Politecnico di Torino

Master's Degree in
Mechanical Engineering

Master's Degree Thesis

A.Y. 2021/2022

Control variables accuracy in Dynamic Programming for Hybrid Electric Vehicles

Advisors:

Prof. Ezio Spessa

Prof.ssa Daniela Misul

Dott. Ing. Federico Miretti

Candidate:

Alex De Franco

1 Summary

Introduction

Optimal control theory is an approach that have replaced the classical analysis to determine the acceptable system's performance. This approach implies to design a system to minimize a performance criterion. In Optimal Control theory it is necessary to define a set of variables called states and a set of inputs called controls to control a process. After that, an appropriate performance measure is chosen to translate the system's physical requirements into mathematical terms with the aim to find a time variation of the inputs that lead the system to perform under a minimized performance criterion. This time variation can be expressed by a relationship called optimal control law or optimal policy with the resulting history called optimal trajectory.

The most prominent methods are:

- Pontragyn's maximum principle;
- Bellman's principle of optimality.

The second one is the main topic of this thesis.

1. Dynamic Programming

Dynamic programming is one of the methods used to find a control function that minimize a performance criterion. This leads to an equation with a solution achievable using a digital computer. First of all, an optimal control is defined in the following form:

$$u^*(t) = f\big(x(t),(t)\big) \tag{1.1}$$

where f represents the optimal control law or optimal policy. Notice that being t an argument of f, the optimal control is time-varying. In Dynamic Programming, a concept called *the principle of optimality* is used to find an optimal policy.

1.1 Deterministic problems

In DP problems, a sequence of states is generated under the influence of control. When the problem is on finite horizons, the system has a finite number N of stages. In deterministic systems, the evolution of the problem depends only on the state and the control at time k:

$$x_{k+1} = f_k(x_k,u_k,w_k), \qquad k = 0,1 \dots \dots \dots \dots N-1 \tag{1.2}$$

- k is the time index;
- x_k is the state of the system or state space at time k;

- u_k is the control variable or control space selected from a set $U_k(x_k)$;
- w_k is an external disturbance;
- f_k is a function that models the state updating from *k* to *k+1;*
- *N* is the number of stages.

Another part of the problem is given by an additive function called *costfunction*. This one is expressed as $g_k(x_k,u_k)$ and represents the cost incurred at time k accumulated over the time. Taking the initial state x_0, the total cost of a sequence is:

$$J(x_0;u_0,\dots\dots u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k,u_k,w_k)$$

(1. 3)

where $g_N(x_N)$ represents the terminal cost at the end of the process. To obtain the optimal value, a cost minimization is necessary with respect to the control constrains over all the sequence. Consequently, the optimal cost function is:

$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0,\dots\dots,N-1}} J(x_0;u_0,\dots\dots u_{N-1})$$

(1. 4)

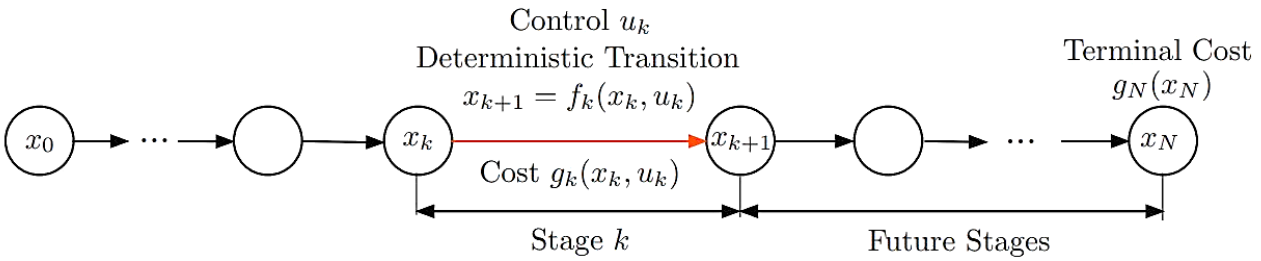

Figure 1.1: illustration of the main elements of a deterministic problem (Bersekas, 2019)

1.2 Stochastic problems

In many problems, the state x_k can take the values from a discrete state (i.e. Integers). In these problems the transition between the states is specified in terms of probabilities, reason why to know which is the probability at time *k* that the next state will be x_{k+1} is necessary. It can be expressed as (Bertsekas, 2005):

$$p_{i,j}(u,k) = P\{x_{k+1} = j \mid x_k = i,\ \ u_k = u\}.$$

(1. 5)

The description of the transition in terms of a discrete-time system equation is possible, and in particular

$$P\{\omega_k = j \mid x_k = i, u_k = u\} = p_{i,j}(u,k)$$

(1. 6)

is the probability distribution of the random parameter ω_k. On the other hand, a transition probabilities starting with a discrete-state system can be also written. This means that a discrete-state system can equivalently be written in terms of a difference equation or in terms of transition probabilities. However, a better way a priori doesn't exist, but it depends on the complexity of the problem. Moreover, in this thesis a deterministic approach is adopted to resolve the problem.

1.3 The principle of optimality

The Figure 1.2: is an example of one or two *optimal paths* for a multistage decision process:

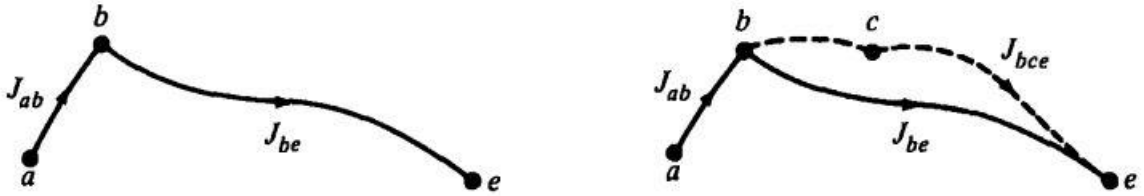

Figure 1.2: possible optimal paths from a to e *(Kirk, 2004)*

In the left figure, J_{ab} and J_{be} represent the costs to cover their respective segments. The minimum cost J_{ae}^* from a to e is expressed as:

$$J_{ae}^* = J_{ab} + J_{be}$$

(1. 7)

In conclusion if a-b-e is the optimal path from a to e, b-e is the optimal path form b to e.
To prove this, two possible optimal paths to go from a to e (right figure) are assumed. Consequently, it would be:

$$J_{bce} < J_{be}$$

(1. 8)

and

$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J^*_{ae}.$$

(1. 9)

The eq. (1. 9) is satisfied only if the condition that a-b-e is the optimal path from a to e is violated. Thus, the first conclusion is proved.

Bellman (Dynamic Programming., 1957) has called this property *the principle of optimality*:

"an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The principle of optimality is at the base of DP algorithm. In terms of optimal state and control space, It can be stated (Bersekas, 2019):

"Let $\{u_0^*, \dots \dots \dots, u_{N-1}^*\}$ be an optimal control sequence, which together with x_0 determines the corresponding state sequence $\{x_0^*, \dots \dots \dots, x_N^*\}$. Consider the subproblem whereby we start at x_k^* at time k and wish to minimize the "cost-to-go" from time k to time N,

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N)$$

Over $\{u_k, \dots \dots \dots, u_{N-1}\}$ with $u_m \in U_m(x_m)$, $m = k, \dots, N-1$. Then the truncated optimal control sequence $\{u_k^*, \dots \dots \dots, u_{N-1}^*\}$ is optimal for this subproblem."

This statement suggests that to determine the optimal cost function apart going backwards from the last stage is possible, solving gradually the "tail subproblem". Particularly, after solving the problem, the solution of shorter time length is chosen.

1.4 Example: Scheduling Problem (Bersekas, 2019)

In this paragraph, the principle of optimality is applied to solve the scheduling problem represented in the Figure 1.3.

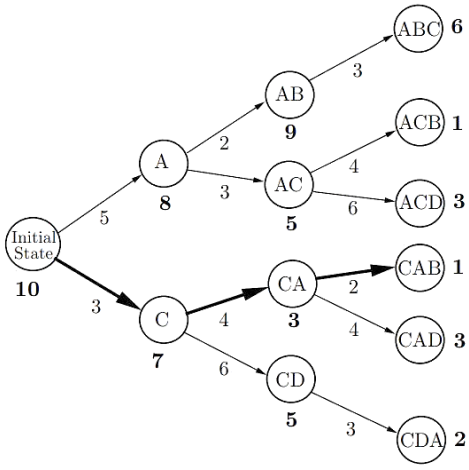

Figure 1.3: Deterministic Scheduling Problem

Notice that the cost of each decision is shown next to the corresponding arc. The constrains of this problem are:

- the operation B can be done only after operation A;
- the operation D can be done only after operation C.

Following the principle of optimality, the tail subproblem must be resolved and consequently the portion of an optimal schedule must be optimal. The resolution's procedure starts from the last stage to arrive at the original problem. The results are reported in the Table 1-1, Table 1-2 and Table 1-3.

State	Possible operation	Cost	Terminal Cost	Optimal cost
AB	C	3	6	9
AC	B D	4 6	1 3	5
CA	B D	2 4	1 3	3
CD	A	3	2	5

Table 1-1: tail subproblem of length 2

State	Possible operation	Cost	Optimal Cost of the previous subproblem	Optimal cost
A	AB AC	2 3	9 5	8
C	CA CD	4 6	3 5	7

Table 1-2: tail subproblem of length 3

State	Possible operation	Cost	Optimal cost of the previous subproblem	Optimal cost
Initial State	A C	5 3	8 7	10

Table 1-3: original problem of length 4

The optimal schedule is represented by CABD with an optimal cost of 10. This is clearly shown in the Figure 1.3 following the thick line.

2 Energy Management in Hybrid Electric Vehicle

Hybrid electric vehicles (HEVs) are characterised by the presence of one or more electric machines and a reversible energy storage device. In comparison to conventional vehicles, HEVs have a reduction of fuel consumption and pollutant emissions. Moreover, thanks to the presence of the additional elements mentioned above, HEVs can exploit regenerative breaking, idle off capability, power assist ability and potential for engine downsizing. However, the introduction of an additional storage device involves new degrees of freedom due to the splitting of energy sources. Consequently, to find an efficient way to manage the power demand between the engine and the battery is more complicated. To resolve this problem, a control layer called *energy management strategy* is necessary (*Figure 2.1* (Serrao, Onori, & Rizzoni, 2011)).

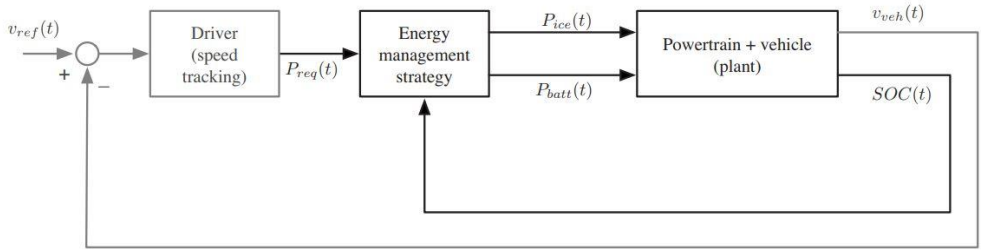

Figure 2.1: energy management system in a HEV

The main target of the control is to guarantee a minimum fuel consumption possibly coupled with a reduction of emissions over a driving cycle without a penalty in the performance. Another target can be the maximization of battery life. With respect to the pattern shown in *Figure 2.1*, the first element is the vehicle speed controller represented by the human driver and modelled like simple feedback controller in simulation. Through the speed controller, the total power P_{req} that the powertrain must provide to follow a velocity profile is demanded. After that, the energy management system decides how to split the power between the engine (P_{ice}) and the battery (P_{batt}). In this study, the vehicle is considered as a dynamic system with two decoupled parameters: the vehicle speed and the state of charge of the battery (SOC). Thanks to decoupling, only the state of charge is considered as a state variable of the energy management problem, while the speed is controlled independently. Moreover, speed transients are neglected because they are much faster with than the state of charge variations. Following these hypotheses, the energy management problem can be seen as an optimal control problem where the aim is to minimize a performance index (fuel consumption) over a driving cycle

through a sequence of instantaneous control actions (power split values). There are a lot of strategies and techniques available in literature to resolve the problem. The main methods are listed below:

- *Numerical methods for global optimization*, i.e. Dynamic Programming.
- *Numerical methods for local optimization*, i.e. Model predictive control.
- *Analytical optimization methods*, i.e. Pontryagin's minimum principle.
- *Instantaneous minimization methods*, i.e. ECMS.
- *Heuristic methods*, i.e. Rule-based control.

2.1 Energy Management Problem and DP

In Hybrid electric vehicles, as said before, the optimal control problem is solved through a sequence of controls that lead to minimize a performance index. Consequently, the cost is defined as (Serrao, Onori, & Rizzoni, 2011):

$$J\left(x(t_0), u(t), x(t_f)\right) = \phi\left(x(t_0), x(t_f)\right) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt$$

(2. 1)

where

- t is the time;
- $u(t)$ is the control variable;
- $x(t)$ is the state variable;
- $[t_0, t_f]$ is the optimization horizon;
- $L(\cdot)$ is the instantaneous cost;
- $\Phi(\cdot)$ is the terminal cost.

The state variable, as mentioned in chapter *2,* can be represented by the state of charge of the battery (SOC), while the control variables can depend on powertrain architecture and number of energy paths between the energy sources and the wheels. Usually, control variables are defined as:

$$u(t) = \{P_{batt}(t), \rho_1(t), \dots\dots\dots\dots, \rho_{m-1}(t)\}$$

(2. 2)

where

$P_{batt}(t)$ is the output battery power and $\rho_i(t)$ represent the additional variables due to how the battery power is split.

If a quasi-static engine model is considered, fuel consumption can be seen as a function of the engine torque T_{ice} and speed ω_{ice}, while vehicle speed ω_{veh} and power demand P_{req} depend on external inputs. The dynamics of the powertrain components are neglected (see par. 2). Finally, some constraints are applied to optimization including:

- system dynamics;
- initial state value;
- terminal state value;
- instantaneous state limitations;
- instantaneous control limitations.

DP is used to solve the optimal energy management problem. Control variables u_k represent the power split between the engine and the battery (or other rechargeable energy storage system) at successive time steps. Moreover, DP is a numerical method that gives the global optimal solution proceeding backwards in time (i.e., from the end of the driving cycle).

To apply DP to this problem, the system is written in a discrete time-form. After that, a control policy Π is defined as a control sequence in which the optimization horizon is divided into *N* time steps. Following the principle of optimality (*par. 1.3*), the total cost is:

$$J_0(\Pi) = \phi(x_N) + \sum_{0}^{N-1} L(x_k, u_k, t_k)$$

(2. 3)

while the optimal policy can be written as:

$$\Pi^* = arg\min_{\Pi} J_0(\Pi).$$

(2. 4)

The optimal cost-to-go is computed starting from backward iterations and, after that the law Π_k^* is defined for each time value, results are stored into a matrix. After that the entire cycle is simulated, the path with the lowest cost represents the optimal solution (Onori, Rizzoni, & Serrao, 2015). The solution obtained with DP is optimal for a discretized problem, but it can be considered also optimal in a continuous one if the simulation grid is fine enough. However, the use of a backward procedure implies that the solution can be obtained only offline for a well-known driving cycle. Moreover, the online implementation is impractical also due to high computational load.

3 Case study: Parallel Hybrid Vehicle

The vehicle chosen for the strategy's application is comparable to a p2 parallel hybrid compact city car. The parallel hybrid drive train has the engine and the electrical machine that supply their mechanical power directly to the wheels. The main advantages compared to series hybrid are:

- One electrical machine is enough;
- Better efficiency;
- A smaller electrical machine.

On the contrary, the control of the parallel drive train could be more complex.

In p2 architecture, EM is placed between engine and transmission unit with capability to decouple transmission from the engine through an appropriate clutch.

The powertrain architecture (Figure 3.1) and the data (Table 3-1) used for the simulation are showed below:

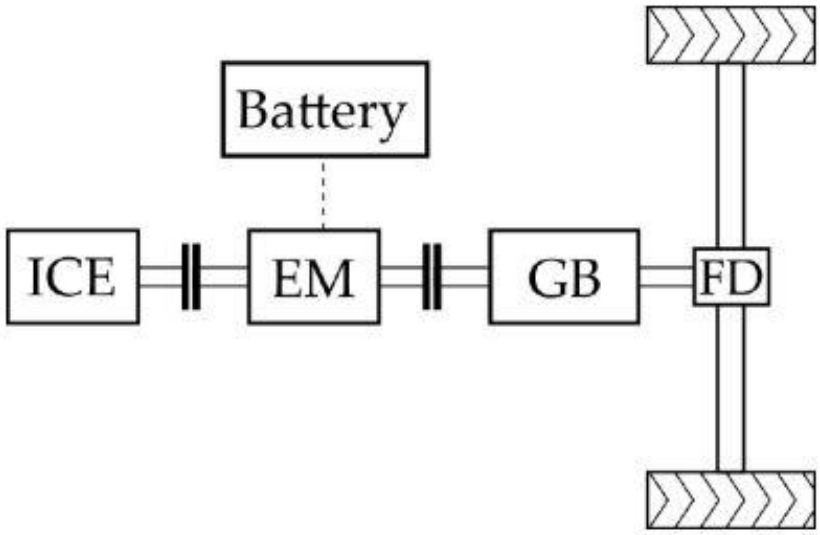

Figure 3.1: p2 parallel hybrid powertrain architecture

Vehicle mass	1080 kg
EM maximum power	60 kW
Engine maximum power	68.5 kW
Battery maximum power	12 kW
Battery energy capacity	5.4 MJ

Table 3-1: vehicle data plate

The vehicle is tested over a WLTC, an emission test cycle. It consists in four different phases (low, middle, high and extra-high) in which duration, distance, velocity and acceleration vary to predict more accurately the exhaust emissions and fuel consumption under real-world driving conditions. The total duration of the cycle is 1800 s with a total distance of about 23 km.

3.1 Model description

In order to apply Dynamic Programming, the first step is to define some parameters from the physical model of the vehicle. First of all, the vehicle speed and acceleration in time are defined as exogenous inputs that are inputs known a priori for each stage of the problem. In fact, velocity and acceleration in time depend on the above-mentioned driving cycle. After that, control and state variables are defined. The model is determined by a *backward* approach in which force follows velocity and the tractive force is calculated starting from the inertia force (Onori, Rizzoni, & Serrao, 2015):

$$F_{tract} = F_{inertia} + F_{roll} + F_{aero} \tag{3.1}$$

where

$F_{roll} + F_{aero}$ represents the resistance vehicle force due to rolling resistance and aerodynamics resistance.

The next step is to determine wheel speed and acceleration:

$$\omega_{wh} = \frac{v}{r_{wh}} \tag{3.2}$$

and

$$\dot{\omega}_{wh} = \frac{a}{r_{wh}} \tag{3.3}$$

with

v and a that are respectively vehicle velocity and acceleration given by driving cycle and r_{wh} that represents wheel radius.

Consequently, wheel torque is determined as:

$$T_{wh} = F_{tract}\, r_{wh}. \tag{3.4}$$

3.1.1 Transmission

Transmission section is modelled starting from the final drive:

$$\omega_{fd} = \tau_{fd}\omega_{wh} \tag{3.5}$$

and

$$\dot{\omega}_{fd} = \tau_{fd}\dot{\omega}_{wh} \tag{3.6}$$

where

ω_{fd} and $\dot{\omega}_{fd}$ represent final drive speed and final drive acceleration and τ_{fd} is the final drive speed ratio.

Then, final drive torque is expressed as:

$$T_{fd} = \begin{cases} \frac{T_{wh}}{\tau_{fd}} + T_{loss,fd} & if\ T_{wh} > 0 \\ \frac{T_{wh}}{\tau_{fd}} - T_{loss,fd} & if\ T_{wh} \leq 0 \end{cases} \tag{3.7}$$

with $T_{loss,fd}$ that represents the final drive loss.

After that, shaft is modelled passing via gearbox:

$$\omega_{shaft} = \omega_{fd}\tau_{gb}(GN) \tag{3.8}$$

and

$$\dot{\omega}_{shaft} = \dot{\omega}_{fd}\tau_{gb}(GN) \tag{3.9}$$

where

ω_{shaft} and $\dot{\omega}_{shaft}$ are shaft velocity and shaft acceleration and τ_{gb} is the gearbox speed ratio expressed as function of gear number.

Gearbox efficiency is calculated from an efficiency map and it depends only by gear number:

$$\eta_{gb} = \eta_{gb}(GN). \tag{3.10}$$

Before to determine shaft torque, gearbox loss must be calculated. These last are expressed as:

$$T_{loss,gb} = \begin{cases} \frac{1-\eta_{gb}}{\eta_{gb}} \frac{T_{fd}}{\tau_{gb}} & if\ T_{fd} > 0 \\ (1-\eta_{gb})\frac{T_{fd}}{\tau_{gb}} & if\ T_{fd} \leq 0 \end{cases}. \tag{3.11}$$

Finally, shaft torque can be calculated as:

$$T_{shaft} = \frac{T_{fd}}{\tau_{gb}} + T_{loss,gb} + J_{gb}\dot{\omega}_{fd}$$

(3. 12)

where

$J_{gb}\dot{\omega}_{fd}$ represents the gearbox inertia.

3.1.2 Control Variables

The control variables are represented by the gear number and by another one called torque split. The gear number depends on the type of vehicle, while torque split, or torque split ratio, is a parameter that allows to determine how the demanded torque T_{req} is split between the torque provided from the engine T_{eng} and the torque provided from the EM T_{EM}. In fact, following the next relation:

$$T_{req} = T_{eng} + T_{EM}.$$

(3. 13)

Previously it is possible to define T_{req} as a function of vehicle's speed and gear number:

$$T_{req} = T_{shaft} + J_{eng}\dot{\omega}_{eng} + J_{EM}\dot{\omega}_{EM}$$

(3. 14)

where

- T_{shaft} is the torque derived from the transmission;
- J_{eng} and J_{EM} represent the moments of inertia of the engine and the EM;
- $\dot{\omega}_{eng}$ and $\dot{\omega}_{EM}$ are the engine and EM acceleration.

Consequently, the torque split can be defined based on engine and EM torques, for example like ratio between T_{eng} and T_{req}, or following other parameters like the battery current i_b. The torque split choice represents the main element of this problem and the main topic of this thesis.

3.1.3 State Variable

The state variable is represented by SOC of the battery. Its dynamics can be derived from a simple battery internal resistance model, shown in Figure 3.2.

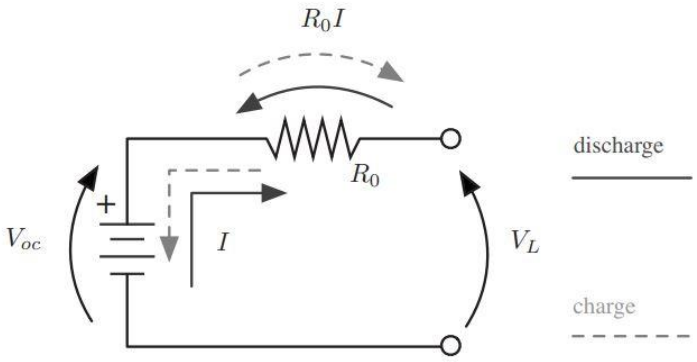

Figure 3.2: battery model

Following the model, the battery power can be written as:

$$V_L(t)I(t) = P_{batt}(t) = V_{oc}(x)I(t) - R_0(x)I^2(t) \tag{3.15}$$

where

- $I(t)$ is the battery current;
- $V_L(t)$ is the load voltage at the terminals;
- $V_{oc}(x)$ and $R_0(x)$ are the open-circuit voltage and the equivalent internal resistance and they are function of the state of charge of the battery (SOC), indicated by (x).

From the previous expression, $I(t)$ can be derived as:

$$I(t) = \frac{V_{oc}(x) - \sqrt{V_{oc}^2(x) - 4R_0(x)P_{batt}(t)}}{2R_0(x)} \tag{3.16}$$

and finally SOC dynamics is determined:

$$\dot{SOC} = \frac{I(t)}{C_{batt}} \tag{3.17}$$

where C_{batt} is the battery energy capacity.

Being $I(t)$ written in function of $P_{batt}(t)$, this one is evaluated as

$$P_{batt}(t) = \begin{cases} \eta_{inv} P_{EM} & if\ P_{EM} \geq 0 \\ \dfrac{1}{\eta_{inv}} P_{EM} & if\ P_{EM} < 0 \end{cases} \tag{3.18}$$

and the power adsorbed or generated by the electrical machine P_{EM} is previously calculated as

$$P_{EM} \begin{cases} \eta_{EM}\omega_{EM}T_{EM} & if\ T_{EM}\omega_{EM} \geq 0 \\ \frac{1}{\eta_{EM}}\omega_{EM}T_{EM} & if\ T_{EM}\omega_{EM} < 0 \end{cases}$$

(3. 19)

where η_{inv} and η_{EM} are respectively inverter and electric machine efficiency.

3.1.4 Constraints

To ensure a proper optimization of the problem, a constraints' set-up that represents the operational constraints of a powertrain must be defined. The first constrain is about engine torque: it cannot exceed its limit torque (that is function of the engine speed)

$$T_{eng} \leq T_{eng,max}(\omega_{eng}).$$

(3. 20)

Turning to the electrical machine torque, it must be included in a range that marks the generation and the motor mode:

$$T_{EM,min} \leq T_{EM} \leq T_{EM,max}.$$

(3. 21)

Moreover, when the vehicle is braking (i.e. $T_{req} < 0$), the electrical machine cannot operate in motor mode (powertrain constraint).

Regarding the battery, its current cannot exceed a pre-set range of charge and discharge:

$$I_{dis,lim} \leq I \leq I_{ch,lim}.$$

(3. 22)

Finally, the last constraints are about SOC:

- The terminal SOC must be equal to the initial value $SOC_i = SOC_f$;
- The SOC must stay into a range on value for the entire simulation.

3.2 Simulation introduction

The simulation environment used for the problems is DynaProg, an open-source MATLAB toolbox that solves multi-stage deterministic optimal decision problems through Dynamic Programming (Miretti, Misul, & Spessa, 2021). The main inputs to give are the state variable *x* (sec. 3.1.3) with related constraints, the control variables (sec. 3.1.2) u_1 and u_2, the number of stages *N*, a function that represents the physical model of the vehicle and velocity and acceleration in time as exogenous inputs *w*. In output, a cost profile is achieved, in this case represented by fuel consumption. The main target of the following simulations is to explore the vehicle's behaviour by varying torque split and discretization levels. The main scripts used for the simulations are reported in the Appendix (chap.7).

4 Simulation Analysis

To start the simulation, the first step is to define a problem's set up. In particular, a grid for state and control variables should be defined. Moreover, some constraints should be imposed, as mentioned in sec. 3.1.4. As concerns the state variable x, the grid is represented by a range in which the SOC must stay and for all the simulations the chosen range is the following: $0.4 \leq x \leq 0.7$ with a discretization equal to 0.001. The discretization's choice is very important because the more the grid is fine, the more the cost-to-go is accurate. The main constraint imposed to x is about the initial and the final state. In fact, initial and final state must be equal ($SOC_i = SOC_f$) and in this case the x initial state is 0.6. Consequently, the x final state must be equal to 0.6, but in order not to make the problem overconstrained, the x final state can stay in the following range: $0.599 \leq x_{final} \leq 0.601$. An example of the plot results is shown below in Figure 4.1:

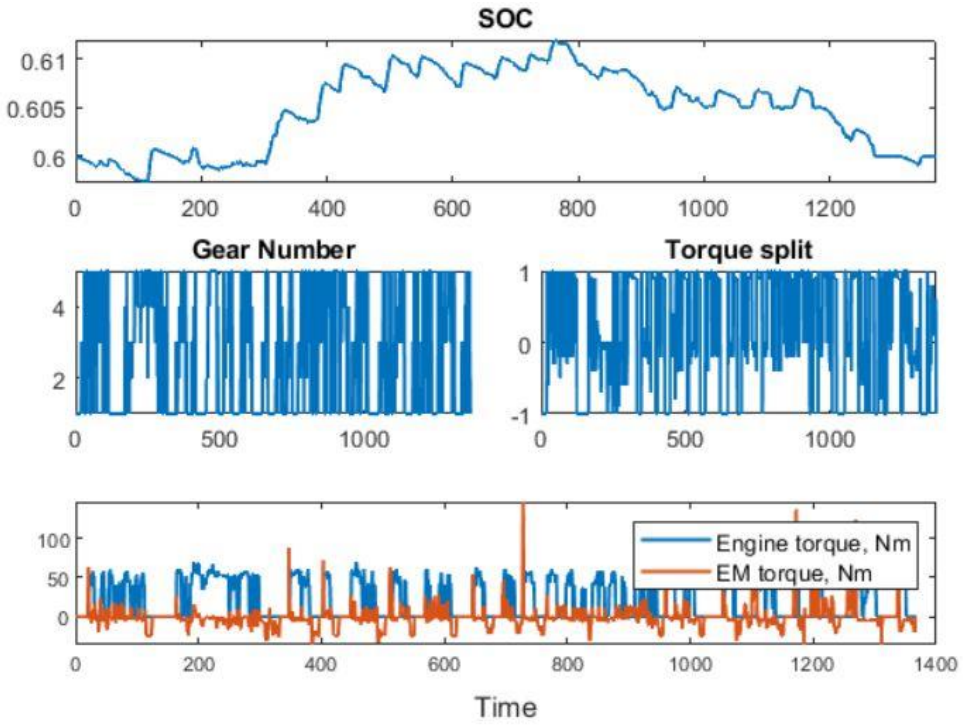

Figure 4.1: plot of the simulation results on Dynaprog

The three plots have the time of the driving cycle on the x-axis and the value of SOC, gear number, torque split and Engine and EM torque on the y-axis. Engine and EM torque plot can be replaced by Fuel Consumption plot as an alternative. In this respect, the expected cost in terms of Fuel Economy must stay between 4 and 5 l/100, that is comparable to a compact city car's result.

Regarding to the control variables, the u_1 $grid$ is represented by the gear number and it is equal to 5. So, it has u_1 $grid = [1\ 2\ 3\ 4\ 5]$. On the other hand, the u_2 $grid$ depends on torque split and the discretization levels. All the simulations are run over five discretization levels equal to 11, 21, 41, 81

and 121, while the tested torque split ratios are nine, by implicating nine different vehicle models to be simulated. The models, indicated by letters from A to H, are the following:

- model A: the engine torque T_{eng};
- model B: the e-machine torque T_{em};
- model C: the battery current i_b;
- model D: the normalized engine torque τ_{eng};
- model E: the normalized e-machine torque τ_{em};
- model F: the normalized battery current ι_b;
- model G: the engine torque-split factor α_{eng};
- model H: the e-machine torque-split factor α_{em};

4.1 Model A: the engine torque T_{eng}

The first simulated model has the torque split depending on the engine torque T_{eng}. The main target is to explore the entire operative range of the engine and consequently the torque split is equal to the operative range of the engine toque. According to the vehicle data provided for all the simulations, the maximum engine torque is equal to **130,5 *Nm***, while the minimum is fixed to **0 *Nm***. As a result, the range is:

$$0\ Nm \leq u_2 \leq 130.5\ Nm \tag{4. 1}$$

and this represents exactly the entire operative range of the engine torque $T_{eng} = u_2$.
From the previous relation, the next one can be written as:

$$T_{req} = T_{eng} + T_{EM} = u_2 + T_{EM} \tag{4. 2}$$

and after obtaining T_{req} from the eq. (3. 13), it has:

$$T_{EM} = T_{req} - u_2. \tag{4. 3}$$

After that, all the necessary constrains (sec. 3.1.4) are inserted as unfeasibility and the simulation is running up to the five discretization levels (chap.4). Thanks to increasing of discretization, the torque split grid is finer and it can get more accurate results.

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-1:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.599246	0.69	3.78
21	0.599006	0.66	3.64
41	0.599019	0.65	3.57
81	0.599003	0.65	3.54
121	0.599006	0.64	3.52

Table 4-1: simulation results model A

From these results, notice that with increasing discretization, fuel consumption and fuel economy are lower. On the other hand, the final SOC moves away from the target, while staying inside the acceptable range.

The vehicle's behaviour in terms of power provided of each part is shown in Figure 4.2:

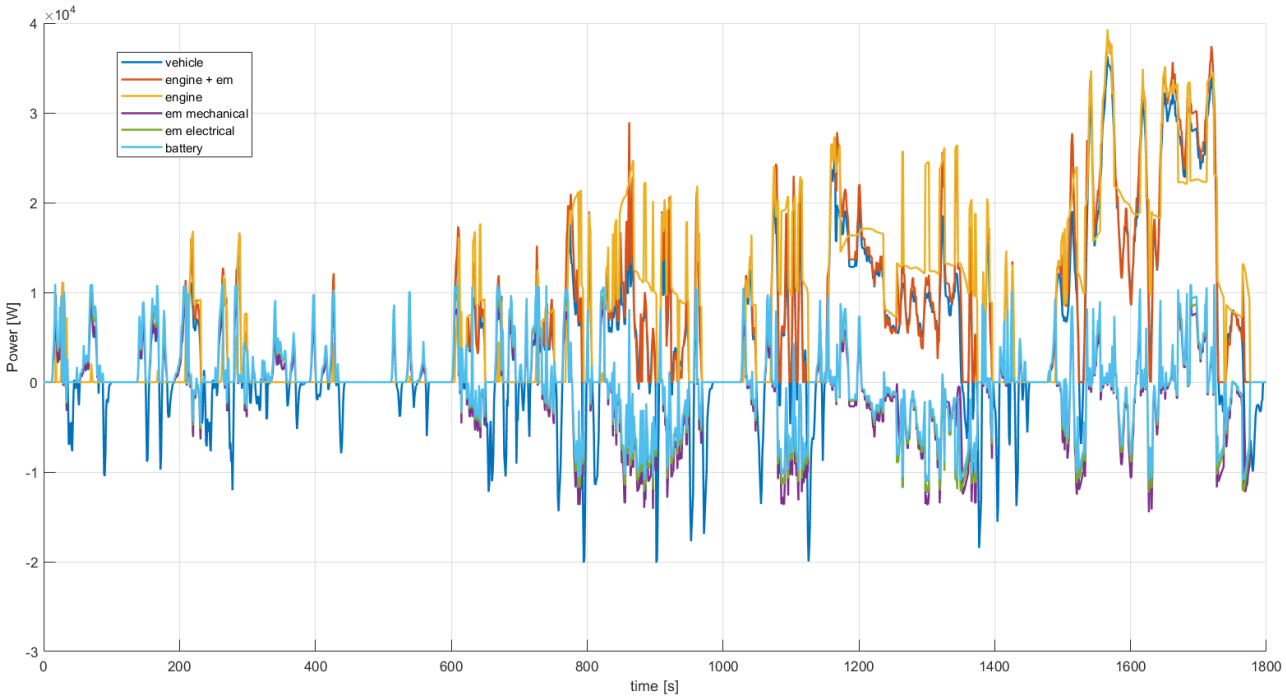

Figure 4.2: power distribution plot model A

In this model, power split's management is good especially in the "*high phase*" of the driving cycle, as shown by Figure 4.3:

Figure 4.3: High phase detail model A

The model provides good results in terms of fuel consumption and fuel economy, remaining in line with the initial expectations.

Another interesting aspect to analyze concerns the gear number variation with to respect torque split. In Figure 4.4, this can be appreciated:

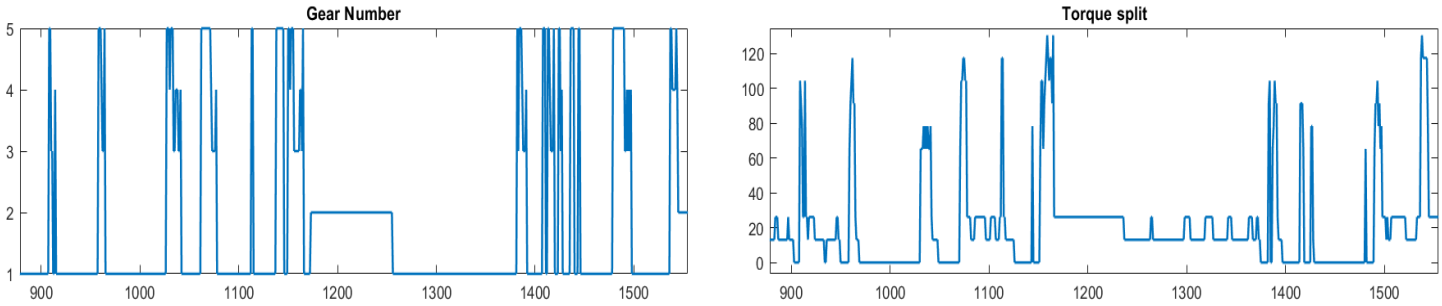

Figure 4.4: gear number variation model A

Gear number is directly proportional to torque split. In fact, when torque split increases, the gear number increases too. This is expected because torque split depends directly on the engine torque that consequently influences the gear number's choice.

The main advantage of this model is the possibility to explore all the entire operative range of the engine, but a lot of operative points are not investigated because of the discretization. On the contrary, if we increased the discretization, the computational time would be very high. Moreover, this model does not give any information about the vehicle powerflow except for the pure electric mode when $u_2 = 0$.

4.2 Model B: the e-machine torque T_{em}

Unlike the model A (par. 4.1), this model has the torque split that depends on the electrical machine torque T_{em}. In particular, u_2 *grid* is equal to the entire operative range of the electrical machine and, from the data provided, it is from **-300 *Nm*** to **300 *Nm***. Consequently, the electrical machine T_{em} can be explored through its entire operative range making the grid increasingly finer thanks to the five discretization levels. In this case, the main constraints are imposed to the engine and to the battery in addition to the powertrain constraint (sec. 3.1.4). Finally, it has:

$$-300\ Nm \leq u_2 \leq 300\ Nm \tag{4.4}$$

with $T_{em} = u_2$.

From the eq. (4. 4), the next one is determined:

$$T_{req} = T_{eng} + T_{EM} = T_{eng} + u_2 \tag{4.5}$$

and obtaining T_{req} from eq. (3. 14), it has

$$T_{eng} = T_{req} - u_2. \tag{4.6}$$

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-2:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.600047	0.85	4.65
21	0.599389	0.82	4.49
41	0.599273	0.81	4.43
81	0.599108	0.80	4.37
121	0.599102	0.78	4.29

Table 4-2: simulation results model B

Also in this model, increasing discretization corresponds to lower fuel consumption and fuel economy. The final SOC is very near to the target, especially in the first discretization level.

The results about power's performance are shown below in Figure 4.5:

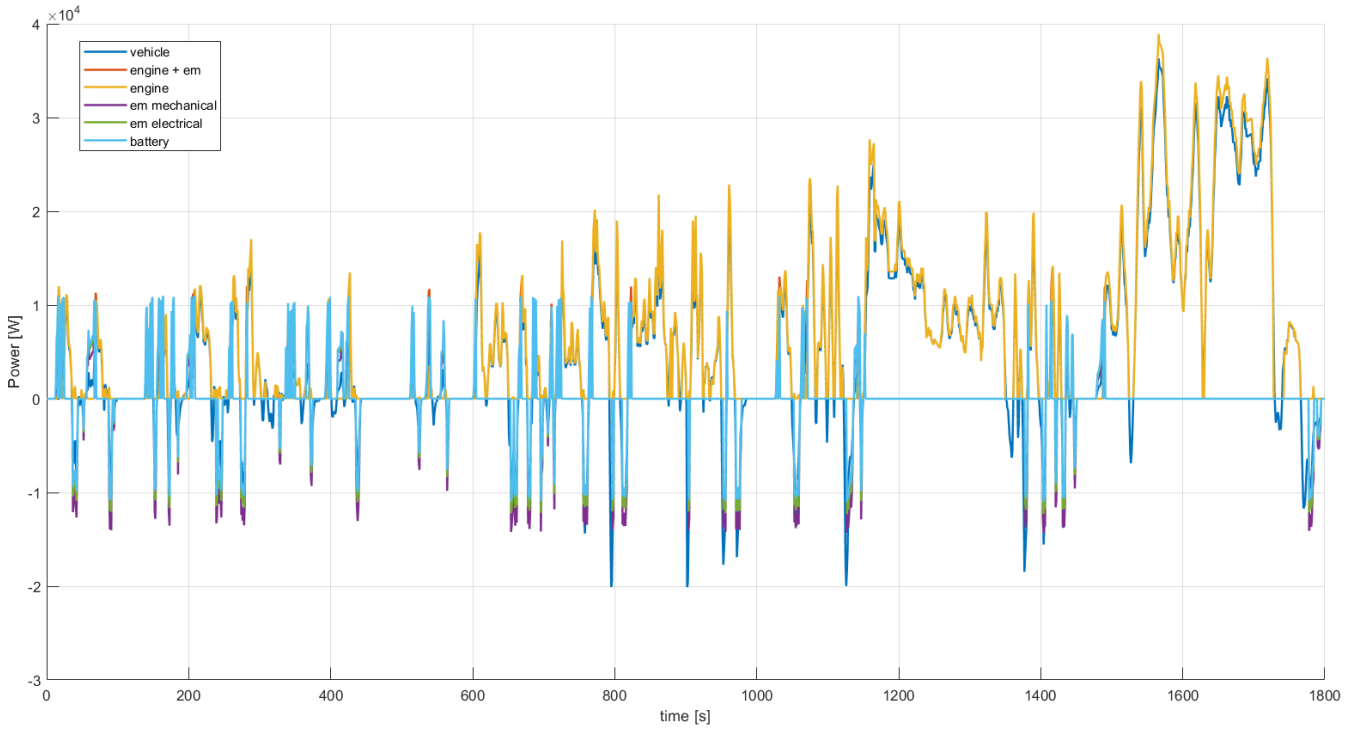

Figure 4.5: power plot results model B

In this case, power split is less evident and probably this can imply a small increase in fuel consumption. An example of this is detectable in the high phase detail (Figure 4.6) of the cycle, where the power provided by the engine is predominant:

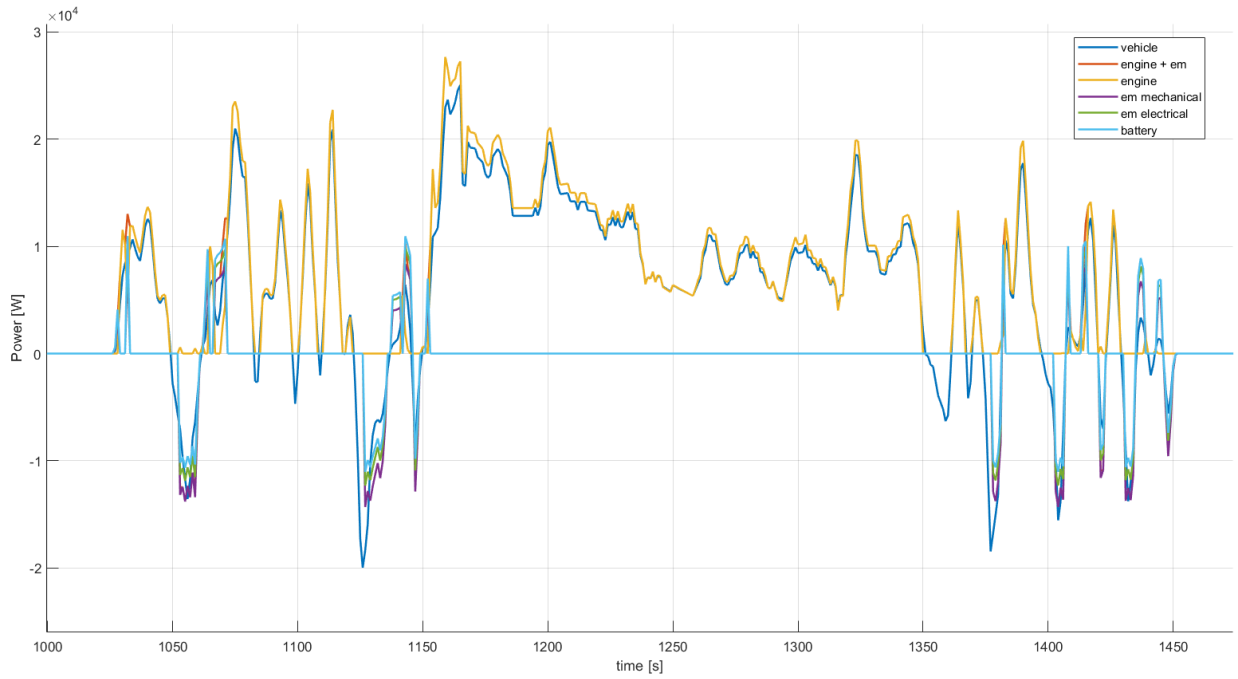

Figure 4.6: High phase detail model B

Anyway, Fuel consumption and Fuel Economy remain in line with the expected results.

Regarding gear number variation, Figure 4.7 shows the obtained results:

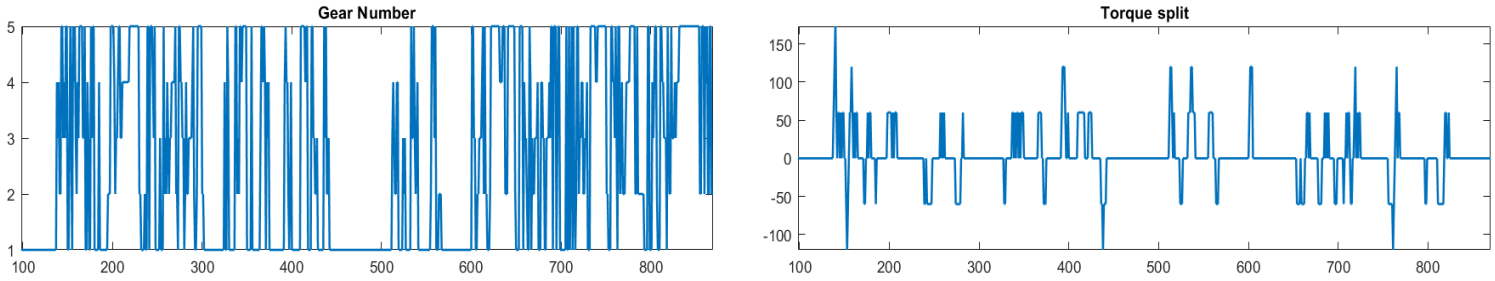

Figure 4.7: gear number variation model B

In this case, there is not direct proportionality between torque split and gear number but there are some considerations to do. First of all, when torque split reaches its maximum or minimum value, the gear number is low probably due to pure electric or battery charge mode. Then, when torque split is zero (EM torque = 0), there is not a specific correlation between the two parameters and the gear number variation probably depends only on the engine torque.

One advantage of this model is to explore all the operative range of the electric machine torque but, same way as the model A, a lot of points are not investigated. At the same time, with increasing discretization too much, the computational time would be very high, more than the model A in this case. Finally, the pure thermal mode can be identified when $u_2 = 0$, but there is not any information about powerflow.

4.3 Model C: the battery current i_b

The model C has the torque split depending on the battery current i_b. In particular, $u_2\ grid$ is exactly equal to the entire admissible operative range of the battery current and in this case, it has:

$$-54\ A\ \leq u_2 \leq 54\ A.$$

(4. 7)

However, this choice implies a substantial change in the vehicle physical model. In fact, model A and model B have the battery current i_b expressed as a function of battery power P_{batt} as shown in the eq. (3. 16), while in this case, the battery power P_{batt} must be written as a function of the battery current i_b. Consequently, the equations become:

$$i_b = u_2$$

(4. 8)

and

$$P_{batt} = V_{oc} i_b + R_0 i_b^2.$$

(4. 9)

From here, the electrical machine power P_{EM} is calculated through the "electrical torque T_{el}" defined as $\frac{P_{EM}}{\omega_{EM}}$ and used as input into an efficiency map. Finally, T_{EM} is obtained and T_{eng} can be found following the eq. (3. 13) and (3. 14).

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-3:

Discretization Level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.599444	0.89	4.87
21	0.599167	0.89	4.87
41	0.599028	0.89	4.86
81	0.599028	0.89	4.85
121	0.599028	0.89	4.85

Table 4-3: simulation results model C

The results show how fuel consumption and fuel economy are almost completely unaffected by discretization. Instead, final SOC moves away from the target with increasing discretization.
The power distribution is represented by Figure 4.8:

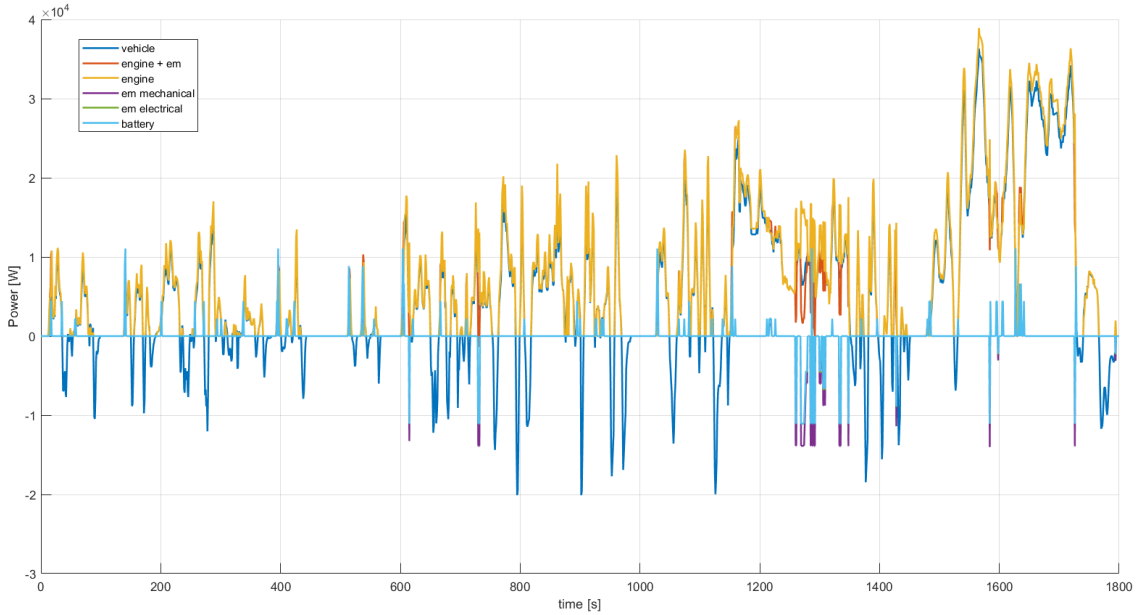

Figure 4.8: power plot results model C

In this model, power split's management is quite good, but the engine power turns out to be sometimes predominant. The high phase detail is shown below in Figure 4.9:

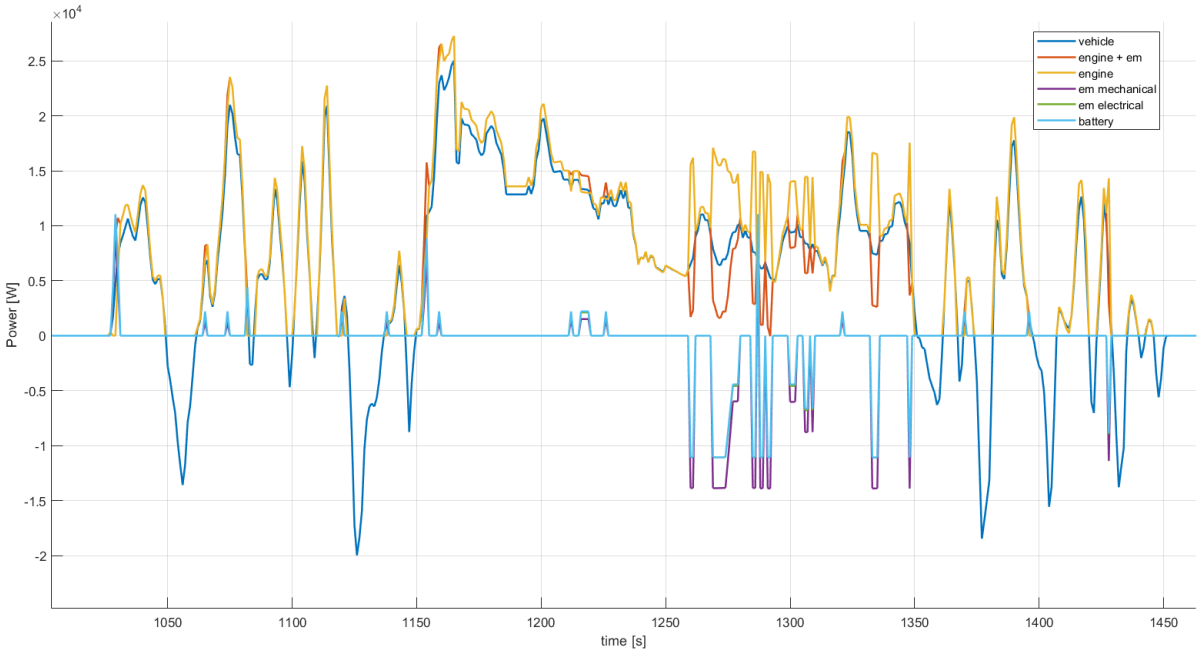

Figure 4.9: High phase detail model C

Fuel consumption and Fuel Economy are slightly higher, but they are in line with the expected one and the vehicle's behaviour remains consistent.

Results about gear number variation are shown by Figure 4.10:

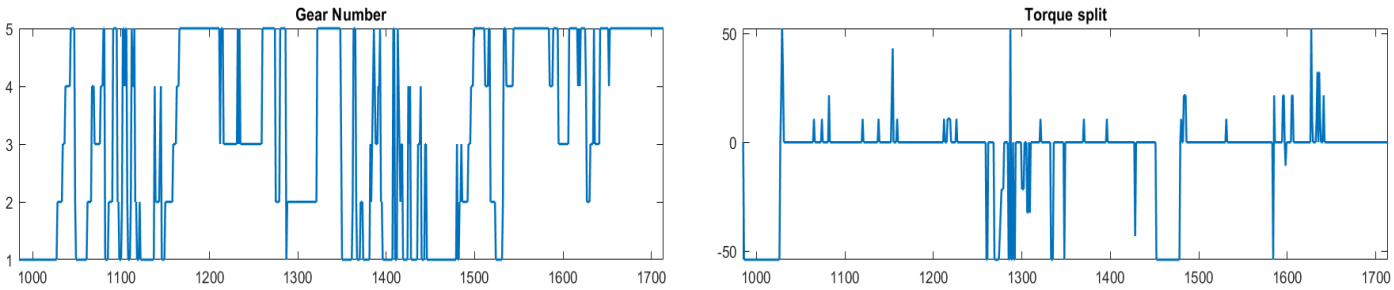

Figure 4.10: gear number variation model C

The correlation between torque split and gear number is very similar then model B. In fact, also in this case, there is not a direct proportionality and when torque split reaches its maximum or minimum values, the gear number is low. On the other hand, when torque split is zero, battery current is zero and this also implies EM torque equal to zero. Consequently, gear number variation depends only on the engine torque.

The best element of this model is surely the simulation of the entire operative range of the battery current. Moreover, using current as torque split implies a simplification of the vehicle physical model since a backwards calculation is not necessary. However, as in other models, the computational time

would be very high when the discretization increased. In addition, this model does not give any information about powerflow.

4.4 Model D: the normalized engine torque τ_{eng}

The torque split of this model depends on the engine torque once again, but in this case referred to the normalized engine torque, written below:

$$u_2 = \tau_{eng} = \frac{T_{eng}}{T_{eng,max}}. \tag{4.10}$$

The idea is to explore the entire operative range of the engine torque (as seen in par. 4.1), but in a more compact way. In fact, thanks to this "normalized" formulation, the u_2 bounds become:

$$0 \leq u_2 \leq 1 \tag{4.11}$$

with $u_2 = 0$ that represents the minimum engine torque (**0** ***Nm***) and $u_2 = 1$ that represents the maximum engine torque (**130.5** ***Nm***).

Consequently, the vehicle physical model is modified and in particular

$$T_{eng} = u_2 T_{eng,max} \tag{4.12}$$

and

$$T_{EM} = T_{req} - u_2 T_{eng,max} \tag{4.13}$$

with T_{req} calculated from eq. (3. 14).

The constraints are the same imposed in the model A and they mainly concern electrical machine, battery and powertrain.

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-4:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.599013	0.69	3.78
21	0.599043	0.66	3.64
41	0.599005	0.65	3.58
81	0.599002	0.65	3.54
121	0.599002	0.64	3.53

Table 4-4: simulation results model D

The simulation results show the discretization addiction to Fuel consumption and Fuel economy. Moreover, final SOC is a bit far from the final target (while remaining inside the permitted range), since the first level of discretization and this could influence simulation accuracy.

The plot results about power's performances are shown below in Figure 4.11:

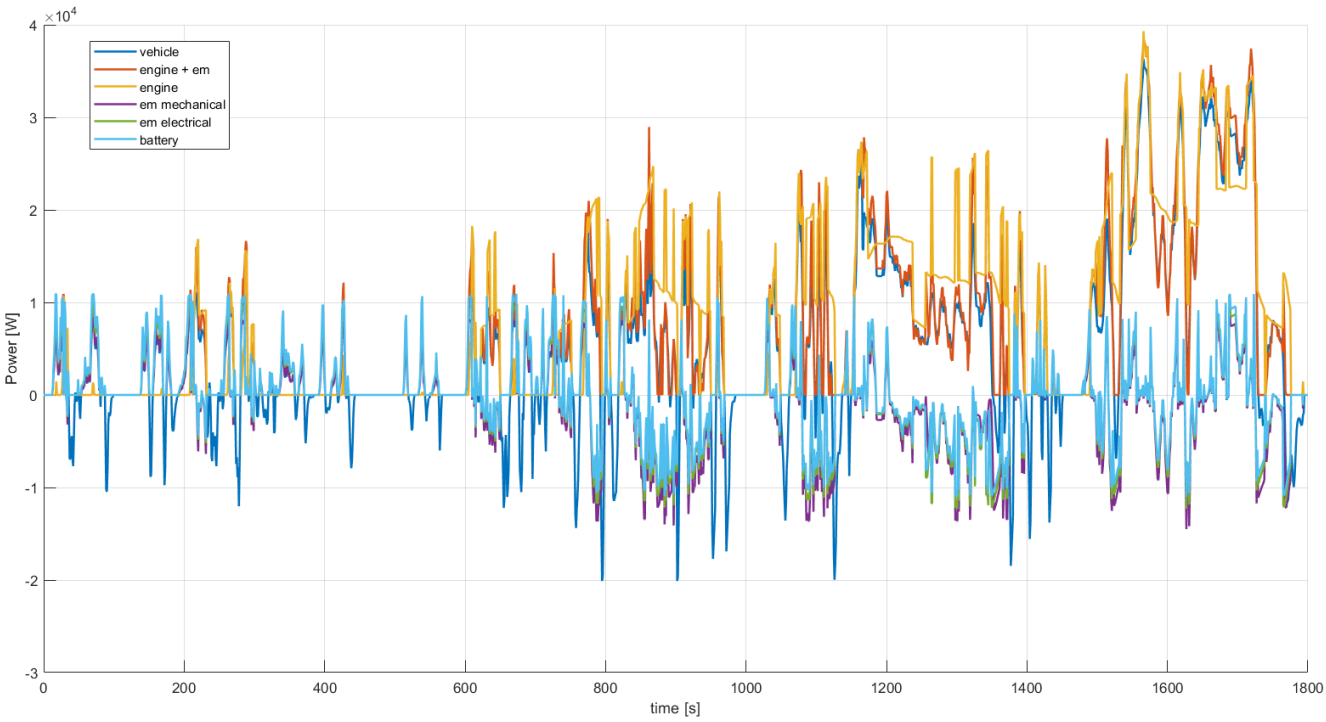

Figure 4.11: power plot results model D

There is a very good power split management that implies a reduction in terms of fuel consumption and consequently Fuel Economy.

"High phase detail" is shown in Figure 4.12:

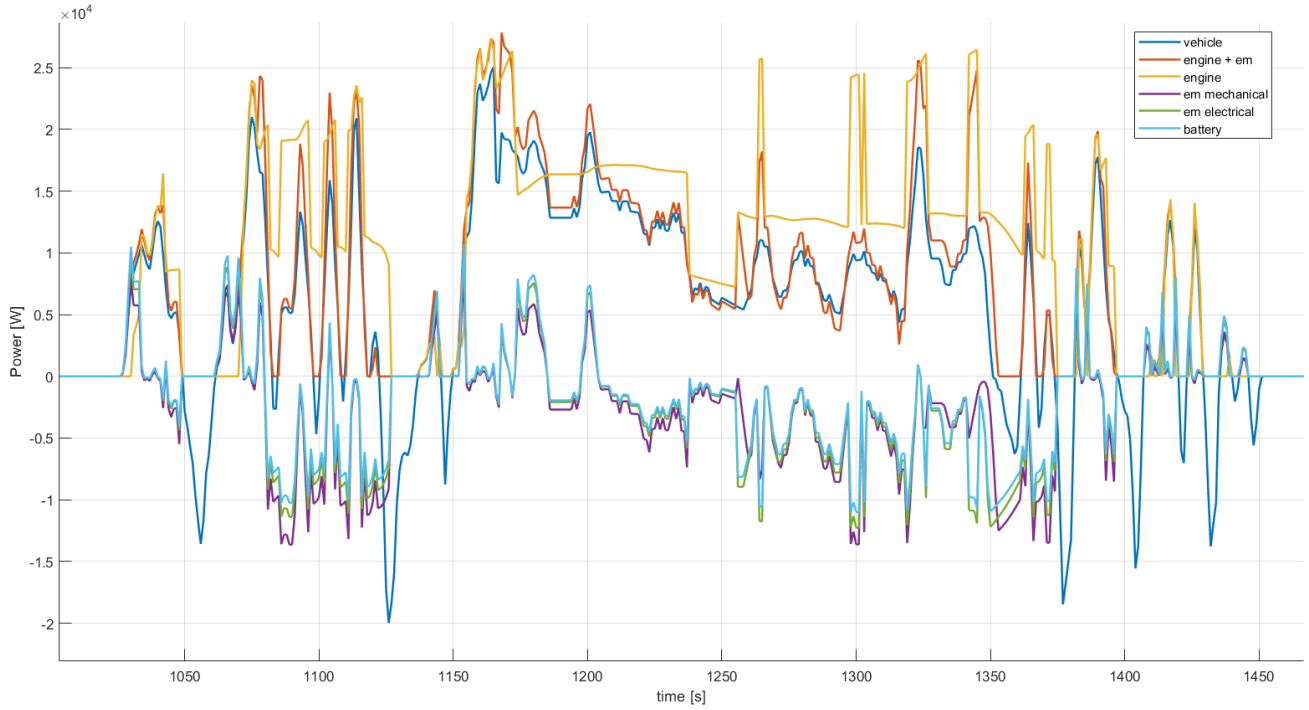

Figure 4.12: High phase detail model D

The Fuel economy's results seem a bit lower than the expected, but still acceptable. In fact, the vehicle's behaviour is coherent and probably this fuel consumption reduction is due to a lower accuracy combined with the torque split's proper choice.

As concerns gear number variation, results are represented by Figure 4.13:

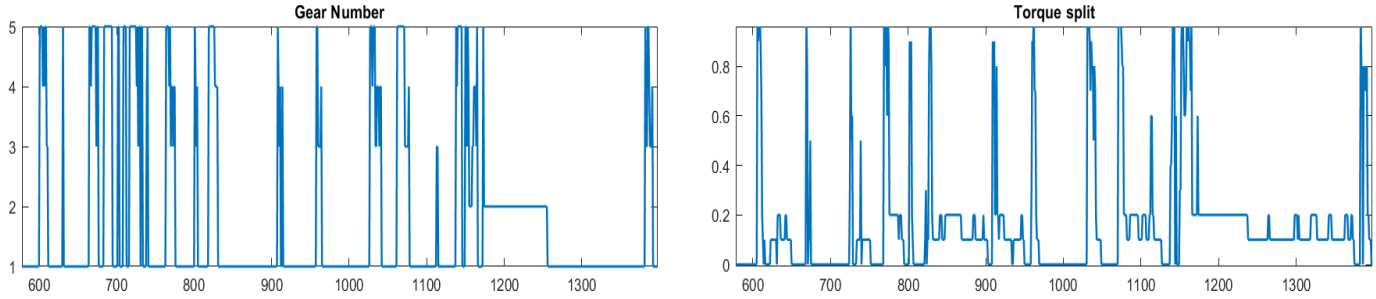

Figure 4.13: gear number variation model D

Essentially, gear number variation and torque split are directly proportional as in model A. In fact, the torque split of this model is directly referred to the engine torque ("normalized" in this case) that influences the gear number variation. The higher the torque split is, the higher gear number is and vice versa.

The main advantage of this model is the capability to investigate a lot of operative points in the engine torque range. Indeed, the normalized torque allows to expand the number of investigated points using the same discretization levels, without affecting the computational time. However, also in this model,

if the discretization increased, the computational time could be higher. And then, when $u_2 = 0$, pure thermal mode is identified, but there is no other information about powerflow.

4.5 Model E: the normalized e-machine torque τ_{em}

In this model, the torque split is represented by a "normalized e-machine torque" along the lines of the model D. In this case, obviously, reference is made to the electrical machine torque and its entire operative range. The normalized e-machine torque is equal to

$$u_2 = \tau_{em} = \frac{T_{EM}}{T_{EM,lim}} \tag{4. 14}$$

when $T_{EM,lim}$ is referred to maximum (300 *Nm)* and minimum (-300 *Nm)* electrical machine torque. Consequently, the u_2 grid is

$$-1 \leq u_2 \leq 1 \tag{4. 15}$$

with $u_2 = -1$ that represents the minimum e-machine torque and $u_2 = 1$ that represents the maximum e-machine torque.

This formulation implies some modifies in the vehicle physical model, in particular:

$$T_{EM} = u_2 T_{EM,lim} \tag{4. 16}$$

and

$$T_{eng} = T_{req} - u_2 T_{EM,lim} \tag{4. 17}$$

with T_{req} calculated from eq. (3. 14).

The unfeasibility is the same of model B and it mainly concerns engine, battery and powertrain.

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-5:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.600572	0.79	4.35
21	0.599092	0.77	4.20
41	0.599095	0.75	4.10
81	0.59906	0.74	4.04
121	0.599022	0.73	4.01

Table 4-5: simulation results model E

The results indicate a correlation between Fuel Consumption and Fuel economy and discretization. Regarding final SOC, the target is good achieved in the first discretization level, while in the others it moves a little bit away, remaining in the admitted range.

The results about power's performances are represented in Figure 4.14:

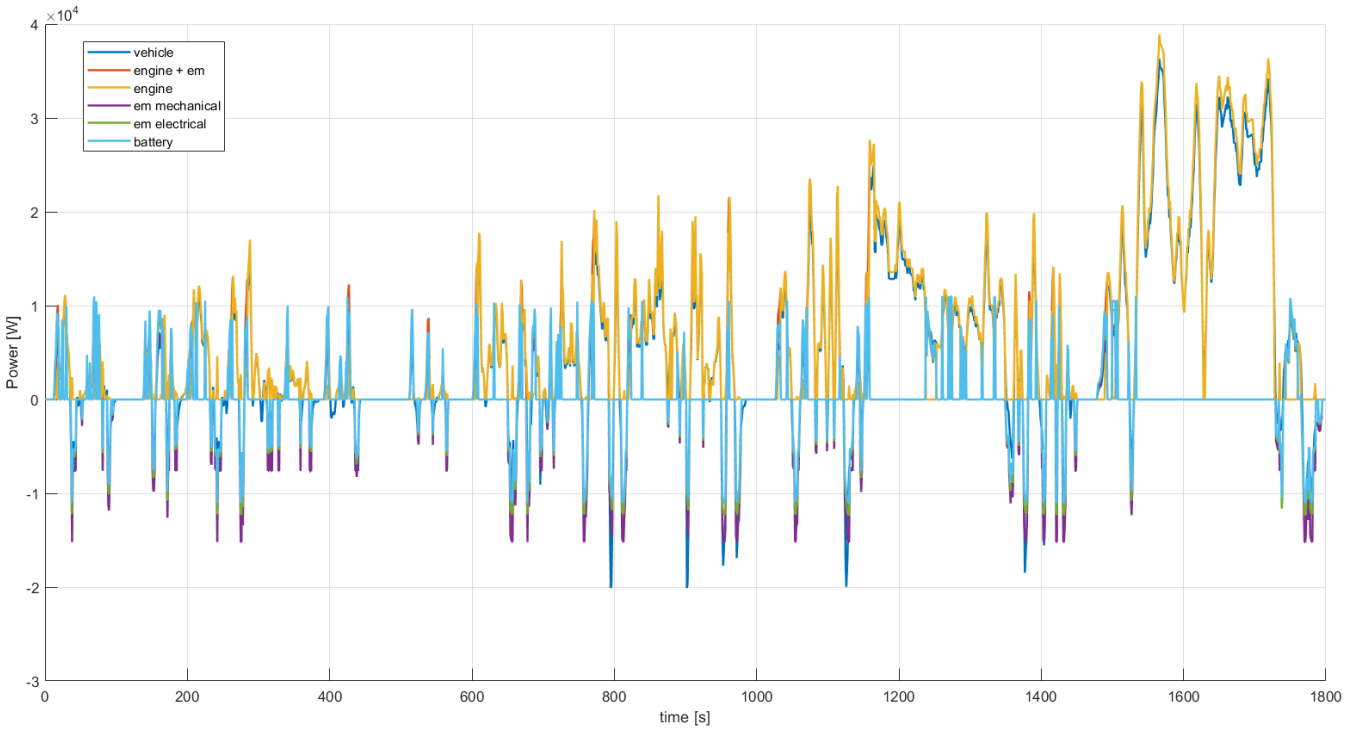

Figure 4.14: power plot results model E

In this case, power split management takes to have predominant engine power in some zones of the cycle. This is evident in the cycle High phase (Figure 4.15):

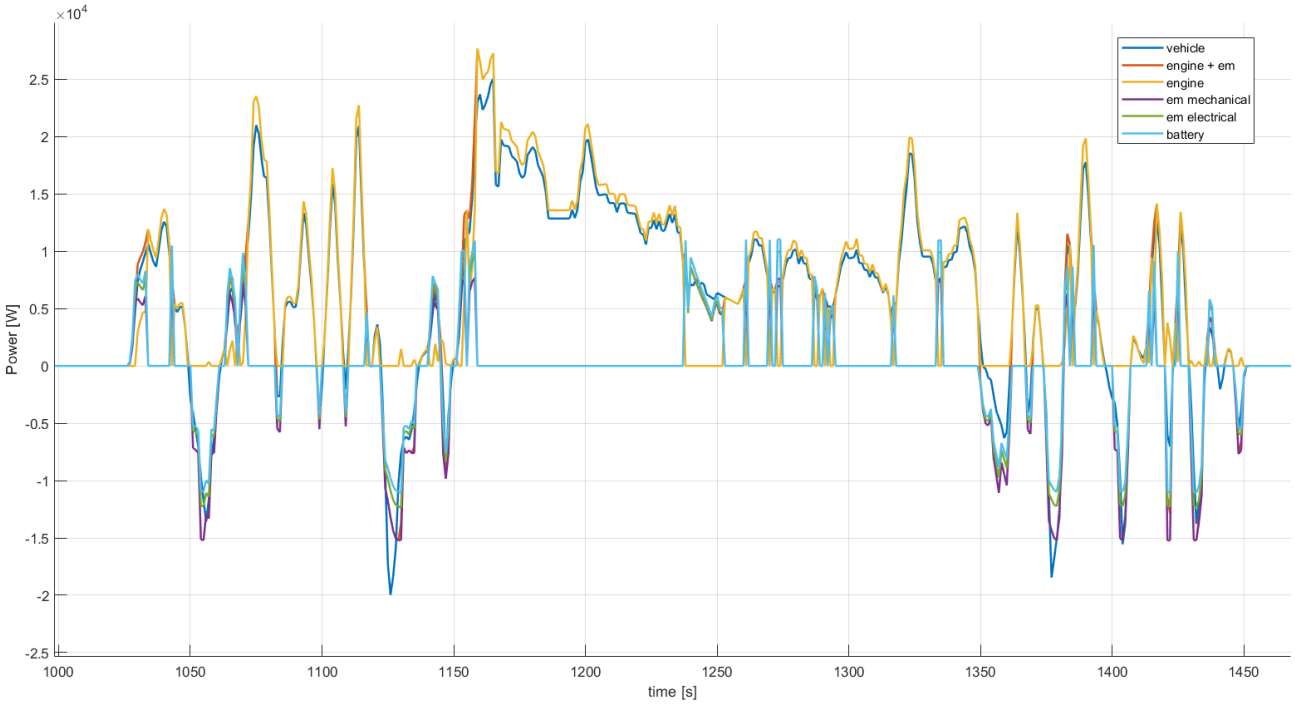

Figure 4.15: High phase detail model E

Notwithstanding the above, the vehicle's behaviour is very good and Fuel consumption and Fuel economy are perfectly in line with the expected one.

Regarding gear number variation, results are plotted below in Figure 4.16:

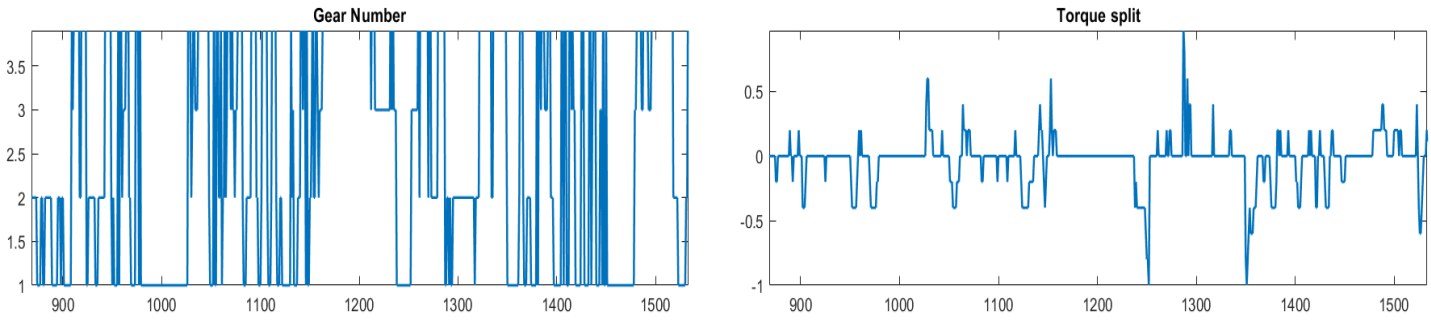

Figure 4.16: gear number variation model E

As seen in model B (sec. 4.2), also in this case there is not direct correlation between the parameters. Consequently, when torque split is around its maximum or minimum value, gear number is low due to pure electric or battery charging mode while a torque split's value equal to zero (EM torque = 0) implies a low or a high gear number probably depending only on the engine torque.

This model can investigate all the entire operative range of the electrical machine, increasing the number of points simulated. Keeping the same discretization levels, the computational time is not particularly affected, but with increasing discretization, the computational time could be higher. Moreover, this model can detect the pure thermal mode when $u_2 = 0$ and it can differentiate the

electrical machine behaviour: when $u_2 < 0$ EM operates in generator mode, while when $u_2 > 0$ the e-machine is in motor mode. However, the model E does not give any information about the rest of the powerflow.

4.6 Model F: the normalized battery current ι_b

In this model, the torque split is referred to the normalized battery current. It is calculated from the ratio between the battery current i_b and $i_{b,lim}$, that represents all the entire operative range of the current. Therefore, the next formulation is written:

$$u_2 = \iota_b = \frac{i_b}{i_{b,lim}}.$$

(4. 18)

However, current's limits are not fixed values, but they depend on the SOC, the state variable x. In fact, it has:

$$i_{b,inf}(x) \leq i_b \leq i_{b,\sup}(x)$$

(4. 19)

and consequently, the normalized battery current ι_b is written as a function of both state and control variables. Meanwhile, thanks to the "normalization", the u_2 grid is:

$$-1 \leq u_2 \leq 1$$

(4. 20)

when $u_2 = -1$ represents $i_{b,inf}$ and $u_2 = 1$ represents $i_{b,\sup}$.

Finally, after defining the battery current as

$$i_b = u_2 i_{b,lim},$$

(4. 21)

all the other parameters are determined through the eq.(4. 9) and following the rest of the procedure seen in par. 4.3. In this case, constraints mainly concern engine, electrical machine and powertrain. The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-6:

Discretization level	Final SOC	Fuel consumption [kg]	Fuel economy [l/100km]
11	0.599444	0.89	4.87
21	0.599167	0.89	4.87
41	0.599028	0.89	4.86
81	0.599028	0.89	4.85
121	0.599028	0.89	4.85

Table 4-6: simulation results model F

In this model, the discretization addiction to Fuel consumption and Fuel economy is void. On the other hand, final SOC is very near to the target in the first discretization level, while it moves away in the others, remaining inside the range anyway.

As regards power's performances results, they can be seen on Figure 4.17:

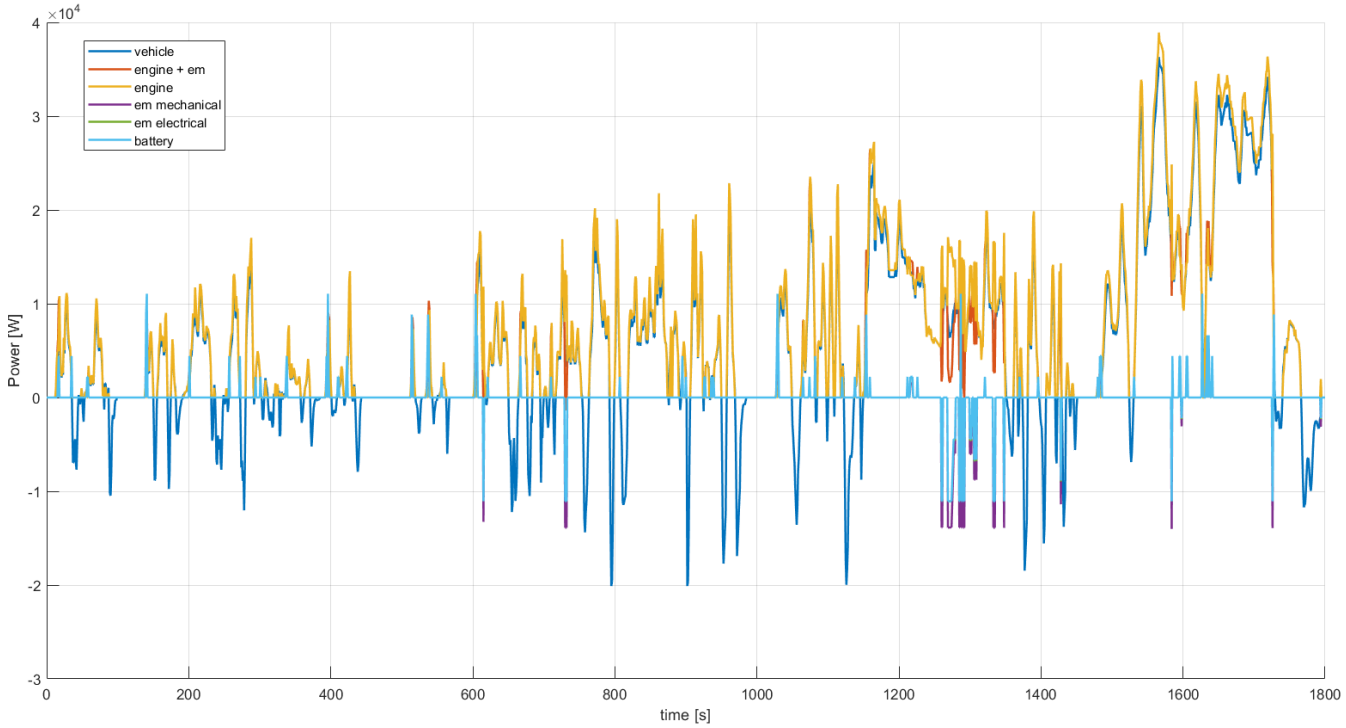

Figure 4.17: power plot results model F

As shown by figure, engine power is predominant in some zones, by implying a less effective power split management. This can be detected in the "High phase detail" represented by Figure 4.18:

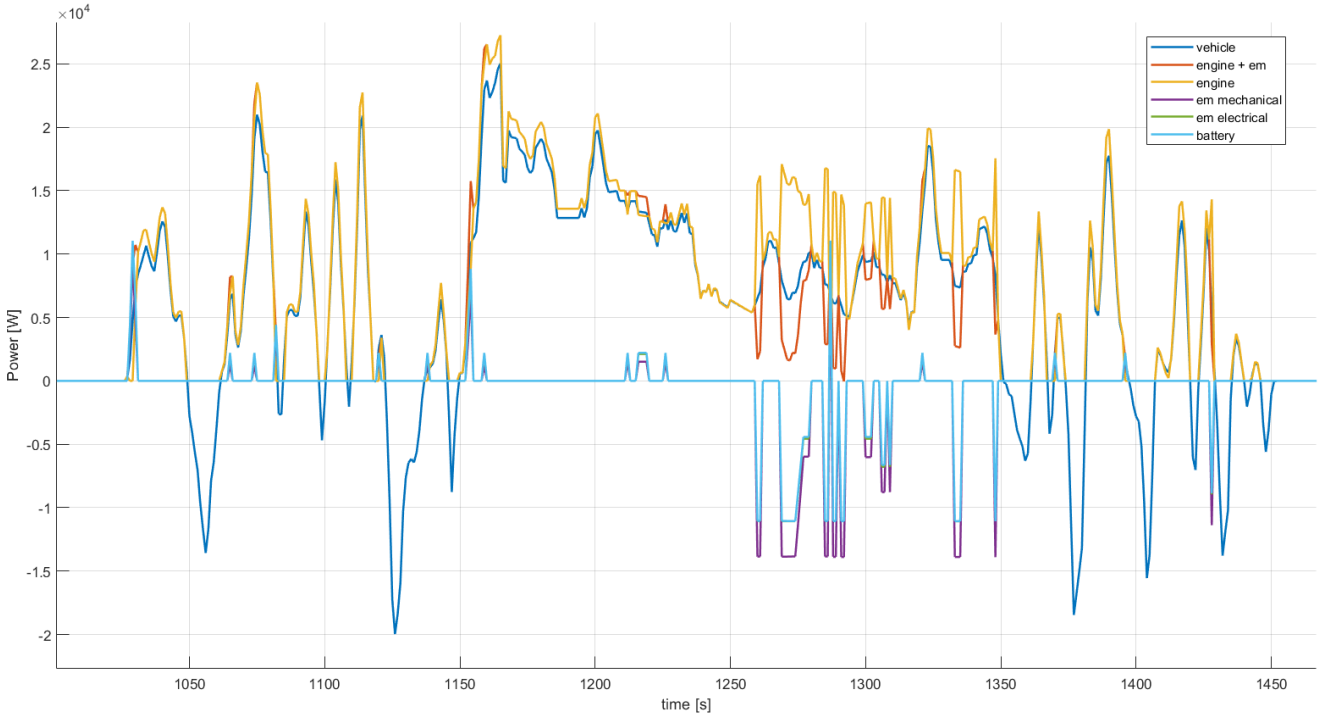

Figure 4.18: High phase detail model F

The power split management implies a small increase in Fuel consumption and Fuel economy that are still in line with the initial expectations anyway. Moreover, the good vehicle's behaviour suggests how this Fuel consumption increase could depend on torque split's proper choice.

Gear number variation expressed as a function of torque split can be seen in Figure 4.19:

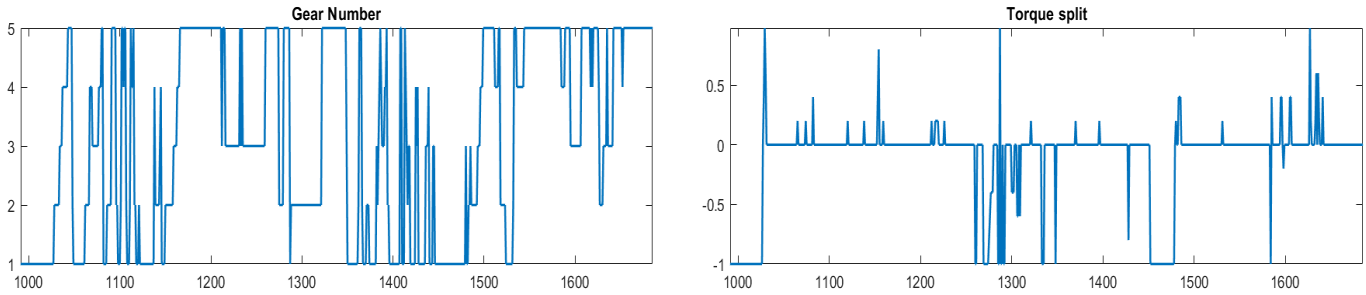

Figure 4.19: gear number variation model F

The correlation between torque split and gear number is not clear. As described in model C (sec. 4.3), when torque split takes maximum or minimum values, gear number is low, while torque split equal to zero (battery current = 0) implies different gear number's choices probably determined by the engine torque.

The best feature of this model concerns the torque split dependence on both control and state variable. As described above, the $i_{b,lim}$ depends on the SOC allowing to have as sort of closed loop control in the model. Moreover, there is a model simplification, as seen in par. 4.3. On the contrary, as a

drawback, increased discretization means increased computational time. In addition, this model does not give any information about powerflow.

4.7 Model G: the engine torque-split factor α_{eng}

After analysing some models with "normalized parameters", in this one torque split is expressed in another different way. In fact, torque-split factor α_{eng} is defined such as the ratio between the engine torque T_{eng} and the demanded torque T_{req}:

$$u_2 = \alpha_{eng} = \frac{T_{eng}}{T_{req}}. \tag{4.22}$$

Using this expression, the percentage of T_{eng} with respect to T_{req} is determined. Consequently, the vehicle's physical model changes:

$$T_{eng} = \alpha_{eng} T_{req} \tag{4.23}$$

and via eq. (3. 13) and eq. (3. 14), it has

$$T_{EM} = T_{req}(1 - \alpha_{eng}). \tag{4.24}$$

The u_2 grid is referred to the T_{eng} operative range and it is

$$0 \leq u_2 \leq 1 \tag{4.25}$$

with $u_2 = 0$ that represents the minimum engine torque (0 *Nm)* and $u_2 = 1$ that implies $T_{eng} = T_{req}$. The constraints are imposed on engine, electrical machine, battery and powertrain.

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-7:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.600999	0.76	4.16
21	0.601	0.75	4.13
41	0.600999	0.75	4.11
81	0.600997	0.75	4.10
121	0.600999	0.75	4.10

Table 4-7: simulation results model G

The results show that the discretization addiction is practically negligible in Fuel Consumption while is slightly apparent in Fuel economy. Instead, final SOC is away from the target of 0.6 in each discretization level, but it remains inside the range reaching the upper bound (0.601) in the second discretization level.

Power's performances results are represented by Figure 4.20:

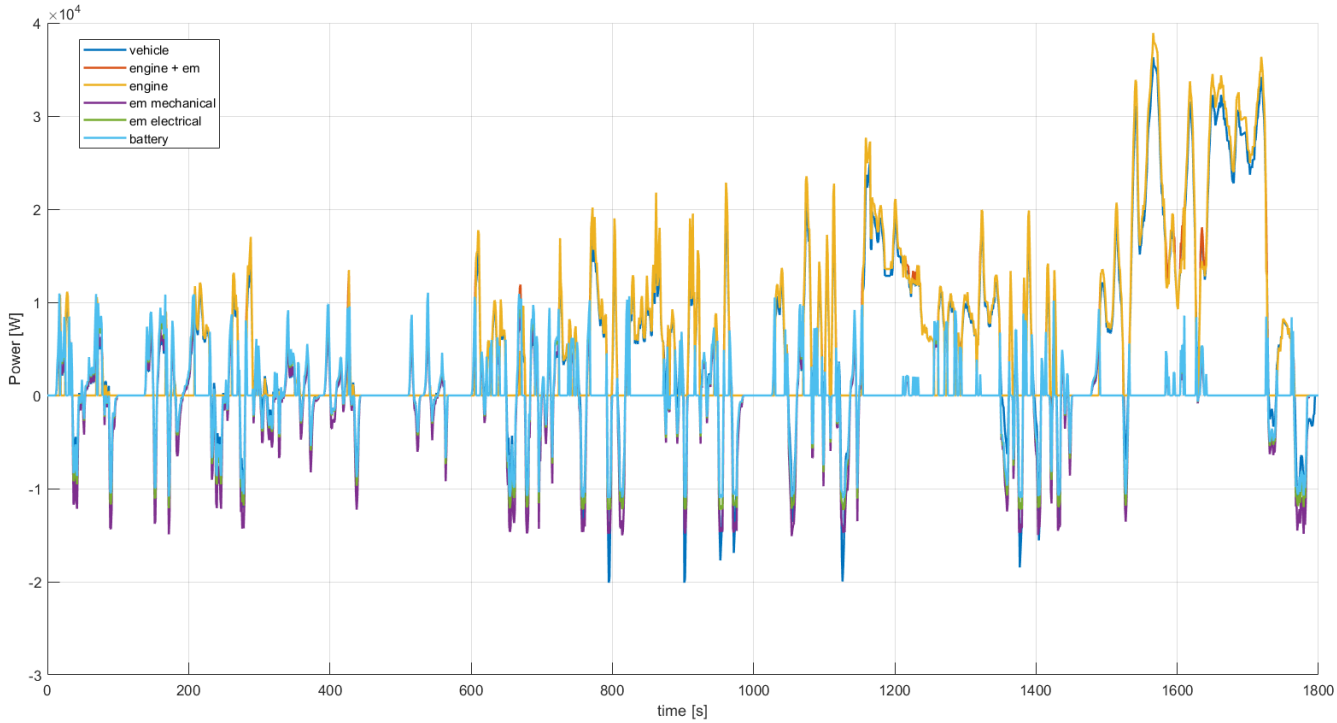

Figure 4.20: power plot results model G

Power split is managed very well and this surely influences Fuel Consumption and Fuel economy.

The "High phase detail" is shown below by Figure 4.21:

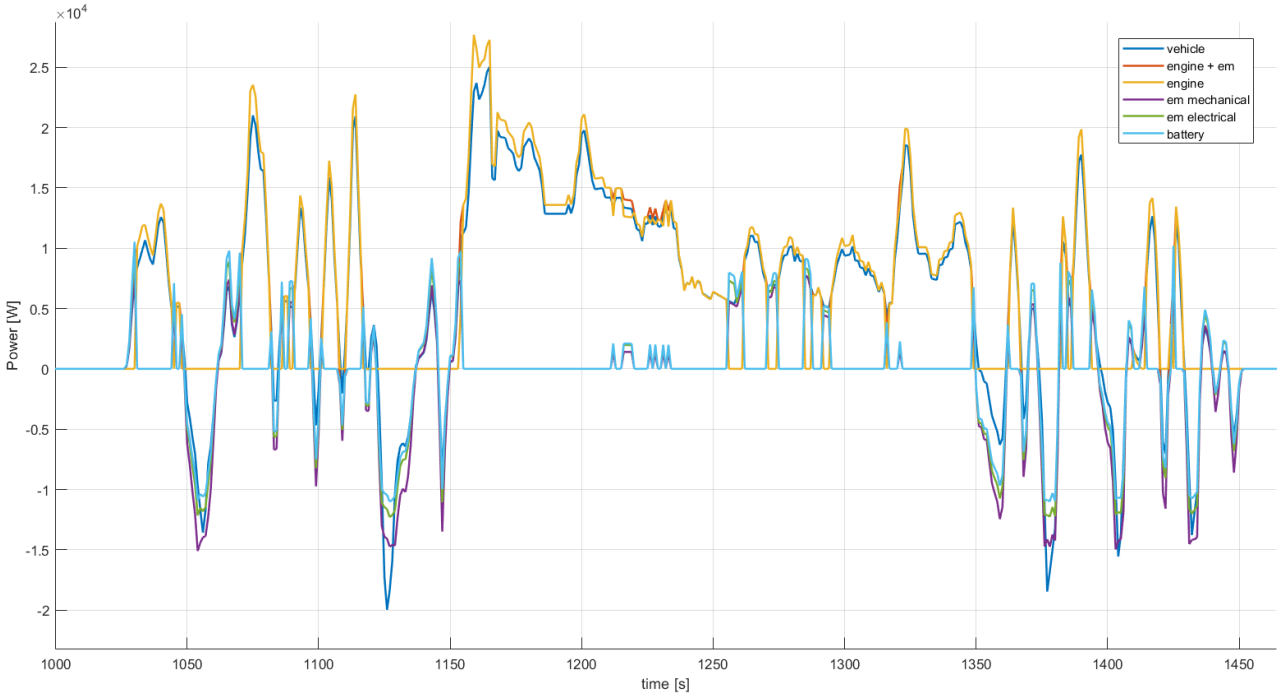

Figure 4.21: High phase detail model G

In this picture, there are some points in which regenerative breaking is detectable. This last is very important because it allows to store the necessary energy that the electrical machine provides during power split. Fuel Consumption and Fuel economy are perfectly in line with the initial expectations. Results about gear number variation are shown in Figure 4.22:

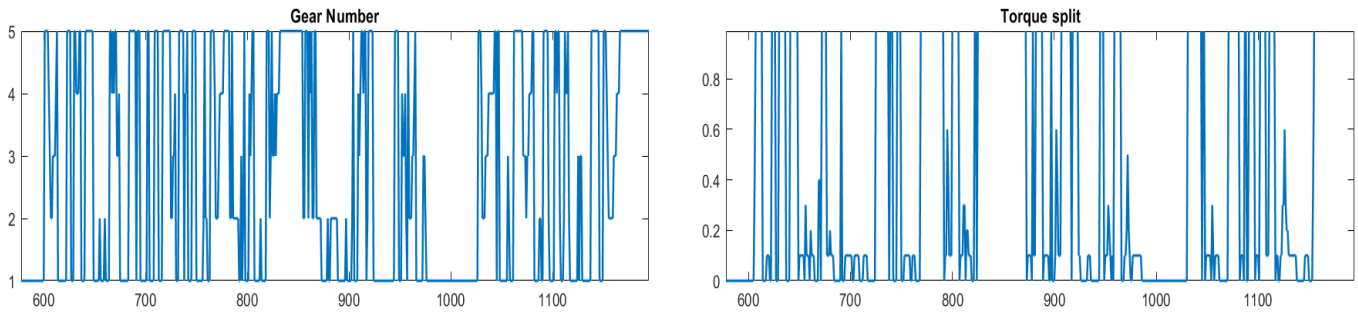

Figure 4.22: gear number variation model G

Gear number variation profile is consistent with the initial expectations. In fact, being torque split expressed as function of engine torque, the expected results should have been like model A and model D. As a confirmation of this, gear number is directly proportional to torque split as described in previous cases (sec. 4.1 and sec. 4.4).

The main strength of this model is to give all the information about powerflow. In fact, it is possible to distinguish pure electric mode when $u_2 = 0$, pure thermal mode when $u_2 = 1$ and power split mode when $0 < u_2 < 1$. In addition, this torque split makes the model more complete thanks to the

presence of T_{req}, in integration with constraints. On the other hand, this model presents some limits. First of all, there are some unexplored operative points of the engine torque and then the electrical machine could be underexploited, never reaching its maximum value.

4.8 Model H: the e-machine torque-split factor α_{EM}

As seen in the previous section, also in this model it is defined a torque split factor. However, this time it is expressed like the ratio between the electrical machine torque T_{EM} and the demanded torque T_{req}:

$$u_2 = \alpha_{EM} = \frac{T_{EM}}{T_{req}}. \tag{4.26}$$

In this way, the percentage of T_{EM} with respect to T_{req} is determined. Continuing to describe the vehicle's physical model, it has:

$$T_{EM} = \alpha_{EM} T_{req} \tag{4.27}$$

and via eq. (3. 13) and eq. (3. 14)

$$T_{eng} = (1 - \alpha_{EM}) T_{req}. \tag{4.28}$$

The $u_2\ grid$ is expressed as follow:

$$-1 \leq u_2 \leq 1 \tag{4.29}$$

with $u_2 = -1$ that represents the electrical machine operating as generator and $u_2 = 1$ that represents the electrical machine operating as motor.

The imposed constraints are about engine, electrical machine, battery and powertrain.

The results about final SOC, fuel consumption and fuel economy are reported on the Table 4-8:

Discretization level	Final SOC	Fuel Consumption [kg]	Fuel economy [l/100km]
11	0.599034	0.77	4.19
21	0.599111	0.76	4.16
41	0.599029	0.75	4.13
81	0.599012	0.75	4.11
121	0.599001	0.75	4.11

Table 4-8: simulation results model H

In these results, the discretization addiction is very little both in terms of Fuel consumption and Fuel Economy. As regards final SOC, the results shown how it moves away from the target getting closer to the lower bound (0.599).

Power plot results are shown in Figure 4.23:

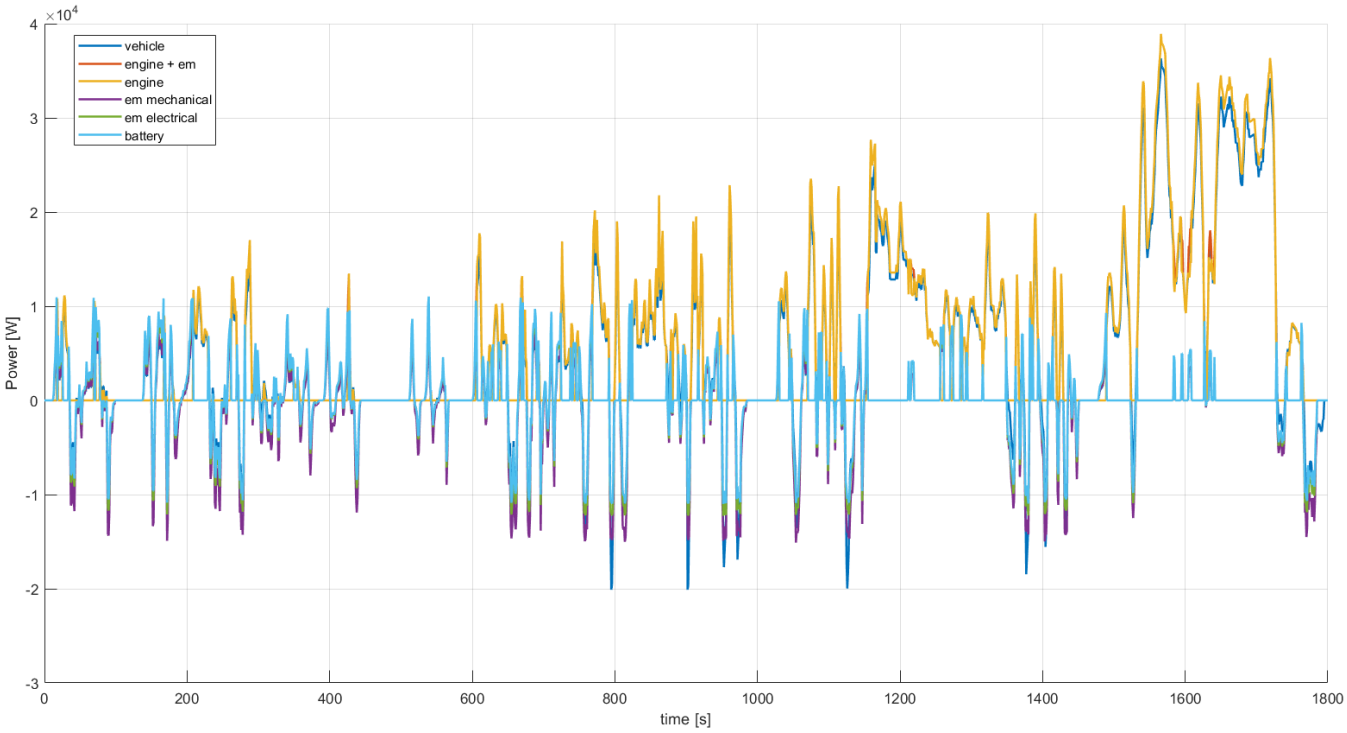

Figure 4.23: power plot results model H

Power split's good management has a positive influence on the Fuel consumption and Fuel Economy. An example of this is detectable in the Figure 4.24:

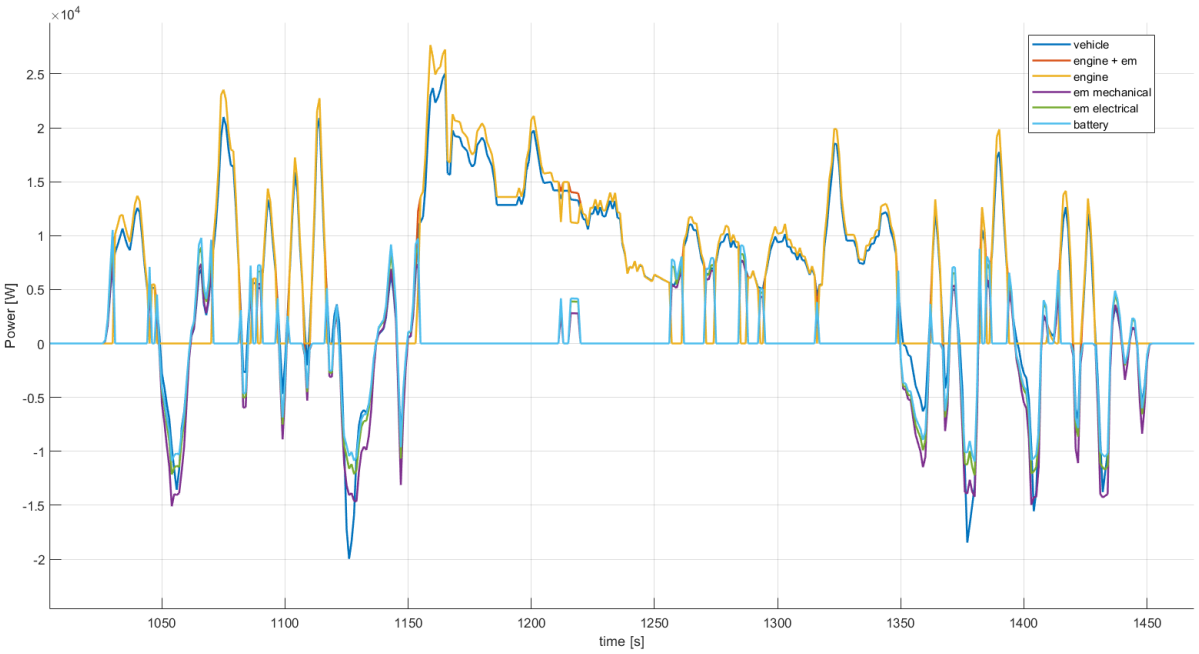

Figure 4.24: High phase detail model H

In this plot, the electrical machine's support is clear, with some zones in which DP choses pure electric mode. Consequently, the impact on Fuel Consumption and Fuel economy is very positive making the obtained values strongly in line with the expected.

Gear number variation is visible in Figure 4.25:

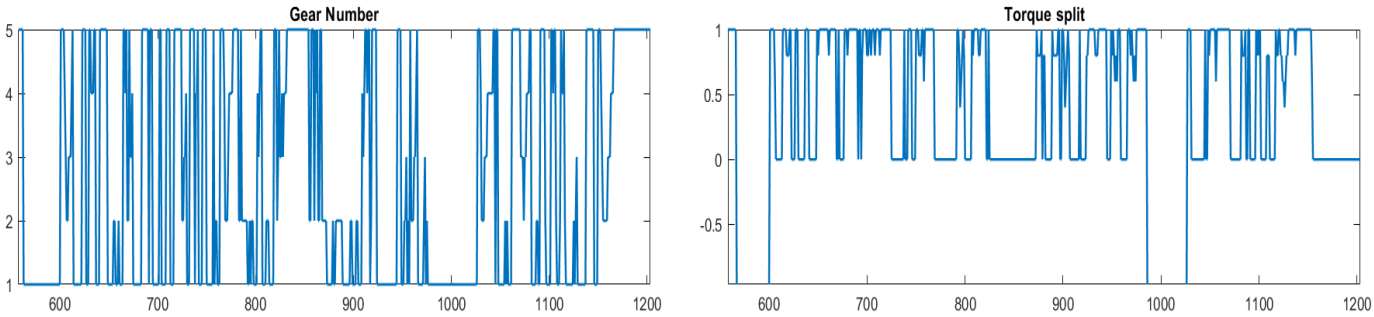

Figure 4.25: gear number variation model H

The results indicate non-correspondence between torque split and gear number. This is expected after watching model B and model E's behaviour. In fact, as seen in previous cases, torque split is newly expressed as function of EM torque and, as a result, gear number is low when torque split's values are maximum or minimum while gear number takes different values when torque split is zero (pure thermal mode).

The main advantage of this model is the possibility to identify all the powerflow mode. In fact, this torque gives information about pure electric, pure thermal and power split. Moreover, it can distinguish when electrical machine operating as generator or motor and this implies also to identify the battery charging mode. However, model H carries on some drawbacks: there are some unexplored operative points of the electrical machine torque and then the engine could be underexploited, never reaching its maximum value.

5 Model comparison

After analysing each single model, the next step is to compare and identify differences and similarities between them. Moreover, the following comparison could give some information about what model is more accurate. Each model is compared on the same resulting parameters of every single analysis: final SOC, Fuel consumption and Fuel economy. Obviously, each parameter is considered with the same discretization level.

5.1 Final SOC comparison

The first compared parameter is final SOC. This last is very important because the nearer is to the target the more accurate is the simulation. The comparison results are reported in Table 5-1:

	Final SOC							
Discretization Level	Model A	Model B	Model C	Model D	Model E	Model F	Model G	Model H
11	0.599246	0.600047	0.599444	0.599013	0.600572	0.599444	0.600999	0.599034
21	0.599006	0.599389	0.599167	0.599043	0.599092	0.599167	0.601	0.599111
41	0.599019	0.599273	0.599028	0.599005	0.599095	0.599028	0.600999	0.599029
81	0.599003	0.599108	0.599028	0.599002	0.59906	0.599028	0.600997	0.599012
121	0.599006	0.599102	0.599028	0.599002	0.599022	0.599028	0.600999	0.599001

Table 5-1: final SOC comparison

From the analysis, the models that came closest to the target of 0.6 are model B, model C, model E and model F. Each one of these reaches the nearest value in the first discretization level while increasing discretization they move away. Model B and model E are indirectly linked to SOC because torque split is expressed as a function of EM torque, from which i_b and consequently SOC is calculated (sec. 3.1.3). On the other hand, model C and model F have torque split expressed as a function of i_b and this directly link it to SOC. Moreover, model C and model F show practically the same results.

Anyway, the other models show good results, always remaining in the pre-set range

$$0.599 \leq SOC \leq 0.601.$$

5.2 Fuel Consumption and Fuel Economy comparison

As mentioned in sec. 3.2, Fuel Consumption and Fuel Economy represent the cost function of the problem and the aim is to minimize them as much as possible. On the other hand, a good "compromise" between cost's minimization and simulation accuracy is required and this reason is fundamental in the models' comparison.

The comparison in terms of Fuel Consumption (Table 5-2) and Fuel Economy (Table 5-3) is reported below:

	Fuel Consumption kg							
Discretization Level	Model A	Model B	Model C	Model D	Model E	Model F	Model G	Model H
11	0.69	0.85	0.89	0.69	0.79	0.89	0.76	4.19
21	0.66	0.82	0.89	0.66	0.77	0.89	0.75	4.16
41	0.65	0.81	0.89	0.65	0.75	0.89	0.75	4.13
81	0.65	0.80	0.89	0.65	0.74	0.89	0.75	4.11
121	0.64	0.78	0.89	0.64	0.73	0.89	0.75	4.11

Table 5-2: Fuel Consumption comparison

	Fuel Economy [l/100 km]							
Discretization Level	Model A	Model B	Model C	Model D	Model E	Model F	Model G	Model H
11	3.78	4.65	4.87	3.78	4.35	4.87	4.16	4.19
21	3.64	4.49	4.87	3.64	4.20	4.87	4.13	4.16
41	3.57	4.43	4.86	3.58	4.10	4.86	4.11	4.13
81	3.54	4.37	4.85	3.54	4.04	4.85	4.10	4.11
121	3.52	4.29	4.85	3.53	4.01	4.85	4.10	4.11

Table 5-3: Fuel Economy comparison

At first sight, model A and model D seem to guarantee lower fuel consumption than the others. This is due to a very good power split management that allows to make engine torque less predominant (sec. 4.1 and 4.4). On the other hand, torque split expressed as a function of the engine torque T_{eng} makes some operative points not investigable, implying a lower fuel consumption. As concerns final SOC, it is not perfectly on target affecting the simulation accuracy for sure. Model C and model F show the highest fuel consumption caused by a lower use of power split as seen in sec. 4.3 and 4.6. In this case, the main limits stem from the fact that torque split can't investigate all the operative points of the battery current range and this leads to increase fuel consumption. However, as mentioned above (sec. 5.1), in this case final SOC is very near to the target and this make the models very promising.

The best compromise between accuracy and fuel consumption is surely represented by model B and model E. In fact, these last have final SOC near to the target like models C and F but their fuel consumption is lower. Probably this is due to torque split expressed as a function of EM torque (sec. 4.2 and 4.5) that allows to provide a good investigation of the engine torque operative range and at the same time a good power split management is done.

Model G and model H represent a very good compromise between results and computational time and they give the possibility to identify all the powerflow configuration. This last represents the real strength of these models.

Finally, all the models give acceptable results, in line with the initial expectations. In fact, Fuel Economy is included in the range $4\frac{l}{100km} \leq Fuel\ Economy \leq 5\frac{l}{100km}$, except models A and D that give good results anyway.

6 Conclusions

The analysis shows that probably model B and model E seem to be a bit more promising than the others. In particular, model E is better as regard computational time because of its torque split's grid. Consequently, discretization could be increased providing better results. However, improvement of vehicle's physical model could give more accurate results. An example of this could be the suggest provided by (Sundström, Guzzella, & Soltic, 2010) in which EM torque is get through a linear interpolation taking three torque split points and three torque's values, chosen after imposing some conditions.

Another way to improve the model is to also include pollutants emissions evaluation as output. To do this, the model is split and an external system for emissions is created. Moreover, a new state variable must be added and this is the TWC temperature.

In this thesis, a p2 parallel hybrid architecture is modelled, but the analysis of other architectures, for example a series configuration, is possible. Obviously, expected results, parameters' choice and vehicle's physical model would be different.

Finally, remember that Dynamic Programming is not implementable online and Dyna-Prog's use is only referred to a pre-design concept. Despite this, DP can provide a very interesting preliminary view over the problem and this allows to save time and resources before the online implementation is done. Moreover, because of model's simplicity, DP could be used to determine different cost function only varying just a few parameters and this maybe represent the main strength of all entire subject.

7 Appendix

In this appendix, a little view over the used MATLAB script is shown. There is some information about vehicle model, torque split of each model and DP main script.

7.1 Vehicle model

The first script represents how vehicle's physical model is implemented on MATLAB, starting from wheels:

```
%% Vehicle Model
% Wheels
% Wheel speed (rad/s)
wheelSpd = w{1} ./ wh.radius;
% Wheel acceleration (rad/s^2)
wheelAcc = w{2} ./ wh.radius;
% Tractive Force (N)
vehResForce =  veh.f0 + veh.f1 .* w{1} + 0.4020 .* w{1}.^2;
vehForce = (w{1}~=0) .* (vehResForce + veh.mass.*w{2});
% Wheel torque (Nm)
wheelTrq = vehForce .* wh.radius;
% Final Drive
% Final drive input speed (rad/s)
fdSpd = fd.spdRatio .* wheelSpd;
% Final drive input acceleration (rad/s^2)
fdAcc = fd.spdRatio .* wheelAcc;
% Final drive input torque (Nm)
fdTrq  = wheelTrq ./ fd.spdRatio ...
    + fd.loss.*(wheelTrq>0) - fd.loss.*(wheelTrq<=0);
```

After that, gearbox and transmission are implemented below:

```
% Gearbox
gbSpRatio = gb.spdRatio(u{1});

% Crankshaft speed (rad/s)
shaftSpd  = gbSpRatio .* fdSpd;
% Crankshaft acceleration (rad/s^2)
shaftAcc = gbSpRatio .* fdAcc;
% Gearbox efficiency (-)
gbEff = gb.effMap(u{1});
gbEff = min(max(gbEff, eps), 1);
% Gearbox inertia
```

```
gbInertia = gb.inertia .* shaftAcc;
% Crankshaft torque (Nm)
gbLoss = (fdTrq>0 & gbEff>eps)  .* (1-gbEff) .* fdTrq ./ (gbEff .* gbSpRatio) ...
     + (fdTrq>0 & gbEff==eps)  .* fdTrq .* gbSpRatio ...
    + (fdTrq<=0) .* (1-gbEff) .* fdTrq ./ gbSpRatio;

shaftTrq  = fdTrq ./ gbSpRatio + gbLoss + gbInertia;
```

The torque split section with the eq. (3. 14) is described below:

```
% Torque Split
% Engine inertia torque (Nm)
engResTrq  = shaftAcc * eng.inertia;
% Electric motor drag torque (Nm)
emResTrq  = shaftAcc * em.inertia;
% Total required torque (Nm)
reqTrq = engResTrq.*(u{2}~=0) + emResTrq + shaftTrq;
```

The remaining part is expressed as a function of model. In fact, following the eq. (3. 13), there are different scripts for each chosen torque split (sec. 7.3).
Constrains and unfeasibility are the same mentioned in sec. 3.1.4 and they are always imposed following the simulated model.

7.2 DP main

In this section, main DP launch script is shown. This script gives the possibility to select the model, to define state and control variables and related grids, to load the driving cycle and vehicle data and to provide all the other elements necessary for the problem (exogenus inputs, N_{int}, etc..).
Following the section mentioned above is reported:

```
% Main
clear

% Select model
model = "H";

%% Set up the problem
```

```
% State variable grid
SVnames = 'SOC';
x_grid = {0.4:0.001:0.7};
% Initial state
x_init = {0.6};
% Final state constraints
x_final = {[0.599 0.601]};

% Control variable grid
CVnames = {'Gear Number', 'Torque split'};
u1_grid = [1 2 3 4 5];
u2_grid = linspace(-1,1,11);
u_grid = {u1_grid, u2_grid};

% Load a drive cycle
load WLTP3 % contains velocity and time vectors
dt = time_s(2) - time_s(1);
% Create exogenous input
w{1} = speed_kmh./3.6;
w{2} = [diff(w{1})/dt; 0];

% Number of stages (time intervals)
Nint = length(time_s);

% Generate and store vehicle data
load vehData

% Select HEV model file
funString = "@(x, u, w) hev" + model + "(x, u, w, veh, wh, fd, gb, eng, em, batt)";
funString = "fun = " + funString;
eval(funString)

% Create DynaProg object
prob = DynaProg(x_grid, x_init, x_final, u_grid, Nint, ...
 fun, 'ExogenousInput', w);
```

The second part of this script is about the problem solving and it gives cost function and related profile, additional inputs saved as profiles and plot of the results as seen in chapter 4.

```
%% Solve
% Solve the problem

prob = run(prob);
```

```
prob.Time = [time_s; time_s(end)+dt];
prob.StateName = SVnames;
prob.ControlName = CVnames;
prob.CostName = 'Fuel Consumption, g';


profiles.ExogenousInput = prob.ExogenousInput;
profiles.StateProfile = prob.StateProfile;
profiles.ControlProfile = prob.ControlProfile;
profiles.CostProfile = prob.CostProfile;
profiles.AddOutputsProfile = prob.AddOutputsProfile;
profiles.Time = prob.Time;
save("profiles" + model + ".mat", '-struct', 'profiles');
% Plots
figure
t = plot(prob);
```

7.3 Torque splits

In this section, the scripts regarding all the torque splits seen in this thesis are reported.

7.3.1 Model A

```
% Torque Split

engSpd = shaftSpd.*ones(size(reqTrq));

% Torque provided by engine
engTrq  =(shaftSpd>0).*u{2};
brakeTrq  = (shaftSpd>0) .* u{2};
engTrq = engTrq.*ones(size(reqTrq));

% Torque provided by electric motor
emTrq  = (shaftSpd>0).*(reqTrq>0).* reqTrq - engTrq;

pwtUnfeas = (reqTrq<0 & emTrq>0);

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emTrq = emTrq ./ 1.58;

% Engine
% Fuel and pollutants mass flow rates
```

```
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% EM
% Electric motor efficiency
emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
emTrq(emTrq<emMinTrq)=emMinTrq(emTrq<emMinTrq);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                   (emTrq>=0) .* (emTrq > emMaxTrq);

% BATTERY
battPwr = emElPwr;
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr      =      battColumbicEff      .*      (battVoltage-sqrt(battVoltage.^2      -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0).* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});


% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas  =  (battPwr<=0).*(battCurr  <  maxChrgBattCurr)  +  (battPwr>0).*(battCurr  >
maxDisBattCurr) + x_new{1}>0.7;
```

7.3.2 Model B

```
% Torque Split

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emSpd = emSpd.*ones(size(reqTrq));



% Torque provided by electric motor
emTrq =u{2}.*ones(size(reqTrq));
% Torque provided by engine
engTrq  = reqTrq - u{2};
engTrq(engTrq<0)=0;
brakeTrq  = (reqTrq<=0) .* reqTrq - u{2};

pwtUnfeas = (reqTrq<0 & emTrq>0);

emTrq = emTrq ./ 1.58;

% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);



% EM
% Electric motor efficiency
% emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;



% BATTERY
battPwr = emElPwr;
```

```
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr      =      battColumbicEff      .*      (battVoltage-sqrt(battVoltage.^2      -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0) .* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});


% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas  =  (battPwr<=0).*(battCurr  <  maxChrgBattCurr)  +  (battPwr>0).*(battCurr  >
maxDisBattCurr);
```

7.3.3 Model C

```
% Torque Split

% BATTERY
if vehForce == 0
    battCurr =0;
else
    battCurr= u{2};
end
% Battery voltage
battVoltage = batt.ocv(x{1});
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery Power
battPwr = battVoltage.*battCurr + battR.*(battCurr.^2);
% Assume inverter efficiency of 0.95
battPwr = (battPwr<0)  .* battPwr ./ 0.95...
  + (battPwr>=0) .* battPwr .* 0.95;
```

```
emElPwr = battPwr;


% EM
% Electric motor efficiency
emSpd = shaftSpd .* 1.58;
elTrq = emElPwr./emSpd;
emSpd = emSpd.*ones(size(elTrq));
emEff = (shaftSpd~=0) .* em.genEffMap(emSpd, elTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
elMaxTrq = em.maxElTrq(emSpd);
elMinTrq = em.minElTrq(emSpd);
elUnfeas = (isnan(emEff)) + (elTrq<0)  .* (elTrq < elMinTrq) +...
                   (elTrq>=0) .* (elTrq > elMaxTrq);
% Calculate electric power consumption
emTrq = (emElPwr>=0) .* (emElPwr./emSpd).*emEff + (emElPwr<0) .* emElPwr./(emEff.*emSpd);
emTrq(isnan(emTrq))=0;
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                   (emTrq>=0) .* (emTrq > emMaxTrq);




% Torque provided by engine
engTrq  = (shaftSpd>0) .* (reqTrq>0)  .* reqTrq - emTrq;
brakeTrq  = (shaftSpd>0) .* (reqTrq<=0) .* reqTrq - emTrq;
engTrq(engTrq<0)=0;


pwtUnfeas = (reqTrq<0 & emTrq>0);

% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
```

```
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);

% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};

battUnfeas = x_new{1}>0.7;
```

7.3.4 Model D

```
% Torque Split
% Engine inertia torque (Nm)

engSpd = shaftSpd.*ones(size(reqTrq));
% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);

% Torque provided by engine
engTrq  = (shaftSpd>0) .* u{2}.*engMaxTrq;
brakeTrq  = (shaftSpd>0) .* u{2}.*engMaxTrq;
engTrq(engTrq<0)=0;
% Torque provided by electric motor
emTrq  = (shaftSpd>0) .*(reqTrq>0).* reqTrq - u{2}.* engMaxTrq;

pwtUnfeas = (reqTrq<0 & emTrq>0);

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emTrq = emTrq ./ 1.58;

% Engine
% Fuel and pollutants mass flow rates
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% EM
% Electric motor efficiency
emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;
% Limit Torque
```

```
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
emTrq(emTrq<emMinTrq)=emMinTrq(emTrq<emMinTrq);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                   (emTrq>=0) .* (emTrq > emMaxTrq);

% BATTERY
battPwr = emElPwr;
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr      =      battColumbicEff      .*      (battVoltage-sqrt(battVoltage.^2      -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0).* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});


% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas  =  (battPwr<=0).*(battCurr  <  maxChrgBattCurr)  +  (battPwr>0).*(battCurr  >
maxDisBattCurr) + x_new{1}>0.7 ;
```

7.3.5 Model E

```
% Torque Split

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emSpd = emSpd.*ones(size(reqTrq));
```

```
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
emlimTrq = emMaxTrq;

% Torque provided by electric motor
emTrq  =u{2}.* emlimTrq;
% Torque provided by engine
engTrq  = reqTrq - u{2}.*emlimTrq;
brakeTrq  = (reqTrq<=0) .* reqTrq - u{2}.*emlimTrq ;
engTrq(engTrq<0)=0;
pwtUnfeas = (reqTrq<0 & emTrq>0);

emTrq = emTrq ./ 1.58;

% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);

% EM
% Electric motor efficiency
% emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;


% BATTERY
battPwr = emElPwr;
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
```

```
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr = battColumbicEff .* (battVoltage-sqrt(battVoltage.^2 -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0) .* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});

% New battery state of charge
x_new{1} = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas = (battPwr<=0).*(battCurr < maxChrgBattCurr) + (battPwr>0).*(battCurr >
maxDisBattCurr);
```

7.3.6 Model F

```
% Torque Split

% BATTERY
ibattlim = (u{2}<0).* -batt.minCurr(x{1}) + (u{2}>=0).* batt.maxCurr(x{1});

if vehForce ==0
    battCurr=0;
else
    battCurr= u{2}.* ibattlim;
end

% Battery voltage
battVoltage = batt.ocv(x{1});
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery Power
battPwr = battVoltage.*battCurr + battR.*(battCurr.^2);
% Assume inverter efficiency of 0.95
battPwr = (battPwr<0) .* battPwr ./ 0.95...
  + (battPwr>=0) .* battPwr .* 0.95;

emElPwr = battPwr;


% % Constraints
```

```
% battUnfeas = (battPwr<=0).*(battCurr < maxChrgBattCurr) + (battPwr>0).*(battCurr >
maxDisBattCurr);

% EM
% Electric motor efficiency
emSpd = shaftSpd .* 1.58;
elTrq = emElPwr./emSpd;
emSpd = emSpd.*ones(size(elTrq));
emEff = (shaftSpd~=0) .* em.genEffMap(emSpd, elTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
elMaxTrq = em.maxElTrq(emSpd);
elMinTrq = em.minElTrq(emSpd);
% Calculate electric power consumption
emTrq = (emElPwr>=0) .* (emElPwr./emSpd).*emEff + (emElPwr<0) .* emElPwr./(emEff.*emSpd);
emTrq(isnan(emTrq))=0;
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                    (emTrq>=0) .* (emTrq > emMaxTrq);


% Torque provided by engine
engTrq  = (shaftSpd>0) .* (reqTrq>0)  .* reqTrq - emTrq;
brakeTrq  = (shaftSpd>0) .* (reqTrq<=0) .* reqTrq - emTrq;
engTrq(engTrq<0)=0;


pwtUnfeas = (reqTrq<0 & emTrq>0);

% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);

% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
```

```
battUnfeas = x_new{1}>0.7;
```

7.3.7 Model G

```
% Torque Split

% Torque provided by engine
engTrq  = (shaftSpd>0) .* (reqTrq>0)  .* u{2}.*reqTrq;
brakeTrq  = (shaftSpd>0) .* (reqTrq<=0) .* (1-u{2}).*reqTrq;
% Torque provided by electric motor
emTrq  = (shaftSpd>0) .* (1-u{2}) .* reqTrq;

pwtUnfeas = (reqTrq<0 & emTrq>0);

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emTrq = emTrq ./1.58;

% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);

% EM
% Electric motor efficiency
emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                  (emTrq>=0) .* (emTrq > emMaxTrq);
```

```
% BATTERY
battPwr = emElPwr;
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr      =      battColumbicEff      .*      (battVoltage-sqrt(battVoltage.^2      -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0).* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});

% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas  =  (battPwr<=0).*(battCurr  <  maxChrgBattCurr)  +  (battPwr>0).*(battCurr  >
maxDisBattCurr);
```

7.3.8 Model H

```
% Torque Split

% Torque provided by engine
engTrq  = (shaftSpd>0) .* (reqTrq>0)  .* (1-u{2}).*reqTrq;
brakeTrq  = (shaftSpd>0) .* (reqTrq<=0) .* (1-u{2}).*reqTrq;
% Torque provided by electric motor
emTrq  = (shaftSpd>0) .* u{2} .* reqTrq;

pwtUnfeas = (reqTrq<0 & emTrq>0);

% Torque-coupling device (ideal)
emSpd = shaftSpd .* 1.58;
emTrq = emTrq ./1.58;
```

```
% Engine
% Fuel and pollutants mass flow rates
engSpd = shaftSpd.*ones(size(engTrq));
fuelFlwRate = eng.fuelMap(engSpd, engTrq); % fuel flow rate, g/s
fuelFlwRate(engTrq==0) = 0;

% Maximum engine torque
engMaxTrq = eng.maxTrq(engSpd);
% Costraints
engUnfeas = (engTrq > engMaxTrq) | (engTrq > 0 & engSpd < eng.idleSpd & u{1}~=1) | (engTrq
> 0 & engSpd > eng.maxSpd);

% EM
% Electric motor efficiency
emSpd = emSpd.*ones(size(emTrq));
emEff = (shaftSpd~=0) .* em.effMap(emSpd, emTrq) + (shaftSpd==0);
emEff(isnan(emEff)) = 1;
% Calculate electric power consumption
emElPwr =  (emTrq<0) .* emSpd.*emTrq.*emEff + (emTrq>=0) .* emSpd.*emTrq./emEff;
% Limit Torque
emMaxTrq = em.maxTrq(emSpd);
emMinTrq = em.minTrq(emSpd);
% Constraints
emUnfeas = (isnan(emEff)) + (emTrq<0)  .* (emTrq < emMinTrq) +...
                   (emTrq>=0) .* (emTrq > emMaxTrq);

% BATTERY
battPwr = emElPwr;
% Assume inverter efficiency of 0.95
battPwr = (battPwr>0)  .* battPwr ./ 0.95...
  + (battPwr<=0) .* battPwr .* 0.95;

% columbic efficiency
battColumbicEff = (battPwr>0) + (battPwr<=0) .* batt.coulombic_eff;
% Battery internal resistance
battR = batt.eqRes(x{1});
% Battery voltage
battVoltage = batt.ocv(x{1});

% Battery current
battCurr      =      battColumbicEff      .*      (battVoltage-sqrt(battVoltage.^2      -
4.*battR.*battPwr))./(2.*battR);
battCurr = real(battCurr);
% Maximum charge current
maxChrgBattCurr = (battPwr<=0).* batt.minCurr(x{1});
maxDisBattCurr = (battPwr>0).* batt.maxCurr(x{1});
```

```
% New battery state of charge
x_new{1}  = - battCurr ./ (batt.maxCap * 3600) .* dt + x{1};
% Constraints
battUnfeas  =  (battPwr<=0).*(battCurr  <  maxChrgBattCurr)  +  (battPwr>0).*(battCurr  >
maxDisBattCurr);
```

8 References

Bellman, R. (1957). *Dynamic Programming.*

Bersekas, D. P. (2019). *Reinforcement Learning and Optimal Control.*

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control* (third edition ed., Vol. 1).

Kirk, D. E. (2004). *Optimal control theory, an introduction.*

Miretti, F., Misul, D., & Spessa, E. (2021). DynaProg: Deterministic Dynamic Programming solver for finite horizon multi-stage decision problems.

Onori, S., Rizzoni, G., & Serrao, L. (2015). *Hybrid Electric Vehicles: Energy Management Strategies.*

Serrao, L., Onori, S., & Rizzoni, G. (2011). A Comparative Analysis of Energy Management Strategies for Hybrid Electric Vehicles.

Sundström, O., Guzzella, L., & Soltic, P. (2010). Torque-Assist Hybrid Electric Powertrain Sizing: From Optimal Control Towards a Sizing Law.