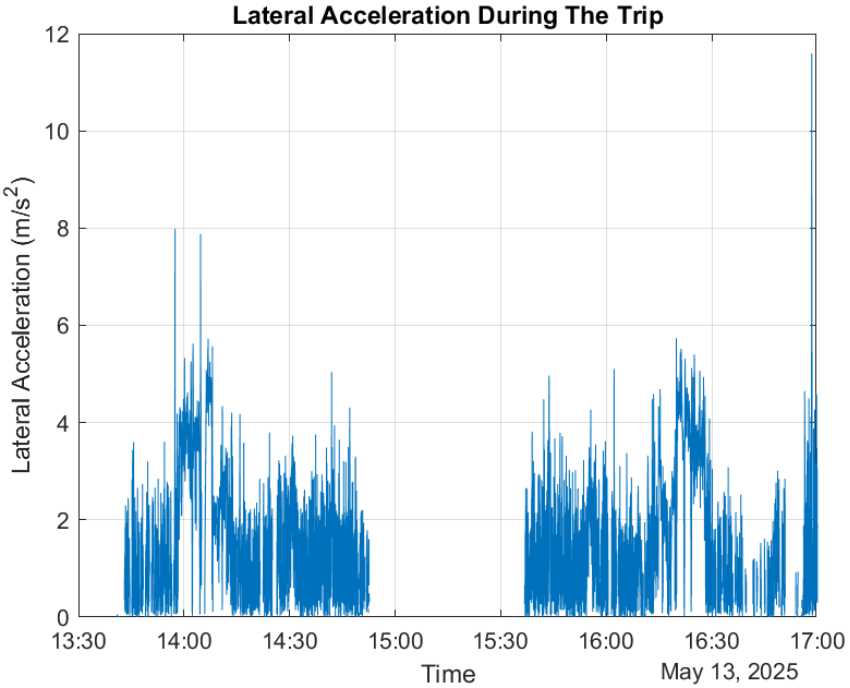

Figure 10: Lateral Acceleration During the Trip

3.4 Road Profiling and Suspension Modeling

Figure 11 is raw data collected by the logger and the inputted into a graph showcasing the vertical acceleration during the trip. The data shows a realistic visualization of the real world, with acceleration which ranges between -0.5 m/s^2 and 0.3 m/s^2. However, the logger also collected data during its standstill in Marstrand, similarly as with the altitude, putting its reliability at question.

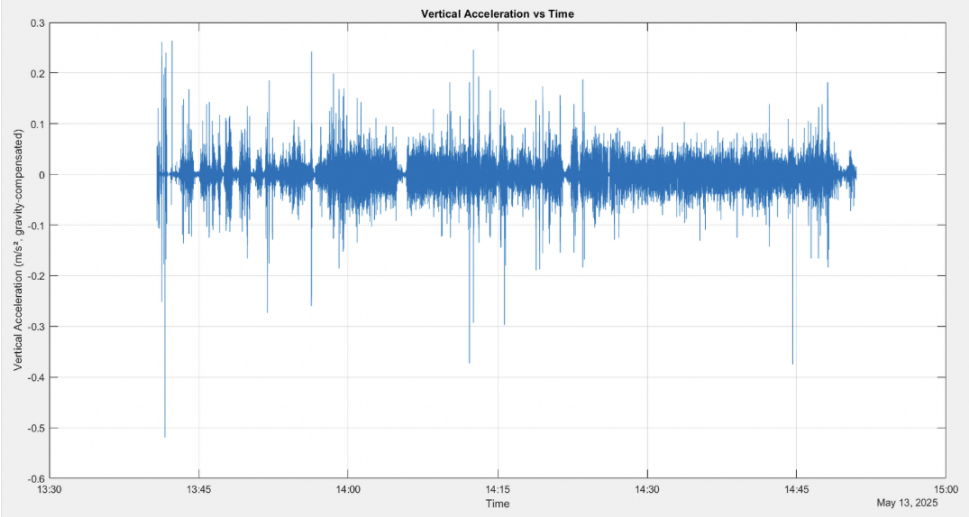

Figure 11: Vertical Acceleration During the Trip

With the use of the Haversine formula and the logged altitude data, figure 12 was created to get an estimation of what the road profile could look like if it was a straight unchanging road. The roads surface is mostly flat, with some noise present throughout, indicating a well kept road. The road profile was then used as the input for the suspension system.

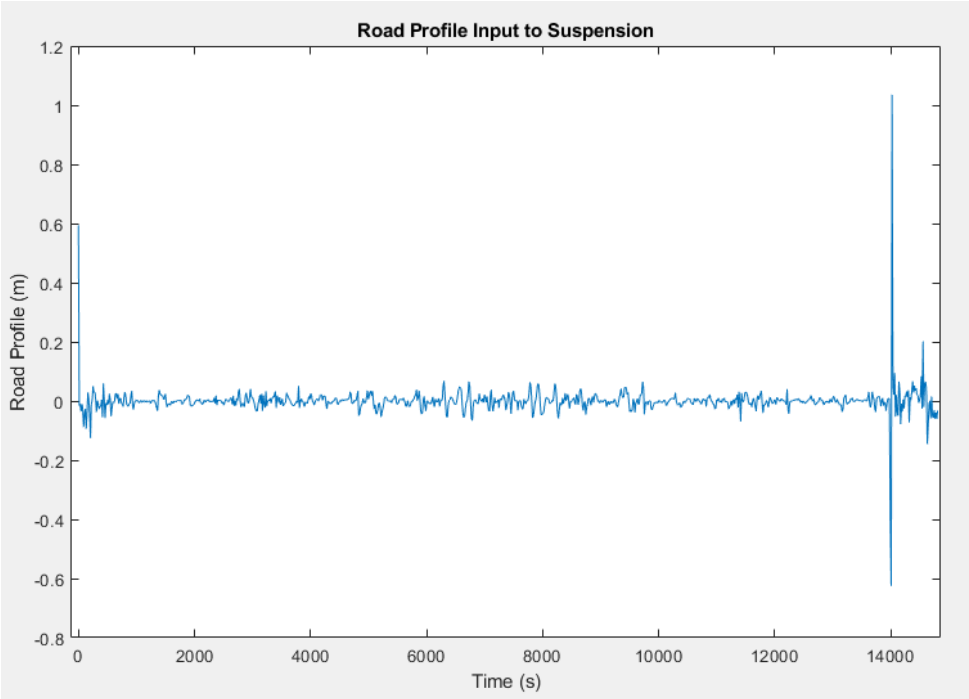

Figure 12: Road Profile

In figure 13 the vertical acceleration of the sprung mass, that is, the vehicles body above the suspension system is plotted. From this plot, it can be seen that the vehicles body experienced close to zero vertical acceleration throughout the trip, except at the end where there are a noticeable spike. This indicated

that the driver and passengers experience a smooth ride, except for that segment, and that the sprung mass, suspension system, worked optimally.

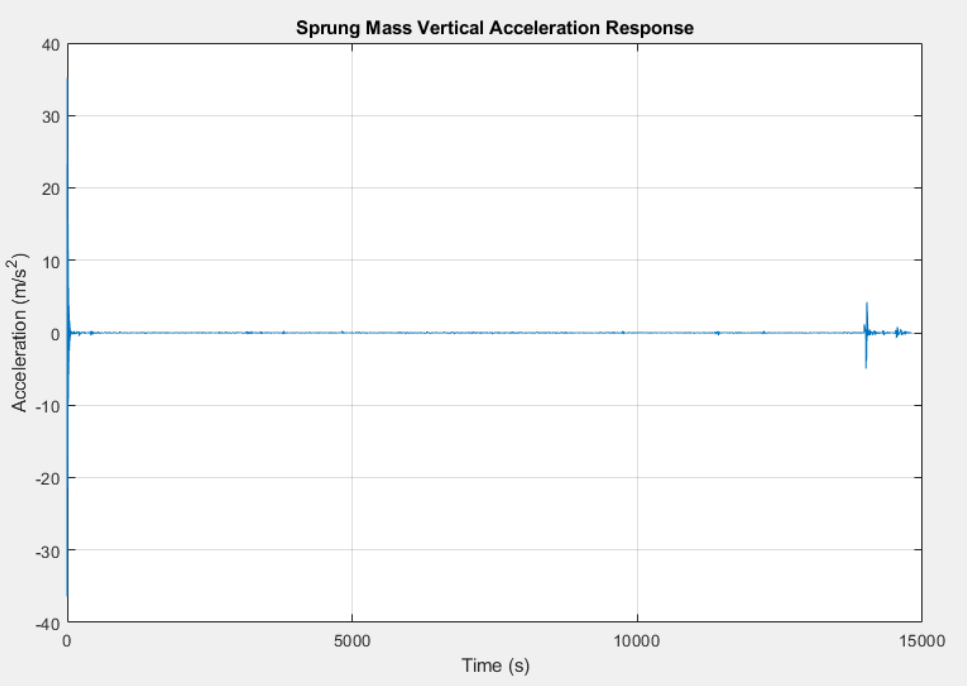

Figure 13: Ride Comfort

4 Discussion

4.1 Lateral Acceleration

The lateral acceleration data collected during rural driving indicate significant variations in side forces, particularly during sharp turns or when avoiding uneven road segments. These variations highlight the challenges in maintaining vehicle stability on rural roads, which are often narrower and have a more abrupt curvature than urban or highway roads.
Key observations:
Two major activity clusters are visible: the first between 13:45 and 15:00, corresponding to the outbound and return segments of the trip. Peaks in lateral acceleration can be seen when driving around bends on country roads, which confirms the increased lateral loads on the suspension system. The frequency of lateral movements indicates that drivers need to regularly adjust the steering to maintain the lane position, especially on less maintained road sections. A higher magnitude of lateral acceleration represents lower ride comfort and can increase the risk of passenger discomfort or vehicle instability.
Influence:
Suspension components, especially antiroll bars and lateral damping mechanisms, are subject to greater stress in rural environments. These findings can guide improvements in chassis tuning and lateral force mitigation strategies to enhance handling and safety in such conditions.

4.2 Braking & Throttle Patterns

The left map visualizes the distribution of braking (red triangles) and throttle (green triangles) events throughout the driving route from Chalmers University to Marstrand and back.
Key observations:
Areas with a high incidence of braking accidents are concentrated along main roads in the city of Göteborg and rural areas, especially near intersections and roundabouts. The throttle inputs were dominated on open highway and rural stretches, particularly in the northern half of the journey, suggesting more consistent acceleration on less traffic-heavy stretches. In some mid-way areas, red and green

dots clearly alternated, reflecting frequent stop-and-go behavior, which was caused by the traffic control equipment or heavy traffic volume at the time.
Explanation:
Speed adjustments must be made more frequently during the urban mobility phase due to dynamic traffic and pedestrian activities. In contrast, the duration of uninterrupted throttle application on a country road is longer, consistent with a more sedate speed cruise. These data show that brake interventions on rural roads are fewer but more severe, while brake interventions on urban roads are more but less severe.

4.3 Road Quality and Vibration

By post-processing the vertical acceleration data collected using the vehicle-mounted data logger, the quality of the road was quantitatively measured and the dynamics of vertical movements was evaluated. This process was performed by preprocessing the vertical acceleration signal and applying double integration to find the vertical road displacement input $z_r(t)$ by minimizing noise and long-term errors. The acquired signal was used as input to a quarter car model, a linear suspension model that represents the dynamics of sprung and unsprung masses.
The ride pattern revealed by the simulation (Figure 13) shows that the vertical acceleration experienced by the sprung mass was minimal during this rural road run, indicating the importance of suspension isolation.
Using this critical information, suspension systems can be evaluated and fault detected, and vertical dynamics data modeling allows vehicle manufacturers (OEM) to,

- Providing support for new active suspension or adaptive damping strategies.
- Estimate the repair needs of the suspension components based on exposure to the road profile.
- Route-specific ride vibration characteristics can be determined.

Furthermore, this model can be embedded in the vehicle's ECU to help determine real-time ride quality and perform OTA diagnostics, which is beneficial for both functional and product development of connected fleets. Also, while this study focused primarily on OEM use cases, this same data driven systems are also important for organizations such as fright/logistic or ride-sharing companies. This model integration allows operators to identify comfort issues, prioritize vehicle maintenance, and increase vehicle utilization, thereby improving passenger experience and efficiency.

4.4 Scalability in a Real Industrial Setting

Ideally, the kind of approach presented in this report would be scaled to a whole fleet of vehicles with a logger in each car. In a real use case the logger would come from the factory installed directly on a chassis to ensure that no extraneous vibrations are registered. This way the data will also match the model as well as possible.
The sheer amount of data coming from a fleet of logger-equiped (with logging at up to 100Hz) cars will pose significant challenges. Edge computing (i.e. processing and filtering the data in the car's computer) could be used to significantly reduce the amount of data that needs to be sent to the OEM and processed. This would also greatly reduce the network requirements in terms of bandwidth and cost. To completely remove network considerations you could transfer data when the car connects to Wi-fi although this could pose ethical problems as you are using the clients bandwith to transfer a lot of data and it also poses cybersecurity risks.
The OEM will also incur significant costs with cloud storage to store all of this data.
Scaling will also provide great opportunities. The large number of vehicles will be able to better show how OTA updates affect driving and behave in widely different environments across the several countries where these cars are used. OTA updates could also be tailored for different regions, countries or even at the user level for the user's specific driving style and preferences.

5 Conclusion

This project primarily demonstrates how data-driven insights can be obtained for both vehicle design and fleet operation by combining real-world data with dynamic models.
Based on the driving data collected during our rural drive, our team studied several critical performance aspects of a connected passenger vehicle: vehicle lateral handling, ride comfort and braking energy

demand. Using the vertical acceleration dataset obtained during this drive and integrated it with the quarter car suspension model, we studied how the vehicle's suspension system reacts to the textures of a rural road.
From a data driven engineering perspective, this project presents a modular reference architecture: that is, real-time data, then embedded inference or cloud computing, then centralized analytics, and finally product and service iteration. This is related to the strategies of connected vehicles in a connected fleet, including region aware tuning, zero downtime, and improved user experience.
This approach is most important in line with the emerging digital twin concept for connected vehicles. Each vehicle becomes a data-providing unit for a larger system model, and this is an important part of a connected fleet.
This project ultimately demonstrates how the development of next-generation vehicles can be implemented in a rapid, strategic, and customer-centric manner, using connected fleet data with combining modeling and simulation. That is, this lays the following foundations for future works like,

- Fleet monitoring (cloud-based)
- Real-time analysis (onboard)
- Vehicle health assessment (AI-enhanced)

By installing data-driven diagnostics in vehicles, fleet operators can also benefit from this process by reducing unplanned downtime, prioritizing services based on usage conditions, and improving ride comfort. All of this helps to achieve transportation efficiency and competitive advantage.

6 References

Segeren, Mark. (2023). *Trends in automotive braking*. Lapinus.
https://www.lapinus.com/applications/automotive/our-thinking/trends-in-automotive-braking/

7 Appendix

Figure 14: Eight of the group members outside of Chalmers library, just before the trip to Marstrand.

Figure 15: The group members in Marstrand.

```
clear all ;
close all ;
clc ;

%%

filename = 'CleanData.txt';
fid = fopen(filename);

gps = [];       % [timestamp, lat, lon, alt, speed, sats]
accel = [];     % [timestamp, ax]

while ~feof(fid)
    line = fgetl(fid);
    if isempty(line), continue; end
    parts = strsplit(line, ',');
    id = str2double(parts{1});

    if id == 0 && numel(parts) >= 7
        ts = str2double(parts{2});
        lat = str2double(parts{3});
        lon = str2double(parts{4});
        alt = str2double(parts{5});
        spd = str2double(parts{6});
        sats = str2double(parts{7});
        gps(end+1,:) = [ts, lat, lon, alt, spd, sats]; %#ok<SAGROW>
    elseif id == 1 && numel(parts) >= 6
        ts = str2double(parts{2});
        us = str2double(parts{3});
        ax = str2double(parts{4});
        accel(end+1,:) = [ts + us/1e6, ax]; %#ok<SAGROW>
    end
end

fclose(fid);

% Normalize time
startTime = gps(1,1);
gpsTimeMin = (gps(:,1) - startTime) / 60;
accelTimeMin = (accel(:,1) - startTime) / 60;

% Calculate distance between GPS points using haversine formula
lat = deg2rad(gps(:,2));
lon = deg2rad(gps(:,3));
R = 6371000; % Earth radius in meters
dlat = diff(lat);
dlon = diff(lon);
a = sin(dlat/2).^2 + cos(lat(1:end-1)) .* cos(lat(2:end)) .* sin(dlon/2).^2;
c = 2 * atan2(sqrt(a), sqrt(1-a));
distances = R * c;
cumulativeDistance = [0; cumsum(distances)];

% Trip on real map
```

```

figure;
geoplot(gps(:,2), gps(:,3), 'b-','LineWidth',2);
geobasemap('streets');

% Plot 1: GPS path colored by time
figure;
scatter(gps(:,3), gps(:,2), 10, gpsTimeMin, 'filled');
xlabel('Longitude');
ylabel('Latitude');
title('GPS Path Colored by Time');
colorbar;
ylabel(colorbar, 'Time (min)');
axis equal; grid on;

% Plot 2: Distance over time
figure;
plot(gpsTimeMin, cumulativeDistance / 1000); % km
xlabel('Time (min)');
ylabel('Distance Traveled (km)');
title('Distance Traveled Over Time');
grid on;

% Plot 3: Altitude over time
figure;
plot(gpsTimeMin, gps(:,4));
xlabel('Time (min)');
ylabel('Altitude (m)');
title('Altitude During the Trip');
grid on;

% Plot 4: Speed over time
figure;
plot(gpsTimeMin, gps(:,5));
xlabel('Time (min)');
ylabel('Speed (m/s)');
title('Speed During the Trip');
grid on;

% Plot 5: Satellite coverage
figure;
plot(gpsTimeMin, gps(:,6));
xlabel('Time (min)');
ylabel('Number of Satellites');
title('GPS Satellite Coverage During the Trip');
grid on;

% Plot 6: GPS with altitude color
figure;
scatter(gps(:,3), gps(:,2), 10, gps(:,4), 'filled');
xlabel('Longitude');
ylabel('Latitude');
title('GPS Path Colored by Altitude');
colorbar;
ylabel(colorbar, 'Altitude (m)');
axis equal; grid on;

% Print total distance
totalDistKm = cumulativeDistance(end) / 1000;
fprintf('Total distance traveled: %.2f km\n', totalDistKm);
```

```
%--- Settings & thresholds ---
filename        = 'DATAcleanoneway.txt'; % DATAcleanoneway.txt or DATAclean.txt
accelSensorID   = 1;      % accelerometer rows
brake_thresh    = -2.5;   % real braking (<   2  .5 m/s  ) filtering
throttle_thresh = +1.0;   % throttle (> +1.0 m/s  ) filtering

 %--- 1. Read & split data ---
M        = readmatrix(filename);
sensorID = M(:,1);
t_s      = M(:,2);
t_us     = M(:,3);

% GPS records
```

```
gpsIdx   = (sensorID == 0);
gpsTime  = t_s(gpsIdx) + t_us(gpsIdx)*1e-6;
lat      = M(gpsIdx,3);
lon      = M(gpsIdx,4);
speed_kmh = M(gpsIdx,6)   % m/s     km/h

% accel records
accIdx   = (sensorID == accelSensorID);
ax       = M(accIdx,4);
axTime   = t_s(accIdx) + t_us(accIdx)*1e-6;

%  z e r o base  times for annotation if needed
t0       = gpsTime(1);

% --- 2. Detect    real    braking peaks & throttle samples ---
% find  m i l d brake  runs
mildBrakeMask = ax < -1.0;
d = diff([0; mildBrakeMask; 0]);
starts = find(d ==  1);
ends   = find(d == -1) - 1;

brakeIdx = [];
for i = 1:numel(starts)
    runIdx = starts(i):ends(i);
    [peakDecel, loc] = min(ax(runIdx));
    if peakDecel < brake_thresh
        brakeIdx(end+1) = runIdx(loc); %#ok<SAGROW>
    end
end

% throttle events
throttleIdx = find(ax > throttle_thresh);

% Map each event to  n e a r e s t time  GPS fix
nB = numel(brakeIdx);  nT = numel(throttleIdx);
brakeLat = nan(nB,1);  brakeLon = nan(nB,1);
thrLat   = nan(nT,1);  thrLon   = nan(nT,1);
for i = 1:nB
    dt = axTime(brakeIdx(i)) - t0;
    [~, j] = min(abs((gpsTime - t0) - dt));
    brakeLat(i) = lat(j);
    brakeLon(i) = lon(j);
end
for i = 1:nT
    dt = axTime(throttleIdx(i)) - t0;
    [~, j] = min(abs((gpsTime - t0) - dt));
    thrLat(i) = lat(j);
    thrLon(i) = lon(j);
end

% --- 3. Plot all three subplots ---
figure('Units','normalized','Position',[0.05 0.1 0.9 0.7]);

% Common zoom limits
latBuf = 0.05; lonBuf = 0.05;
latLim = [min(lat)-latBuf, max(lat)+latBuf];
lonLim = [min(lon)-lonBuf, max(lon)+lonBuf];

% 3.1 Accel/Brake only
subplot(1,2,1);
geoscatter(lat, lon, 5, [0.8 0.8 0.8], 'filled');  % base track in gray
hold on;
geoscatter(brakeLat, brakeLon, 50, 'r', 'filled', 'Marker','v');
geoscatter(thrLat,   thrLon,   50, 'g', 'filled', 'Marker','^');
geobasemap streets;
geolimits(latLim, lonLim);
title('Braking (red) & Throttle (green)');
hold off;

% 3.2 Velocity only
subplot(1,2,2);
h2 = geoscatter(lat, lon, 15, speed_kmh, 'filled');
geobasemap streets;
```

```
geolimits(latLim, lonLim);
cb2 = colorbar;
cb2.Label.String = 'Speed (km/h)';
title('Speed Along Route');
```

```
clear; close all; clc;


% Load and parse file
filename = 'DATAclean.txt';
raw = readlines(filename);

gnss = [];
acc = [];

for i = 1:length(raw)
    row = str2double(split(raw(i), ','));
    if row(1) == 0
        gnss(end+1,:) = row(2:end); % timestamp, lat, lon, ampl, speed, sats
    elseif row(1) == 1
        acc(end+1,:) = row(3:end);  % timestamp, x, y, z
    end
end

% Sort GNSS and Acc data by timestamp
gnss = sortrows(gnss, 1); % sort by first column (time)
acc  = sortrows(acc, 1);

% Remove duplicates (optional, but safe)
[~, uniq_idx_gnss] = unique(gnss(:,1), 'stable');
gnss = gnss(uniq_idx_gnss,:);

[~, uniq_idx_acc] = unique(acc(:,1), 'stable');
acc = acc(uniq_idx_acc,:);

% Constants
mass = 1870 + 75*4; % kg (vehicle + 4 passengers)
g = 9.81;

% Convert time (assume UNIX seconds, not microseconds!)
gnss_time = gnss(:,1) - gnss(1,1); % [s]
acc_time = acc(:,1) - acc(1,1);   % [s]

% Check increasing time
if ~issorted(gnss_time) || ~issorted(acc_time)
    error('Timestamps are not increasing!');
end

% Convert GNSS speed from km/h to m/s
speed_ms = gnss(:,5) / 3.6;

% Interpolate speed to acc time
speed_interp = interp1(gnss_time, speed_ms, acc_time, 'linear', 'extrap');
speed_interp = fillmissing(speed_interp, 'linear');
speed_interp = smoothdata(speed_interp, 'movmean', 5);

% Clamp unrealistic speeds
speed_interp(speed_interp < 0 | speed_interp > 80) = NaN;

% Accelerometer x-axis in m/s
acc_x = acc(:,2) * g;

% Braking condition: strong deceleration and moving
brake_threshold = -1.0; % m/s
min_speed = 2;          % m/s
is_braking = acc_x < brake_threshold & speed_interp > min_speed;

% Compute kinetic energy drop (  (  mv  ))
brake_energy = zeros(size(acc_time));
energy_step = zeros(size(acc_time));

prev_v = speed_interp(1);
for i = 2:length(speed_interp)
    curr_v = speed_interp(i);
```

```
    if is_braking(i) && ~isnan(curr_v) && ~isnan(prev_v) && curr_v < prev_v
        delta_E = 0.5 * mass * (prev_v^2 - curr_v^2);
        if delta_E > 0
            energy_step(i) = delta_E;
            brake_energy(i) = brake_energy(i-1) + delta_E;
        else
            brake_energy(i) = brake_energy(i-1);
        end
    else
        brake_energy(i) = brake_energy(i-1);
    end
    prev_v = curr_v;
end

% Brake wear thresholds
pad_lifespan_energy = 1e8; % J
warning_limit = 0.6 * pad_lifespan_energy;

% Plotting
figure('Position', [100, 100, 1600, 900]);

% 1. Braking frequency
subplot(4,1,1);
histogram(acc_time(is_braking), 'BinWidth', 60);
xlabel('Time (s)');
ylabel('Brake Events');
title('Braking Frequency Over Time');
grid on;

% 2. Braking Force Distribution
subplot(4,1,2);
brake_force = -acc_x .* mass;
brake_force(~is_braking) = NaN;
histogram(brake_force, 'BinWidth', 1000);
xlabel('Braking Force (N)');
ylabel('Count');
title('Braking Force Distribution');
grid on;

active_idx = find(energy_step > 0);

if ~isempty(active_idx)
    x_min = max(0, acc_time(active_idx(1)) - 10);  % start a bit earlier
    x_max = acc_time(active_idx(end)) + 10;        % end a bit after
else
    x_min = 0;
    x_max = max(acc_time);
end

% 3. Cumulative Braking Energy
subplot(4,1,3);
plot(acc_time, brake_energy, 'r', 'LineWidth', 2);
hold on;
yline(warning_limit, 'k--', '60% Wear Threshold');
yline(pad_lifespan_energy, 'k-', 'Limit');
ylim([0, 1.2 * max([brake_energy(end), warning_limit])]); % <-- dynamic y-axis limit
xlim([0, 12000]); % <-- Manually limit x-axis to 18000 seconds (5 hours)
xlabel('Time (s)');
ylabel('Cumulative Braking Energy (J)');
title('Cumulative Braking Energy (Wear Indicator)');
legend('Accumulated Energy', '60% Wear', 'Limit');
grid on;

% 4. Energy per timestep
subplot(4,1,4);
plot(acc_time, energy_step, 'b');
xlabel('Time (s)');
ylabel(' Energy  (J)');
title('Braking Energy Rate per Time Step');
grid on;

% Optional zoom to active area
nonzero_idx = find(energy_step > 0);
```

```
if ~isempty(nonzero_idx)
    xlim([acc_time(nonzero_idx(1)), acc_time(nonzero_idx(end))]);
end

fprintf('Total cumulative braking energy: %.2e J\\n', brake_energy(end));
```

```
% Load and parse the data file
filename = 'DATAclean';
data = readtable(filename, 'Delimiter', ',', 'ReadVariableNames', false);

% Extract GPS data (type 0 records)
gps_data = data(data.Var1 == 0, :);
timestamps = gps_data.Var2;
latitudes = gps_data.Var3;
longitudes = gps_data.Var4;
altitudes = gps_data.Var5;
speeds = gps_data.Var6;
satellites = gps_data.Var7;

% Convert Unix timestamps to MATLAB datetime
dates = datetime(timestamps, 'ConvertFrom', 'posixtime','TimeZone','UTC');
dates.TimeZone = 'Europe/Stockholm';

% Create a map of your trip
figure;
geoplot(latitudes, longitudes, 'b-', 'LineWidth', 2);
geobasemap('streets');
title('The Trip Route');

% Plot altitude over time
figure;
plot(dates, altitudes);
xlabel('Time');
ylabel('Altitude (m)');
title('Altitude During The Trip');
grid on;

% Plot speed over time
figure;
plot(dates, speeds);
xlabel('Time');
ylabel('Speed (km/h)');
title('Speed During The Trip');
grid on;

% Plot satellite count
figure;
plot(dates, satellites);
xlabel('Time');
ylabel('Number of Satellites');
title('GPS Satellite Coverage During The Trip');
grid on;

% Calculate distance traveled (approximate)
earth_radius = 6371000; % meters
lat_rad = deg2rad(latitudes);
lon_rad = deg2rad(longitudes);
delta_lat = diff(lat_rad);
delta_lon = diff(lon_rad);
a = sin(delta_lat/2).^2 + cos(lat_rad(1:end-1)) .* cos(lat_rad(2:end)) .* sin(delta_lon
    /2).^2;
c = 2 * atan2(sqrt(a), sqrt(1-a));
distances = earth_radius * c;
total_distance = sum(distances);

fprintf('Total distance traveled: %.2f km\n', total_distance/1000);

% Plot cumulative distance
figure;
cumulative_distance = [0; cumsum(distances)]/1000; % convert to km
plot(dates, cumulative_distance);
xlabel('Time');
ylabel('Cumulative Distance (km)');
title('Distance Traveled Over Time');
```

```
grid on;

% --- Compute lateral acceleration (a = V^2 / R) ---
% Convert speed from km/h to m/s
speeds_ms = speeds * (1000 / 3600);

% Approximate radius of curvature using 3-point method
R = NaN(length(lat_rad), 1); % Preallocate
for i = 2:length(lat_rad)-1
    % Get 3 consecutive points
    lat1 = lat_rad(i-1); lon1 = lon_rad(i-1);
    lat2 = lat_rad(i);   lon2 = lon_rad(i);
    lat3 = lat_rad(i+1); lon3 = lon_rad(i+1);

    % Convert to x, y
    x1 = earth_radius * lon1 * cos((lat1 + lat2)/2);
    y1 = earth_radius * lat1;
    x2 = earth_radius * lon2 * cos((lat1 + lat2)/2);
    y2 = earth_radius * lat2;
    x3 = earth_radius * lon3 * cos((lat2 + lat3)/2);
    y3 = earth_radius * lat3;

    % Determinant method
    A = [x1 y1 1; x2 y2 1; x3 y3 1];
    D = 2 * det(A);
    if abs(D) < 1e-6
        R(i) = NaN; % Degenerate case (points nearly colinear)
        continue;
    end
    a = x1^2 + y1^2;
    b = x2^2 + y2^2;
    c = x3^2 + y3^2;
    center_x = det([a y1 1; b y2 1; c y3 1]) / D;
    center_y = det([x1 a 1; x2 b 1; x3 c 1]) / D;
    R(i) = sqrt((x2 - center_x)^2 + (y2 - center_y)^2); % Radius at midpoint
end

% Lateral acceleration
a_lat = speeds_ms.^2 ./ R;

% Plot
figure;
plot(dates, a_lat);
xlabel('Time');
ylabel('Lateral Acceleration (m/s^2)');
title('Lateral Acceleration During The Trip');
grid on;
```

```
clc; clear;

% Parameters for quarter-car suspension model
ms = 1470;     % Sprung mass (kg)
mu = 165;      % Unsprung mass (kg)
ks = 32500;    % Suspension stiffness (N/m)
cs = 1600;     % Suspension damping (Ns/m)
kt = 22500;    % Tire stiffness (N/m)

% State-space model
A = [0 0 1 0;
     0 0 0 1;
     -ks/ms ks/ms -cs/ms cs/ms;
     ks/mu -(ks+kt)/mu cs/mu -cs/mu];

B = [0; 0; 0; kt/mu];
C = [0 0 -ks/ms cs/ms];  % Output: sprung mass acceleration
D = 0;
sys = ss(A, B, C, D);

%% Load logger data
filename = 'DATAclean.txt';
data = readmatrix(filename);

% Extract GPS rows (ID = 0)
gps_data = data(data(:,1) == 0, :);
```

```
timestamps = gps_data(:,2);
latitude = gps_data(:,3);
longitude = gps_data(:,4);
elevation = gps_data(:,5);  % Elevation in meters

% Estimate travel distance using haversine formula
R = 6371000; % Earth radius in meters
lat_rad = deg2rad(latitude);
lon_rad = deg2rad(longitude);
dlat = diff(lat_rad);
dlon = diff(lon_rad);
a = sin(dlat/2).^2 + cos(lat_rad(1:end-1)) .* cos(lat_rad(2:end)) .* sin(dlon/2).^2;
c = 2 * atan2(sqrt(a), sqrt(1-a));
dist = R * c;
dist_cum = [0; cumsum(dist)];

% Resample elevation data over uniform distance
res_dist = linspace(0, dist_cum(end), 1000);

% Remove duplicate distances
[dist_cum_unique, unique_idx] = unique(dist_cum, 'stable');
elevation_unique = elevation(unique_idx);

% Interpolate elevation
res_elev = interp1(dist_cum_unique, elevation_unique, res_dist, 'linear');


% Convert to road profile (differentiated & normalized)
res_elev = res_elev(:);           % Force column vector
z_r = diff([0; res_elev]);

z_r = z_r - mean(z_r);
z_r = 0.05 * z_r / std(z_r);  % Normalize to ~5 cm RMS

% Time base from average speed
V = 25 / 3.6;  % Assume 25 km/h in m/s
t = res_dist / V;

% Interpolate road profile for uniform time step
Fs = 1000;
t_uniform = (0:1/Fs:t(end))';
z_r_interp = interp1(t, z_r, t_uniform, 'linear', 'extrap');

% Run simulation
[y, t_out] = lsim(sys, z_r_interp, t_uniform);

% Plot results
figure;
subplot(3,1,1);
plot(res_dist/1000, res_elev);
xlabel('Distance (km)'); ylabel('Elevation (m)');
title('Elevation Profile from GPS');

subplot(3,1,2);
plot(t_uniform, z_r_interp);
xlabel('Time (s)'); ylabel('Road Profile (m)');
title('Road Profile Input to Suspension');

subplot(3,1,3);
plot(t_out, y);
xlabel('Time (s)'); ylabel('Acceleration (m/s^2)');
title('Sprung Mass Vertical Acceleration Response');
grid on;
```