An ECMS-based powertrain control of a parallel hybrid electric forklift

Catalin Stefan Teodorescu
Flanders Make
Leuven, Belgium
stefan.teodorescu@flandersmake.be

Steve Vandenplas
Flanders Make
Leuven, Belgium
steve.vandenplas@flandersmake.be

Bruno Depraetere
Flanders Make
Leuven, Belgium
bruno.depraetere@flandersmake.be

Keivan Shariatmadar
Advanced Engineering
Dana – Belgium NV
Brugge, Belgium
keivan.shariatmadar@dana.com

Thomas Vyncke
Advanced Engineering
Dana – Belgium NV
Brugge, Belgium
thomas.vyncke@dana.com

Joost Duflou
Dept. Mechanical Engineering
KU Leuven
Leuven, Belgium
joost.duflou@kuleuven.be

Ann Nowé
Artificial Intelligence Lab
Vrije Universiteit Brussel
Brussels, Belgium
ann.nowe@vub.ac.be

***Abstract*—In this paper we focus on the supervisory control problem of a parallel hybrid electric vehicle (HEV): minimize fuel consumption while ensuring self-sustaining State-of-Charge (SoC). We reapply the state of the art methodology by comparing optimal results of Dynamic Programming (DP) against a real-time control candidate. After careful selection, we opted for an Equivalent Consumption Minimization Strategy (ECMS) based approach for the following reasons: (i) results are quite remarkable with less than 5% fuel usage increase when compared to DP; (ii) simple and intuitive tuning of control parameters; (iii) readily usable for code generation (prototyping).**

Topics that distinguish this article from others in the literature include: (i) the usage of trapezoidal rule of integration implementing DP and ECMS; consequently, the offline simulation results are intended to be more precise and representative when compared against the more common, often used rectangular rule; (ii) a particular post-processing procedure of the recorded driving cycle data based on physical interpretation; it allows consistent offline simulations with quite high sampling period (in the order of seconds); (iii) tuning of control parameters in such a way that control system is robust towards new, unknown, unpredictable but closely resembling driving cycles.

In particular, we focus on the supervisory control of a forklift truck. The real-time control is able to compute: (i) the power split (i.e. a balanced usage between an internal combustion engine and a supercapacitor); (ii) the drivetrain control (i.e. automatic gear shifting and clutching). Numerous numerical implementation issues are discussed along our presentation.

***Index Terms*—automotive application, hybrid electric vehicles, energy management, supervisory control, dynamic programming, real-time control**

I. Introduction

Hybrid electric vehicles (HEV) are attractive nowadays because of their ability to achieve reduced fuel consumption while maintaining similar performance when compared to conventional vehicles. Some of the key features that make HEV successful can be cited:

This work was supported by the Belgian IWT (Agency for Innovation by Science and Technology) through the project PERPETUAL (Personalized Products Emerging from Tailored User-Adapting Logic) nr. 110041.

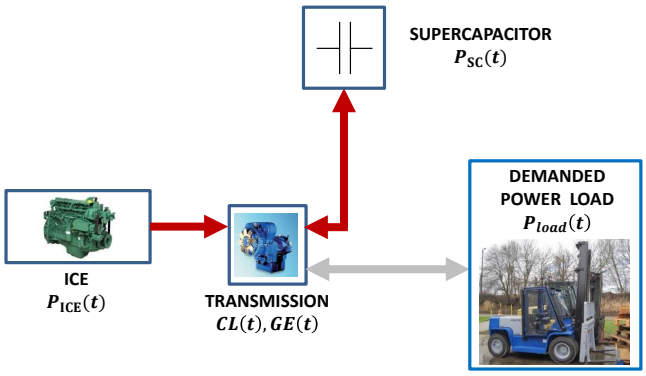


Fig. 1: Schematic diagram of the powertrain of an off-highway hybrid forklift truck; for simplicity reasons, not all components are illustrated (e.g. the differential, motor reduction are omitted).

- ability to use multiple power sources. Most often one finds two: a downsized conventional internal combustion engine (ICE) together with an electric power source;
- usage of smaller ICE leads to reduced particulate matter emissions and consequently compliance with regulations worldwide;
- ability to store regenerative power, instead of wasting it as conventional vehicles do. Ideally, once the driver pushes the brake pedal all this power is stored.

However, there is a cost to be paid: these complex machines require more advanced control than conventional vehicles.

This article is dedicated to the energy management problem, also called supervisory control in [1]–[3], for a specific parallel HEV depicted in Fig. 1. This choice for a parallel HEV is also reflected in the references section.

The post-transmission parallel HEV forklift illustrated in Fig. 1 uses two power sources that can be used simultaneously: an ICE and a supercapacitor (SC). The arrows indicate power

distribution, with the following conventions: negative regenerative power can be transmitted from the load (i.e. wheels of the vehicle) towards the SC which can store it for later usage; positive generative power can be transmitted from ICE towards transmission (TE), which can be used: (i) to charge the SC, and/or (ii) to drive the load; positive generative power can be transmitted from the SC towards TE and then along the path towards the load. To summarize, note the power flow can be uni- or bi-directional between these components.

A. Optimal control problem

The optimal fuel consumption problem can be formulated as follows:

$$\min_{u(t\in[t_0,t_e])} J = \int_{t_0}^{t_e} \dot{m}(\tau, u(\tau))\,\mathrm{d}\tau \tag{1a}$$

subject to:

$$|P_{\mathrm{SC}}(t)| \le P_{\mathrm{SC}}^{\max} \tag{1b}$$

$$SoC(t_e) \in \Omega(SoC^{\mathrm{ref}}, \varepsilon) \tag{1c}$$

$$\dot{SoC}(t) = -\frac{P_{\mathrm{SC}}(t)}{E_{\mathrm{SC}}^{\max}} \tag{1d}$$

$$SoC^{\min} \le SoC(t) \le SoC^{\max} \tag{1e}$$

$$GE(t) = GE_{\mathrm{req}}(t - \Delta t_{\mathrm{GS}}) \tag{1f}$$

$$CL(t) = CL_{\mathrm{req}}(t - \Delta t_{\mathrm{CL}}) \tag{1g}$$

power balance static equality equations including losses along powertrain path (1h)

instantaneous torque and speed inequality constraints on each powertrain component (1i)

where the control actions

$$u(t) = [P_{\mathrm{SC}}(t),\, GE_{\mathrm{req}}(t - \Delta t_{\mathrm{GS}}),\, CL_{\mathrm{req}}(t - \Delta t_{\mathrm{CL}})] \tag{2}$$

constitute the optimization variables. The internal state variables

$$x(t) = [SoC(t),\, GE(t),\, CL(t)]\,. \tag{3}$$

The cost function J in (1a) is defined using the integral of fuel rate $\dot{m}$ over a finite time-horizon. The values t_0 and t_e represent the starting and the ending time, respectively, of the driving cycle load data. The latter is specified in terms of speed and torque profiles $(\omega_{\mathrm{load}}(t), T_{\mathrm{load}}(t))$.

The constraint (1b) imposes bounds on the power delivered to and from the SC. For simplicity, we considered symmetric bounds, where $P_{\mathrm{SC}}^{\max}$ corresponds to maximum power.

Let us now detail one-by-one, the constraints from (1c), (1d), (1e). They are responsible of State-of-Charge (SoC), defined as the ratio between available electric energy $E_{\mathrm{SC}}(t)$ and maximum electric energy $E_{\mathrm{SC}}^{\max}$:

$$SoC(t) = \frac{E_{\mathrm{SC}}(t)}{E_{\mathrm{SC}}^{\max}} \in [0, 1] \tag{4}$$

A simplified electric circuit model of SC is used, by assuming the voltage to be constant. The electric energy is governed by the equation:

$$\dot{E}_{\mathrm{SC}}(t) = -P_{\mathrm{SC}}(t) \tag{5}$$

The constraint (1c) expresses a desired self-sustaining SoC property: we want that by the end of the driving cycle both power sources are available and, in particular, the electrical power source is not lost. Moreover it should be in an ε-neighborhood of a desired reference value SoC^{ref}; we call it an Ω-set. This is a softer constraint, easier to cope with in real-time (RT) control, when compared to the alternative of using an equality between $SoC(t_e)$ and SoC^{ref}.

The equation (1d) expresses the dynamics of the SoC.

The inequality constraint (1e) is a guarantee that electric energy inside the SC stays within admissible bounds. Too low SoC values put the SC in a situation where it cannot deliver full power $P_{\mathrm{SC}}^{\max}$ any more. Often $SoC^{\min}$ is chosen around 0.25, while $SoC^{\max}$ equal to 1 corresponds to fully charged SC.

In order to introduce (1f)-(1g) we need to give an insight on the transmission model. It consists of a clutch in series with multiple parallel gears that can be acted one at a time. This is depicted in Fig. 2. The role of the clutch is to isolate (decouple) the ICE on the driveline path, thus allowing the vehicle to run with only one power source, namely the SC.

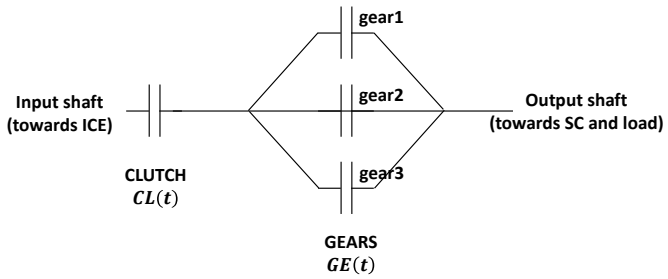


Fig. 2: Transmission model.

The equation (1f) expresses the fact that gear status $GE(t)$ is a delayed version of the request signal $GE_{\mathrm{req}}(t)$ that was made in the past; both take discrete values; Δt_{GS} is the overall time needed to do gear shifting (in the order of seconds).

The equation (1g) translates the relation between clutch status $CL(t)$ and the request $CL_{\mathrm{req}}(t)$ that was made in the past; both take binary values corresponding to these states:

(i) engaged, allowing for power to be transmitted from the ICE towards SC and load;

(ii) disengaged, leaving the ICE isolated from the rest of the powertrain; consequently, the HEV can only be driven in electric-mode.

For this transmission, the overall time Δt_{CL} required by clutch action to become effective is such that $\Delta t_{\mathrm{CL}} \le \Delta t_{\mathrm{GS}}$.

The set of constraints (1h) ensure drivability requirement (see [4]).

Finally, the last set of constraints (1i) make sure that each powertrain component is running in its admissible operating range. For safety reasons, the torques and speeds on the shafts cannot exceed specific values specified by each manufacturer.

B. Methodology

The state of the art methodology in [1], [3], [5]–[10] for the design of real-time control consists of 3 steps that will be recalled hereafter.

1) Control-oriented plant model: The powertrain model consists of a succession of mechanical components (like the differential, transmission, clutches, gears, motor reduction, etc.) and electrical ones (like supercapacitor, etc.) that are coupled together. Using input-output power balance static equations that also take into account power losses, one can build a simplified model. It is representative for steady-state behavior, while neglecting transient dynamics (e.g. ICE dynamics due to inertia). For control purpose, it is more useful to build a so-called backward-facing model (see [11, §12]): given instantaneous driving cycle load, i.e. a fixed pair of $(\omega_{\text{load}}(t), T_{\text{load}}(t))$ values, the model should be able to provide all feasible combinations in terms of internal states and control actions of the powertrain components. Then, it is the task of the supervisory control to choose the most optimal one. In practice, lookup tables are used to build these models, most of them consisting of efficiency loss maps (ICE, TE, SC, etc.) that are experimentally measured.

The table below summarizes specifications of the parallel HEV used, namely a modified series forklift truck Kalmar DCE120-12.

Diesel ICE	160 kW max power
Supercapacitor (SC)	104 kW max power
Transmission	DANA TE15, 3 steps automatic
Mass of vehicle	15600 kg
Max payload vehicle	12000 kg

2) Model-based control designs: Supervisory control can be divided into two classes:

(i) offline, like Dynamic Programming (DP), and
(ii) online, real-time implementable.

The DP algorithm in [12] is a step-by-step procedure consisting of a systematic exploration of all possible solutions in state-space. Gradually, only the optimal trajectories are being kept while the others are being disregarded. Eventually the *global optimal control* solution is computed.

In particular, generic matlab functions implementing DP exist. E.g., in [13], [14] they were applied to the parallel HEV control problem.

Concerning the second class, numerous real-time strategies are presented in the literature, some of which include:

- a *gear-shifting and clutching strategy* in [15], based on a predefined map having points that are optimized offline using DP;
- using an approach similar to the so-called data mining techniques, authors in [5] extract rules by exploring and seeking for structure inside the DP results. Criticism includes the fact that in general, there is no guarantee that clear patterns exist at all. Moreover, assuming *a priori* they do exist: (i) these patterns could be very difficult to be identified and later on modeled, (ii) once identified on a particular HEV, there is no guarantee that they will be present in combination with another driving cycle or another vehicle, so this methodology might lack scalability.
- heuristic control in [3], [9]; $\mathcal{H}_\infty$ control in [9].

In addition to them, the family of Equivalent Consumption Minimization Strategy (ECMS) is quite popular nowadays. We mention the works of [1], [6], [7], [9], [10], [16] and the book [3]: the main difference comes from the choice of tuning the control parameters. In this paper, we address the same topic and focus exclusively on ECMS.

3) Selection of control designs: The widely used methodology that is also applied in this article can be summarized as follows. On the one hand we have DP which is used:

- for benchmarking purpose: it provides the *global optimal control* action; later on, the proposed real-time control strategies are compared against DP in order to check their performance;
- to check feasibility and compatibility between driving cycles data and the HEV model: if DP cannot reproduce a given driving cycle then we know for sure that there is no point in looking further for a real-time control strategy;
- to better understand and gain insight of the best control action. As mentioned above, some authors attempted to replicate it for online usage, by extracting rules.

Apart running slow, the main drawback of DP is the fact of being acausal, making it unsuited for RT usage: it needs knowledge of the future (driving cycle) in order to take decisions concerning the present. In practice, this information might not be available. Moreover, DP control decisions are duty cycle-dependent and once applied on another driving cycle, there is no guarantee that decisions are still optimal.

Contrary to DP, ECMS requires knowledge of the present, instantaneous driving cycle data only, no future preview is needed. It is formulated as an instantaneous optimization problem:

$$\min_{u(t)} \; P_{\text{ICE}}(t) + s(t)\, P_{\text{SC}}(t) \tag{6}$$
$$\text{subject to: (1b)} - \text{(1i)}$$

Decisions are being made at each time instant by weighting together available powers from the two power sources.

Before introducing $s(t)$ let's make a step backwards in time. Historically, the original optimization problem (1a)-(1i) was approached using DP and, equivalently, by the Pontryagin's minimum principle. ECMS is based on the latter by introducing additional assumptions (see [2]), notably to the nature of the so-called equivalence factor $s(t)$.

The topic regarding the choice of $s(t)$ gave rise to extensive research. In particular, authors in [2], [3] proposed to use a simple formula:

$$s(t) = s_0 + s_1(SoC(t) - SoC^{\text{ref}}) \tag{7}$$

where control parameters (s_0, s_1) are driving cycle dependent. We are now ready to summarize the features of ECMS that make it successful:

- is a strong candidate for near-optimal, real-time control (shown later on in this paper, we achieve less than 5% fuel usage increase when compared to DP);
- provides an elegant transition from offline DP towards an online counterpart; it relies on a mathematical framework with some additional physical insight;
- tuning is simple and intuitive; in particular, a dedicated control gain parameter s_1 can be used to reinforce self-sustaining SoC;
- is readily implementable using a rapid prototyping;
- when compared against rule-based, heuristic and SDP competitors, it might require less measurement data (driving cycles) for tuning the control parameters.

This paper is organized as follows. In section II we discuss some key numerical implementation issues concerning an offline simulator. It will be used to reliably compare DP and ECMS. Then, section III is dedicated to the real-time control. Simulations follow in section IV and the paper ends with conclusions.

II. An Offline Simulator

It is important to design a relatively simple control system that is representative with respect to physical phenomena. The physics behind the *real plant* is inherently based on dynamical equations. Some powertrain components have fast transients (like the electric current inside the SC, power transmission through the differential, etc.) while others have longer transients (like gear shifting process, actuating the clutch, etc.).

On the other hand, as presented in section I, we opt for a control-oriented plant model that accounts only for the steady-state behavior, while ignoring the transitory dynamics. In other words, in order for this simplified plant model to be representative and meaningful at any successive time instants t_k and t_{k+1}, the powertrain needs to be in steady-state at t_k and t_{k+1}.

To summarize, the advantages of the steady-state control-oriented model are:

- is able to reproduce, in average, the physical behavior of powertrain along a given driving cycle;
- is fairly complex, built on experimentally measured maps stored using lookup tables; consequently we expect the model-based real-time control to be easily applied later on, on prototyping platform (e.g. dSpace).

One drawback is that sampling time Δt needs to be kept quite high (8). In particular, it should satisfy the constraint:

$$\Delta t \geq \max\left(\Delta t_{\text{GS}}, \Delta t_{\text{CL}}\right) \tag{8}$$

If on the contrary, one decides to arbitrarily decrease Δt irrespective of (8), then a finer model able to capture the transitory would be needed. This is beyond the scope of our work.

As a consequence of (8), we are faced with the problem of working with large Δt (in the order of seconds). This might potentially lead to a degradation of simulated results. Later on in this section, we will focus on improving the precision of numerical calculations.

1) Control-oriented plant model: Now that we have highlighted some of the general features of the plant model, in this section we would like to emphasize the operational side.

Effectively, the plant model takes into account all steady-state mechanical and electrical constraints associated to each powertrain component. Most of their characteristics can be found in datasheets provided by manufactures. The rest (e.g., efficiency loss maps of ICE, TE, SC, etc.) need to be experimentally measured.

The outcome of the plant model is that it maps one pair of $(\omega_{\text{load}}(t), T_{\text{load}}(t))$ together with all range of feasible internal state variables, to ICE fuel rates $\dot{m}(t)$. Consecutively, this information will be used to calculate the cost function (1a).

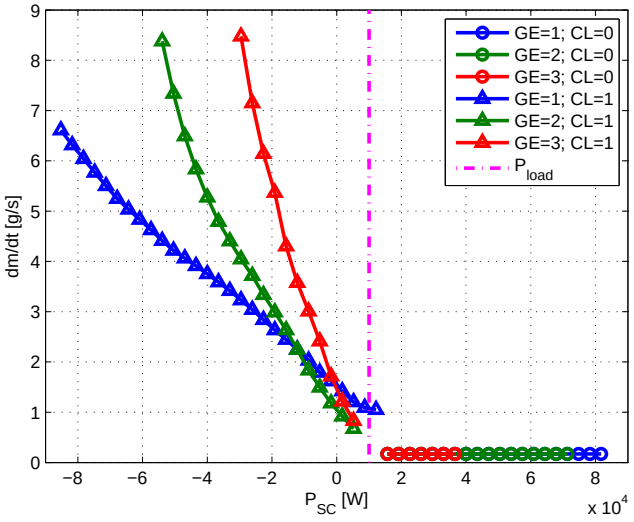


Fig. 3: Fuel usage: instantaneous steady-state ICE fuel rate $\dot{m}(t)$ profile curves calculated for $\omega_{\text{load}}(t) = 2$ [rad/s] and $T_{\text{load}}(t) = 5000$ [Nm]. This gives a desired load power $P_{\text{load}}(t) = 10^4$ [W].

Fig. 3 illustrates an example of fuel rate profile curves (the y-axis), depending on the power of the supercapacitor (the x-axis). Each curve corresponds to a different combination of gear and clutch. The vertical magenta dashed line is used to indicate the amount of demanded power and is used as an indicator to split the figure into multiple regions:

(i) the majority of the profile curves situated on the right-hand side of the magenta dashed line correspond to lowest fuel consumption: the clutch is disengaged and ICE set to idle fuel consumption, while the HEV runs in *electric only* mode; all excessive power needs to be dissipated by the driver who needs to push the brake pedal; unlike for passenger cars, it is undesired to turn off the ICE;

(ii) the profile curves situated between the magenta dashed line and zero SC power correspond to simultaneously using the ICE and SC in order to drive the vehicle: the HEV runs

in *electric assist* mode; the clutch is now engaged; note the abrupt jump with respect to ICE idle, notably because ICE has to supply sufficiently high power in order to compensate for losses along powertrain path;

(iii) profile curves in the region of negative $P_{\text{SC}}(t)$ correspond to simultaneously charging the SC and also driving the load; the HEV runs in *SC charging* mode.

Fig. 4 illustrates fuel rate for different load points. Note that P_{SC}-axis is common to Fig. 3.

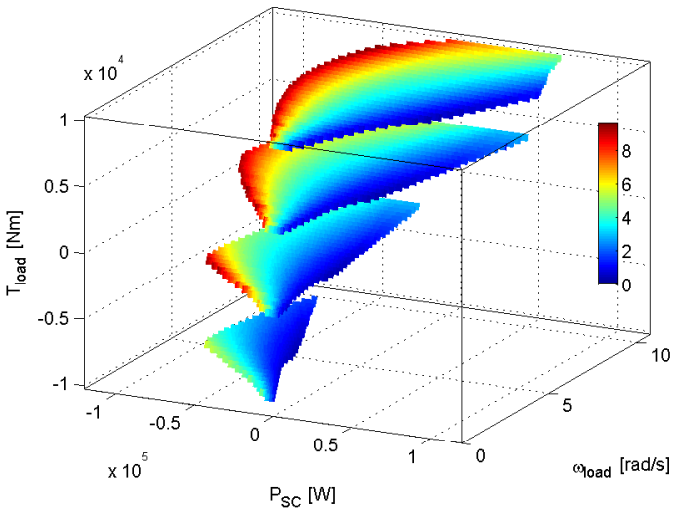


Fig. 4: Fuel usage: instantaneous steady-state ICE fuel rate $\dot{m}(t)$ map (expressed in grams per second) for a wide range of $(\omega_{\text{load}}(t), T_{\text{load}}(t), P_{\text{SC}}(t))$ values. Simulation conditions: $GE(t) = 3$; $CL(t) \in \{0, 1\}$.

2) Driving cycles data: The availability of a representative set of driving cycles data is one of the *prerequirements* of HEV control design. Their characteristics need to be compatible with the vehicle and specifically will be used for:

- tuning of control parameters: in practice it may happen that some control parameters are better suited for some driving cycles, while others are better applicable to other driving cycles. One way to cope with this situation is to perform clustering of data;
- validation and testing of control design.

Clustering: Generally speaking the idea of associating a set of tuned control parameters to clustered load data, is not new (see [16]).

In particular for our use case, the forklift is executing in daily practice a few typical maneuvers, some of which include:

- the so-called Y-cycles: repetitive forward-backward movements, picking and placing payloads from one place to another.
- the so-called R-cycles: long ride cycles where the vehicle is moving for longer distances between different places. Once the destination is reached, the vehicle should start executing Y-cycles.

More details on the clustering procedure, including feature selection can be found in [17].

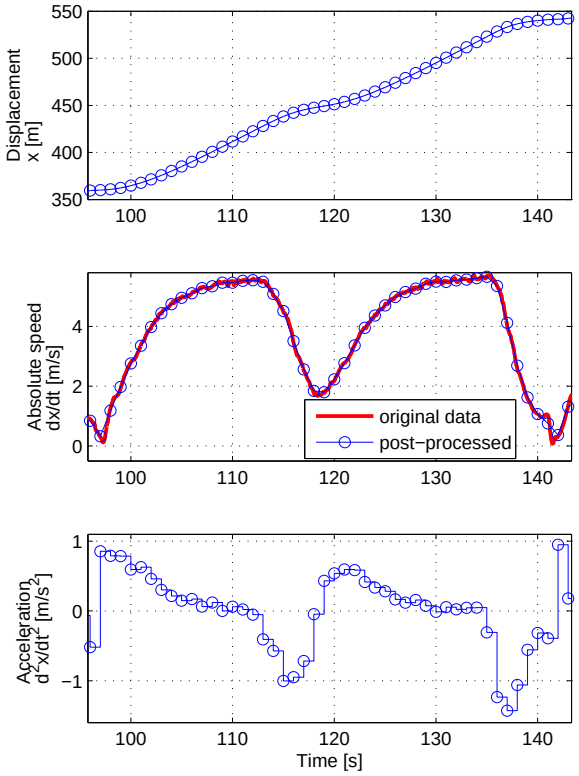


Fig. 5: Recorded data: R-cycle pattern executed by a forklift. Post-processing of original data.

Post-processing of recorded data and Interpretation: Absolute speed profiles measured at the wheels are readily available on any vehicle. They are typically constituted of successive bumps which vary in amplitude and width. An example is given in Fig. 5 showing data collected during actual driving. By post-processing these signals, one can extract other profile curves like displacement, acceleration and eventually obtain the load data $(\omega_{\text{load}}(t), T_{\text{load}}(t))$, with $t \in [t_0, t_{\text{e}}]$.

For the offline simulator we are now faced with two challenges:

- recorded data can be noisy: the solution comes by post-processing (typically use zero-phase filtering);
- the relatively high sampling time Δt from (8) requires to downsample the measured signal. Effectively how to proceed is a challenging task as will be explained hereafter.

In Fig. 5 one curve corresponds to an original measured speed profile. It has a low sampling period (in the order of milliseconds) while the offline simulator uses a sampling period Δt that is 10^3 times higher (in the order of seconds). The question is how to extract meaningful physical relations between the displacement, speed and acceleration profiles at the given sampling period Δt.

The first step is to calculate averages of the original speed profile $v(t)$ by using a sliding window of width Δt. Consequently, local oscillatory phenomena influence is mitigated. These average points will constitute the downsampled speed profile $\bar{v}(t)$. Moreover, in view of the other signals that need to be calculated later on, let us introduce an interpretation aspect: in-between any successive points of $\bar{v}(t)$, we make the assumption that speed is linear (see its linear profile in Fig. 5). One benefit of this windowing technique is that it can be shown that total traveled distances are almost the same, for $v(t)$ and $\bar{v}(t)$: the longer (with respect to time) these signals are, the closer the total traveled distances get to each other.

Now we can define $\omega_{\text{load}}(t) = \bar{v}(t)/R$, where R is the rolling radius of the wheel.

The second step is to calculate the downsampled acceleration signal. Using the interpretation introduced above, acceleration $\bar{a}(t_k)$ is readily given by the slope between two successive speeds $\bar{v}(t_k)$ and $\bar{v}(t_{k+1})$, where t_k is a discrete time instant:

$$\bar{a}(t_k) = \frac{\bar{v}(t_{k+1}) - \bar{v}(t_k)}{t_{k+1} - t_k}, \quad k \in \mathbb{Z}_{\geq 1}$$

Consequently, we interpret that between two successive samples, the acceleration is kept constant (see its stairs profile in Fig. 5).

Now we can calculate $T_{\text{load}}(t_k) = (m\bar{a}(t_k) - \sum_i F_i(t_k))\, R$, where F_i are external forces like rolling friction, air drag, etc.; m is total mass of vehicle.

The third step is to calculate the downsampled displacement signal. Using the same interpretation as above, displacement $\bar{d}(t_k)$ is calculated using the formula:

$$\bar{d}(t_{k+1}) = \bar{d}(t_k) + (\bar{v}(t_k) - \bar{a}(t_k)t_k)\,\Delta t + \frac{\bar{a}(t_k)}{2}\left(t_{k+1}^2 - t_k^2\right)$$

Consequently, according to our interpretation, between two successive samples the displacement follows a smooth profile given by a second order polynomial in time (see its profile, in Fig. 5).

The displacement signal will be used later on to calculate an estimate of the average fuel consumption (see further on, Fig. 7).

3) Backward Dynamic Programming: this iterative algorithm consists in moving one step backward in time and selecting among a population of candidate trajectories, the optimal one. This concept is illustrated in Fig. 6 where some candidate trajectories have been crossed out: they are non-optimal. For our problem, the cost function from (1a) can be used to derive the partial cost between two successive iterations:

$$J^*(t_k, x_k; x_{\text{e}}) = \min_{u_k} \left\{ \int_{t_k}^{t_{k+1}} \dot{m}(\tau, x(\tau), u(\tau))\, \mathrm{d}\tau + J^*(t_{k+1}, x_{k+1}; x_{\text{e}}) \right\}$$

where x_e represents state vector at final time; the integral can be numerically calculated using the rectangular or trapezoidal rule $\int_{t_k}^{t_{k+1}} \dot{m}(\tau)\, d\tau \simeq \Delta t\,(\dot{m}(t_k) + \dot{m}(t_{k+1}))\,/2$. Recall that

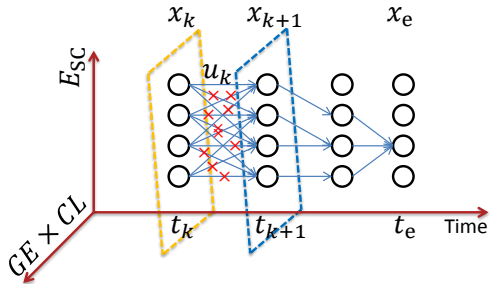


Fig. 6: Backward DP concept.

$u(t_k)$ and $x(t_k)$ were defined in (2) and (3), respectively. J^* is the optimal cost function. It is interesting to notice that a consequence of choosing the trapezoidal integration method is that next state variable depends not only on past information (x_k, u_k), but also on the next decision u_{k+1}, namely

$$x_{k+1} = f(x_k, u_k, u_{k+1})$$

Consequently, this makes its implementation atypical with respect to standard DP.

III. Real-time control: an ECMS-based approach

The ECMS optimization problem has already been formulated in (6)-(7). In addition, we choose to normalize the weighted power quantities that appear inside the cost function, thus avoiding to work with big numbers. For the ease of reading, we rewrite it explicitly:

$$\min_{u(t)} \; L(t) = \frac{\dot{m}(t)}{\dot{m}^{\max}} + \left(s_0 + s_1(SoC(t) - SoC^{\text{ref}})\right) \frac{P_{\text{SC}}(t)}{P_{\text{SC}}^{\max}}$$
subject to: (1b) – (1i)

where $\dot{m}^{\max}$ is in the order of tenths of grams/second; $P_{\text{SC}}^{\max}$ is in the order of hundreds of kilowatts; (s_0, s_1) are tuning parameters and in particular it is interesting to nothe that one can associate a physical meaning to s_1: it acts as the stiffness of a virtual spring that is stretched between the current value of $SoC(t)$ and the desired SoC^{ref}. Consequently, the higher s_1 is set, the more the $SoC(t)$ will have tendency to keep in the neighborhood of SoC^{ref}. On the other hand, s_0 can be related to an average behavior of $SoC(t)$; e.g. the state-of-charge will have tendency to run to its upper saturation limit if s_0 is set too high, at the expense of more fuel being used.

Selection of numerical integration methods

The SoC and the total fuel mass need to be calculated by numerical integration. Since the first one appears as internal state variable in the optimization problem while the latter appears in the cost function, results are expected to differ when switching from one integration method to another. In our work, we have tested two methods, namely rectangle rule (which is the simplest one to implement) and the trapezoidal rule.

$$L(t_k) = \frac{\dot{m}(t_k)}{\dot{m}^{\max}} + \left(s_0 + s_1 \left(\frac{E_{\text{SC}}(t_{k-1}) - P_{\text{SC}}(t_{k-1})\,\Delta t}{E_{\text{SC}}^{\max}} - SoC^{\text{ref}} \right) \right) \frac{P_{\text{SC}}(t_k)}{P_{\text{SC}}^{\max}} \tag{9}$$

$$L(t_k) = \frac{\dot{m}(t_k)}{\dot{m}^{\max}} + \left(s_0 - s_1 \left(\frac{E_{\text{SC}}(t_{k-1})}{E_{\text{SC}}^{\max}} - SoC^{\text{ref}} - \frac{P_{\text{SC}}(t_{k-1})\Delta t}{2\,E_{\text{SC}}^{\max}} \right) \right) \frac{P_{\text{SC}}(t_k)}{P_{\text{SC}}^{\max}} + s_1 \frac{\Delta t}{2\,E_{\text{SC}}^{\max}} \frac{(P_{\text{SC}}(t_k))^2}{P_{\text{SC}}^{\max}} \tag{10}$$

1) Rectangular rule: The discrete time-evolution of the electrical energy is:

$$E_{\text{SC}}(t_k) = E_{\text{SC}}(t_{k-1}) - P_{\text{SC}}(t_{k-1})\,\Delta t$$

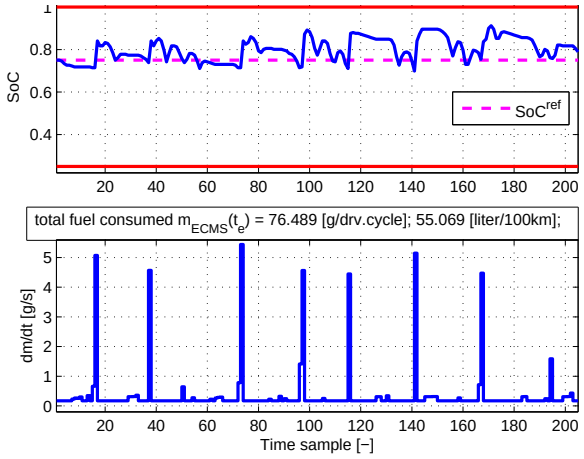
and the cost function becomes (9). Note that $L(t_k)$ is linear in the optimization variable $P_{\text{SC}}(t_k)$.

2) Trapezoidal rule: Compared to the previous integration method, trapezoidal rule is just slightly more complex and needs not only past information, but also from the present in order to compute the time-evolution of discrete variables. In particular, the electrical energy:

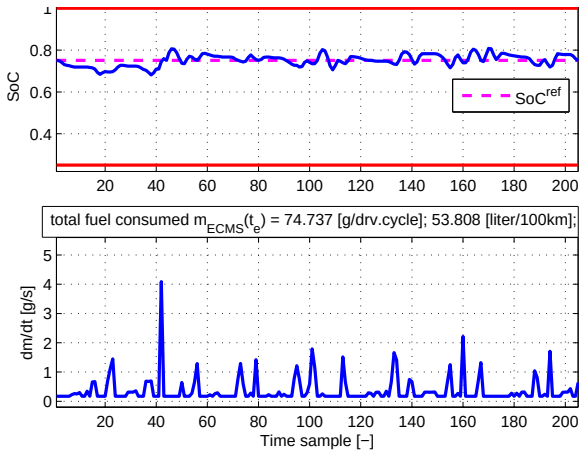
$$E_{\text{SC}}(t_k) = E_{\text{SC}}(t_{k-1}) - \frac{P_{\text{SC}}(t_k) + P_{\text{SC}}(t_{k-1})}{2}\Delta t$$

and the cost function becomes (10). Note that $L(t_k)$ has not only a linear term, but also a quadratic term in the optimization variable $P_{\text{SC}}(t_k)$.

Comparison: In Fig. 7 we compare the SoC and fuel rate $\dot{m}$ profile curves computed using the two numerical schemes. The differences in results are quite visible mainly due to the sampling time Δt which needs to be quite high in order to fulfill constraint (8). The SoC results of the trapezoidal method appear to be smoother than results using the rectangular method. The same remark applies for $\dot{m}$ where prominent spikes appear on the rectangular method. Comparing the total fuel used $m_{\text{ECMS}}(t_{\text{e}})$ (expressed in grams per driving cycle), we see that values are quite close to each other, as one can read in the figures. To ease the reading, an estimated average is also calculated (expressed in liter per 100 kilometers). As expected, we conclude that the choice of the integration method will consequently influence the precision of the calculations and the numerical implementation schemes of both DP and ECMS. Details will be provided further on.



(a) rectangular method



(b) trapezoidal method

Fig. 7: Comparison of two ECMS implementations: each uses a different numerical integration method, while all the other simulation conditions are similar: $(s_0, s_1) = (0.5, 1)$; $\Delta t = 1$ second; $SoC^{\text{ref}} = 0.75$; load data consists of the R-cycle from Fig. 5. Horizontal lines correspond to boundaries $SoC^{\min} = 0.25$ and $SoC^{\max} = 1$, respectively.

IV. Benchmark: ECMS versus DP

A natural question that arises is how well the real-time control ECMS perform when compared against the best possible solution calculated by DP. We apply the methodology recalled in section I-B. Our main focus will be on the sensitivity analysis of the tuning parameters (s_0, s_1).

Cluster-based tuning of (s_0, s_1)*:* We have already introduced the Y-cycles and the R-cycles. Here, we are interested in associating to each of these clusters, a set of (s_0, s_1) that allows the real-time control to work in near-optimal conditions.

Results of Fig. 8 can be used as a performance indicator of how well ECMS copes with DP. (s_0, s_1) candidates are placed on a uniform grid and for each of these pair of points we make a fair comparison between ECMS and DP results by checking that initial conditions (at time t_0) and final conditions (at time t_{e}) of simulated trajectories coincide. Specifically, first we run ECMS forward in time, then use its final conditions to initialize DP. Then we run DP backwards in time and once t_0 is reached we select that optimal trajectory that starts at the same initial points with ECMS.

We begin our simulations by first disregarding the self-sustaining SoC constraint (1c), while keeping all the rest

active. This will provide us an indication of how well, in general, the ECMS is able to get close to the DP results. Fig. 8a illustrates the relative total fuel consumption increase defined according to the formula:

$$100\ \left(m_{\text{ECMS}}(t_{\text{e}}) - m_{\text{DP}}(t_{\text{e}})\right) / m_{\text{DP}}(t_{\text{e}})$$

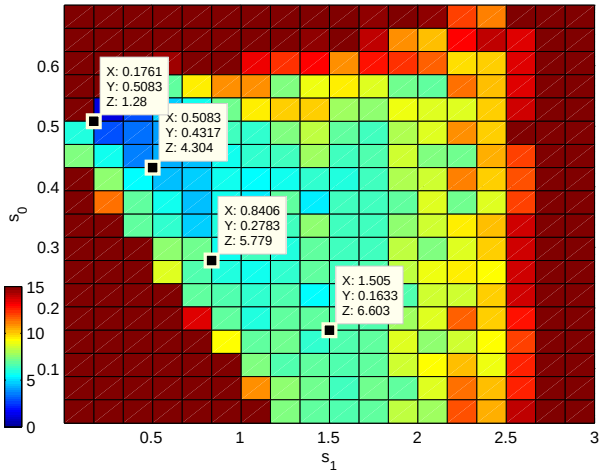
and is calculated in percentage (%). ECMS is able to achieve less than 5% fuel usage increase when compared to DP, which is quite remarkable. However, not all of these (s_0, s_1) are able to ensure the self-sustaining SoC, therefore some of them still need to be disregarded.

Next, we include the self-sustaining SoC constraint (1c) in our simulations, together with all the other constraints: the Ω-set is chosen to be the interval $[-0.001, 0.05] + SoC^{\text{ref}}$. Results are illustrated in Fig. 8b. Note that a region of arbitrary width, situated on the left part, has been grayed out: the reason is to allow s_1 to act as increased stiffness, thus leveraging the self-sustaining ability to handle new, unknown, unpredictable driving cycles. This simple mechanism acts as a robustness margin.

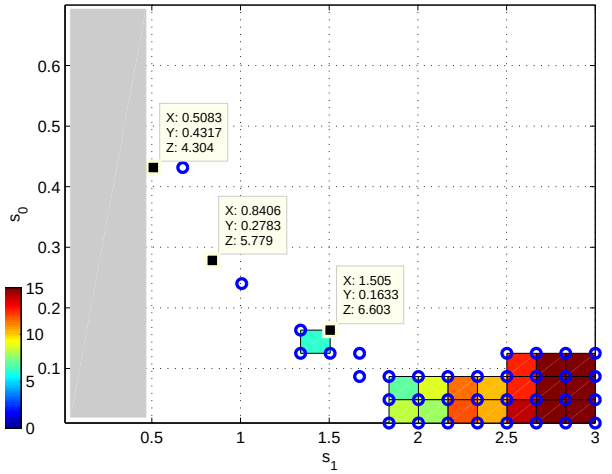
In the end, the control designer will select only one optimal point (s_0, s_1) by visual inspection of Fig. 8b and consequently associate it to the specific cluster. All the other information is disregarded.

V. Conclusions

This article presented a methodological approach that allows a control designer to measure the quality and performance of real-time powertrain control. In particular, we showed how to implement an ECMS supervisory control which, due to its performance in simulation when compared with DP, appears to be a strong candidate towards rapid prototyping on a HEV forklift truck.



(a) without guaranteeing self-sustaining SoC



(b) guaranteeing self-sustaining SoC

Fig. 8: Benchmark: Relative fuel consumption increase between ECMS and DP, expressed in percentage. Simulation conditions: $\Delta t = 1$ second; $SoC^{\text{ref}} = 0.75$; load data consists of the R-cycle from Fig. 5; both ECMS and DP simulations are implemented using trapezoidal numerical integration method

References

[1] A. Sciarretta and L. Guzzella, "Control of hybrid electric vehicles," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 60–70, April 2007.

[2] L. Guzzella and A. Sciarretta, "Model-based supervisory control for energy optimization of hybrid-electric vehicles," in *The Control Handbook: Control System Applications*, 2nd ed., W. S. Levine, Ed. Boca Raton: CRC Press, 2011, ch. I.4.

[3] ——, *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*, 3rd ed. Berlin: Springer, 2013.

[4] C.-C. Lin, H. Peng, and J. W. Grizzle, "A stochastic control strategy for hybrid electric vehicles," in *American Control Conf.*, Boston, MA, June-July 2004, pp. 4710–4715.

[5] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang, "Power management strategy for a parallel hybrid electric truck," *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 6, pp. 839–849, November 2003.

[6] A. Sciarretta, M. Back, and L. Guzzella, "Optimal control of parallel hybrid electric vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 3, pp. 352–363, May 2004.

[7] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, "A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management," *Eur. J. Control*, vol. 11, no. 4–5, pp. 509–524, December 2005.

[8] T. Hofman, M. Steinbuch, R. van Drutten, and A. Serrarens, "Rule-based energy management strategies for hybrid vehicles," *Int. J. Electric and Hybrid Vehicles*, vol. 1, no. 1, pp. 71–94, 2007.

[9] P. Pisu and G. Rizzoni, "A comparative study of supervisory control strategies for hybrid electric vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 506–518, May 2007.

[10] X. Ye, Z. Jin, X. Hu, Y. Li, and Q. Lu, "Modeling and control strategy development of a parallel hybrid electric bus," *Int. J. Automotive Technol.*, vol. 14, no. 6, pp. 971–985, 2013.

[11] C. Mi, M. A. Masrur, and D. W. Gao, *Hybrid Electric Vehicles: Principles and Applications with Practical Perspectives*. John Wiley & Sons, 2011.

[12] D. Bertsekas, *Dynamic programming and optimal control*, 3rd ed. Belmont, MA: Athena Scientific, 2005, vol. 1.

[13] O. Sundström and L. Guzzella, "A generic dynamic programming matlab function," in *IEEE Multi-conference on Systems and Control*, Saint Petersburg, Russia, July 2009, pp. 1625–1630.

[14] O. Sundström, D. Ambühl, and L. Guzzella, "On implementation of dynamic programming for optimal control problems with final state constraints," *Oil & Gas Science and Technology - Rev. IFP*, vol. 65, no. 1, pp. 91–102, 2010.

[15] H.-D. Lee, S.-K. Sul, H.-S. Cho, and J.-M. Lee, "Advanced gear-shifting and clutching strategy for a parallel-hybrid vehicle," *IEEE Industry Application Magazine*, vol. 6, no. 6, pp. 26–32, November/December 2000.

[16] B. Gu and G. Rizzoni, "An adaptive algorithm for hybrid electric vehicle

energy management based on driving pattern recognition," in *Proc. of ASME International Mechanical Engineering Congress and Exposition*, Chicago, Illinois, USA, November 2006, pp. 249–258.

[17] K. Rutten, B. Piccart, C. S. Teodorescu, and B. Depraetere, "Application of random forests to create task-based control for a parallel hybrid forklift - a case study," in *ENBIS - the European Network for Business and Industrial Statistics*, Prague, Czech Republic, September 2015.