Assignment #1: Quasi-static HEV model and Rule Based Control

Table of Contents

Group information

Group number: 12

Students:

- Emanuele Landolina, s297069
- Giuseppe Maria Marchese, s348145
- Walter Maggio, s343988

Load the cycle and vehicle data

In this initial phase, the necessary folders are added to the MATLAB path to ensure that all required functions and tools are available for the script.

The **vehicle data** and the **WLTP driving cycle** are then loaded into the workspace. From the WLTP cycle, the main variables (**time vector**, **vehicle speed**, and **vehicle acceleration)** are extracted and stored for further use in the simulation.

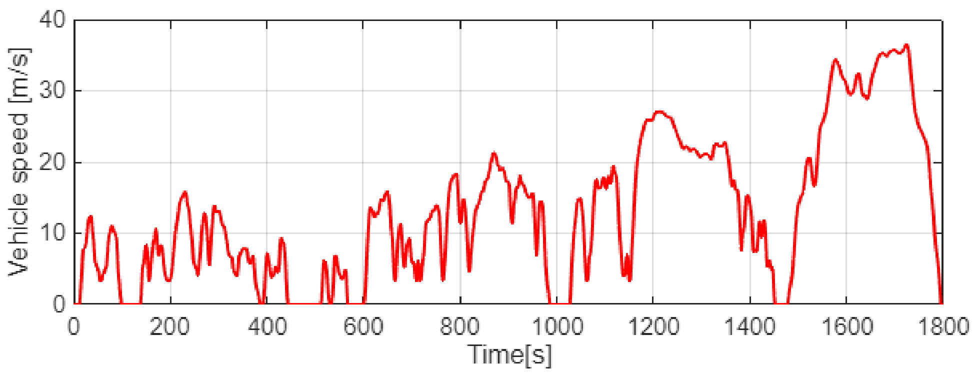
To gain a better understanding of the driving cycle being analyzed, two plots are generated:

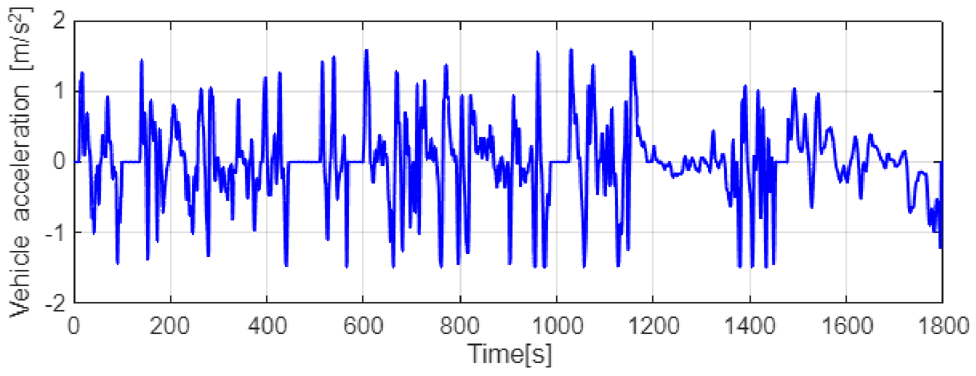
- **Vehicle Speed vs. Time**
- **Vehicle Acceleration vs. Time**

These plots provide a clear visualization of the WLTP cycle's dynamics, helping to interpret the cycle's behavior and the demands it will impose on the vehicle during the simulation.

```
clear all
close all
clc


addpath("utilities");
addpath("models");
addpath("data");

%load vehicle data
vehData_raw = load("vehData.mat");

%load mission data
mission = load("WLTP");

%time vector
time_vector = mission.time_s;

%Velocity and acceleration
vehAcc = mission.acceleration_m_s2;       %[m/s^2]
vehSpd = mission.speed_km_h/3.6;        %[m/s]
```

```
%% Vehicle speed and acceleration

%plot mission varibles
t = tiledlayout(2,1);

nexttile(1)
plot(time_vector,vehSpd,"r",LineWidth=1.2)
xlabel("Time[s]")
ylabel("Vehicle speed [m/s]")
grid on

nexttile(2)
plot(time_vector,vehAcc,"b",LineWidth=1.2)
xlabel("Time[s]")
ylabel("Vehicle acceleration [m/s^2]")
grid on
```





The WLTP cycle is divided into several parts. The first phase simulates urban driving, characterized by low speeds and gentle accelerations, resulting in a relatively low power demand. As the cycle progresses into the extra-urban phase, both speed and acceleration increase, leading to a higher power requirement. The final phase represents highway driving, which is the most demanding in terms of power due to high speeds and rapid accelerations.

In addition to the plotting phase, a **scaling operation** is performed. The **vehicle data** provided are adjusted to align with the specific parameters assigned to the group.

```
%vehicle characteristic [Group 12]
engine_power = 66000;     %[W]
E_machine_power = 70000; %[W]
battery_energy = 945;     %[Wh]

%Scaled vehicle data
vehData = scaleVehData(vehData_raw,E_machine_power,engine_power,battery_energy);
```

Thermostat controller

Simulation loop

This section presents the simulation of the thermostat controller, whose objective is to optimize the chosen objective function, in this case the **fuel consumption**.

The simulation starts by setting the initial conditions:

- **State of Charge (SOC):** 0.6
- **Engine state:** OFF (discharge mode), corresponding to the boolean value `engState = 0`

The simulation is performed using a **for-loop** iterated over the time interval of the **WLTP cycle**.

The exogenous inputs are taken from the **vehicle data structure** and the **WLTP cycle profile**. The main functions used during the simulation are:

- `thermostatControl`: computes the control variables (**engine speed** and **torque**) and the **boolean** variable **`engState`**, which determines if the engine is ON or OFF; a detailed description is provided in the appropriate section.
- `hev_model`: evaluates the **controller performance** and **vehicle behavior** based on the current state and control actions.

```
%Initialating SOC and engineState
SOC(1) = 0.6;
engState(1) = 0; % engine is off


%Setup for simulation
for n = 1:length(time_vector)
    [engSpd(n), engTrq(n),engState(n+1)] = ...
        thermostatControl(SOC(n), engState(n),vehSpd(n),vehAcc(n),vehData);
    [SOC(n+1), fuelflwRate(n), unfeas(n), prof(n)] = ...
        hev_model(SOC(n), [engSpd(n), engTrq(n)], [vehSpd(n), vehAcc(n)],
vehData);
end
```

Post-Processing: main results

To analyze the performance of the controller and obtain the objective function of this project two metrics are evaluated:

- `fuelConsumption` is the total fuel consumption for the whole mission (in kg). It is evaluated performing an integral of the fuel flow rate over time by using the function trapz. the arguments of the function are the `time_vector` and the `prof_struct, that is` a scalar structure where all the information about the vehicle state are saved instant by instant during the simulation. The main objective was to remain below 1 kg of fuel consumption for the entire mission, and this target was fully achieved. This result validates the correct functioning of the designed thermostat controller.

```
prof_struct = structArray2struct(prof);

fuelConsumption = trapz(time_vector,prof_struct.fuelFlwRate)/1000;
fprintf("Fuel consumption: %.3f kg\n", fuelConsumption);
```

```
Fuel consumption: 0.902 kg
```

- `FuelEconomy` represents the distance-specific fuel consumption for the entire mission, expressed in l/100 km, to provide a clearer and more explicit result. To compute this, the mission distance is first evaluated by integrating the speed profile over time, and then the fuel economy is calculated, taking into account the fuel density (`fuel_rho`).

$$fuelEconomy = \frac{fuelConsumption}{fuel\_rho \cdot distance} \cdot 100$$

```
distance = trapz(time_vector,vehSpd)/1000;
fuel_rho = vehData.eng.fuelDensity; %[Kg/l]
fuelEconomy = (fuelConsumption/(fuel_rho*distance))*100;
fprintf("Fuel economy: %.3f l/100km\n", fuelEconomy );
```

```
Fuel economy: 4.937 l/100km
```

- `finalSOC` represents the final state of charge (SOC) of the battery, which indicates the battery's status at the end of the mission and furthermore the quantity of energy that can be used.

```
finalSOC = SOC(end);
 fprintf("Final SOC: %.3f\n", finalSOC );
```

```
Final SOC: 0.610
```

Save the results

The obtained results are saved in a .mat file.

```
save("results.mat", "prof", "fuelConsumption", "fuelEconomy", "finalSOC")
```

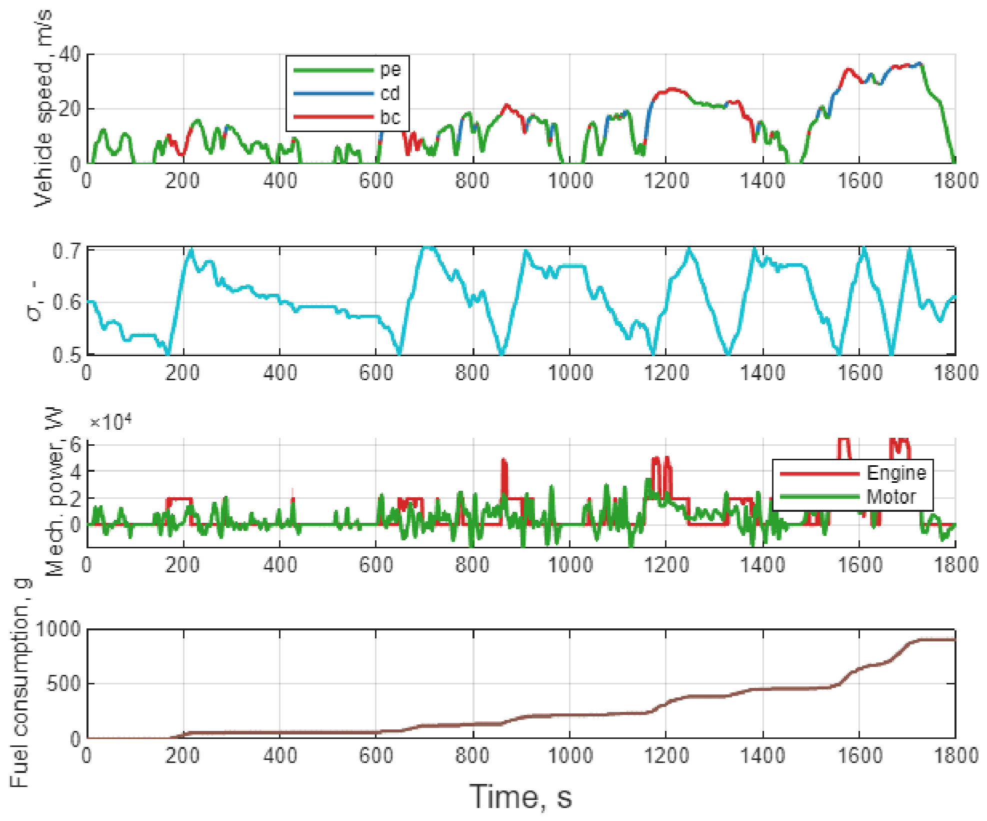
Post-Processing: plots and analysis

To evaluate the controller's performance, several plots are generated to verify that the system parameters respect the given boundaries:

- **Battery State of Charge (SOC):** displays the charge/discharge trend;
- **Battery Power:** shows the power requested and delivered;
- **Engine Operating Points;**
- **Generator Operating Points;**
- **Power Profiles.**

The following graphs provide detailed vision about how to controller works.

```
mainProfiles(prof);
```



The main profiles consist of four key plots that should be analyzed together to fully understand the control system's behavior throughout the entire cycle.

Plot 1: Vehicle Speed vs Time

This plot shows the vehicle speed throughout the WLTP cycle (0-1800s), color-coded based on three operating modes:

- **Green (pe):** pure electric mode;
- **Blue (cd):** charge-depleting mode;
- **Red (bc):** battery charging mode (engine active to recharge the battery).

During the **urban phase (0–600s)**, the vehicle operates mostly in pure electric mode (green), with low speeds and frequent stops, typical of city driving.

In the **extra-urban phase (600–1200s)**, speeds increase and stabilize, with a mix of electric and engine-supported operation. During the transitions from extra-urban phase to highway's one, there is an intensive charge depleting mode, during which the vehicle draws down the battery's stored energy to power the electric motor.

In the **highway phase (1200–1800s)**, vehicle speed is higher, and there are clear segments of battery charging (red). This indicates that the engine is used more aggressively to recharge the battery.

Plot 2: State of Charge vs Time

This plot shows the SOC evolution, showing the charge and discharge trends between two set limits of 0.5 and 0.7.

The starting condition was SOC=0.6,indicating 60% of the battery energy is available.

During urban phase SOC decreases as the vehicle relies on electric power. At around 200s, SOC starts rising, signaling the engine's first intervention, which is triggered by the threshold controller. When SOC reaches 0.5, the engine is activated to recharge the battery and prevent excessive depletion.

During extra-urban and highway driving phases, the SOC fluctuates due to alternating charging and discharging cycles, influenced by power demand and engine operation. In the highway phase, the high power demand leads to a rapid depletion of the battery charge, resulting in frequent charging cycles. This repeated cycling accelerates battery aging and negatively impacts its long-term performance and durability.

Plot 3: Mechanical Power vs Time

This plot displays the mechanical power contributions from the electric motor (green line) and the internal combustion engine (red line).

During urban phase, power is mostly supplied by the electric motor while the ICE remains off or is barely used (three times over 600s), for the reasons analyzed previously. The thermostat here ensures that when the ICE is switched on, it is operating efficiently. Since the engine is activated occasionally, the thermostat's role is to ensure it's running at the most fuel-efficient operating conditions, minimizing fuel consumption during those brief moments of operation.

When the vehicle switch into extra-urban phase, the engine starts contributing more, supporting battery charging and assisting propulsion during higher demand moments. It is really important to remember that the vehicle is a series hybrid vehicle, so the ICE cannot provide the power directly to the wheels so it must respects the battery power's limits.

In highway phase, the engine power increase significantly, running steady and at higher loads, which is ideal for efficient operation. The thermostat optimizes fuel consumption by ensuring the engine operates in a range that minimizes fuel waste and maximizes power output. This is where the engine is most efficient, and the thermostat ensures it remains in that optimal range, particularly during sustained, high-load conditions.

Plot 4: Fuel consumption vs Time

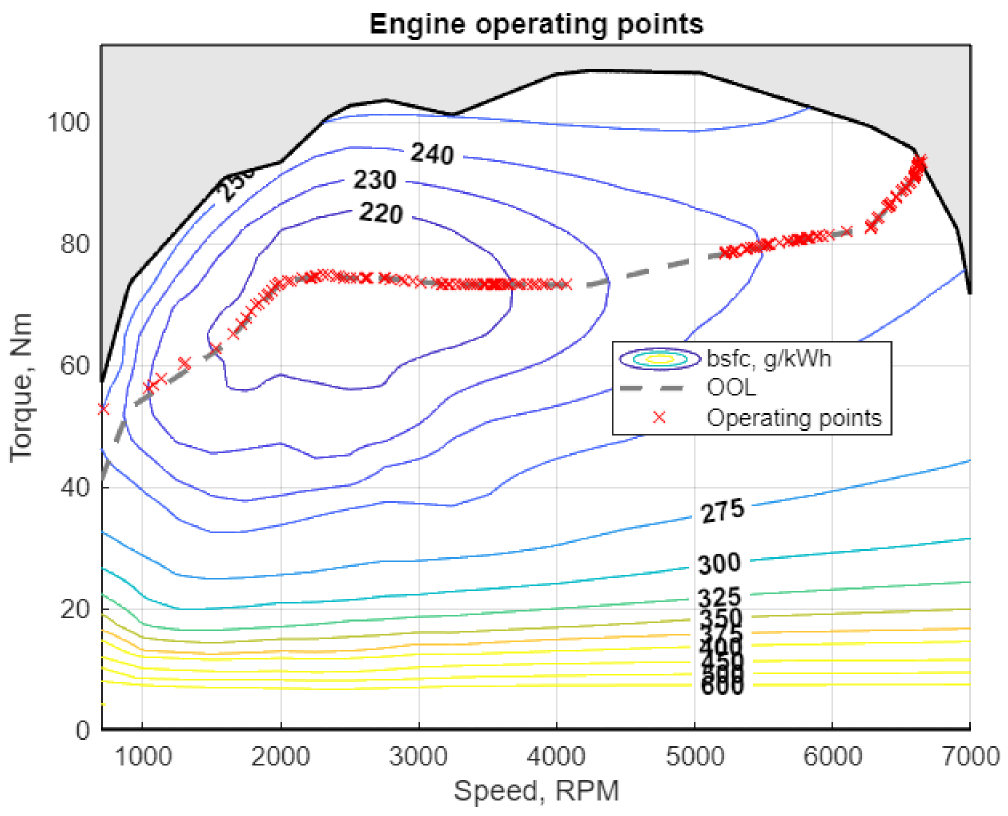
Overall, the fuel consumption trend increases proportionally to engine power. Notably, the highway phase has a significant impact, accounting for approximately 65% of the total fuel consumption during the mission. This underscores the importance of optimizing the controller under such conditions to achieve substantial fuel savings.

However, the total fuel consumption remains below one kilogram (902 g), indicating that the controller effectively manages fuel resources across the varied driving conditions of the WLTP cycle.

BSFC engine map

The following plot presents the Brake Specific Fuel Consumption (BSFC) engine map, where the working points throughout the entire cycle are displayed. This provides a clear representation of how the controller operates and the strategies adopted to achieve the goal of minimizing fuel consumption.

```
engMapWithPF(vehData.eng,prof,"bsfc");
```



It is evident that the engine does not operate solely at the two predefined working points set in the controller. Instead, multiple operating points are distributed across a wide range of engine speeds and torque levels to accommodate varying power demands, particularly when constraints such as battery limitations are reached.

As explained in the section on the thermostat controller, the engine primarily operates along the Optimal Operating Line (OOL), ensuring maximum efficiency. Notably, the majority of operating points fall within the region where BSFC is below 220 g/kWh, effectively minimizing fuel consumption.

However, a significant number of points are located in the high engine speed region, where increased power demand leads to higher BSFC values, contributing substantially to overall fuel consumption.

Overall, two distinct operating zones can be identified:

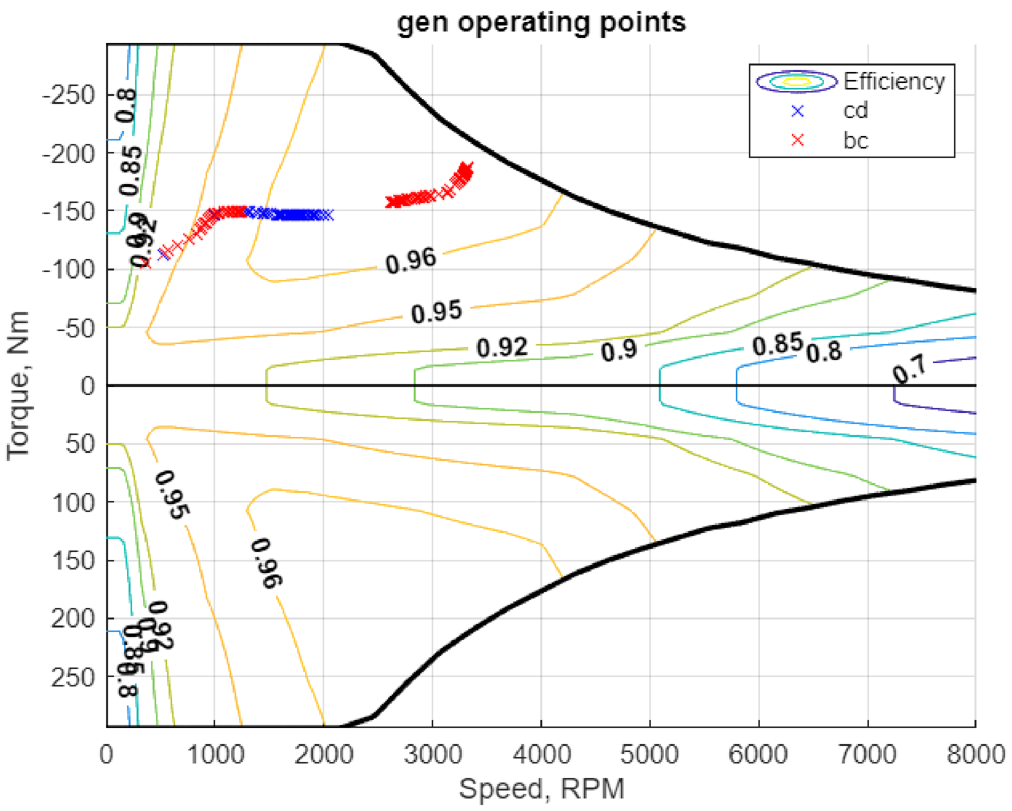
1. One around the optimal power region, where efficiency is maximized.
2. One near maximum power, where fuel consumption increases.

Optimizing the controller to utilize mid-range engine speeds could further reduce fuel consumption, enhancing overall efficiency.

Generator efficiency map

In the following plot, the generator efficiency map is presented, with all the working points plotted. It is possible to distinguish two operational modes: **battery charging** (**bc**) and **charge depleting** (**cd**), which uses the engine to not exceed the battery limits.

```
emMapWithPF(vehData.gen,prof,"gen","all");
```

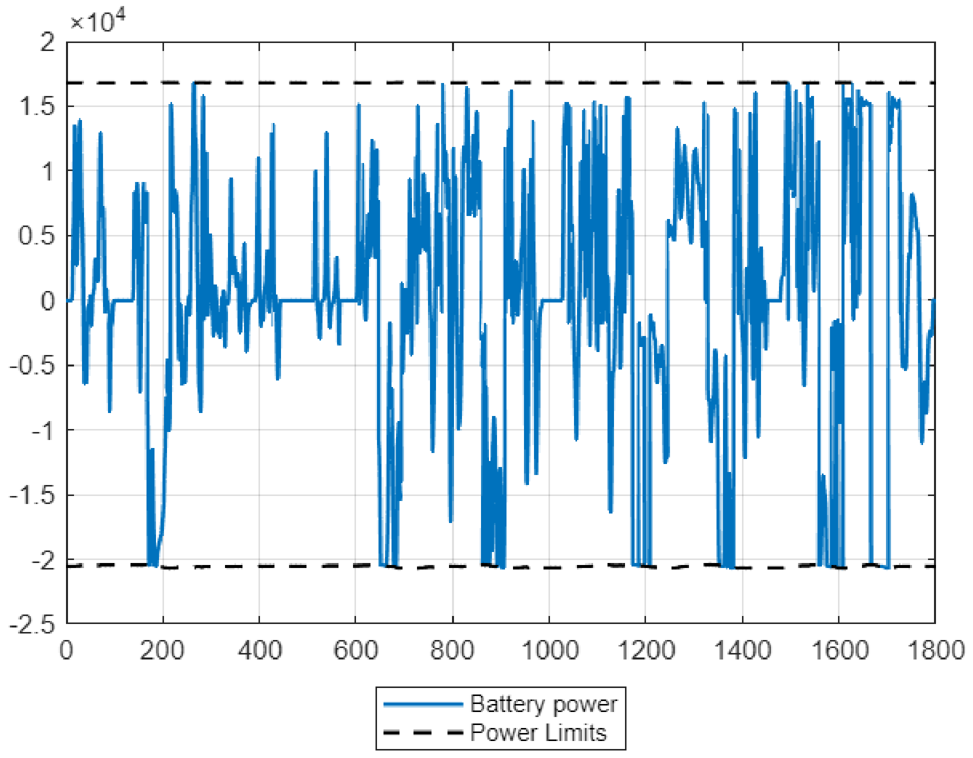


Generally, the generator operates within the maximum efficiency zone (`effGen = 0.96`). However, there is a non-negligible number of points within lower efficiency zones. These could be moved to higher efficiency zones by limiting the engine's operation at low speeds, thereby improving the overall efficiency of the system.

Battery power vs Time

The next plot illustrates the power supplied to and drawn from the battery. This analysis was conducted to evaluate its behavior, identify potential issues, and verify whether the battery limits specified in the datasheet (`Pb,min` and `Pb,max`) were respected.

```
%Battery power - Time plot
prof_struct = structArray2struct(prof);
figure()
plot(time_vector,prof_struct.battVolt.*prof_struct.battCurr,Color =
"#0072BD",LineWidth=1.5)
hold on
plot(time_vector,vehData.batt.maxPwr(SOC(1:length(time_vector))),"k--",LineWidth=
1.5)
plot(time_vector,vehData.batt.minPwr(SOC(1:length(time_vector))),"k--",LineWidth=
1.5)
grid on
legend("Battery power", "Power Limits","Location","southoutside")
```

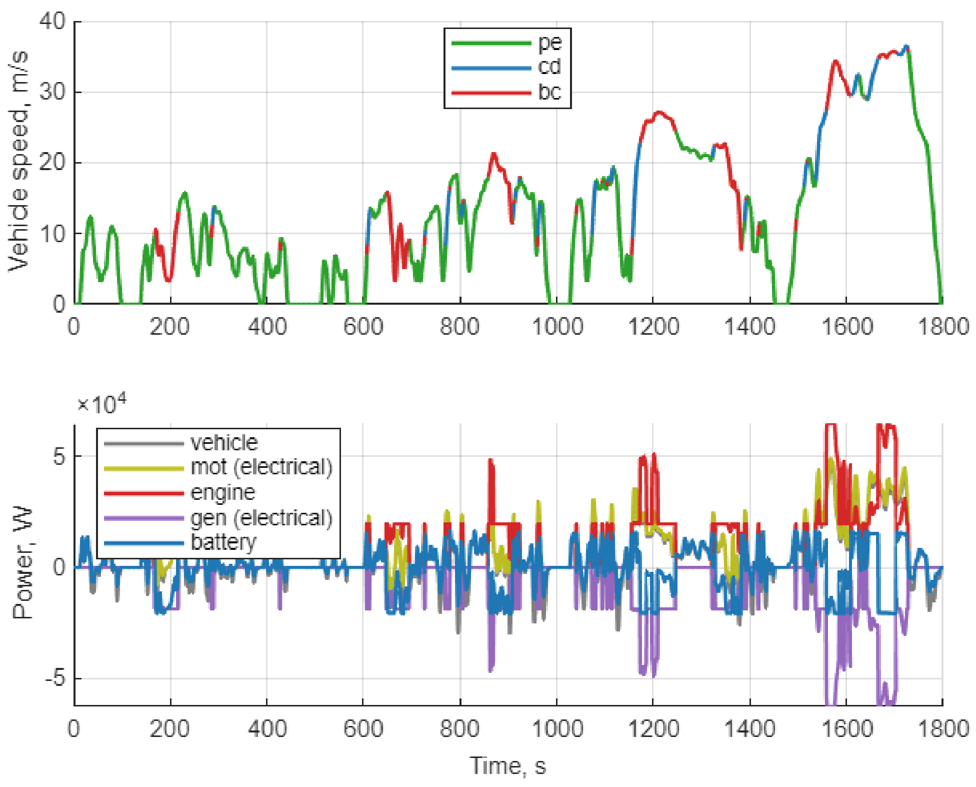


Analyzing the battery's behavior, it is evident that power peaks are significantly present during both charging and discharging conditions, which consequently lead to very high current peaks. This operating condition, particularly during charging, accelerates the chemical degradation of the battery, contributing to its rapid aging. To preserve the battery and enhance its durability, it is strongly recommended to limit these peaks and prioritize a constant power operating condition.

Power profiles

Next, the power profiles of the all mechanical organs are plotted and are related to the velocity profile of the vehicle.

```
powerProfiles(prof,"all");
```



In the first figure, a vehicle speed-time diagram illustrates the different **powertrain operating modes**: **PE** (pure electric), **CD** (charge depleting = engine + eMotor), and **BC** (battery charge). The controller aims to **maximize** the use of the **electric mode**, which generally has higher efficiency, to minimize fuel consumption. As a result, the powertrain primarily operates in pure electric mode during discharge, minimizing engine usage whenever possible.

A similar trend is observed in the second figure, which illustrates the power contributions of all powertrain components. This visualization serves as a valuable tool for understanding SOC behavior by analyzing component interactions. For instance, peaks exceeding SOC limits can be attributed to regenerative braking, as they coincide with both negative acceleration and negative battery power at that time.

Thermostat controller improved

In this second part of the report, an **advanced thermostat controller** is implemented. The objective of the improved controller is to **enhance battery longevity** and optimize its utilization, particularly during charging.

Simulation loop

The simulation parameters and implementation are the same used in paragraph Simulation loop.

```
%Initialating SOC and engineState
SOC_opt(1) = 0.6;
engState_opt(1) = 0; % engine is off


%Setup for simulation
for n = 1:length(time_vector)

    [engSpd_opt(n), engTrq_opt(n),engState_opt(n+1)] = ...
        thermostatControl_opt(SOC_opt(n),
engState_opt(n),vehSpd(n),vehAcc(n),vehData);

    [SOC_opt(n+1), fuelflwRate_opt(n), unfeas_opt(n), prof_opt(n)] = ...
        hev_model(SOC_opt(n), [engSpd_opt(n), engTrq_opt(n)], [vehSpd(n),
vehAcc(n)], vehData);
end
```

Post-Processing: main results

As previously done in paragraph post-processing, two metrics are used to evaluate the controller's performance and then discussed in the successive paragraph comparison:

- `fuelConsumption`, which effectively confirms the proper operation of the controller.

```
prof_struct_opt = structArray2struct(prof_opt);

fuelConsumption_opt = trapz(time_vector,prof_struct_opt.fuelFlwRate)/1000;
fprintf("Fuel consumption: %.3f kg\n", fuelConsumption_opt);
```

```
Fuel consumption: 0.881 kg
```

- `fuelEconomy`.

```
distance = trapz(time_vector,vehSpd)/1000;
fuel_rho = vehData.eng.fuelDensity; %[Kg/l]
fuelEconomy_opt = (fuelConsumption_opt/(fuel_rho*distance))*100;
fprintf("Fuel economy: %.3f l/100km\n", fuelEconomy_opt );
```

```
Fuel economy: 4.823 l/100km
```

Also the `finalSOC` at the end of the mission is reported.

```
finalSOC_opt = SOC_opt(end);
fprintf("Final SOC: %.3f\n", finalSOC_opt );
```

```
Final SOC: 0.707
```

Save results

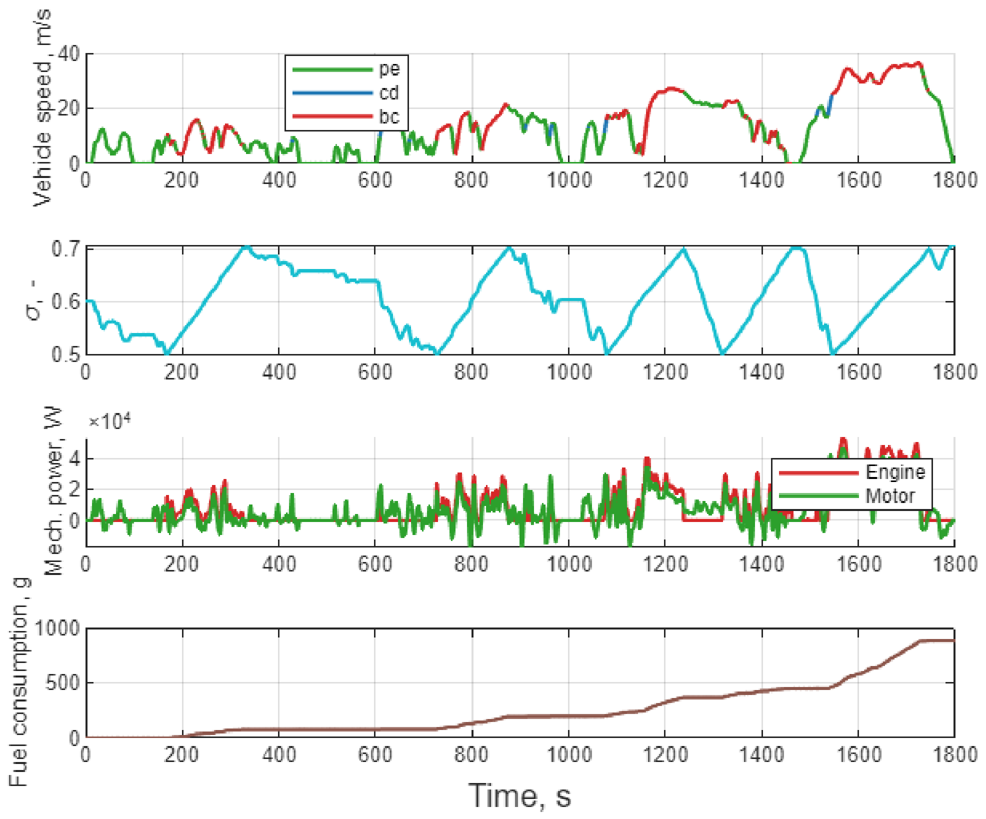
The obtained results are saved in a .mat file.

```
save("results_extra.mat", "prof_opt", "fuelConsumption_opt", "fuelEconomy_opt",
"finalSOC_opt")
```

Post-Processing: plots and analysis

A detailed vision of how the controller works is provided in the following plot

```
mainProfiles(prof_opt);
```



Plot 1: Vehicle Speed vs Time

During the **urban phase (0–600s)**, the vehicle primarily operates in pure electric mode, except during the first charging cycle. Overall, this phase is characterized by low velocity and, consequently, low power demand.

In the **extra-urban phase (600–1200s)**,vehicle speeds increase and stabilize, leading to a combination of electric and engine-supported operation. A higher frequency of charging cycles is observed, corresponding to an increase in average velocity and higher acceleration demands.

A substantial difference emerges when compared to the standard controller. Specifically, the charge-depleting (CD) phase is significantly reduced, as high power demand peaks are managed during charging cycles. Generally, due to the extended duration of charging cycles, it is more likely that high power peaks are handled within these cycles, which, although not explicitly considered in this simulation, can contribute to reducing overall energy consumption.

With the standard controller, power peaks are managed by switching the engine ON to supply the required power while ensuring that battery power limits are not exceeded. Consequently, the engine may not operate at its optimal thermal state because during the OFF state the temperature decrease. In contrast, handling these peaks during battery charging enables the engine to be utilized when it is already warmed up, thus improving its efficiency.

In the **highway phase (1200–1800s)**, vehicle speed and acceleration are higher, leading to greater battery energy depletion and more frequent charging cycles. As previously mentioned, the charge-depleting zones are significantly reduced and replaced by battery charging phases, where the motor operates with higher efficiency.

Plot 2: State of Charge vs Time

This plot illustrates the evolution of the State of Charge (SOC), highlighting the charging and discharging trends within the defined limits of 0.5 and 0.7.

The initial condition is SOC = 0.6, indicating that 60% of the battery's energy is available.

As discussed in the section dedicated to the controller, battery **charging is performed at constant power** and, consequently, at constant current. This is evident in the SOC profile, where the charging phases exhibit an almost **linear trend**. As shown in the equation below, SOC is directly proportional to the current

$$\sigma = -\int \frac{i_{batt}}{Q} dt = \frac{i_{batt}}{Q} \cdot \Delta t + \sigma_0$$

Overall, the sustaining mode is effectively maintained, as the SOC consistently oscillates between the upper and lower thresholds, performing the complete charging or discharging into the thresholds limits.

Plot 3: Mechanical Power vs Time

This plot illustrates the mechanical power output of the electric motor (green line) and the internal combustion engine (ICE) (red line).

During the **urban phase**, power is predominantly supplied by the electric motor, while the **ICE remains off** or is rarely activated (only once over 600 s), as previously analyzed. As the cycle progresses, the engine is engaged more frequently and operates at higher power levels due to the characteristics of the WLTP cycle.

Overall, the **engine is not utilized under high-power conditions**, suggesting potential opportunities for engine downsizing in the design phase or a revision of the control strategy to fully exploit the engine's capabilities.

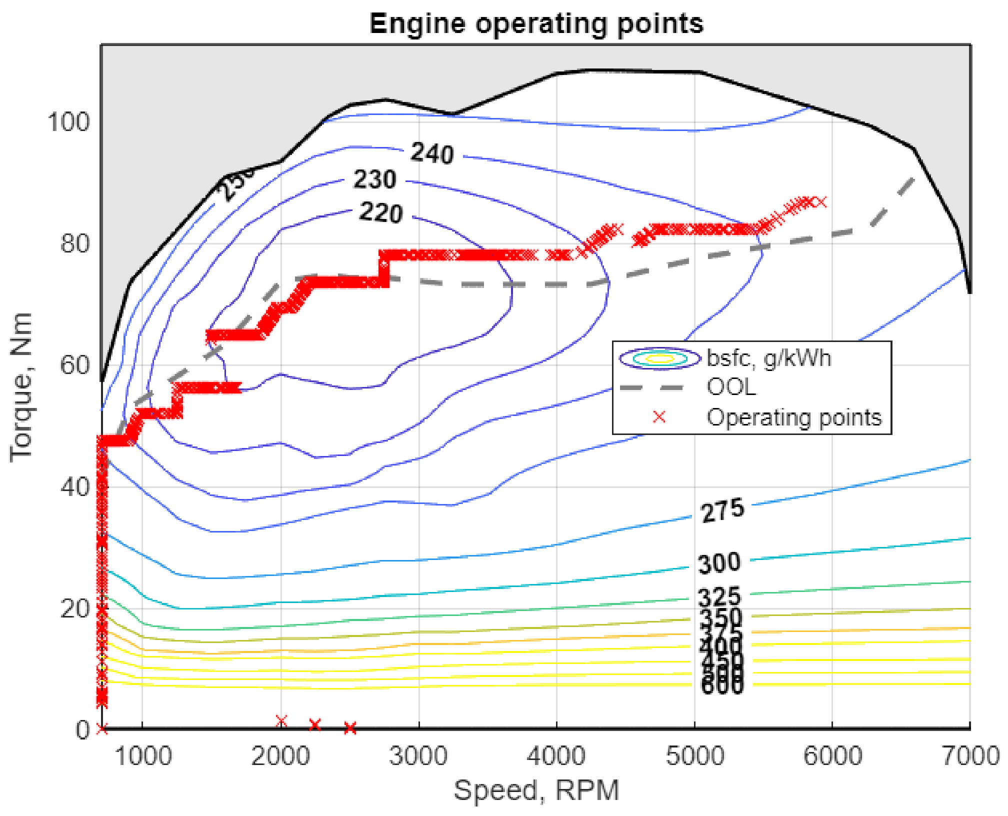
Plot 4: Fuel consumption vs Time

Overall, fuel consumption increases proportionally with engine power. As discussed in the paragraph on the standard controller, the **highway phase** plays a significant role, contributing approximately **63% of the total fuel consumption** during the mission.

Despite this, total fuel consumption remains below one kilogram (881 g), demonstrating that the controller efficiently manages fuel resources across the diverse driving conditions of the WLTP cycle.

BSFC engine map

```
engMapWithPF(vehData.eng,prof_opt,"bsfc");
```



As in the first case, the engine operates at multiple points to meet the power demand while remaining in the most efficient region of the map.

The main difference in this scenario is that the **operating points** do **not always align with the OOL**. This is primarily due to the interpolation method, which introduces step-like behavior, mainly because of its linear nature and the way it is constructed.

Notably, the **engine never operates at full power,** indicating that its potential is not fully exploited.

Another important observation is that many **operating points** that occur at **idle** speed (700 RPM) **with high torque may not correspond to optimal efficiency**.

Something similar happens to few points that works at high speed and very low torque, due to low power demand, in a very low efficiency region.

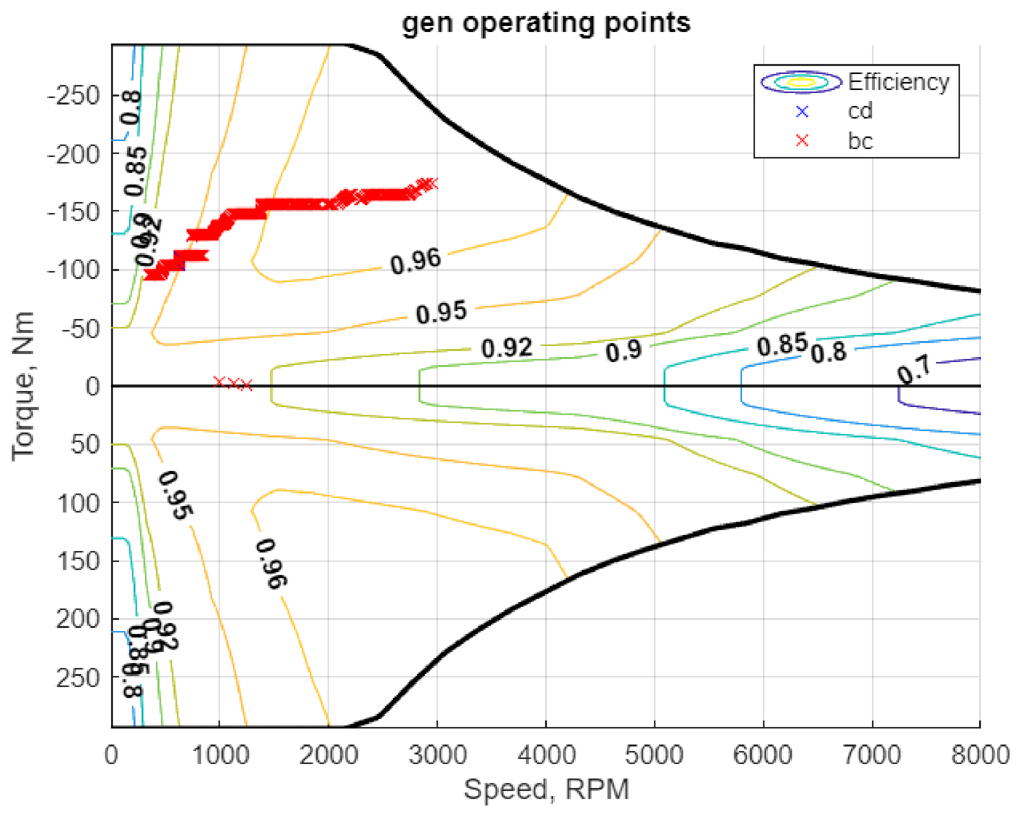
These problems suggest the need for further investigation, to understand why sometimes that happen.

Optimizing the controller to fix these operating points and the non-fully-exploited engine could further increase the overall efficiency and reduce the fuel consumption.

Efficiency generator Map

In the following plot, the generator efficiency map is presented, with all the working points plotted. It is possible to distinguish two operational modes: **battery charging** (**bc**) and **charge depleting** (**cd**), which uses the engine to not exceed the battery limits.

```
emMapWithPF(vehData.gen,prof_opt,"gen","all");
```



The generator primarily operates in the highest efficiency region; however, many points fall into lower efficiency areas. This is mainly due to operation at idle or, more generally, under low power demand conditions.

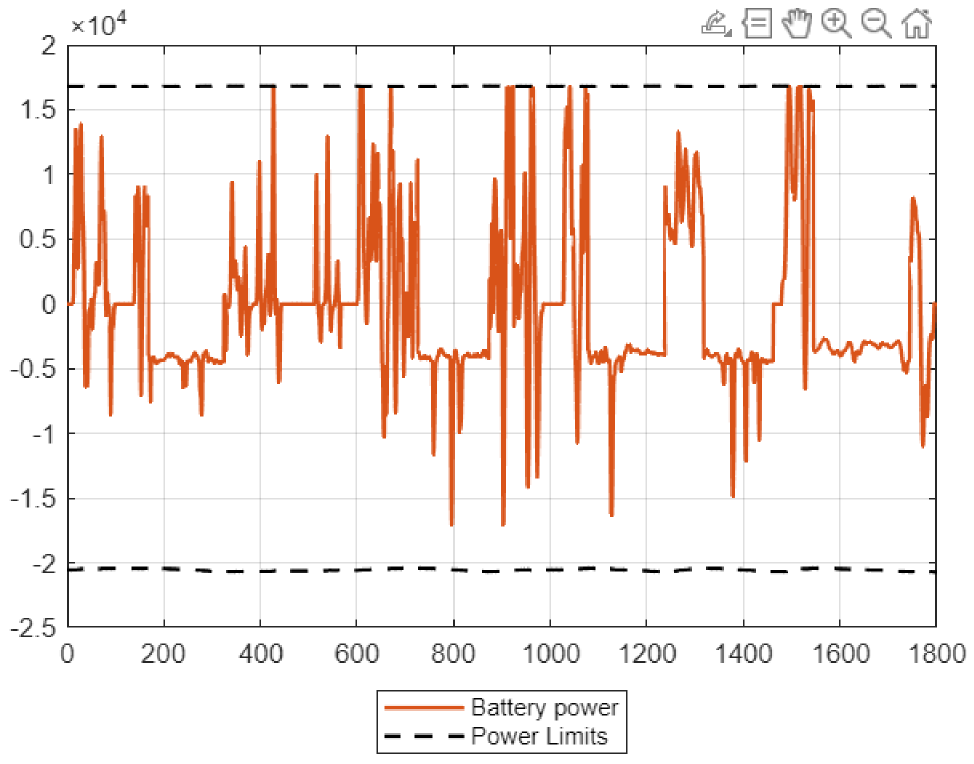
Some improvements could be made by limiting the low power operating points, moving them at higher power.

Battery power vs Time

The following plot illustrates the power supplied to and drawn from the battery. This analysis was conducted to assess battery performance, identify potential issues, and verify compliance with the specified limits as defined in the datasheet.

```
%Battery power - Time plot
figure()
```

```
plot(time_vector,prof_struct_opt.battVolt.*prof_struct_opt.battCurr,Color =
"#D95319",LineWidth=1.5)
hold on
plot(time_vector,vehData.batt.maxPwr(SOC_opt(1:length(time_vector))),"k--",LineWi
dth=1.5)
plot(time_vector,vehData.batt.minPwr(SOC_opt(1:length(time_vector))),"k--",LineWi
dth=1.5)
grid on
legend("Battery power", "Power Limits","Location","southoutside")
```



In discharging conditions, the power exhibits frequent peaks, particularly during the first to middle part of the mission. Additionally, the output power limit is reached several times, indicating a stressful usage under certain conditions.

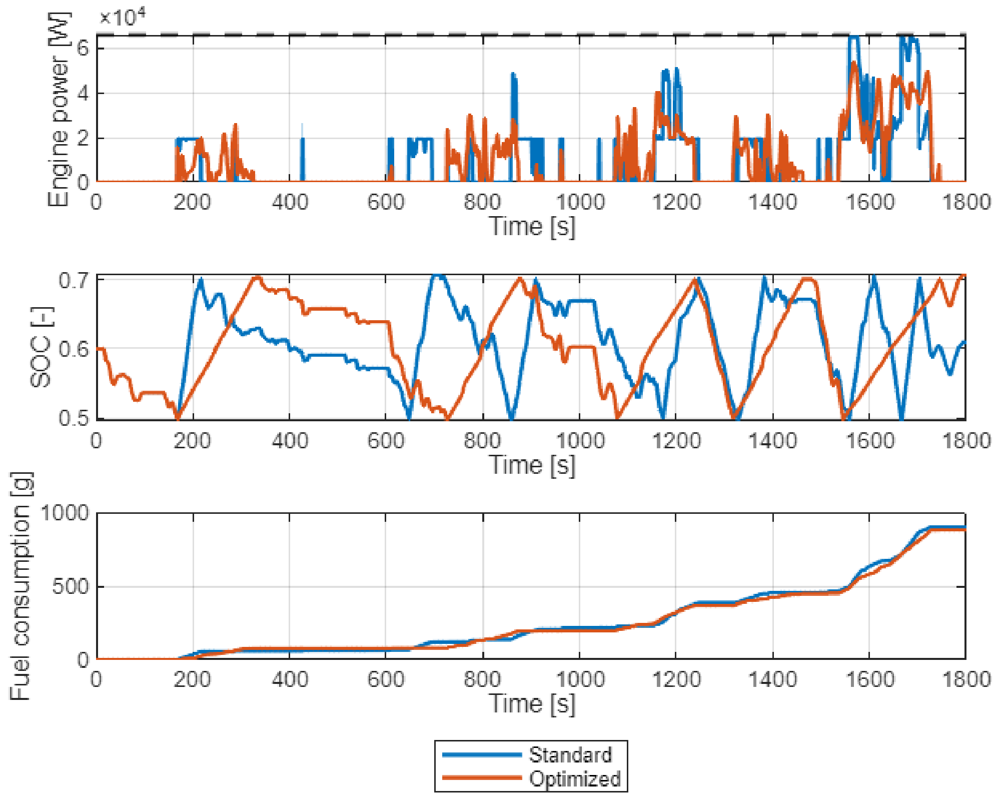
In charging conditions same peaks are still present due to regenerative braking of the tractive motor but the overall charging cycle is performed at constant power. This approach enhances the reliability and lifespan of the battery, which is one of the primary objectives of this controller.

Comparison between the two versions

The following plot compares the engine power, State of Charge (SOC), and fuel consumption between the two versions.

```
figure()
tloy = tiledlayout("vertical");

nexttile
plot(time_vector,prof_struct.engSpd.*prof_struct.engTrq,LineWidth=1.5)
```

```
hold on
plot(time_vector,prof_struct_opt.engSpd.*prof_struct_opt.engTrq,LineWidth=1.5)
yline(vehData.eng.maxPwr,"k--",LineWidth=1.5)
grid on
xlabel("Time [s]")
ylabel("Engine power [W]")


nexttile
plot(time_vector,prof_struct.battSOC,LineWidth=1.5)
hold on
plot(time_vector,prof_struct_opt.battSOC,LineWidth=1.5)
grid on
xlabel("Time [s]")
ylabel("SOC [-]")


nexttile
plot(time_vector,cumtrapz(time_vector, prof_struct.fuelFlwRate),LineWidth=1.5)
hold on
plot(time_vector,cumtrapz(time_vector,
prof_struct_opt.fuelFlwRate),LineWidth=1.5)
grid on
xlabel("Time [s]")
ylabel("Fuel consumption [g]")
lg = legend("Standard", "Optimized");
lg.Layout.Tile = 'South';
```



Starting with **engine power**, the standard version uses the engine more frequently than the optimized version, but for shorter durations. Additionally, in the final part of the mission, the standard version utilizes the maximum engine power, while the optimized version reaches a maximum value of 53 kW, 13 kW below the limit. However, operating at a lower power level than the maximum enhances fuel consumption by working at a lower BSFC value.

When analyzing the **SOC** of both versions, it is evident that the optimized version performs fewer recharge cycles than the standard one throughout the mission, due to longer recharging times. It is also important to note that the slopes of the SOC differ between the two versions when charging occurs. The slope is directly proportional to the C-rate, and a higher C-rate, as seen in the standard controller, results in a steeper slope compared to the optimized version, where the C-rate is limited to 5. Furthermore, the final SOC values differ between the two controllers. The standard version ends with a SOC of 0.61, while the optimized version reaches 0.71.

Regarding **fuel consumption**, the optimized controller demonstrates a slightly lower consumption:

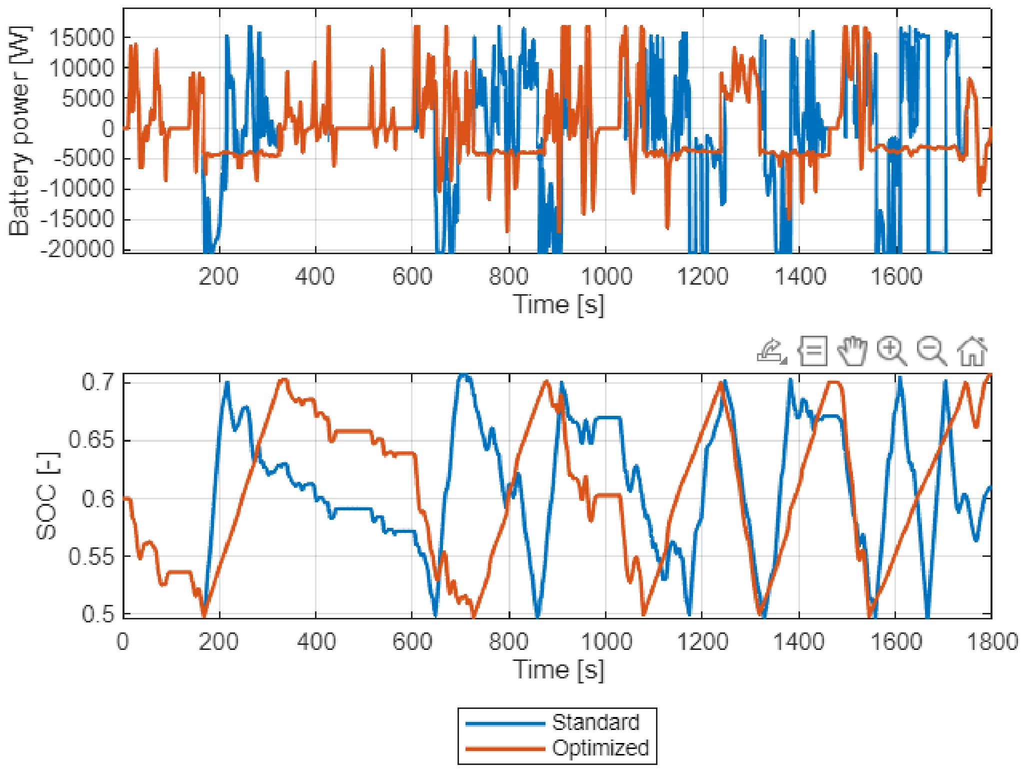
- Standard controller: 902 g
- Optimized controller: 881 g

Although the primary goal of the optimized controller is to enhance the battery's lifespan, a small improvement in fuel consumption is observed. The fuel consumption trends are generally similar until the start of the highway phase of the mission (0–1200 s). After this point, the standard controller tends to operate the engine at higher power levels, whereas the optimized controller stays at lower power levels, never reaching the maximum. This leads to a minor reduction in consumption.

A factor that must be considered is that the final SOC of the two versions differs, with the optimized version having a higher SOC. This indicates a considerable amount of energy available for further use, which may contribute to the efficiency of the optimized controller in the long term.

```
figure()
tloy =tiledlayout("vertical");

nexttile
plot(time_vector,prof_struct.battVolt.*prof_struct.battCurr,LineWidth=1.5)
hold on
plot(time_vector,prof_struct_opt.battVolt.*prof_struct_opt.battCurr,LineWidth=1.5
)
grid on
xlabel("Time [s]")
ylabel("Battery power [W]")

nexttile
plot(time_vector,prof_struct.battSOC,LineWidth=1.5)
hold on
plot(time_vector,prof_struct_opt.battSOC,LineWidth=1.5)
grid on
xlabel("Time [s]")
```

```
ylabel("SOC [-]")

lg = legend("Standard", "Optimized");
lg.Layout.Tile = 'South';
```



Comparing the power at the battery output between the two versions, the differences outlined in the previous paragraphs become evident. The key distinctions between the two versions are primarily observed in the charging phases. The standard controller exhibits several peaks with higher frequency, frequently reaching the battery's limit. In contrast, the optimized controller never reaches the limit and instead performs the charging phase at constant power, as demonstrated in the SOC graph below.

In discharging conditions, a similar pattern is observed, with the optimized controller showing less frequent peaks and an average power value lower than that of the standard controller.

The thermostat controller function

In the context of a series hybrid configuration, a **rule-based controller** is designed to manage the engine responsible for charging the battery and supplying the necessary power to the motor. The controller operates in **two layers**: the first layer verifies the State of Charge (SOC), while the second layer assesses the power demand from the tractive electric motor. The controller function receives the following **inputs**:

- **`SOC`** from the previous iteration, which is compared with predefined thresholds to determine whether the engine should be activated;
- **`last_eng_State`**, indicating the engine status from the previous iteration, used to assess whether a charging cycle is in progress and whether charging or discharging is required within the defined thresholds;
- **`vehSpeed`**, the vehicle speed at a given moment;
- **`vehAcc`**, the vehicle's acceleration at the same instant. These inputs, together with the speed, are fed into the `hev_drivetrain` function, which outputs the shaft speed and torque of the electric motor, enabling the calculation of the power demand (`PwrDemand`).
- **`vehData`** structure encompasses all relevant vehicle and mechanical data.

The function **outputs** the following control variables:

- **`engSpeed`**: The speed at which the engine should operate during the current iteration.
- **`engTorque`**: The torque at which the engine should operate during the current iteration.
- **`eng_State`**: The engine state, which is determined based on the current iteration to indicate whether the engine is on, off, or operating in a particular mode.

Controller logic

The hybrid vehicle operates in **charge-sustaining mode** due to the limited capacity of the battery. Therefore, the **first layer** of the controller performs a check on the State of Charge (**SOC**) to ensure it stays within predefined thresholds. The upper limit of the SOC is set to **`0.7`**, while the lower limit is set to **`0.5`**. This ensures that the battery charge is maintained within the required range, preventing overcharging or excessive depletion.

In the first control rule, the engine is activated (**`eng_State = 1`**) if at least one of the following conditions is met:

- The **SOC** falls below the minimum threshold of **0.5**.
- The **SOC** is below the upper threshold (**0.7**) *and* the engine was already ON in the previous iteration (**`last_eng_State = 1`**).

If neither condition is satisfied, the engine remains OFF (**`eng_State = 0`**).

The primary objective of the controller is to enhance fuel efficiency by minimizing consumption, therefore in the second rule the controller have to define the operating point of the engine, to this scope two points are previously selected:

- **Optimal operating point**: is the point on the OOL that exploit the minimum bsfc.
- **Maximum power operating point**: is the working point that exploit maximum engine power.

Considering the engine state (`eng_State`) and the power requested by the motor (`PwrDemand`), the controller determines the target engine power (`engPwrCmd`). If the first rule specifies that the engine should be turned on and the power demand is lower than the optimal engine power (`engPwrOpt`), the controller sets `engPwrCmd` to this optimal value. Otherwise, if the engine is turned on in the first case but the power demand exceeds `engPwrOpt`, the operating point is set to the maximum power (`engPwrMax`). In all other cases, the engine power command is set to zero.

Battery power constraints

Discharging power limit

If, at the DC bus equilibrium, the power supplied by the battery exceeds the discharge power limit, the *engPwrCmd* value is adjusted as follows to ensure the battery power remains in its limit:

$$engPwrCmd = \frac{1}{\eta_{gen}} \cdot (PwrDemand - P_{batt,max})$$

Where $P_{batt,max}$ represents the maximum battery output power, which depends on the state of charge and is retrieved from the provided 1-D LookUp Table.

If the adjusted engine power command falls below the optimal power level, the engine and generator would operate at low efficiency. To prevent this, the operating point is set to the optimal power, with any excess energy used to recharge the battery. This strategy aims to minimize fuel consumption.

Charging power limit

If, at the DC bus equilibrium, the power supplied to the battery exceeds the charge power limit, the *engPwrCmd* value is adjusted as follows to ensure the battery power remains in its limit:

$$engPwrCmd = \frac{1}{\eta_{gen}} \cdot (PwrDemand - P_{batt,min})$$

Where $P_{batt,min}$ represents the maximum battery input power, which depends on the state of charge and is retrieved from the provided 1-D LookUp Table.

If the newly computed `engPwrCmd` value falls below the minimum exploitable power of the OOL, it is set to zero. This scenario likely occurs during regenerative braking, where the motor is already charging the battery. In such conditions, operating the engine at low power levels is inefficient due to the reduced efficiency of both the engine and the generator. This strategy aligns with the primary objective of minimizing fuel consumption.

Engine torque and speed

Given the controller's objective of minimizing fuel consumption throughout the mission, the control variables (`engSpeed` and `engTorque`) are determined based on the Optimal Operating Line (OOL), which theoretically represents the points of maximum efficiency for each engine speed. Going deep into the procedure:

- If `engPwrCmd` is greater than zero, it is used as an input to the `vehData.eng.oolSpd` Look-Up Table, which returns the corresponding speed (`engSpeed`) on the OOL for the target power. Once `engSpeed` is determined, `engTorque` is computed using the relation:

$$engTorque = \frac{engPwrCmd}{engSpeed}$$

- If `engPwrCmd` is zero and a charging cycle is active (`engState = 1`), the engine is set to idle mode (`engSpeed = idle, engTorque = 0`) to remain ready for the next simulation step.
- Otherwise, the engine is switched off (`engSpeed = 0, engTorque = 0`) to avoid unnecessary fuel consumption.

```
function [engSpeed, engTorque, eng_State] = ...
    thermostatControl(SOC, last_eng_State,vehSpeed,vehAcc,vehData)


%Define threshold of SOC

SOChi = 0.7; %Maximum SOC threshold
SOClo = 0.5; %Minimum SOC threshold


%Define a fix efficiency point of the generator
effGen = 0.9; %Corresponds to one of the worst efficiency point
              %of the generator, selected to be conservative



%Define the optimal engine operating point to minimize the bsfc
engSpd_vect = linspace(vehData.eng.idleSpd,vehData.eng.maxSpd,1000); %vector of
% engine speed which contains all the points from idle speed to the maximum

engOOLTrq_vect = vehData.eng.oolTrq(engSpd_vect); %For each point of the
       % engine speed vector it extract the correspondent torque on the OOL

bsfcOOL_vector = vehData.eng.bsfcMap(engSpd_vect,engOOLTrq_vect); % Extract the
                    % bsfc value referred to each point of speed and torque
vector

% The following two lines evaluate the speed and the torque which
% corresponds to the engine's optimal operating point for the minimum bsfc.
% The function FIND is used to extract the index of the minimum bsfc on the
% bsfcOOL_vector, at which corresponds the value of optimal engine speed
% and torque on engSpd_vect and engOOLTrq_vect.
engSpdOpt = engSpd_vect(find(bsfcOOL_vector == min(bsfcOOL_vector(:)),1));
engTrqOpt = engOOLTrq_vect(find(bsfcOOL_vector == min(bsfcOOL_vector(:)),1));

engPwrOpt = engSpdOpt*engTrqOpt; %Engine power at the optimal operating point

engPwrOOLMin = min(engSpd_vect.*engOOLTrq_vect); % Compute the minimum engine
    %power that can be providded using the OOL points

genSpdOpt = (1/vehData.gen.tcSpdRatio)*engSpdOpt; %Generator speed at
                                                  %optimal operating point
```

```
genTrqOpt = -1*vehData.gen.tcSpdRatio*engTrqOpt; %Generator torque at
                                                 %optimal operating point
genPwrOpt = vehData.gen.effMap(genSpdOpt,genTrqOpt)*engPwrOpt; %Generator power
%                                                   at optimal operating point



%Define the maximum engine operating point and generator power

engTrqMax_vector = vehData.eng.maxTrq(engSpd_vect); %For each point of
% the engine speed vector it extract the correspondent maximum torque


engPwrMax_vector = engSpd_vect.*engTrqMax_vector; %Define the vector which
            % contains the maximum power for each speed and torque vector

engPwrMax = max(engPwrMax_vector); %The maximum power erogated by the engine




%Phase 1: Check if SOC is in between thresholds

%In this section the first rule is implemented.
% The engine state from the previous cycle (last_eng_State) and the SOC are
% monitored to determine whether the engine should be turned on. The engine
% is activated if at least one of the following conditions is met: 1. The
% SOC is below the minimum SOC threshold. 2. The engine state in the
% previous cycle was ON, and the SOC is below
%    the maximum SOC threshold.
% If neither condition is met, the engine is turned off.

if or( SOC<=SOClo, and(last_eng_State > 0, SOC<SOChi))
    eng_State = 1; %Engine ON
else
    eng_State = 0; %Engine OFF
end



%PHASE 2: Assess the demanded Power and then choose if the engine is in PwrOpt or
%PwrMax

% Torque and speed of motor shaft are computed using the function
% hev_drivetrain
[shaft_speed, shaft_torque] = hev_drivetrain(vehSpeed, vehAcc, vehData);


% The power demanded by the traction electric motor is computed in two
% different ways: 1. In traction mode (torque ≥ 0), the power requested at
% the shaft is
%    divided by the efficiency to account for electromagnetic losses during
%    the conversion from electrical to mechanical energy.
% 2. In regenerative braking mode (torque < 0), the efficiency is
% multiplied
%    by the power requested at the shaft, as the energy flow is from the
%    wheels to the battery.
if shaft_torque >= 0
   PwrDemand = shaft_speed*shaft_torque*(1/
vehData.mot.effMap(shaft_speed,shaft_torque));
else
   PwrDemand =
shaft_speed*shaft_torque*(vehData.mot.effMap(shaft_speed,shaft_torque));
end


% The target engine power (engPwrCmd) is initialized to 0.
engPwrCmd = 0;

% This section implements the second rule of the controller. If the engine
% needs to be turned on (eng_State = 1), the controller selects one of two
% operating points: 1. If the power requested by the traction motor is less
% than or equal to
%    the power supplied by the generator when the engine operates at its
%    optimal point, the engine power command is set to the optimal point.
% 2. Otherwise, if the requested power exceeds the optimal level,
%    the engine operates at maximum power.
if PwrDemand <= genPwrOpt && eng_State == 1
    engPwrCmd = engPwrOpt;

elseif PwrDemand > genPwrOpt && eng_State == 1
    engPwrCmd = engPwrMax;
end


% PHASE 3: BATTERY POWER LIMITS In this section, two constraints are set to
% ensure that the charging and discharging power limits are not exceeded.

% Constraint on discharging: If the power requested from the battery
% exceeds the limit, the engine provides the necessary power to keep the
% battery within its limits. Additionally, another condition is applied: If
% the power requested from the engine is lower than the optimal power and
% the SOC is lower than the higher SOC threshold, the engine operating
% point is set to the optimal level, and any excess power is used to
% recharge the battery. This is done to increase the of the controller
% strategy and decrease the consumption.

if engPwrCmd < (1/effGen) * (PwrDemand - vehData.batt.maxPwr(SOC))
    engPwrCmd = (1/effGen) * (PwrDemand - vehData.batt.maxPwr(SOC));
    if engPwrCmd<engPwrOpt && SOC<=SOChi
        engPwrCmd = engPwrOpt;
    end

% Constraint on charging: If the charging power supplied to the battery
% exceeds the limit, the engine power is reduced to stay within the limits.
% Additionally, if the engine power command is lower than the minimum power
% value on the OOL, it is probably that the motor is in regenerative
% braking. In this case, the engine is turned off to reduce fuel
```

```matlab
% consumption.
elseif engPwrCmd >= (1/effGen) * (PwrDemand - vehData.batt.minPwr(SOC))...
        && eng_State == 1
    engPwrCmd = (1/effGen) * (PwrDemand - vehData.batt.minPwr(SOC));
    if engPwrCmd<engPwrOOLMin
        engPwrCmd = 0;
    end
end




% ENGINE POWER LIMITS

% Maximum value on the is limited to the maximum power that can be erogated
% by the engine
engPwrCmd = min(engPwrCmd,vehData.eng.maxPwr);

% Negative values of engine power are non-physical; therefore, the minimum
% value of the engine power command is limited to 0.
engPwrCmd = max(engPwrCmd,0);





% PHASE 4: ENGINE SPEED AND TORQUE

% In this phase, the controller outputs are computed based on the engine
% power command. If the target power is greater than 0, the engine speed is
% calculated using the OOL 1-D lookup table (vehData.eng.oolSpd), with the
% engine power command as input. The torque is then computed as Power /
% Speed. If the engine power command is equal to 0, two conditions are
% considered: 1. If a charging cycle is running (eng_State = 1), the engine
% is set to idle
%    condition and is not turned off, so it is ready for the next cycle.
% 2. Otherwise, in all other cases, the engine is turned off.
if engPwrCmd > 0
    engSpeed = vehData.eng.oolSpd(engPwrCmd);
    engTorque = engPwrCmd/engSpeed;
elseif engPwrCmd == 0 && eng_State == 1
    engSpeed = vehData.eng.idleSpd;
    engTorque = 0;
else
    engSpeed = 0;
    engTorque = 0;
end





% CONSTRAINTS ON ENGINE SPEED AND TORQUE

% The minimum value of engine torque is set to 0 because negative values
% are non-physical.
engTorque = min(engTorque,vehData.eng.maxTrq(engSpeed));
```

```
% In case of the computed engine speed is lower than engine speed and
% torque is higher than 0 (So, not in idle condition set by the controller)
% new values of engine speed and torque are computed trying to minimizing
% the bsfc for the requested power. To do that vector of toque values is
% computed dividing the engine power command by the engine speed vector
% (swipe from idle to maximum speed).
if engSpeed < vehData.eng.idleSpd && engTorque > 0
    engTrq_vect = engPwrCmd./engSpd_vect;
    for l = 1:length(engSpd_vect)
        if engTrq_vect(l)<=vehData.eng.maxTrq(engSpd_vect(l))
            bsfc_vect(l) = vehData.eng.bsfcMap(engSpd_vect(l),engTrq_vect(l));
        else
            bsfc_vect(l) = NaN;
        end
    end
    index = find(bsfc_vect == min(bsfc_vect(:)));
    engSpeed = engSpd_vect(index(1));
    engTorque = engPwrCmd/engSpeed;
end

% Emergency control: In case of non-physical conditions (e.g., negative
% power command to the engine), control variables (speed and torque) are
% set to zero to prevent errors.
if any([engPwrCmd < 0, engTorque < 0, engSpeed < 0])
   engTorque = 0;
   engSpeed = 0;
end

% If SOC is negative display an error
if SOC < 0
   disp("Error! Negative SOC")
end

end
```

Variant of the thermostat controller function

As discussed in the previous paragraph, the developed thermostat controller heavily relies on the battery, leading to frequent and high power peaks that ultimately reduce its reliability and lifespan. Moreover, the most critical phase of battery operation is the recharging process, where maintaining a constant current is advisable to minimize degradation. The objective of the improved controller is to **enhance battery longevity** and optimize its utilization, particularly during charging. Based on these considerations, the controller dynamically determines the engine's target power to ensure that charging occurs at a constant power level, maintaining a **C-rate of 5**.

The C-rate value is selected based on the optimal value established for standard, commercially available small-sized hybrid vehicles.

The function implementation is done maintaining the same inputs and outputs as those of the thermostat controller. This choice is justified by the same reasons outlined in thermostat controller.

Controller logic

The enhanced rule-based controller operates, similar to the standard approach, on two layers. In the first layer, the controller monitors the state of charge (**SOC**) and determines whether the engine should be **ON** (`eng_State = 1`) or **OFF** (`eng_State = 0`) at the current instant, following the same conditions as the thermostat controller.

In the second layer, if `eng_State = 1`, the target engine power (`engPwrCmd`) is computed using the following equation:

$$P_{batt,charg} = V_{ocv}(SOC) \cdot Q \cdot C$$

$$P_{cmd} = P_{demand} + P_{batt,charg}$$

Where:

- $V_{ocv}(SOC)$ is the oper circuit voltage that depend on the SOC. It is computed using the provided 1-D LookUp Table
- Q is the battery capacity in $[Ah]$
- C is the charging C-rate
- P_{demand} is the power requested by the tractive electric motor (`PwrDemand`)
- P_{cmd} is the target engine power (`engPwrCmd`)

Otherwise the `engPwrCmd` is set to 0.

Engine Speed and Torque

Under the assumption that the Optimal Operating Line (OOL) does not encompass all possible power levels deliverable by the engine but only a restricted range, a new algorithm has been developed to ensure that a feasible operating point can be defined for any power value within the engine's limits.

Given a **target power value** (`engPwrCmd`) **greater than zero** and a speed vector (`engSpd_vect`) ranging from idle to maximum speed, the corresponding torque values required to achieve the desired power are computed for each speed value and stored in a torque vector (`engTrq_vect`). When plotted on a speed-torque graph, these points form an iso-power curve. For each speed-torque pair in the vector, the Brake Specific Fuel Consumption (BSFC) value is extracted from the provided 2-D lookup table. The optimal operating point is then identified by determining the index of the **minimum BSFC**, from which the corresponding speed (`eng_Speed`) and torque (`eng_Torque`) values are selected as the controller outputs. In summary, for a given target engine power, the algorithm determines the torque-speed pair that minimizes BSFC while also aiming to reduce the overall mission cost.

If **`engPwrCmd` is zero** but the first layer of the controller determines that the **engine** should remain **ON**, the speed is set to **idle**, and the torque is set to zero to keep the engine ready for the next instant. Conversely, if the **engine state is OFF**, all **output variables** are set to **zero**.

```matlab
function [engSpeed, engTorque, eng_State] = ...
    thermostatControl_opt(SOC, last_eng_State,vehSpeed,vehAcc,vehData)


%Define threshold of SOC

SOChi = 0.7; %Maximum SOC threshold
SOClo = 0.5; %Minimum SOC threshold


%Define a fix efficiency point of the generator
effGen = 0.9; %Corresponds to one of the worst efficiency point
              %of the generator, selected to be conservative


%PHASE 1: Check if SOC is in between thresholds

%In this section the first rule is implemented.
% The engine state from the previous cycle (last_eng_State) and the SOC are
% monitored to determine whether the engine should be turned on. The engine
% is activated if at least one of the following conditions is met: 1. The
% SOC is below the minimum SOC threshold. 2. The engine state in the
% previous cycle was ON, and the SOC is below
%    the maximum SOC threshold.
% If neither condition is met, the engine is turned off.

if or( SOC<=SOClo, and(last_eng_State > 0, SOC<SOChi))
    eng_State = 1; %Engine ON
else
    eng_State = 0; %Engine OFF
end


% PHASE 2: Define the engine power that have must be delivered by the engine

%Power requested by the battery for charging at constant current
C = 5; %Charging Rate

% To charge the battery at a constant current, the power delivered to the
% battery must also remain approximately constant. Therefore, the voltage
% is
% extracted from the 1-D Lookup Table of Open Circuit Voltage. Then, by
multiplying
% this voltage with the C-rate and the nominal capacity, the charging power
% (battPwrCharg) is obtained.
battPwrCharg = vehData.batt.ocv(SOC)*C*vehData.batt.nomCap;


% Torque and speed of motor shaft are computed using the function
% hev_drivetrain
[shaft_speed, shaft_torque] = hev_drivetrain(vehSpeed, vehAcc, vehData);
```

```
% The power demanded by the traction electric motor is computed in two
% different ways: 1. In traction mode (torque ≥ 0), the power requested at
% the shaft is
%    divided by the efficiency to account for electromagnetic losses during
%    the conversion from electrical to mechanical energy.
% 2. In regenerative braking mode (torque < 0), the efficiency is
% multiplied
%    by the power requested at the shaft, as the energy flow is from the
%    wheels to the battery.
if shaft_torque >= 0
   PwrDemand = shaft_speed*shaft_torque*(1/
vehData.mot.effMap(shaft_speed,shaft_torque));
else
   PwrDemand =
shaft_speed*shaft_torque*(vehData.mot.effMap(shaft_speed,shaft_torque));
end

% The target engine power (engPwrCmd) is initialized to 0.
engPwrCmd = 0;

% If a charging cycle is running, the target engine power (engPwrCmd) is set
% to the sum of the power requested or generated by the motor and the power
% required to charge the battery.
if eng_State == 1
    engPwrCmd = PwrDemand+battPwrCharg;
end



% CONSTRAINTS SECTION



% PHASE 3: BATTERY POWER LIMITS In this section, two constraints are set to
% ensure that the charging and discharging power limits are not exceeded.

% Constraint on discharging: If the power requested from the battery
% exceeds the limit, the engine provides the necessary power to keep the
% battery within its limits. Additionally, another condition is applied: If
% the power requested from the engine is lower than the optimal power and
% the SOC is lower than the higher SOC threshold, the engine operating
% point is set to the optimal level, and any excess power is used to
% recharge the battery. This is done to increase the of the controller
% strategy and decrease the consumption.

%if engPwrCmd < (1/effGen) * (PwrDemand - vehData.batt.maxPwr(SOC))
%    engPwrCmd = (1/effGen) * (PwrDemand - vehData.batt.maxPwr(SOC));

engPwrCmd = max(engPwrCmd,(1/effGen) * (PwrDemand - vehData.batt.maxPwr(SOC)));

% Constraint on charging: If the charging power supplied to the battery
% exceeds the limit, the engine power is reduced to stay within the limits.
% Additionally, if the engine power command is lower than the minimum power
% value on the OOL, it is probably that the motor is in regenerative
% braking. In this case, the engine is turned off to reduce fuel
% consumption.
%elseif engPwrCmd >= (1/effGen) * (PwrDemand - vehData.batt.minPwr(SOC))...
%         && eng_State == 1

%    engPwrCmd = (1/effGen) * (PwrDemand - vehData.batt.minPwr(SOC));

%end
engPwrCmd = min(engPwrCmd,(1/effGen) * (PwrDemand - vehData.batt.minPwr(SOC)));

% ENGINE POWER LIMITS

% Maximum value on the is limited to the maximum power that can be erogated
% by the engine
engPwrCmd = min(engPwrCmd,vehData.eng.maxPwr);

% Negative values of engine power are non-physical; therefore, the minimum
% value of the engine power command is limited to 0.
engPwrCmd = max(engPwrCmd,0);

% PHASE 4: ENGINE SPEED AND TORQUE

% In this phase, the controller outputs are computed based on the engine
% power command. If the target power is greater than 0, a vector of engine
% torque (engTrq_vect) is generated for each point in the engine speed vector,
% considering the engine power command (engPwrCmd). Each point lies on an
% iso-power curve in the Speed-Torque graph.
% Using the 2-D lookup table of brake-specific fuel consumption (BSFC),
% the BSFC is computed for each speed-torque pair. Ensure that values outside
% the engine limits are not considered. The index corresponding to the minimum
% BSFC is then extracted, and the command variables are determined from the
% engine speed and torque vectors.
% If the engine power command is equal to 0, two conditions are considered:
% 1. If a charging cycle is running (eng_State == 1), the engine is set to idle
%    and is not turned off, keeping it ready for the next cycle.
% 2. Otherwise, in all other cases, the engine is turned off.

engSpd_vect = linspace(vehData.eng.idleSpd,vehData.eng.maxSpd,1000); %vector of
% engine speed which contains all the points from idle speed to the maximum

if engPwrCmd > 0
    engTrq_vect = engPwrCmd./engSpd_vect;

    for l = 1:length(engSpd_vect)
        % If the engine torque exceeds the maximum allowable value, the BSFC is
        % set to NaN (Not a Number) to exclude it from the selection of the
        % minimum BSFC.
        if engTrq_vect(l)<=vehData.eng.maxTrq(engSpd_vect(l))
            bsfc_vect(l) = vehData.eng.bsfcMap(engSpd_vect(l),engTrq_vect(l));
        else
            bsfc_vect(l) = NaN;
        end
    end
    index = find(bsfc_vect == min(bsfc_vect(:)));
    engSpeed = engSpd_vect(index(1));
    engTorque = engTrq_vect(index(1));

elseif eng_State == 1 && engPwrCmd == 0
    engSpeed  = vehData.eng.idleSpd;
    engTorque = 0;

else
    engSpeed = 0;
    engTorque = 0;
end




% Emergency control: In case of non-physical conditions (e.g., negative
% power command to the engine), control variables (speed and torque) are
% set to zero to prevent errors.
if any([engPwrCmd < 0, engTorque < 0, engSpeed < 0])
   engTorque = 0;
   engSpeed = 0;
end

% If SOC is negative display an error
if SOC < 0
   disp("Error! Negative SOC")
end


end
```

Conclusion

This study analyzed and improved the control strategy of a quasi-static HEV model using a rule-based thermostat controller. The initial controller effectively managed fuel consumption, maintaining a total fuel consumption below 1 kg (902 g) while ensuring the battery's state of charge remained within predefined thresholds. However, the battery experienced frequent and high-power peaks, potentially accelerating its degradation.

To address this issue, an optimized thermostat controller was implemented. This improved version prioritized battery longevity by ensuring charging cycles occurred at a constant power level while still maintaining fuel efficiency. The results showed a slight improvement in fuel consumption, reducing it to 881 g, while also achieving a final state of charge of 0.71, compared to 0.61 in the standard controller.

Comparing the two versions, the optimized controller demonstrated a more stable power distribution, reduced peak loads on the battery, and improved fuel efficiency, particularly during highway driving.

In conclusion, to further enhance the performance of the improved controller, several additional adjustments could be considered:

- Shifting low-efficiency engine operating points to more optimal conditions.
- Better utilization of the entire engine power range or considering engine downsizing.

These refinements could further improve overall efficiency and fuel consumption while maintaining the benefits of the optimized control strategy.