Assignment #4: dynamic programming

Table of Contents

Group information

Group number: 12

Students:

- Giuseppe Maria Marchese, s348145
- Emanuele Landolina, s349706
- 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.

```
clear all
close all
clc

% Add path
addpath("utilities");
addpath("models");
addpath("data");
addpath("additional utilities")

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

```
%load mission data
mission = load("WLTP.mat");
%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 characteristic [Group 12]
engine_power = 66000;    %[W]
E_machine_power = 70000; %[W]
battery_energy = 945;    %[Wh]
beta = 6.67e-6;

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

Test cycle

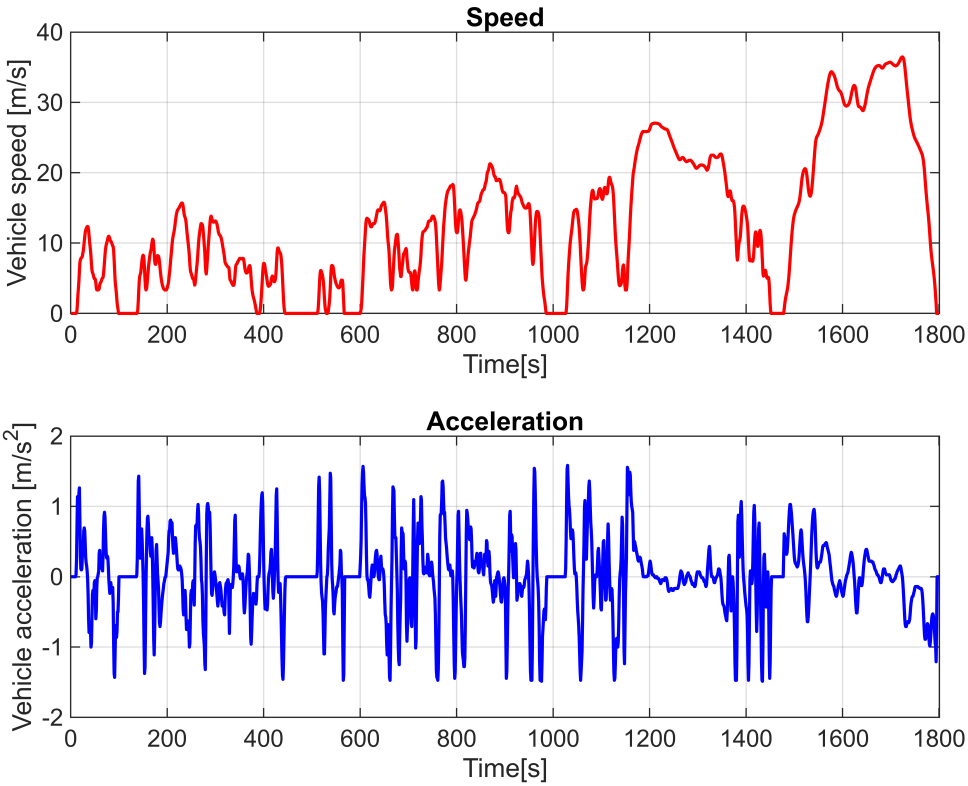
To test the Dynamic Programming, the WLTP cycles is used.

```
%% Vehicle speed and acceleration

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

nexttile(1)
plot(time_vector,vehSpd,"r",LineWidth=1.2)
title("Speed")
xlabel("Time[s]")
ylabel("Vehicle speed [m/s]")
xlim([0 length(time_vector)])
grid on

nexttile(2)
plot(time_vector,vehAcc,"b",LineWidth=1.2)
title("Acceleration")
xlabel("Time[s]")
ylabel("Vehicle acceleration [m/s^2]")
xlim([0 length(time_vector)])
grid on
```



The **WLTP** cycle is divided into several parts. The first phase simulates urban driving, characterized by low speeds and not so frequent 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. It is noticeable that in none of the phases are present very high dynamics, with few stops and so not much energy recovered from braking manouvres.

EMS definition

Both the requested fuel-optimal and aging-aware EMS have been realized through one revised `hev_cell_model`. They are both describe in the following paragraphs.

Aging-aware EMS

The provided **`hev_cell_model`** has been modified to implement a **new objective function** for the optimization problem. The aim is to perform a **multi-objective optimization** that **minimizes fuel consumption while limiting the impact on battery reliability and aging**. Fuel consumption is addressed by considering the fuel flow rate, whereas battery aging is evaluated based on the current. This choice is motivated by the fact that high C-rates, corresponding to high current levels, accelerate the degradation of battery chemistry, ultimately reducing performance.

Accordingly, the following objective function is implemented:

$$L_k = \alpha \frac{\dot{m}_{\mathrm{f},k}}{\dot{m}_{\mathrm{f,max}}} + (1-\alpha)\frac{|i_{\mathrm{b},k}|}{i_{\mathrm{b,max}}}$$

The mass fuel flow rate is normalized using the maximum value available from the engine map. Similarly, the battery current is normalized based on the maximum allowable current during both discharging and

charging. A weighting factor, denoted as **alpha**, is introduced to balance the contribution of each objective in the minimization process. This factor **determines the relative importance of fuel consumption versus battery aging in the overall optimization.**

Fuel-optimal EMS

To obtain an EMS that accounts only for the fuel consumption optimization, the previous discussed `hev_cell_model` has been modified only putting $\alpha = 1$, in order to eliminate the term related to battery current. The resultant objective function is shown:

$$L_k = \frac{\dot{m}_{\mathrm{f},k}}{\dot{m}_{\mathrm{f,max}}}$$

Objective function implementation

The implementation on the `hev_cell_model` is done as follows:

```
%%% Stage cost
%
%
% maxBattCurr = ones(size(battCurr));
% maxBattCurr(battCurr >= 0) = maxDisBattCurr;
% maxBattCurr(battCurr < 0) = maxChrgBattCurr;
%
%
% stageCost  = alfa.*(fuelFlwRate./mf_max)+(1-alfa).*abs(battCurr./maxBattCurr);
```

First of all, it is important to note that **`fuelFlwRate`** and **`battCurr`** are **matrices** that store the corresponding values for each control candidate. In particular, the entries in `battCurr` can be either positive or negative. Moreover, the maximum battery current differs depending on whether the battery is charging or discharging.

For this reason, **`maxBattCurr`** is **also defined as a matrix**, initially filled with ones. Then, using logical indexing, the values corresponding to positive battery currents are set to the maximum discharge current, while those corresponding to negative values are set to the maximum charging current.

Finally, the stage cost formulation described above is implemented based on these normalized values.

Dynamic Programming

To implement the dynamic programming method, the **DynaProg Toolbox** is used. This implementation requires preparation on both the input side and the model function side(`hev_cell_model`).

Simulations are conducted for multiple values of the weighting factor **α** to analyze its influence on the results. To achieve a good trade-off between computational cost and accuracy, **α is discretized from 0 to 1 with a step size of 0.025**. The `simulationRun` function returns a structure of size `1 × length(alfa_vect)`, where all results corresponding to each α value are stored.

To avoid rerunning all simulations every time, the results and the corresponding α values are **saved in a `.mat` file**. This file can be loaded for future use, significantly reducing computation time.

A detailed explanation of the DynaProg Toolbox implementation is provided in the simulationRun function chapter.

```
%engine map discretization
```

```
resEng = 50;

%SOC grid creation
stepSOC = 0.001;
SOCgrid = {0.4:stepSOC:0.8};

% Set initial SOC
SOCinitial = {0.6};

% Set final SOC range
SOCfinal = {[0.599 0.601]};

% Mission data cell
missionData = {vehSpd,vehAcc};

% Definition of vector with multiple alpha
alfa_vect = 0:0.025:1;

% Run the simulation
%results_multipleAlfa =
simulationRun(resEng,SOCgrid,SOCinitial,SOCfinal,alfa_vect,missionData,vehData);

% Save results
%save("multipleAlfa_simulation.mat","results_multipleAlfa","alfa_vect")
```

Alpha influence analysis

In this section, we analyze the simulation results obtained for different values of α.

To visualize the influence of α on vehicle performance, two plots have been created:

- **SOC evolution vs Time**
- **Fuel Consumption and Distance to EoL vs Alpha α**

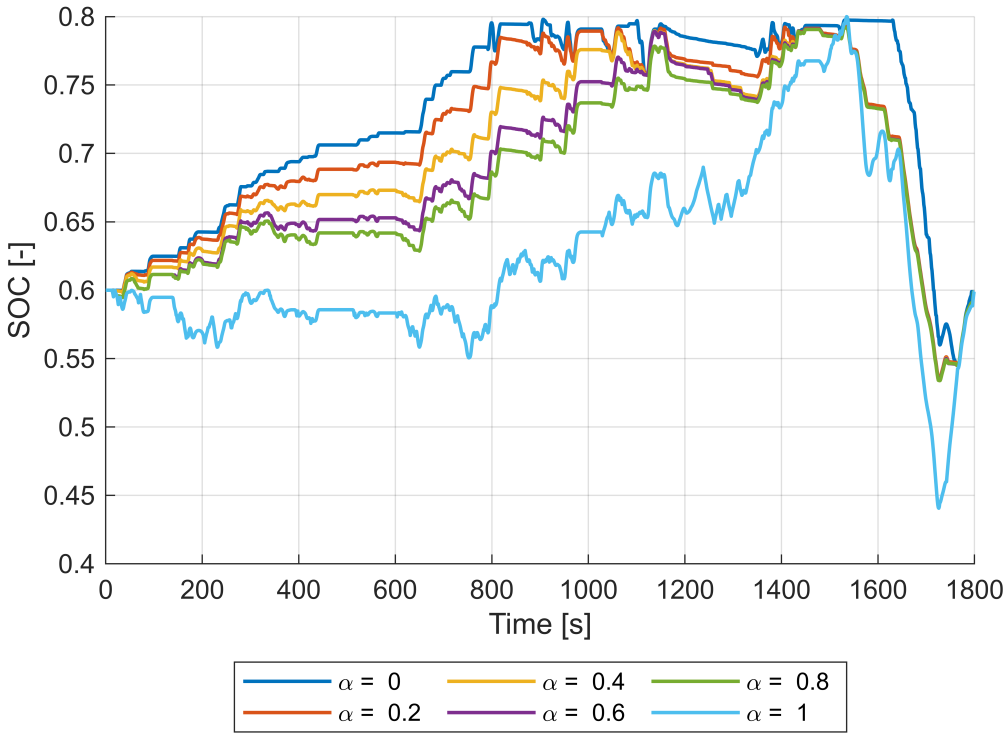
They provide valuable insights into how different α, which balance fuel consumption and battery usage in the objective function, affects the overall behavior of the system.

```
% Load simulation results
results_multipleAlfa =
load("multipleAlfa_simulation.mat","results_multipleAlfa");
results_multipleAlfa = results_multipleAlfa.results_multipleAlfa;
alfa_vect = load("multipleAlfa_simulation.mat","alfa_vect");
alfa_vect = alfa_vect.alfa_vect;

% Extract SOC evolution and construct plot label
for k = 1:length(results_multipleAlfa)
    SOC_evolution(k,:) = results_multipleAlfa(k).StateProfile{1};
    alfa_label(k) = join(["\alpha = ",string(alfa_vect(k))]);
end
```

SOC Evolution vs Time

```
figure()
hold on
plot(time_vector,SOC_evolution(1:8:41,2:end),LineWidth=1.3)
grid on
axis([0 time_vector(end) 0.4 0.8])
xlabel("Time [s]")
ylabel("SOC [-]")
legend(alfa_label(1:8:41),"NumColumns",4,"Location","southoutside")
```

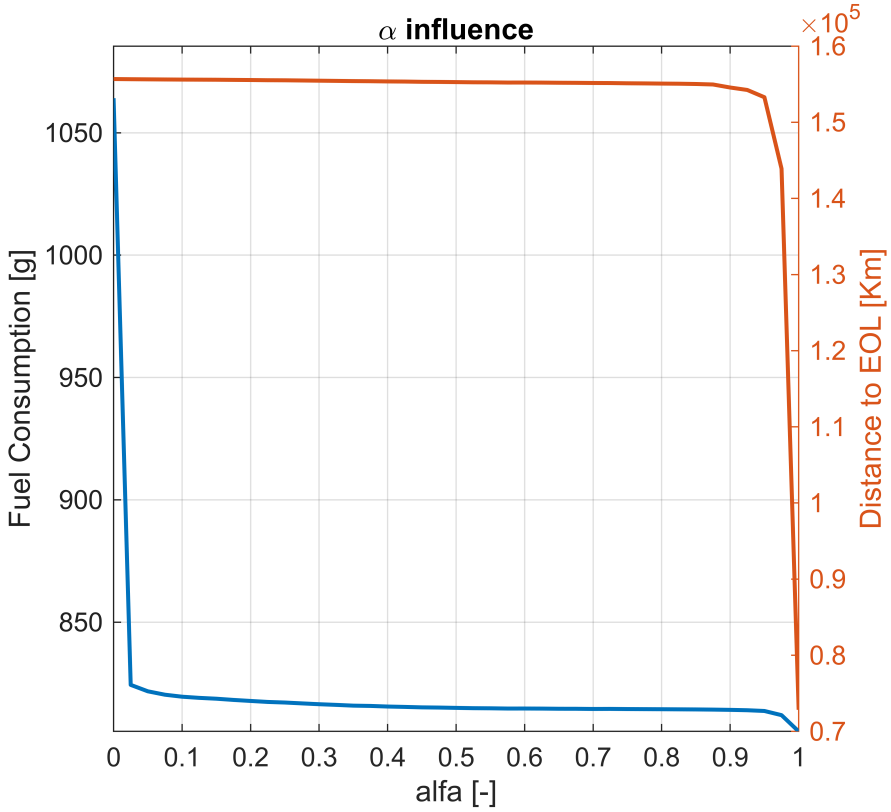


The plot shows the SOC evolution for **six values of α**, ranging from 0 to 1 with a step of 0.2. As **α increases**,assigning **greater weight to fuel consumption in the minimization** strategy, the **battery is used more extensively**, especially in the first part of the WLTP cycle, where a prolonged charge-sustaining phase is observed. In contrast, **lower values of α** (like 0 or 0.2) result in a strategy that **prioritizes battery usage**: the engine fully recharges the battery early on, and then maintain the SOC to high value for almost the totality of the cycle. Only in the final part, where the power demand peaks, the battery is discharged to support the engine.

In addition, since dynamic programming has full knowledge of the driving cycle in advance, it can effectively **exploit strong deceleration phases**, such as the one at the end of the WLTP cycle, **for battery recharging through regenerative braking**. As **α increases**, placing greater emphasis on minimizing fuel consumption, **this regenerative potential is utilized more effectively**, resulting in a significant benefit in terms of fuel savings. Conversely, for **lower values of α**, regenerative braking is less exploited, as the control strategy prioritizes preserving battery health and minimizing degradation, which **discourages aggressive power fluctuations**.

Fuel Consumption and Distance to EoL vs Alpha α

```
alfaAnalysis(results_multipleAlfa,alfa_vect,beta,mission);
```



The second plot highlights the **opposite trends of fuel consumption and the estimated distance to End of Life (EoL)** of the battery as functions of the weighting parameter α. These two quantities exhibit an inverse relationship, as they are respectively proportional to α and 1−α.

Fuel consumption shows a sharp decrease for very small values of α, particularly between α=0 and α≈0.025, where a 2.5% increase in α results in a reduction of nearly 24% in fuel usage. Beyond this point, the curve flattens significantly, stabilizing around 800 grams, with **only marginal reductions (approximately 20 grams) as α increases from 0.05 to 1**.

Conversely, the **distance to EoL remains almost constant across a wide range of α values, up to α=0.95**. This suggests that battery degradation is largely mitigated when α remains below this threshold. However, **as α approaches 1**, meaning battery degradation is no longer considered in the cost function, **the distance to EoL drops sharply**, indicating accelerated battery wear due to the optimization strategy favoring fuel minimization exclusively.

This plot clearly illustrates the **trade-off** introduced by α, emphasizing the need for a balanced tuning of the weighting factor to ensure both fuel efficiency and battery longevity: **α=0.95** represents the best trade-off guaranteeing an acceptable fuel consumption while reducing effectively the battery aging.

For a detailed numerical analysis, we refer the reader to the section numerical analysis.

The metric "Distance to EoL", defined as the distance the vehicle can travel before reaching the SOH limit (set to 80%), is calculated using the following equation:

$$\text{Distance to EoL} = \text{Cycle Distance} \cdot \frac{100 - \text{SOH}_{\text{limit}}}{C_{\text{loss},\%}}$$

Parameters computation and results storage

```
for i = 1:length(results_multipleAlfa)
    prof = structArray2struct(results_multipleAlfa(i).AddOutputsProfile{1});

    % Total fuel consumption
    fuelConsumption_multipleAlpha(i) = trapz(time_vector,prof.fuelFlwRate)/1000;

    % Fuel economy
    distance = trapz(time_vector,vehSpd)/1000;
    fuel_rho = vehData.eng.fuelDensity; %[Kg/l]
    fuelEconomy_multipleAlpha(i) = (fuelConsumption_multipleAlpha(i)/
(fuel_rho*distance))*100;

    % Final SOC
    finalSOC_multipleAlpha(i) = prof.battSOC(end);

    % Distance to EOL
    SOH_limit = 80;
    Closs = 100*trapz(time_vector,abs(prof.battCurr)')*beta/3600;
    distance_eol_multipleAlpha(i) = distance*(100-SOH_limit)/Closs;
end

% Performance of fuel optimal EMS
fuelConsumption = fuelEconomy_multipleAlpha(end);
fuelEconomy = fuelEconomy_multipleAlpha(end);
finalSOC = finalSOC_multipleAlpha(end);
distance_eol = distance_eol_multipleAlpha(end);


% Store results
save("results_base.mat", "fuelConsumption", "fuelEconomy", "finalSOC",
"distance_eol")

save("results_ageing.mat", "fuelConsumption_multipleAlpha",
"fuelEconomy_multipleAlpha", "finalSOC_multipleAlpha",
"distance_eol_multipleAlpha", "alfa_vect")
```

Results comparison

This section compares **four α** values to identify **meaningful outcomes** for the analysis.

The selected values are **`0`**, **`0.05`**, **`0.95`**, and **`1`**. These were chosen to assess the difference between two limit cases and two values recognized as **most significant** due to a sharp change of the trends.

```
% Selected results
alphaSelected_idx = [1 3 39 41];
alphaSelected = alfa_vect(alphaSelected_idx);
totalResults = results_multipleAlfa(alphaSelected_idx);
```

Numerical Analysis

In this paragraph, the most meaningful parameters are compared.

- **Fuel consumption and economy**

```
table(alphaSelected',fuelConsumption_multipleAlpha(alphaSelected_idx)',fuelEconom
y_multipleAlpha(alphaSelected_idx)',
round(100.*(fuelEconomy_multipleAlpha(alphaSelected_idx)'./
max(fuelEconomy_multipleAlpha(alphaSelected_idx)')-1),2),'VariableNames',
["alpha","Fuel Cons. [Kg]","Fuel Economy [l/100km]","FE variation [%]"])
```

ans = 4×4 table

	alpha	Fuel Cons. [Kg]	Fuel Economy [l/100km]	FE variation [%]
1	0	1.0642	5.8275	0
2	0.0500	0.8217	4.5000	-22.7800
3	0.9500	0.8137	4.4560	-23.5400
4	1	0.8053	4.4102	-24.3200

Since fuel consumption is one of the two parameters minimized by the objective function, it is particularly interesting to observe how it varies across the different scenarios. Surprisingly, the **percentage variation** from the **optimal result** (**α=1**) is **very small** even for **α=0.05**, which theoretically **should focus** almost **exclusively** on minimizing **battery aging**. However, in practice, it **achieves** a **high level** of **optimization** for both parameters.

Even more surprising is that the percentage difference between **α=0.05** and **α=0** is almost ten times greater, suggesting that the inclusion of the **fuel consumption term** alone in the objective function is **sufficient** to **drive** its **effective optimization**.

- **SOC deviation**

```
table(alphaSelected',round(finalSOC_multipleAlpha(alphaSelected_idx),3)',round(10
0.*(0.6-finalSOC_multipleAlpha(alphaSelected_idx)')./0.6,2),'VariableNames',
["alpha","Final SOC","Deviation [%]"])
```

ans = 4×3 table

	alpha	Final SOC	Deviation [%]
1	0	0.5990	0.1700
2	0.0500	0.5990	0.1800
3	0.9500	0.5990	0.1800
4	1	0.5990	0.1800

This table demonstrates that **Dynamic Programming** effectively **achieves charge-sustaining operation**, thanks to prior knowledge of the driving mission and its backward-solving approach. By starting from the end of the cycle and working backward, it can optimize the entire operation obtaining a deviation from the target SOC that is always equal to 0.001 (practically null).

- **Distance to EOL**

```
table(alphaSelected',distance_eol_multipleAlpha(alphaSelected_idx)',round(100.*(d
istance_eol_multipleAlpha(alphaSelected_idx)'./
```

```
max(distance_eol_multipleAlpha(alphaSelected_idx)')-1),2),'VariableNames',
["alpha","Distance to EOL","Var. [%]"])
```

ans = 4×3 table

	alpha	Distance to EOL	Var. [%]
1	0	1.5568e+05	0
2	0.0500	1.5564e+05	-0.0300
3	0.9500	1.5329e+05	-1.5400
4	1	7.2838e+04	-53.2100

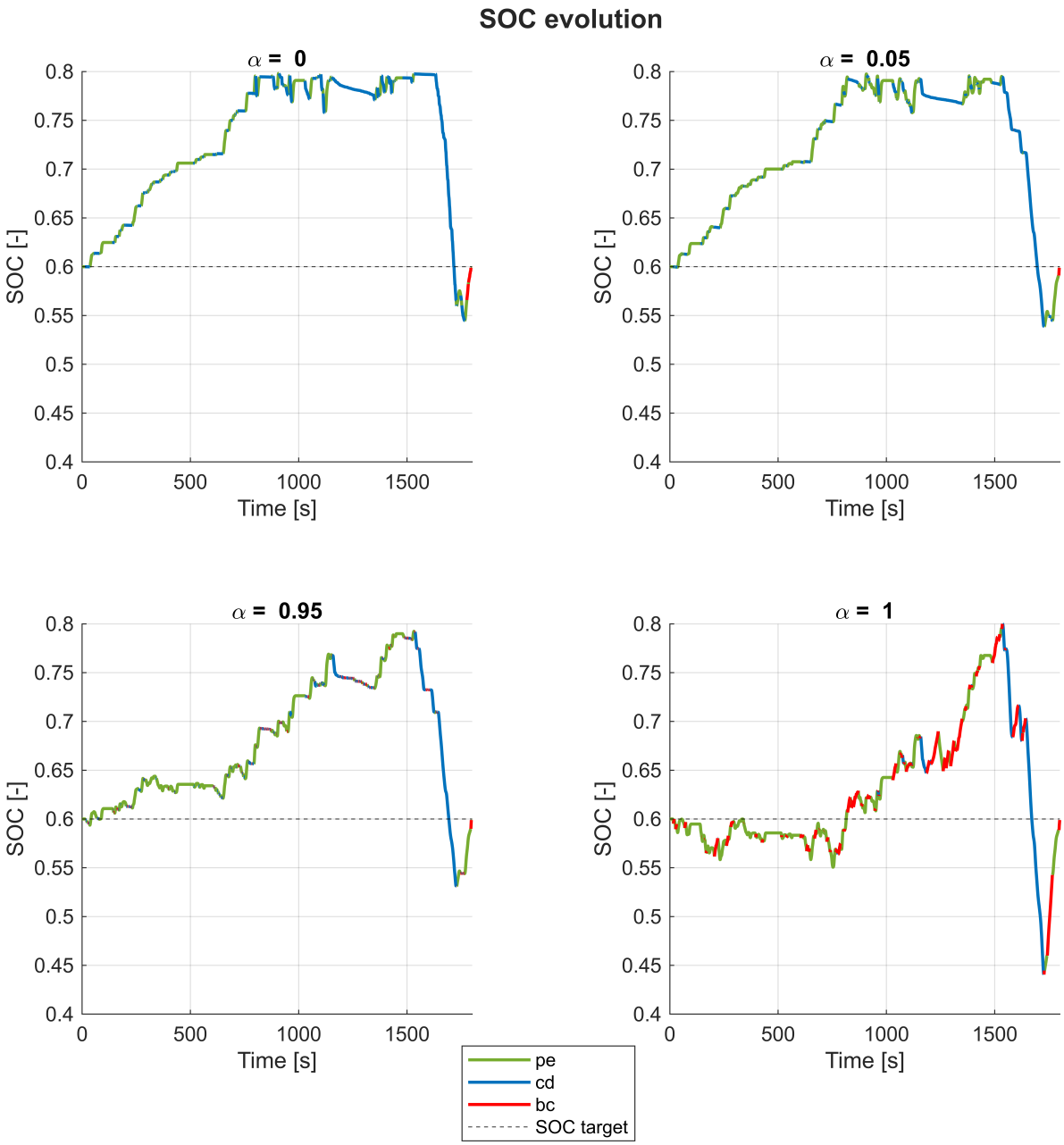
The second key parameter to analyze is the distance from the End of Life (EOL) of the battery. Its trend mirrors that of fuel consumption, but in the opposite direction. Specifically, the **percentage variation** between **α=0** and **α=0.95** is **minimal**, in **opposite** with the **expectations** given the **low weight** assigned to **battery aging** in the objective function. In contrast, the percentage improvement in **α=1** is 53%, highlighting how strongly the **inclusion** of the **battery aging term** alone **drives** its **optimization**.

Graphical comparison

In this paragraph, the most meaningful plots are discussed to extract valuable informations.

SOC vs Time

```
SOCwithPF(totalResults,alphaSelected);
```



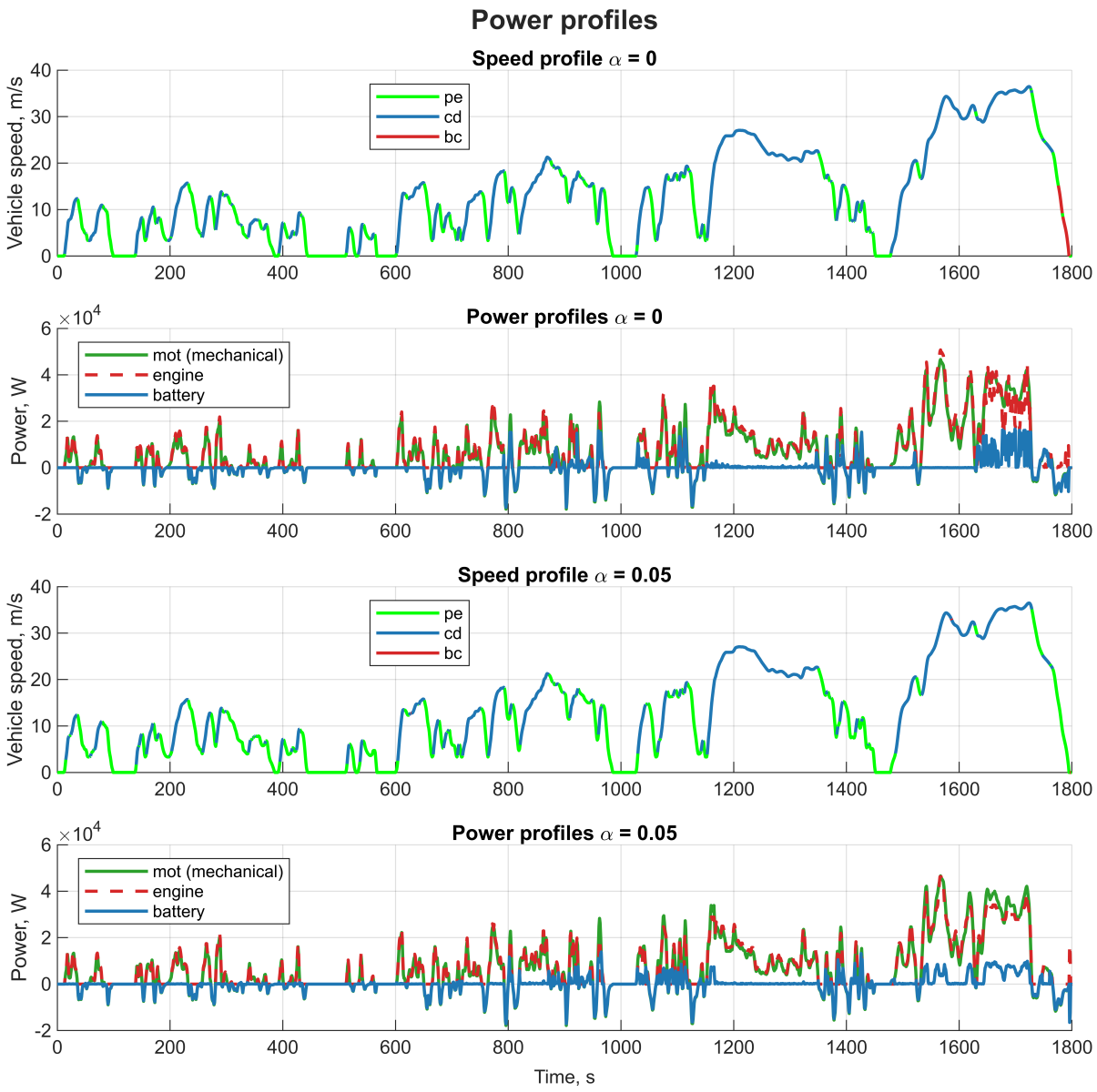
Starting from **α=0**, where **battery usage** is **optimized** to slow its aging, **battery charging** operations are almost **avoided**. In fact, the **battery** is **charged** only when **forced** through **regenerative braking**, while it supplies energy when engine power is insufficient and electric motor use is unavoidable or to reach the proper charge-sustaining operation.

In contrast, with **α=1**, the **fuel consumption** is the parameter to be **minimized**. This results in a **blended behavior** of **pure electric** and **battery charging** modes. **Charge depleting** is **used briefly** (few seconds) when the **maximum power** is **demanded**. In this case, the electric pathway helps the engine in meeting the power demands, continuously increasing and decreasing the battery SOC.

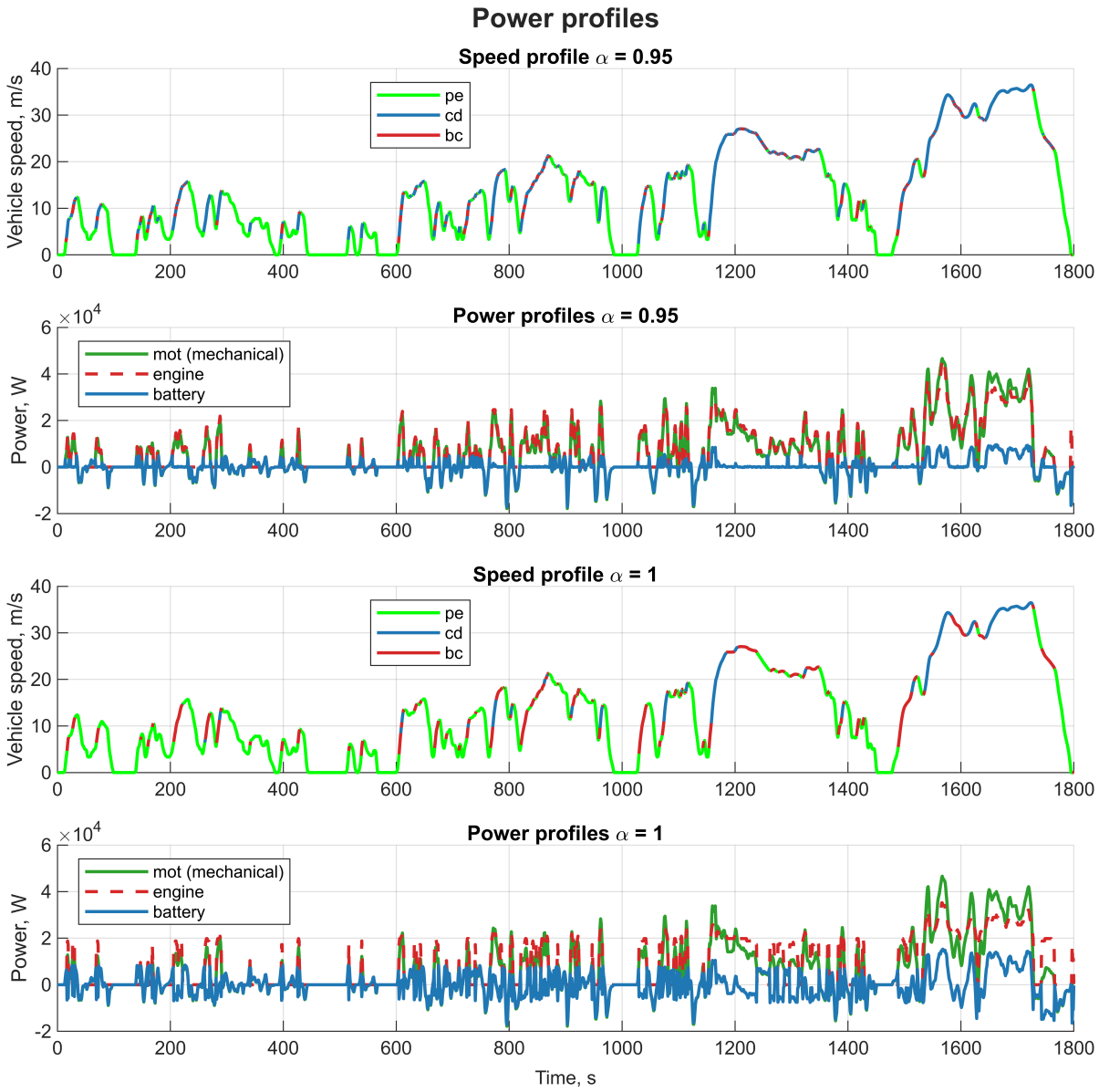
For both **α=0.05** and **α=0.95**, the behavior resembles that of **α=0**, with battery charging operations almost avoided. In fact, for **value closer** to **α=0** the **tendency** is to **save** more **electric energy**, as expected. However the **system** becomes **more flexible** in **different operating modes**, resulting in more readily switching between charge-depleting and battery charging to assist the engine.

Power split vs time

```
powerProfiles_comp(totalResults(1:2),["mot","batt","eng"],alphaSelected(1:2));
```




```
powerProfiles_comp(totalResults(3:4),["mot","batt","eng"],alphaSelected(3:4));
```



In these plots it is interesting to evaluate how the **power demand** is **managed** between **engine** and **battery**.

Battery power usage increases with **higher values** of α, particularly in regions where the power demand is greater. Likewise, **battery charging** operations become **more frequent** as α **increases**, reflecting a greater blending of operating modes and energy source usage. This trend indicates a shift from a **conservative battery management strategy** (at **low α**) toward a more **performance-oriented strategy** (at **high α**), where the electric powertrain is more actively used to support the engine.

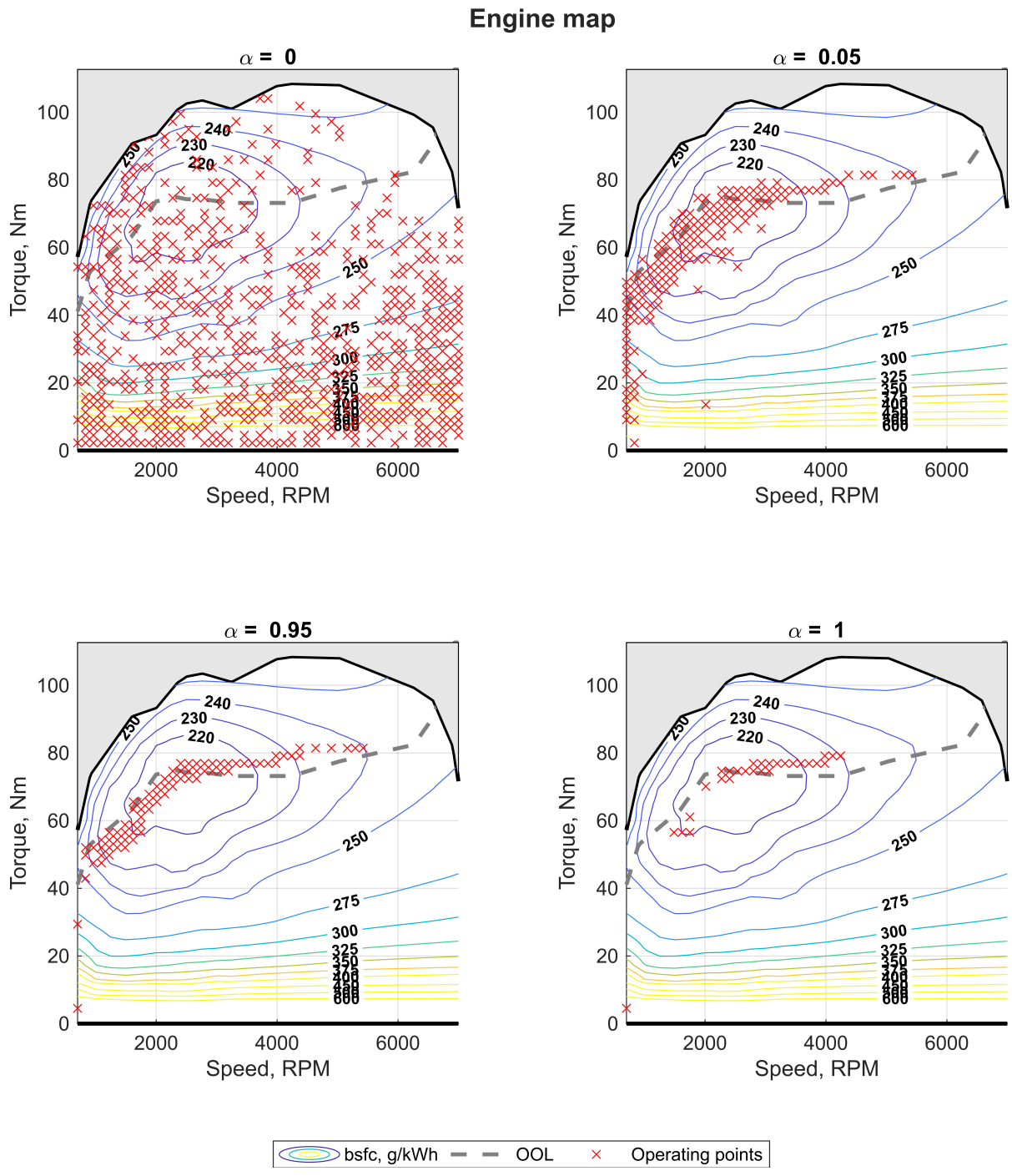
Focusing on the final phase of the cycle, it is particularly interesting to examine how power is managed under different values of α:

- **α = 0**: The engine closely follows the power demand, while a significant portion of energy is supplied by the battery, which is discharged to reach the target State of Charge (SOC).
- **α = 0.05 and α = 0.95**: The battery delivers power more steadily, which helps reduce the engine's power fluctuations. This smoother engine operation contributes to lower fuel consumption.

- **α = 1**: The engine operates at a nearly constant power level, slightly lower than in the other cases, while the battery absorbs the power peaks, effectively smoothing the overall system load.

Engine map

```
mapWithPF(totalResults,alphaSelected,vehData);
```



These plots illustrate the engine map with Brake Specific Fuel Consumption (BSFC) contours, overlaid with the engine's operating points marked as red crosses. These points reflect how the energy management strategy (EMS) controls the engine's behavior. Comparing the four graphs notably the distribution and number of operating points is different. As **α increases**, the EMS increasingly concentrates the engine's

operation within **high-efficiency zones**, gradually **aligning** the operating points with the Optimal Operating Line (**OOL**).

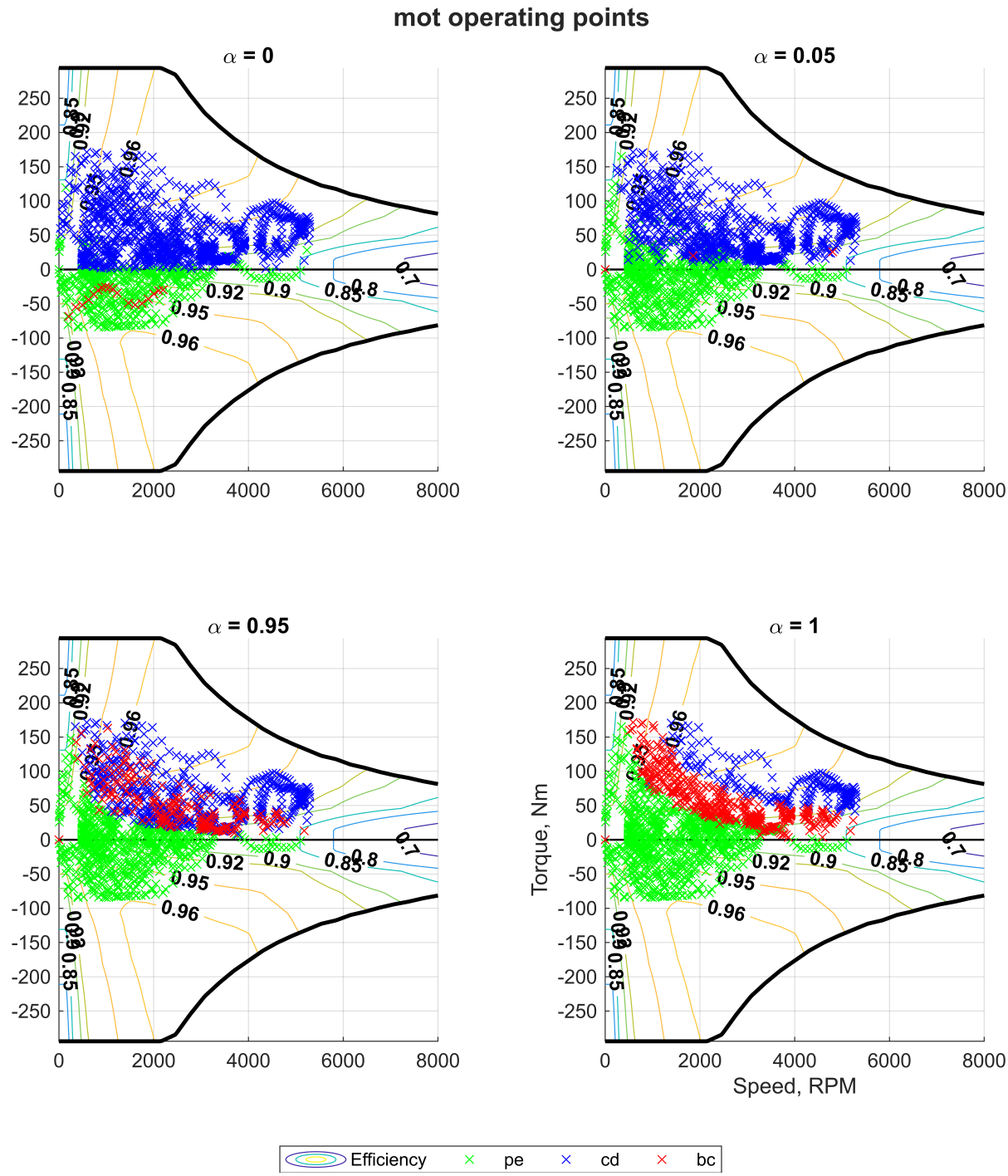
For **α=0**, where fuel consumption is not considered in the cost function, the **engine operates** across a **wide range** of map regions, **including low-efficiency zones**. This widespread dispersion is consistent with the previously observed increase in fuel consumption, up to 24% higher compared to strategies using even a small α value.

When α is increased slightly, such as to **α=0.05**, the control strategy begins to **avoid high-BSFC regions** and starts concentrating points **closer** to the **OOL**. This reflects a shift in priority toward fuel economy. At higher α values, such as **α=0.95**, the number of **operating points** at **idle speed** is **significantly reduced**, contributing to further fuel savings.

Finally, for **α=1**, the **operating points** are densely **clustered** within the **lowest BSFC regions** and are closely **aligned** with the **OOL**.This demonstrates how the EMS, when focused entirely on minimizing fuel consumption, can optimize engine operation very effectively.

Motor operating points

```
emMapWithPF_alpha(vehData.gen,totalResults,alphaSelected,"mot","all");
```



The following plots present the electric motor efficiency maps, overlaid with all operating points categorized into three modes:

- **Pure electric (pe)** – green,
- **Charge-depleting (cd)** – blue,
- **Battery-charging (bc)** – red.

The *pure electric* operating points can be further divided into two quadrants:

- The **positive quadrant** (upper half) corresponds to **traction** using only the electric motor.
- The **negative quadrant** (lower half) corresponds to **regenerative braking**.

For **α = 0**, where the EMS **prioritizes battery health**, the **traction** phase is primarily managed in **charge-depleting mode**. In this mode, all power requests are supplied by the internal combustion engine, while the pure electric mode is reserved exclusively for regenerative braking. Additionally, the **battery charge mode** is activated **during braking** events to help meet the State of Charge (SOC) target at the end of the drive cycle. A few exceptions exist, where pure electric mode is used during the tractive phase; however, these occur only at very low power levels to minimize battery usage.

As **α increases** (like **α = 0.05**), only **minimal changes** are observed. However, some **pure electric** points begin to appear in the **positive quadrant**, indicating an **early tendency** toward **electric traction** to **slightly reduce fuel consumption**.

At **α = 0.95**, a **clearer separation** among **operating modes** emerges. In particular, the **pure electric** mode becomes **more defined**, following iso-power curves. Nevertheless, the distinction between **battery-charging** and **charge-depleting** remains somewhat **mixed** to avoid high current value that can aging the battery.

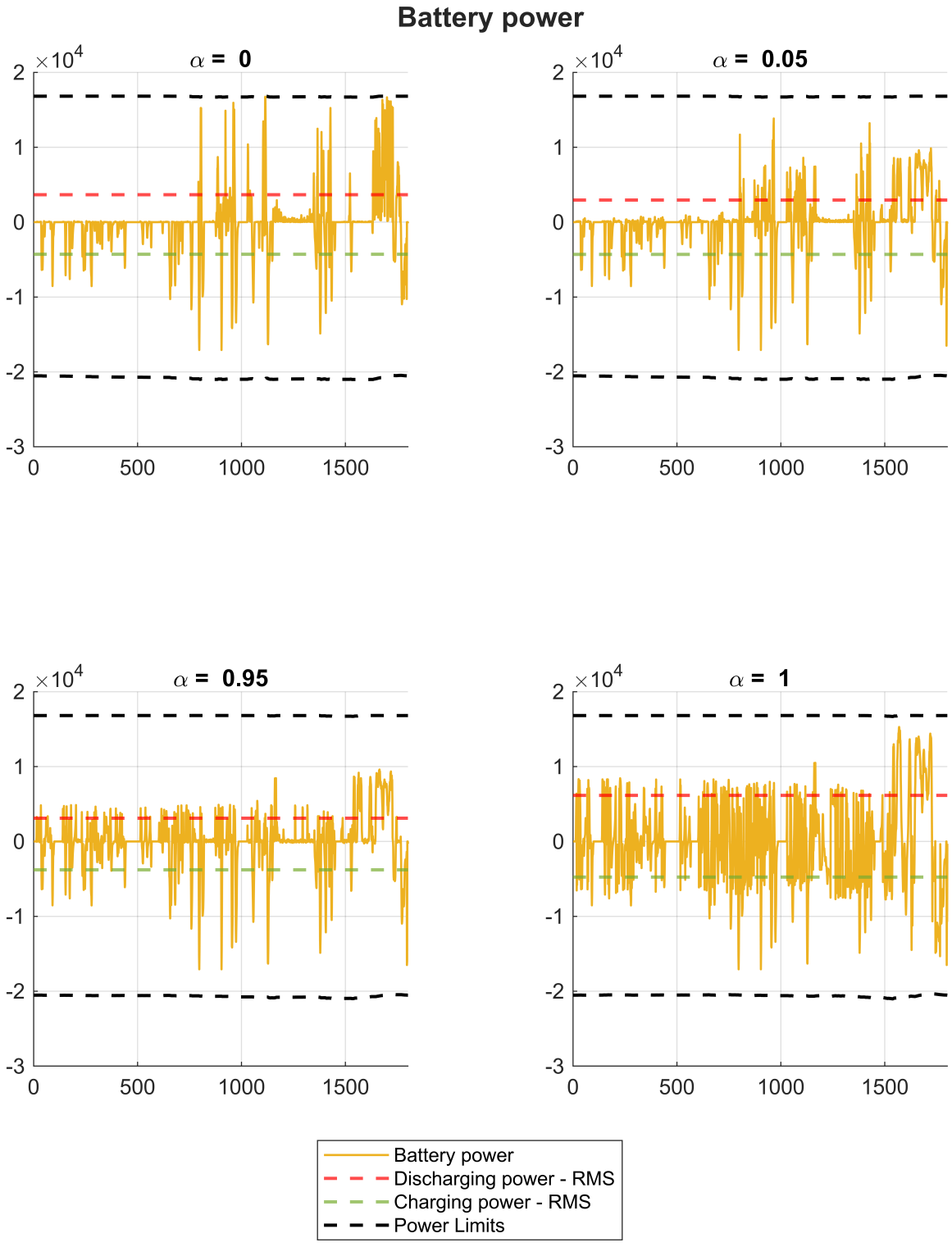
Finally, for **α = 1**, the EMS focuses **entirely** on **fuel consumption minimization**. The operating points become more clearly separated, aligned with iso-power lines. The **motor** handles **traction** in **low-demand** scenarios, while the **engine** activates for **battery-charging** or when the **power demand exceeds** a certain **threshold**.

This strategy maximizes overall system efficiency by:

- Using pure electric mode at low power, avoiding the use of the engine in low efficiency zones;
- Using battery charge mode at medium power to align the engine operating points on the OOL in the best bsfc zones through the use of the generator as variable load;
- Using charge depleting mode to supply the high power request when the battery limits are reached.

Battery power vs Time

```
battPwrPlot(totalResults,alphaSelected,vehData);
```



Battery usage is likely the primary focus of this analysis, as it directly influences the extent of aging that occurs during each cycle. In particular, the battery current should be closely examined. However, since the battery voltage remains relatively constant, fluctuating within a narrow ±10V range (with respect to +200V battery), the power profile serves as a good representation of the current behavior.

Ideally, the **α=0** should show the best behavior, minimizing the total time during which the battery experiences input or output currents. However, the **current peaks** are **few** but **extremely high**, which is **particularly harmful** and **accelerate battery degradation**.

The RMS value during both the charging and discharging phases turns out to be similar, or even higher, in the **α=0** case compared to **α=0.05** and **α=0.95**, highlighting how the **inclusion** of the **battery aging term** alone in the **objective function significantly influenced** the **degradation behavior**.

On the contrary, in the case of **α = 1**, the expected behavior occurs: the **battery** is almost **constantly active**, both for supplying and recharging energy. This scenario does **not account** for **aging**, but instead **focuses** solely on **optimizing** the usage of the **eMotor** and **engine combination**. This is confirmed by the RMS values, which are significantly higher than in the other cases for both charging and discharging phases.

Simulation function

DynaProg Implementation

The aim of this project is to **leverage** the **capabilities** of **Dynamic Programming**, to **achieve** the **global optimization** of the previously described objective function J over a well-known cycle (in this case the WLTP).

Dynamic Programming is structured in two phases:

- **Backward phase**: the tail subproblem is solved and then extended at each stage up to the initial one, storing the optimal cost for every state.
- **Forward phase**: beginning from the initial stage, the optimal control path is constructed by evaluating the previously stored optimal costs at each stage.

To define the dynamic programming problem, the `DynaProg` function is used, which requires several input variables:

- **`x_grid`**: This represents the discretized range of the state variable, which in this case is the State of Charge (SOC). The admissible SOC range is from 0.4 to 0.8, discretized with a step size of 0.001 to ensure high accuracy and convergence within the final SOC limits.
- **`x_initial`**: This sets the initial state of the system. For the simulation, the initial SOC is set to 0.6.
- **`x_final`**: This defines the acceptable range for the final SOC. To ensure charge-sustaining behavior with a maximum deviation of ±0.001, the final SOC boundaries are set to 0.599 and 0.601.
- **`u_grid`**: This is a cell array containing all control candidates, defined in terms of engine speed and torque. The resolution is determined by the input variable `engResMap`, which is set to 50 in this case. This value represents a compromise between accuracy and computational efficiency. Additionally, a value of `0` is added at the beginning of both the engine speed and torque vectors to account for the engine-off condition.
- **`Nint`**: This specifies the number of stages in the optimization problem and is set equal to the number of sampled points from the mission velocity and acceleration profiles.
- **`HevSyS`**: This is the model dynamic function that defines all the inputs and outputs required by the optimization.
- **`w`**: This represents the mission velocity and acceleration data, which are provided to the `DynaProg` function as exogenous inputs.

Some variables at the end of the iterations are cleared to avoid conflict in the next ones.

```
function [simulationResult] =
simulationRun(engResMap,SOCgrid,SOC_0,SOCfinal,alfa_vect,missionData,vehData)
```

```
% Engine control variable discretization

% Engine discretization resolution
resEng = engResMap-1;

engIdleSpd = vehData.eng.idleSpd; % Eng idle speed
engMaxSpd = vehData.eng.maxSpd; % Eng max speed

% 0 is added to consider also engine off condition
engSpd_vect = [0, linspace(engIdleSpd,engMaxSpd,resEng)];

% Engine torque vector
engMaxTrq = max(vehData.eng.maxTrq(engSpd_vect));
engTrq_vect = [0, linspace(0,engMaxTrq,resEng)];

% control variable are compacted in a cell
u_grid = {engSpd_vect,engTrq_vect};



% SOC values discretization
x_grid = SOCgrid;
x_initial = SOC_0;
x_final = SOCfinal;



% Mission data
w = missionData;

% Maximum fuel flow rate
maxFFR = maxFuelFlwRate(vehData.eng);

% Stages number
Nint = length(missionData{1});

for i = 1:length(alfa_vect)

    % Define the current alfa value
    alfa = alfa_vect(i);
    fprintf("alfa = %.2f \n", alfa)

    % Define the model dynamic function
    HevSyS = @(x,u,w,prof) hev_cell_model(x,u,w,alfa,maxFFR,vehData);

    % Setup the dynamic problem
    prob = DynaProg(x_grid, x_initial,
x_final,u_grid,Nint,HevSyS,'ExogenousInput', w);

    % Solve the dynamic problem
    simulationResult(i) = run(prob);

    % Clear used variable
    clear prob; clear HevSys; clear alfa; clear x; clear u;
```

```
end
end
```

Conclusion

The objective of this project was to apply Dynamic Programming to identify optimal energy management strategies for a hybrid electric vehicle (HEV) operating along a WLTP cycle. A multi-objective cost function was introduced to balance **two competing goals**: **minimizing fuel consumption** and **minimizing battery aging**. The balance between these objectives was controlled via a weighting parameter α, whose influence was extensively analyzed.

Among all tested values, four were chosen to compare the controller behaviors at different α (1, 0.95, 0.05, 0). In particular two were at the extremities (1 and 0) and two were chosen as the most representative of the results (0.95 and 0.05)

The Best Trade-off: α = 0.95

Setting α = 0.95 yields a control strategy that comes remarkably close to the fuel-optimal behavior seen at α = 1 while still preserving a significant degree of protection for the battery. At this configuration:

- **Fuel consumption** is only **marginally higher** (less than 1%) than the absolute minimum achieved at α = 1, demonstrating excellent efficiency.
- The **battery's degradation**, measured via the “Distance to EoL” metric, remains **well within acceptable limits**, with only a 1.5% reduction in battery lifespan compared to the most conservative setting (α = 0).
- The **power split** is **smooth** and **balanced**. The battery is used more consistently throughout the cycle, supporting the engine by handling transient power demands and helping it avoid high-load fluctuations.
- The **engine** operates within **high-efficiency zones** of the BSFC map, aligning closely with the Optimal Operating Line (OOL) while avoiding excessive idling or inefficient operation modes.
- The **electric motor works effectively** in both **traction** and **regenerative braking** modes, without extreme current peaks, thus reducing C-rate-induced battery stress.

This setup proves ideal for real-world HEV control, where minimizing fuel consumption is a primary goal, but without compromising battery health excessively. α = 0.95 guaranties a **well-balanced, aging-aware EMS** that ensures both **performance and durability**.

The Fuel-Optimal Strategy: α = 1

When α is set to 1, the controller is fully focused on minimizing fuel consumption, without taking into account battery aging. As expected, this leads to:

- The **lowest possible fuel consumption**, which reflects the fuel-optimal solution.
- The **most aggressive use of the battery**, both in discharging and recharging phases. The battery absorbs nearly all the dynamic fluctuations in power demand, allowing the engine to run at a nearly constant and efficient power level.
- While this enhances overall drivetrain efficiency, it comes at the cost of a **severe reduction in battery life:** over 50% shorter “distance to EoL” compared to aging-aware strategies.
- The **RMS values** of battery power are **significantly higher**, indicating **intense usage**. The current peaks are both frequent and large, particularly harmful from a battery degradation perspective.

- The **SOC trajectory** features frequent **charge-depleting** and **recharging patterns**, consistent with an aggressive hybrid strategy designed to shift energy demands away from the engine.

Despite its excellent efficiency, this approach is **unsustainable for long-term usage** in real HEV systems where battery replacement or degradation has significant cost and performance implications.

Final Remarks

The analysis **confirms** the **strength** of **Dynamic Programming** in solving multi-objective control problems with foresight. It also highlights that:

- Small adjustments in α (e.g., from 0.95 to 1) can produce **meaningful differences** in battery aging without major gains in fuel economy.
- Conversely, even moderate values of α (e.g., 0.05) already yield **substantial fuel savings** compared to purely aging-aware strategies, with only marginal battery wear.

Ultimately, the choice of α depends heavily on the specific application and design goals, but incorporating both terms in the objective function is essential to achieve a well-balanced and effective energy management strategy.

α = 0.95 provides the most balanced and robust solution, offering nearly optimal fuel performance with significantly reduced battery aging. It is thus recommended as the ideal trade-off configuration for practical HEV energy management systems.