Real-time Predictive Control Strategy for a Plug-in Hybrid Electric Powertrain[☆]

Elsevier[1]

Radarweg 29, Amsterdam

Amir Taghavipour[a,], Nasser L. Azad[a], John McPhee[a]*

[a]Systems Design Engineering Department, University of Waterloo, 200 University Ave. W, Waterloo, Ontario

Abstract

Model predictive control is a promising approach to exploit the potentials of modern concepts and to fulfill the automotive requirements. Since, it is able to handle constrained multi-input multi-output optimal control problems. However, when it comes to implementation, the MPC computational effort may cause a concern for real-time applications. To maintain the advantage of a predictive control approach and improve its implementation speed, we can solve the problem parametrically. In this paper, we design a power management strategy for a Toyota Prius plug-in hybrid powertrain (PHEV) using explicit model predictive control (eMPC) based on a new control-oriented model to improve the real-time implementation performance. By implementing the controller to a PHEV model through model and hardware-in-the-loop simulation, we get promising fuel economy as well as real-time simulation speed.

Keywords: Automotive powertrain, plug-in hybrid electric vehicle, power management, explicit model predictive control.

2010 MSC: 00-01, 99-00

[☆]Fully documented templates are available in the elsarticle package on CTAN.

*Corresponding author, Email:ataghavi@uwaterloo.ca

[1]Since 1880.

1. Introduction

Rising fuel costs, stringent legal standards and increasing environmental concerns have made car manufacturers produce vehicles with high fuel efficiency and low emissions. This is possible due to new components and technologies that are introduced in automotive powertrains (e.g. turbo charging, exhaust gas recirculation, continuous variable transmission). Unfortunately, it seems that the control software of powertrains remains backward with respect to their complexity [1]. While most current strategies are based on heuristics and look-up tables, [2, 3] have shown that model predictive control has a large potential for automotive powertrain control design. One of the most attractive solutions for sustainable transportation to car manufacturers is the hybrid electric powertrain. Hybrid electric vehicles exploit energy production and energy storage systems to achieve improved fuel economy with respect to conventional powertrains. For further improvement in fuel economy and emissions performance, plug-in hybrid electric vehicles (PHEVs) were introduced. These vehicles benefit from a larger power storage system which leads to a longer full-electric range in comparison to HEVs. As such, they can significantly reduce the environmental footprint of the vehicle. These vehicles are one step closer to the full electric vehicle (EV) but more attractive to the market with range-anxiety concerns for EVs.

To maximize fuel economy and emissions performance, control strategies are required to estimate the amount of energy to be produced and stored optimally. HEV power management decides on how much power should be produced by the internal combustion engine and how much should be stored/released from the electric drive to achieve the demanded power at the wheels, while enforcing the operating constraints, and to optimize fuel economy at the same time. The PHEV's larger battery provides more flexibility and on the other hand more complexity for the power management system in comparison to HEVs. Several strategies for HEV/PHEV power management have been proposed, including dynamic programming (DP), stochastic dynamic programming (SDP), equiv-

alent fuel consumption minimization (ECMS), and model predictive control (MPC). To fully exploit these strategies' capability for improving fuel economy and emission performance, complete information of the driving schedule is required beforehand. Unfortunately, information about the future driving cycle is not available during conventional driving. Furthermore, planning for the whole future driving cycle is computationally demanding. Even by having the exact driving schedule available at the starting point, DP cannot be implemented in real time, although it can offer the most efficient solution. As a result, rule-based strategies based on DP results are usually implemented to the powertrain controller.

Stochastic models can reduce some of these problems, but the choice of stochastic model and its identification still faces some challenges [4]. Moura et al. derived an optimal power management scheme for a plug-in hybrid vehicle (power-split architecture) based on stochastic dynamic programming [5]. Musardo et al. [6] proposed an adaptive ECMS (A-ECMS) method based on driving condition, which calculates the equivalency factor in ECMS technique for parallel HEVs. Tulpule et al. [7] used the ECMS approach to design a power management strategy in series and parallel PHEVs by considering two operation modes (EV and Blended).

Model predictive control is another approach for designing a power management strategy. The success of MPC in industrial applications is due to its ability to handle processes with many manipulated and controlled variables and constraints in a rather systematic manner [8]. Furthermore, MPC allows an objective function to be optimized by the controller. Other advantageous MPC features are the capability of dealing with time delays [9], of taking advantage from future information [10], and of rejecting measured and unmeasured disturbances [11]. It is noteworthy that MPC embodies both (receding horizon) optimization and feedback adjustment. Model predictive control has been applied to diesel engines control [12], catalyst control [13], transmission control [14], and HEV [15, 16]/ PHEV [17, 18] power management.

Despite the obvious benefits of MPC, its capabilities are limited due to the

computational effort required for solving the online optimization problem of the MPC [19]. In our previous work [20], we compared the performance of A-ECMS strategy to MPC approach for designing a power management strategy for a PHEV. Both strategies improved fuel economy by 10% in comparison to the baseline control strategy, but A-ECMS was approximately 15 % faster than MPC.

This shortcoming can be overcome by using the so-called explicit MPC (eMPC) methods. In eMPC the online optimization problem involved in the MPC is solved off-line using multi-parametric programming approaches and the control variables and the value function of the optimization problem are derived as explicit functions of the system state variables, as well as the critical regions of the state-space where these functions are valid. Such a function is piecewise affine in most cases, so that the MPC controller maps into some polyhedral regions that can be stored as a look-up table of linear gains [8]. The key advantage of explicit MPC is that it can replace the online optimization problem of the traditional MPC with a set of function evaluations, significantly reducing the computational effort required for the implementation [19].

Explicit MPC techniques [21] can be used to synthesize the controller as a piecewise affine function. With this approach, the MPC can be implemented in a micro-controller without the need for an optimization solver and satisfying limitations on memory and computational power characteristic of automotive electronic control units (ECUs).

In practice, explicit MPC is limited to relatively small problems (typically 1-2 inputs, up to 5-10 states, up to 3-4 free control moves). But it allows one to reach very high sampling frequencies and requires a very simple control code to be embedded in the system [8].

Industrial problems addressed through explicit MPC techniques have been reported in technical papers, starting from what is probably the first work in this domain which is traction control [22]. Most applications of explicit MPC have been reported in the automotive domain and electrical power converters.

The hardware-in-the-loop (HIL) systems have become efficient tools for strat-

egy and interface software development [23]. The HIL systems allow a lot of control function development to be done and verified ahead of a vehicle build. Improved software quality and early verification of software leads to reduced vehicle commissioning time if a minimum level of functionality exists before being handed off to the various engineering teams for further development [24].

The authors in [25] applied the HIL approach to a parallel HEV configuration in order to analyze fuel reduction benefits due to hybridization without any influence of vehicle characteristics or engine technology improvement. Petersheim and Brennan [26] down-scaled the electric machine and the battery of an HEV to perform a lab-scale HIL simulation.

The advent of microprocessor-based electronic control units (ECUs) for car engines and powertrain created a need for new tools for testing, calibrating, and validating these ECUs. HIL simulation met this need, and became a key technology for engine ECU testing and calibration [27].

Lee et al. present a formal process for developing such a HIL simulator that uses automatic code generation to streamline the transition of control system designs from pure simulation to a commercial embedded code [28].

The use of HIL simulation for automotive ECU development is not limited to engine applications. In fact, HIL simulation has been used effectively for the development, calibration, and validation of transmission and driveline electronic control units.

In this paper, we propose a near-optimal, real-time implementable solution for a PHEV power management strategy using explicit model predictive control. In [4], the authors used an eMPC solution for a series HEV, but to the best of our knowledge, this is the first time that an explicit model predictive controller is designed and implemented for a power-split PHEV architecture. Due to system complexity, there are some challenges for finding an appropriate control-oriented model. Using eMPC is only practical for relatively small problems because the size of the controls database is exponentially increased by the number of state variables. Therefore, the control-oriented model should be very simple, but accurate enough to capture the complex dynamics of a power-split PHEV

powertrain. Moreover, the control-oriented model and the optimization cost function should be chosen in such a way that they guarantee a feasible solution, optimality, stability and desirable performance for the controller. The proposed control system is a switched discrete-time one. As a result, stability analysis is required to make sure that the control system keeps its performance for all PHEV operating points. Therefore, we introduce an innovative control-oriented model that is very simple and addresses the mentioned issues.

In the next section, we introduce the simulation model. Then, we discuss the power management strategy design and implementation by developing an appropriate control-oriented model. In section 4, we show the polytopes resulting from solving the eMPC and discuss the physical interpretation of different regions. Then we discuss the stability of the closed-loop system. In section 7, we apply the designed controller to the simulation model, which is followed by HIL testing. Finally, we discuss the results and compare them with the MPC approach.

2. Powertrain simulation model

Among the different architectures for a hybrid electric vehicle, the power split configuration seems to be the most efficient for a limited size of battery [29]. In a power split configuration, the engine, the electric motor and the generator are connected to each other by means of 2 planetary gear sets (PGS). Figure 1 shows the schematic of the Toyota Prius plug-in powertrain. The engine shaft and first electric machine are connected to the carrier and the sun gear of PGS 1. The second electric machine is connected to the sun gear of the second PGS.

To derive the dynamics of the system, it is assumed that the mass of the pinion gears is small, there is no friction, no tire slip or efficiency loss in the powertrain. By considering the vehicle longitudinal dynamics and an internal resistance model for the battery, the equation of the system can be written as (1), (2) and (3).

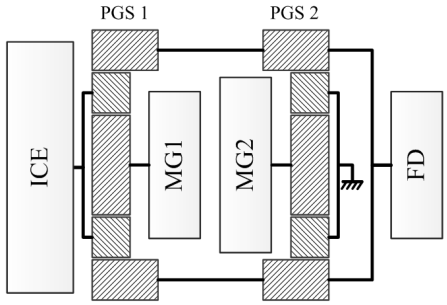

Figure 1: Toyota Prius plug-in powertrain schematic. ICE: Internal combustion engine, MG: Motor/Generator, PSG: Planetary gear set, FD: Final drive

$$\begin{aligned}(\tfrac{I_v^{'}(s_1+r_1)^2}{r_1 I_e^{'} K}+\tfrac{I_v^{'} s_1^2}{r_1 I_g^{'} K}+r_1)(\tfrac{r_2}{s_2})\dot{\omega}_r &= (\tfrac{(s_1+r_1)}{I_e^{'}})T_e+(\tfrac{s_1}{I_g^{'}})T_g \\ +(\tfrac{(s_1+r_1)^2}{r_1 I_e^{'}}+\tfrac{s_1^2}{r_1 I_g^{'}})(\tfrac{r_2}{s_2})T_m &- (\tfrac{(s_1+r_1)^2}{r_1 I_e^{'} K}+\tfrac{s_1^2}{r_1 I_g^{'} K})C \end{aligned} \tag{1}$$

$$\begin{aligned}(\tfrac{I_e^{'} r_1^2 K}{(s_1+r_1) I_v^{'}}+\tfrac{I_e^{'} s_1^2}{(s_1+r_1) I_g^{'}}+s_1+r_1)\dot{\omega}_e &= -(\tfrac{r_1}{I_v^{'}})C \\ (\tfrac{r_1^2 K}{(s_1+r_1) I_v^{'}}+\tfrac{s_1^2}{(s_1+r_1) I_g^{'}})T_e+(\tfrac{r_1 K}{I_v^{'}})T_m &- (\tfrac{s_1}{I_g^{'}})T_g \end{aligned} \tag{2}$$

$$\dot{SoC} = -\frac{V_{oc}-\sqrt{V_{oc}^2-4(T_m\omega_r\eta_m^{-k}-T_g\omega_g\eta_g^{k})R_{batt}}}{2R_{batt}Q_{batt}} \tag{3}$$

where $I_v^{'}=(m/K)R_{tire}^2+(I_m+I_r)K$, $I_g^{'}=I_g+I_s$, and $I_e^{'}=I_e+I_c$. Moreover, for the first planetary gear set we have $r_1\omega_r+s_1\omega_g=(r_1+s_1)\omega_e$ as a kinematic constraint. C denotes the total resistive torque at the driver wheels and can be found as $C=mgf_rR_{tire}+0.5\rho A_f c_d(\omega_r/K)^2R_{tire}^3$ by ignoring the wheel slip effect and gradability. The total resistive torque consists of two terms: one is related to the equivalent tire rolling resistance and the second one considers the effect of the drag force.

In this system, there are 3 state variables: first ring gear speed (ω_r), which is proportional to the vehicle velocity, engine speed (ω_e), and battery state of

charge (SoC). There are 3 inputs: engine (T_e), motor (MG2) (T_m) and generator (MG1) torque (T_g). η_m and η_g represent motor and generator efficiency, respectively. When the battery is discharging k=1. But k=-1 for battery charging. The model parameters are listed in Table 1.

3. Power management strategy design

In this paper, we design a power management strategy for a PHEV by using multi-parametric model predictive control approach. Bemporad et al. [21] presented a technique to determine the linear quadratic regulator for constrained systems through offline multi-parametric linear programming (mp-LP) and multi-parametric quadratic programming (mp-QP). The control law was shown to be piecewise linear and continuous, and could be implemented as a lookup table, i.e., different linear state feedback laws apply to different polyhedral regions. Therefore, the online control computation is reduced to determining the region associated with the current state and then applying the stored control law associated with that region.

First of all we need a control-oriented model to design the controller. The design procedure of the controller can be divided into 2 different stages: offline and online. The objective of the offline procedure is to populate some look-up tables which contain the appropriate control actions for different system operating points. In this procedure, we solve the multi-parametric programming problem with an initial condition. The result is a polytope with a specific control action. Then we have to explore the state space to find the other polytopes and control actions. There may be a large number of look-up tables as the result of solving the optimization problem. In this case, region reduction methods can be used to remove sone redundant constraints in the optimization problem in order to downsize the look-up tables and improve controller speed in the implementation stage. Finally we find the least number of the look-up tables.

The online procedure happens during controls implementation. Here, we need a fast and efficient search algorithm to look up the tables to find which

Table 1: Variables and Parameters Description

Symbol	Unit	Value	Description
I_g	kgm^2	0.1	Generator equivalent inertia
I_s	kgm^2	0.1	Sun gear equivalent inertia
I_e	kgm^2	0.5	Engine equivalent inertia
I_c	kgm^2	0.1	Carrier equivalent inertia
I_m	kgm^2	0.1	Motor equivalent inertia
I_r	kgm^2	0.1	Ring gear equivalent inertia
R_{tire}	m	0.3	Tire average radius
m	kg	1440	Vehicle curb weight
K	–	3.268	Final drive gear ratio
f_r	–	0.02	Equivalent rolling resistance coefficient
A_f	m^2	2.5	Vehicle frontal area
ρ	kg/m^3	1.2	Air density
c_d	–	0.25	Drag coefficient
r_1	–	78	First ring gear teeth No.
s_1	–	30	First sun gear teeth No.
r_2	–	79	Second ring gear teeth No.
s_2	–	30	Second sun gear teeth No.
V_{oc}	V	207.2	Battery open circuit voltage
R_{batt}	Ω	0.85	Battery open circuit resistance
Q_{batt}	Ah	21.72	Battery nominal capacity

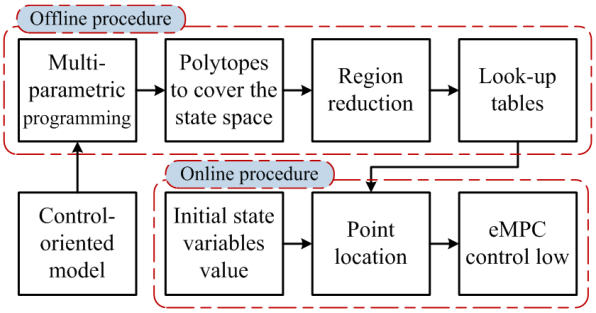

Figure 2: eMPC design procedure

one of those polytopes contains the initial state variable. This algorithm is called point location. Once the corresponding polytope is found, the eMPC law is obtained. These steps are shown in Figure 2.

In the following subsections, we address each step separately.

3.1. Control-oriented model

We have to consider a relatively simple model to take advantage of explicit model predictive control approach. To obtain smaller look-up tables and make it possible to implement the controller to a commercial hardware with a limited amount of memory and computational power, we use the following model inside the controller:

$$Z(k+1) = \mathbf{A}Z(k) + \mathbf{B}U(k) \tag{4}$$

where $Z = \begin{bmatrix} SoE, & E \end{bmatrix}^T$ and $U = \begin{bmatrix} P_{BAT}, & P_{ENG}, & P_{BRK} \end{bmatrix}^T$.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} a_1 & 0 & a_2 \\ a_3 & a_4 & a_5 \end{bmatrix}$$

There are 2 state variables in this model: battery state of energy (SoE) and tractive energy E. Battery SoE is defined as the ratio of battery stored/released

energy to the battery total usable energy. We define the tractive energy E the integral of the power which is required to propel the vehicle, i.e. tractive energy is the power required in a time step to propel the vehicle. We have 2 sources of energy in the powertrain: the battery and the fuel. We define 2 inputs to address these sources as P_{BAT} and P_{ENG} which are the battery and the engine power, respectively. The battery power is the summation of powers from 2 onboard motors. The third input is required to stop the vehicle. As a result, we add braking power to the control actions. The coefficients a_1-a_5 are chosen based on battery pack capacity, efficiencies (electrical and mechanical) as well as the control period that is considered in the design procedure: $a_1 = -\Delta t/E_{BAT}$, $a_2 = \eta_{regen}\Delta t/E_{BAT}$, $a_3 = \eta_{BAT}\Delta t$, $a_4 = \eta_{ENG}\Delta t$, $a_5 = -\Delta t$ where η_{regen}, η_{BAT}, η_{ENG} are the percentage of the regenerative braking power that returns to the battery, electrical motor propulsion efficiency (including battery and power electronic efficiencies), and the engine propulsion efficiency (including the mechanical efficiencies of the powertrain components), respectively. E_{BAT} and Δt are the maximum stored energy in the battery and the desired controller's sampling time.

3.2. Optimization problem formulation

In a hybrid electric vehicle, researchers are interested in improving the fuel economy and emissions performance while maintaining the vehicle drivability. Emissions are not considered in this paper, but reducing the fuel consumption has an important contribution to the emissions performance.

In a PHEV, fuel economy is closely related to the battery depletion trajectory. As a result, we consider a tracking term inside the cost function that SoE should follow at each time step (SoE_{ref}). To address drivability, we consider another term inside the cost function to ensure that the hybrid powertrain provides adequate propulsion power, the driver request (E_{ref}). We also need to reduce the fuel consumption as our main objective. Since the engine fuel consumption is proportional to the engine generated power, we should minimize the engine power as one of the assumed control actions. The cost function along

the prediction horizon can be written as follows:

$$\begin{gathered}\min_U \sum_{j=1}^{N_p} \left\{ Y^T(j)\mathbf{Q}Y(j) + U^T(j)\mathbf{R}U(j) \right\} \\ s.t. \\ \left[\begin{array}{c} SoE_{min} \\ E_{min} \end{array} \right] \leq \left[\begin{array}{c} SoE \\ E \end{array} \right] \leq \left[\begin{array}{c} SoE_{max} \\ E_{max} \end{array} \right] \\ U_{min} \leq U \leq U_{max} \end{gathered} \tag{5}$$

where $Y = \left[\begin{array}{cc} SoE - SoE_{ref}, & E - E_{ref} \end{array} \right]^T$, N_p is the prediction horizon length, and $\mathbf{Q}$ and $\mathbf{R}$ are.

$$\mathbf{Q} = \left[\begin{array}{cc} \omega_1 & 0 \\ 0 & \omega_2 \end{array} \right]$$

$$\mathbf{R} = \left[\begin{array}{ccc} \omega_3 & 0 & 0 \\ 0 & \omega_4 & 0 \\ 0 & 0 & \omega_5 \end{array} \right]$$

The above cost optimization is subjected to the constraints on state variables and control actions. The weighting parameters ω_i $(i = 1, 2, .., 5)$ should be tuned for the best performance.

There are 4 parameters in the cost function: 2 state variables (SoE, E) and 2 setpoints (SoE_{ref}, E_{ref}). To separate the parameters from each other, we have to rewrite (5) as:

$$\begin{gathered}\min_U \sum_{j=1}^{N_p} \left\{ X^T(j)\mathbf{H}X(j) + U^T(j)\mathbf{R}U(j) \right\} \\ s.t. \\ \mathbf{G}U \leq \mathbf{W} + \mathbf{S}X \end{gathered} \tag{6}$$

where $X = \left[\begin{array}{cccc} SoE, & E & SoE_{ref}, & E_{ref} \end{array} \right]^T$, $X_{min} = \left[\begin{array}{cc} SoE_{min}, & E_{min} \end{array} \right]^T$, $X_{max} = \left[\begin{array}{cc} SoE_{max}, & E_{max} \end{array} \right]^T$, and the matrices are:

$$\mathbf{H} = \begin{bmatrix} \omega_1 & 0 & -\omega_1 & 0 \\ 0 & \omega_2 & 0 & -\omega_2 \\ -\omega_1 & 0 & \omega_1 & 0 \\ 0 & -\omega_2 & 0 & \omega_2 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I}_{3\times 3} \\ \mathbf{I}_{3\times 3} \\ \mathbf{O}_{4\times 3} \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} -U_{min} \\ U_{max} \\ -X_{min} \\ X_{max} \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{O}_{6\times 4} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$

In multi-parametric programming, the objective is to find the optimizer U^*, for a whole range of parameters X, i.e. $U^*(X)$ as an explicit function of the parameter X. The cost function is quadratic, so we are solving a multi-parametric quadratic programming (mp-QP) problem. As shown in [30], we wish to solve problem (6) for all X within the polyhedral set of feasible values X_N. According to [31], if we consider the multi-parametric quadratic program (6), then the set of feasible parameters X_N is convex, the optimizer U^* is continuous and piecewise affine (PWA), and the optimal function J^* is continuous, convex and piecewise quadratic.

$$U^*(X) = f_i X + g_i, X \in T_i = \{X \mid h_i X \leq k_i\}; i = 1, ..., N \tag{7}$$

Each $\{T_i\}_{i=1}^N$ defines a polytope which will be referred as a region. Note that the evaluation of the PWA solution (6) of the mp-QP problem provides

the same result as solving the quadratic program, i.e. for any given parameter X, the optimizer $U^*(X)$ is identical to the optimizer obtained by solving the quadratic program (6) for X.

To solve the mp-QP problem we need to solve an active constraint identification problem. A feasible parameter $\hat{X}$ is determined and the associated QP (6) is solved. This will yield the optimiser U^* and active constraints, defined as inequalities that are active at solution. The rows indexed by the active constraints are extracted from the constraint matrices $\mathbf{G}$, $\mathbf{W}$ and $\mathbf{S}$ to form the matrices $\mathbf{G}_A$, $\mathbf{W}_A$ and $\mathbf{S}_A$

Now, we can compute the regions. It is possible to use the Karush-Kuhn-Tucker (KKT) conditions to obtain an explicit representation of the optimiser $U_N(x)$ which is valid in some neighborhood of $\hat{X}$:

$$\begin{gathered} \mathbf{H}U + \mathbf{G}^T\lambda = 0 \\ \lambda^T \left(\mathbf{G}U - \mathbf{W} - \mathbf{S}\hat{X}\right) = 0 \\ \lambda \geq 0 \\ \mathbf{G}U \leq \mathbf{W} + \mathbf{S}\hat{X} \end{gathered} \tag{8}$$

We can find the optimized variable: $U = -\mathbf{H}^{-1}\mathbf{G}^T\lambda$. For inactive constraints, $\lambda = 0$. For active constraints and the corresponding Lagrange multipliers λ_A, inequality constraints are changed to equalities. Substituting for U from (7) into the equality constraints gives:

$$\begin{gathered} -\mathbf{G}_A\mathbf{H}^{-1}\mathbf{G}_A^T\lambda_A + \mathbf{W}_A + \mathbf{S}_A\hat{X} = 0 \\ \implies \lambda_A = -(\mathbf{G}_A\mathbf{H}^{-1}\mathbf{G}_A^T)^{-1}(\mathbf{S}_A\hat{X} + \mathbf{W}_A) \end{gathered}$$

The optimal control trajectory U are given as affine functions of $\hat{X}$

$$U^*(\hat{X}) = \mathbf{H}^{-1}\mathbf{G}_A^T(\mathbf{G}_A\mathbf{H}^{-1}\mathbf{G}_A^T)^{-1}(\mathbf{S}_A\hat{X} + \mathbf{W}_A) = f_i\hat{X} + g_i \tag{9}$$

In the next step, the set of states is determined where the optimizer U^* satisfies the same active constraints and is optimal. Such a region is characterized by two inequalities and is written compactly as $h_iX \leq k_i$ where

$$h_i = \begin{bmatrix} \mathbf{G}f_i - \mathbf{S} \\ (\mathbf{G}_A\mathbf{H}^{-1}\mathbf{G}_A^T)^{-1}\mathbf{S}_A \end{bmatrix}$$

$$k_i = \begin{bmatrix} \mathbf{W} - \mathbf{G}g_i \\ -(\mathbf{G}_A\mathbf{H}^{-1}\mathbf{G}_A^T)^{-1}\mathbf{W}_A \end{bmatrix}$$

Once the controller region is computed, the algorithm proceeds iteratively until the entire feasible state space X_N is covered with controller regions T_i, i.e. $X_N = \cup_{i=1,\ldots,N} T_i$.

3.3. Region reduction

At the implementation stage, a small number of constraints defining a region is preferable, since the controller must quickly check the constraints to find the appropriate control action. Therefore, computation of the minimal representations of the controller regions T_i where h_i and k_i are given, according to (11), can significantly reduce the computational load in most multi-parametric programming solvers [32]. There are a couple of approaches to identify and remove redundant constraints to reduce the number of regions. An typical way to address this problem is to solve n-LPs (in the worst-case, $n-1$ constraints) for each region in order to detect and remove all redundant constraints [33]. Another approach, called ray shooting [34], is suitable for the cases where the fraction of redundant constraints is low. On the other hand, the bounding box approach is most useful for polytopes with many easily detected redundant constraints. The region reduction that is used here is a combination of ray shooting and bounding box in order to find the redundant constraints even faster [35].

3.4. Point location problem

In this part, we consider the point-location or set membership problem for the class of discrete-time control problems with linear state and input constraints for which an explicit time-invariant piecewise state feedback control law over a set of overlapping polyhedral regions is given. The point-location problem comes

into play online when evaluating the control law. One must identify the state space region in which the measured state lies at the current sampling instance. As the number of defining regions grows, a purely sequential search through the regions is too slow to achieve high sampling rates. Hence, it is important to find an efficient online search strategy.

However, due to the combinatorial nature of the problem, the number of state space regions over which the control look-up table is defined grows in the worst case exponentially [21], [31]. Here, the well-known concept of interval trees [36] is used to find a list of candidates that are possible solutions to the point-location problem. Standard interval trees are efficiently ordered binary search trees for determining a set of possibly overlapping one-dimensional line segments that contain a given point or line segment. The mentioned line segments can be found through the bounding box approach. Then, a local search needs to be done on the list of candidates in order to determine the polytope to which the current state variable belongs [35]. The optimization problem has been solved here using multi-parametric toolbox [37]. After solving the mp-QP problem, we obtain lookup tables and control actions. The following section discusses the lookup tables further.

4. Resulting polytopes

Based on the drive cycle, maximum demanded power, and also the battery state of charge, we can discretize the SoE_{ref} and E_{ref} range. Here, we define 9 levels for each. By solving the mp-QP problem, we end up with 81 different sets of polytopes, each containing a definite control action. The total number of polytopes in each set is 3153. Figure 3 shows how the polytopes are distributed for different SoE_{ref} and E_{ref} levels.

By doing region reduction, we can reduce the total number of polytopes to 3123. To get more insight to the problem, we can consider the set corresponding to $E_{ref} = 0$ and $SoE_{ref} = 60$. It consists of 33 polytopes. As shown in Figure 4, the number of polytopes around the reference set points is most concentrated.

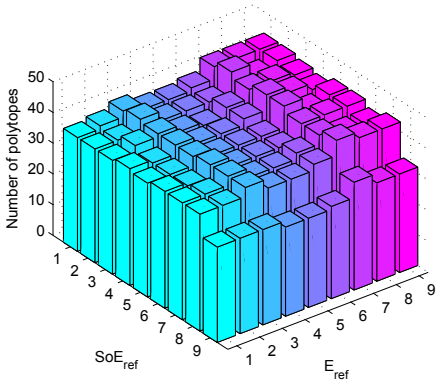

Figure 3: Number of polytopes for different levels of E_{ref} and SoE_{ref}

The reason is that the eMPC controller is supposed to track a predefined level of SoE and E.

Figure 5 shows the control action versus different measured values of SoE and E (initial conditions) at the current sampling instance. It contains 33 polytopes. We can analyze this figure in 4 different regions, labelled I-IV in Figure 4:

(I) $E > E_{ref}$ and $SoE < SoE_{ref}$: In this part, the controller has to increase the battery state of charge and slow down the vehicle. There are 2 ways to do that. One is to increase the engine power in order to charge the battery. The other is to increase the braking power in order to get use of regenerative braking. Figure 5 shows that the controller uses both ways to get to the objective in region (I). The battery power should be negative indicating that it is being charged (Figure 5.a). Moreover, if $\delta E = |E - E_{ref}|$ is high, the braking power will be more (as shown in Figure 5.c)

(II) $E < E_{ref}$ and $SoE < SoE_{ref}$: Since $E < E_{ref}$, the powertrain is required to provide propulsion power from the engine and/or the electric drive. But, in this case, the battery state of energy is less than the reference value ($SoE < SoE_{ref}$), so we cannot use the electric drive to assist the engine to

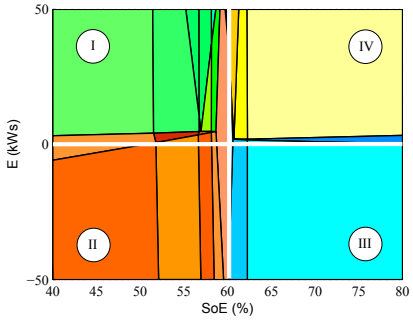

Figure 4: Polytope set for $E_{ref} = 0$ and $SoE_{ref} = 60$

propel the vehicle by depleting the battery further. On the other hand, we cannot use the regenerative braking for charging the battery (zero in Figure 5.c), since we cannot stop the vehicle. As a result, the engine plays a key role in this case. As shown in Figure 5.a $P_{BAT} < 0$, because SoE is less than the reference value. Region (II) is the worst case among all 4 propulsion scenarios.

(III) $E < E_{ref}$ and $SoE > SoE_{ref}$: In this region, we should accelerate the vehicle ($P_{BRK} = 0$). The electric drive can assist the engine since we have enough charge in the battery. Another objective of the controller is to minimize the engine power (to reduce fuel consumption). In region (III), there is no need to increase SoE so the electric drive can take care of propelling the vehicle. Moreover, the engine power is changing based on the magnitude of δE (as shown in Figure 5.b)

(IV) $E > E_{ref}$ and $SoE > SoE_{ref}$: In this case, we neither need propulsion power nor battery charging. So, there is no need to run the engine (for the sake of fuel consumption); as a result the engine power is zero throughout this region. However, we need to stop the vehicle so $P_{BRK} \neq 0$. On the other hand $P_{BAT} > 0$ to deplete the battery to return SoE closer to SoE_{ref}.

Figure 6 shows the continuous cost function over the set of polytopes which is piece-wise quadratic.

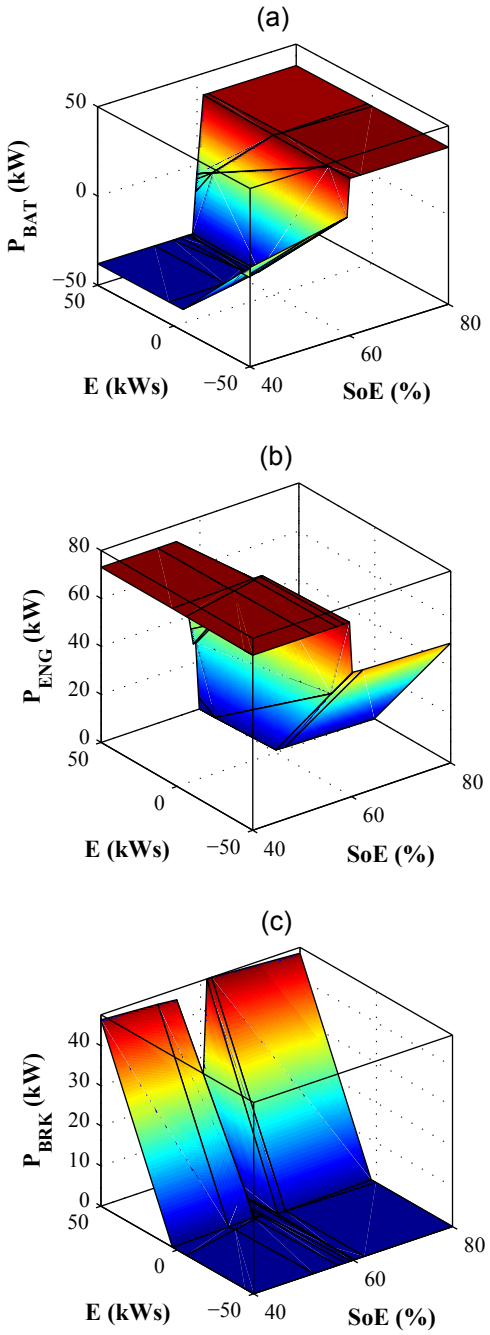

Figure 5: Control actions for $E_{ref} = 0$ and $SoE_{ref} = 60$ based on different initial conditions (a) battery power (b) engine power (c) braking power

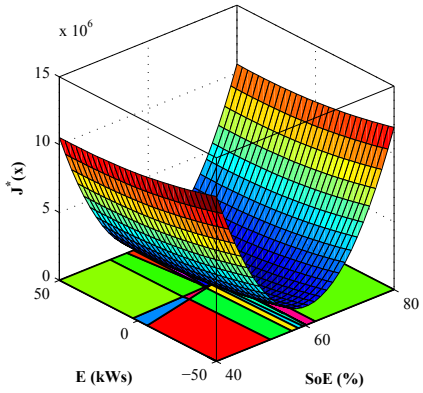

Figure 6: Cost function J^* for $E_{ref} = 0$ and $SoE_{ref} = 60$ based on different initial conditions

5. Stability notes

The main drawback of MPC is that it doesn't guarantee stability in general. MPC might drive the state variables to a part of state space where there is no solution to the finite time optimal control problem which can meet the constraints. As a result, the feasibility and stability of MPC must be investigated. The closed-loop system with MPC is globally asymptotically stable if and only if the optimization problem is feasible [38]. For the eMPC problem, feasibility of the solution is not adequate for proving stability, since we have a switched discrete-time system. So, we have to investigate the stability of the closed-loop system at 3 levels. First, the local stability of the closed-loop system around the equilibrium point in each of 81 sets of polytopes should be proven. Secondly, we have to prove the global stability of the mentioned controller throughout that specific set of polytopes. Finally, the stability of the closed-loop system must be investigated, while the controller switches between different sets of polytopes based on reference SoE and E. In each set of polytopes which belongs to a definite SoE_{ref} and E_{ref}, the controller drives the state variables to the mentioned reference values in finite time steps and $Z_0 = [SoE_{ref}, E_{ref}]^T$ is the equilibrium

point in each set. To prove the local stability of the closed-loop system, we pick the polytope which contains Z_0. The control corresponding to that polytope is:

$$\hat{U} = f_0 \hat{Z} + g_0 \tag{10}$$

By applying the above control to the control-oriented model we can find the closed-loop system equation as:

$$Z(k+1) = (\mathbf{A} + \mathbf{B}f_0)Z(k) + \mathbf{B}g_0 \tag{11}$$

By defining $\tilde{Z} = Z - Z_0$, we transfer the state variables to the equilibrium point. As a result we have:

$$\begin{aligned} \tilde{Z}(k+1) &= (\mathbf{A} + \mathbf{B}f_0)\tilde{Z}(k) \\ +\mathbf{B}g_0 + (\mathbf{A} + \mathbf{B}f_0 - I_{2\times 2})Z_0 &= \tilde{\mathbf{A}}\tilde{Z}(k) + \tilde{\mathbf{B}} \end{aligned} \tag{12}$$

Now, we can investigate the stability of (12) around $\tilde{Z} = 0$. First we show that $\tilde{\mathbf{A}}$ is locally and asymptotically stable for all 81 sets of polytopes. We have a discrete switching system and need to make sure that the spectral radius of $\tilde{\mathbf{A}}$ is less than unity, which is confirmed by the results in Figure 7.

We show that if $\tilde{\mathbf{A}}$ is stable and $\tilde{\mathbf{B}}$ is bounded, then closed-loop system (12) is stable. For a discrete system, if $V_1(\tilde{Z}_k) > 0$ exists and $\Delta V_1(\tilde{Z}_{k+1}, \tilde{Z}_k) = V_1(\tilde{Z}_{k+1}) - V_1(\tilde{Z}_k) < 0$ then the system is exponentially stable in the sense of Lyapunov. Since $\tilde{\mathbf{A}}$ is stable , we can find $P_1 > 0$ and $Q > 0$ such that:

$$\tilde{\mathbf{A}}^T P_1 \tilde{\mathbf{A}} - P_1 + Q = 0 \tag{13}$$

We assume that $V_1(\tilde{Z}_k) = \tilde{Z}_k^T P_1 \tilde{Z}_k$. As a result,

$$\begin{aligned} \Delta V_1(\tilde{Z}_{k+1}, \tilde{Z}_k) &= \tilde{Z}_{k+1}^T P_1 \tilde{Z}_{k+1} - \tilde{Z}_k^T P_1 \tilde{Z}_k \\ = \tilde{Z}_k^T (\tilde{\mathbf{A}}^T P_1 \tilde{\mathbf{A}} - P_1)\tilde{Z}_k &+ \tilde{\mathbf{B}}^T P_1 \tilde{Z}_{k+1} + \tilde{\mathbf{B}}^T P_1 \tilde{Z}_k \end{aligned}$$

If $Q = \mathbf{I}_{2\times 2}$ in (13) we can write:

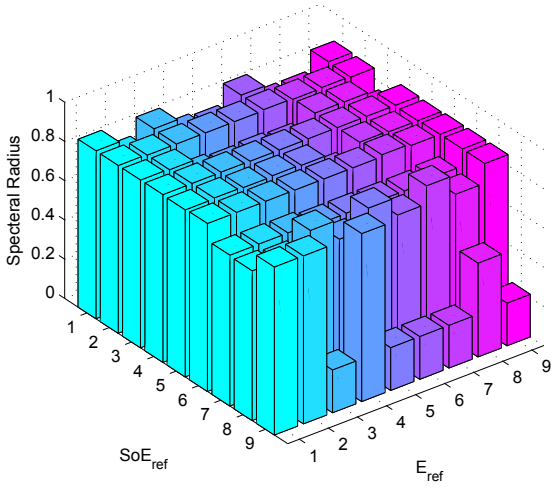

Figure 7: Spectral radius of $\tilde{A}$ for different levels of E_{ref} and SoE_{ref}

$$\Delta V_1(\tilde{Z}_{k+1}, \tilde{Z}_k) = -\tilde{Z}_k^T \tilde{Z}_k + \tilde{\mathbf{B}}^T P_1 \tilde{Z}_{k+1} + \tilde{\mathbf{B}}^T P_1 \tilde{Z}_k \tag{14}$$

Suppose that in (14) we take $P_1 = \mathbf{I}_{2\times 2}$ and $Q > 0$; then we can say $\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \leq I_{2\times 2}$ and $\|\tilde{Z}_k\|$ is monotonically convergent:

$$\|\tilde{Z}_{k+1}\| \leq \|\tilde{\mathbf{A}}\| \|\tilde{Z}_k\| \leq \|\tilde{Z}_k\| \tag{15}$$

Therefore:

$$\Delta V_1(\tilde{Z}_{k+1}, \tilde{Z}_k) \leq -\|\tilde{Z}_k\|^2 + 2\|\tilde{\mathbf{B}}\| \|P_1\| \|\tilde{Z}_k\| \tag{16}$$

If $\tilde{\mathbf{B}}$ is bounded, there is a $\beta > 0$ such that $\|\tilde{\mathbf{B}}\| < \beta \|\tilde{Z}_k\|^2$. As a result:

$$\Delta V(\tilde{Z}_{k+1}, \tilde{Z}_k) \leq \|\tilde{Z}_k\|^2 (1 - 2\beta \|P_1\| \|\tilde{Z}_k\|) \tag{17}$$

For $\|\tilde{Z}_k\| < (1/2\beta\|P_1\|)$,$\Delta V_1(\tilde{Z}_{k+1}, \tilde{Z}_k) < 0$, (12) would be stable. In this problem, $\beta = 10^{-8}$.

Now, we have to investigate the global stability of the closed-loop system for each set of polytopes.

Theorem. The equilibrium $x = 0$ is exponentially stable on sets of polytopes if there exist a function $\bar{V}(x)$ where $(\alpha, \gamma > 0)$:

$$\alpha\|x\|^2 < \bar{V}(x) < \gamma\|x\|^2 \tag{18}$$

with a negative forward difference $\Delta\bar{V}(x_{k+1}, x_k) = \bar{V}(x_{k+1}) - \bar{V}(x_k) < 0$ when $x_k \in T_j \backslash 0$ and $x_{k+1} \in T_i$ [39].

We introduce the following function as a positive definite candidate (since $\mathbf{Q} > 0$) for V:

$$\bar{V}(\tilde{Z}) = \sum_{j=1}^{N_p} \{\tilde{Z}^T(j)\mathbf{Q}\tilde{Z}(j)\} \tag{19}$$

which is a part of the cost function. We previously proved that $\|\tilde{Z}_{k+1}\| \leq \|\tilde{Z}_k\|$ so we can easily get $\Delta\bar{V}(\tilde{Z}_{k+1}, \tilde{Z}_k) < 0$. As a result, the closed-loop system is globally and exponentially stable.

Up to now, we investigated the stability of the closed-loop system of each set of polytopes. However, the controller switches between different sets of polytopes to cover all operating points. A switched system is stable if all individual subsystems are stable and the switching is sufficiently slow, so as to allow the transient effects to dissipate after each switch. In [40], this property is formulated and justified using multiple Lyapunov techniques. In this work, the switching frequency depends on the dynamics of (SoE_{ref}, E_{ref}). As mentioned before, (SoE_{ref}, E_{ref}) are bounded values. As a result, we assume that the following equations govern the dynamics of those reference values.

$$\begin{bmatrix} SoE_{ref}(k+1) \\ E_{ref}(k+1) \end{bmatrix} = \begin{bmatrix} 1/\gamma & 0 \\ 0 & 1/\gamma \end{bmatrix} \begin{bmatrix} SoE_{ref}(k) \\ E_{ref}(k) \end{bmatrix} + \Gamma \tag{20}$$

where γ should be chosen in such a way that guarantees (20) stability and also makes the switching system slower than the control-oriented model. For stability, γ should be greater than unity, so that the poles of (20) are located inside the unity circle in the z-plane. On the other hand, these poles should be far enough from the center of unity circle to slow down the system (20)

response. We assume that ρ is the largest spectral radius of $\tilde{A}$ for all 81 sets of polytopes. If we choose $1/\gamma > \rho$, the switching system will be slower than the control-oriented model. As a result, we choose γ in such a way that the poles of the switching system are located inside the dark ring of Figure 8 in order to make the switched system stable.

We can join the control-oriented model to the switching system:

$$X(k+1) = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{O}_{2\times 2} \\ \mathbf{O}_{2\times 2} & (1/\gamma)\mathbf{I}_{2\times 2} \end{bmatrix} X(k) + \begin{bmatrix} \mathbf{B} \\ \mathbf{O}_{2\times 3} \end{bmatrix} U(k) + \begin{bmatrix} \mathbf{O}_{2\times 1} \\ \Gamma \end{bmatrix} \tag{21}$$

where $1 < \gamma < 1/\rho$. For the closed-loop system, (21) can be transformed to:

$$\begin{bmatrix} \tilde{Z}_{k+1} \\ \tilde{Z}_{0,k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{O}_{2\times 2} \\ \mathbf{O}_{2\times 2} & (1/\gamma)\mathbf{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \tilde{Z}_k \\ \tilde{Z}_{0,k} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{B}} \\ \Gamma \end{bmatrix} \tag{22}$$

Since the spectral radius of $(1/\gamma)\mathbf{I}_{2\times 2}$ is less than unity and Γ is bounded ($\|\Gamma\| < 115$), according to the above discussion there is a $V_2(\tilde{Z}_{0,k}) = \tilde{Z}_{0,k}^T P_2 \tilde{Z}_{0,k} > 0$ that $\Delta V_2(\tilde{Z}_{0,k+1}, \tilde{Z}_{0,k}) =< 0$ where $P_2 > 0$.

For the whole system, we introduce a positive definite V $(P_1, P_2 > 0)$ such that:

$$\begin{aligned} V(\begin{bmatrix} \tilde{Z}_k \\ \tilde{Z}_{0,k} \end{bmatrix}) &= \begin{bmatrix} \tilde{Z}_k & \tilde{Z}_{0,k} \end{bmatrix} \begin{bmatrix} P_1 & \mathbf{O}_{2\times 2} \\ \mathbf{O}_{2\times 2} & P_2 \end{bmatrix} \begin{bmatrix} \tilde{Z}_k \\ \tilde{Z}_{0,k} \end{bmatrix} \\ &= \tilde{Z}_k^T P_1 \tilde{Z}_k + \tilde{Z}_{0,k}^T P_2 \tilde{Z}_{0,k} = V_1 + V_2 \end{aligned} \tag{23}$$

We proved that $\Delta V_1 < 0$ and $\Delta V_2 < 0$, so that $\Delta V = \Delta V_1 + \Delta V_2 < 0$. Now, we can say that closed-loop system (21) is stable.

6. HIL simulation

The electronic control unit (ECU) strategy prove-out is done in successive steps on (1) off-line simulations on a desktop, (2) HIL, (3) dynamometer, and (4)

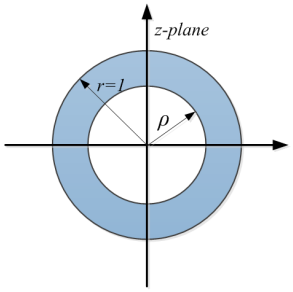

Figure 8: The locus of switching system poles in the z-plane

vehicle, with each step bringing in additional "real" substitutes for the virtual models. The ECU validation procedure in this sequence has some advantages. First, it ensures that component-level testing is done prior to subsystem and system level testing. Second, it capitalizes on the fact that ECUs are usually available much sooner than vehicle hardware prototypes, enabling a large amount of testing to be completed prior to vehicle manufacturing [24].

The off-line simulations used within the early phases of the development process are often called model-in-the-loop (MIL) simulations. For the modeling of the vehicle and functions at the MIL stage, standard tools such as Matlab/Simulink, MapleSim, and Dymola can be used. The next step is software-in-the-loop (SIL) simulation, where the functional model of an ECU is replaced by C-code and coding errors can be found independent of the future ECU hardware. In the next step, the actual hardware of the ECU is available and the tests can be supported by HIL simulation. The HIL simulation consists of two parts: open-loop integration and closed-loop strategy testing [41]. Then open-loop test platform uses a simpler model inside the real-time computer in order to check the functionality of user inputs and low-level I/O interfaces, for instance, push button start or gear shift command. The closed-loop test platform needs the dynamical model of the plant implemented on the real-time computer, which provides feedback information from the plant.

After the software tests are successfully passed, the calibration of the ECUs

can be done on the test-bench or in the vehicle. At this point changes to the functions and system specifications are time consuming, expensive, and in most cases not possible [42].

Since the performance of the ECU is tested in a virtual vehicle environment, the vehicle dynamics model need to be both accurate and real-time capable [42].

As the interactions between the physical and virtual components of a HIL simulator are bidirectional, it is crucial that the time frames of these components match exactly. Therefore, the virtual components must run in real time, placing tight requirements on the HIL microprocessor, operator system, and integration routine.

7. Hardware Description

An HIL simulation setup provides a more realistic environment then MIL simulation for controller evaluation purposes, as it takes into consideration different aspects of the control loop that are neglected in MIL simulations, such as communication issues and controller computational limitations. The two main components in an HIL setup are: 1) an independent processing unit to run the controller procedure, and 2) a powerful real-time processing unit to run the plant model. For our HIL simulation, the designed controller is programmed into an ECU, and the powertrain model is solved by a real-time target to provide the accurate sampling which the controller requires. The communication channel between the ECU and the plant (real-time target) is Control Area Network (CAN) bus. The following sections contain details of the hardware used in this research.

7.1. MotoTron ECU

The HIL simulation results are more reliable when the controller prototype is the same as the controller used in the real plant. For power management system applications, a MotoTron ECU with flash memory size of 2 MB is used to serve as the powertrain controller. This ECU is from the ECM-5554-112

family of controllers from Woodward that uses an 80MHz Motorola MPC5554 processor. The commercial version of this controller is used in automotive and marine applications. The automotive-based design of this ECU makes it an ideal choice for the HIL simulations. To program the controller code into the ECU, the code needs to be compiled by the MotoHawk Green Hills compiler. Then, the generated code can be programmed into the ECU by the MotoTune software. The controller code itself can be complied using Woodward's Green Hills compiler, which compiles the required code directly from a Simulink model.

7.2. PXI Real-time Target

A PXI platform from National Instrument (NI) is used as the real-time target. The processing unit of this computer is PXI-8110, which is powered by a 2.26 GHz quad-core CPU with 2GB of RAM. This PXI platform runs the LabVIEW Real-Time operating system, which responds to an interrupt or performs a task before a specified deadline, as opposed to non-real-time operating systems where tasks are prioritized based on different criteria such as maintaining the hardware/software functionality. Therefore, by making use of such real-time operating systems, the model can be solved with greater consistency, and communication delays can be minimized. To use this platform for solving the powertrain model in real-time, the model has to be converted into a C-code and then into a Digital Link Library (DLL) to be used in the LabVIEW environment.

Major responsibilities of the real-time target are shown in Figure 9. Each core of the real-time target CPU runs a different application. One core is responsible for running the PXI-host communication. This application is solely used to send and receive variables to and from the laptop host via Ethernet connection. The other CPU core runs the CAN communication application. The remaining cores are responsible for solving the powertrain model.

7.3. CAN Bus

The hybrid vehicle has several critical subsystems with individual control modules such as the engine, battery, transaxle and brakes. The controllers

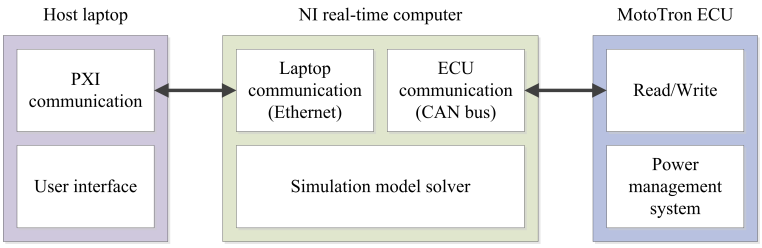

Figure 9: Communication schematic of the HIL setup

communicate with each other and with the vehicle system controller on a CAN-based communication network. The behaviors of these subsystems are strongly influenced by their individual controllers. Not all of these control modules were connected in the HIL setup. Those controllers that were not connected as hardware pieces were simulated as models along with the plant dynamics on the HIL system. So, the communication between controllers and the controller functionalities had to be modeled carefully to ensure a good compromise between functional accuracy and real-time constraints [24]. On a CAN bus, each of the nodes are directly connected to the bus, and there is no central control unit to regulate the communications. Instead, the CAN bus is a serial message-based protocol, where each node can send and receive messages when the bus is free. When two nodes start to send a message simultaneously, the message with higher priority prevails, and the lower-priority message waits until the bus is free. The priority of each message is identified by an arbitration ID, where lower IDs have the higher priority. The arbitration ID also serves as the name tag for each message. When a node transmits a message on the CAN bus, the message is received by every node on the bus. Each node can then ignore the message, or do a specific task based on the ID and the contents of the message. The other part of a CAN message is the data frame. A CAN data frame is defined byte-wise, i.e., the message consists of groups of bytes that contain an integer number. Thus, to send a variable, it should be scaled to an integer number, based on its range and required accuracy.

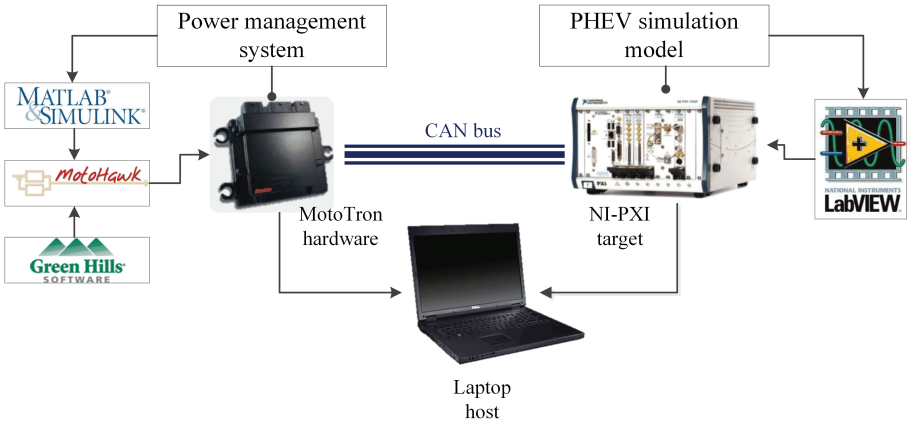

Figure 10: Schematic of the HIL setup

As shown in Figure 10, the simulation model is implemented to the PXI target via LabVIEW. The MotoTron ECU is communicating with the PXI target via CAN bus. The power management system that is designed in Matlab/Simulaink can be included in the MotoTune toolbox that is provided by MotoHawk software and is compiled by the MotoHawk Green Hills compiler. We can monitor the HIL test procedure in the laptop host.

The power management system requires two readings from the plant: the current battery state of charge and the current demanded tractive energy. The two measurements are calculated by the real-time target by solving the powertrain model. Also, it takes two readings from SoE_{ref} and E_{ref}. The real-time target then sends these four pieces of information in a single CAN message to the ECU. The controller processes the information and calculates P_{BAT}, P_{ENG}, and P_{BRK} and sends them back to the real-time target in another message.

Table 2 shows the variables, and the position of the variable in the CAN messages for ECU-PXI communication.

In the base CAN frame format (CAN 2.0 A protocol), the identifier portion of the message (arbitration ID) contains 11 bits following the start bit. The main data frame can contain up to 8 bytes (64 bits). Combined with all other

Table 2: CAN message definition for the HIL simulation

CAN message				MotoHawk		LabVIEW	
message name	arbitration ID	message length	variable name	start bit	bit length	start bit	bit length
PXI to ECU	1	7 bytes	E_{ref}	48	8	8	8
			SoE_{ref}	40	8	16	8
			E	24	16	24	16
			SoE	8	16	40	16
ECU to PXI	2	7 bytes	P_{BAT}	40	16	8	16
			P_{ENG}	24	16	24	16
			P_{BRK}	8	16	40	16

regulatory bits, a CAN message is comprised of up to 108 bits. Depending on the bit-rate of the CAN channel, a limited number of messages can be sent on a CAN bus. In this HIL setup, the CAN channels work with a bit rate of 500 kbps (kilo-bits per second); therefore, the maximum capacity of each CAN channel is roughly 4600 messages per second. The communication program on the real-time target runs at every 1ms and sends a message (PXI to ECU) in each run of the loop. The controller program also runs every 5ms and sends one message (ECU to PXI). Thus, 1200 messages are sent in each second, and this load occupies 26% of the CAN channel capacity.

8. Controls implementations

After finding the polytopes and the corresponding control actions, we need to implement the controller on the simulation model by using low-level controls. Basically, we have to change the provided power to torque and speed for different components. Figure 11 shows the procedure that is done at each control time step. At the beginning, we have E_{ref} and SoE_{ref} as well as initial SoE and demanded energy that are given to the eMPC controller. By using point location algorithm, we can find the appropriate controls among the polytopes. We are

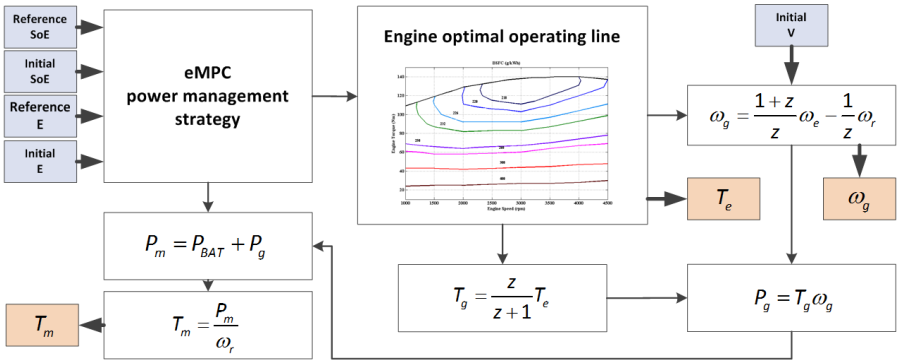

Figure 11: Low-level controls implementation

looking for T_e, T_m, and ω_g. On the propulsion side, we have P_{BAT} and P_{ENG}. Once we got P_{ENG}, we can use the optimal operating line of the engine, which gives us the most efficient operating point for the given P_{ENG}. Now, we have the engine speed and torque for the optimum operating point. By having T_e we can control the engine throttle to the desired engine torque [43]. The engine torque setpoint can directly be given to the engine low-level controller.

If we measure the vehicle velocity, we will be able to get the MG1 speed setpoint by using the speed constraint relation on the first planetary gear set ($z = \frac{s_1}{r_1}$ where s_1 and r_1 are the number of sun and ring gear teeth). Meanwhile, if we use a static torque relation on the planetary gear set, we can find the MG1 torque based on the engine torque. Now MG1 power is calculated and we can find the MG2 power, since we have the P_{BAT} from eMPC controller. By measuring the MG2 speed at the current time step, we are able to find the last setpoint value, which is MG2 torque. Now, we can implement the controller to the simulation model.

9. Controls Implementation Notes for HIL

For implementing the eMPC power management system onto the ECU, a database with the size of 1.5 MB plus the eMPC search algorithm should be

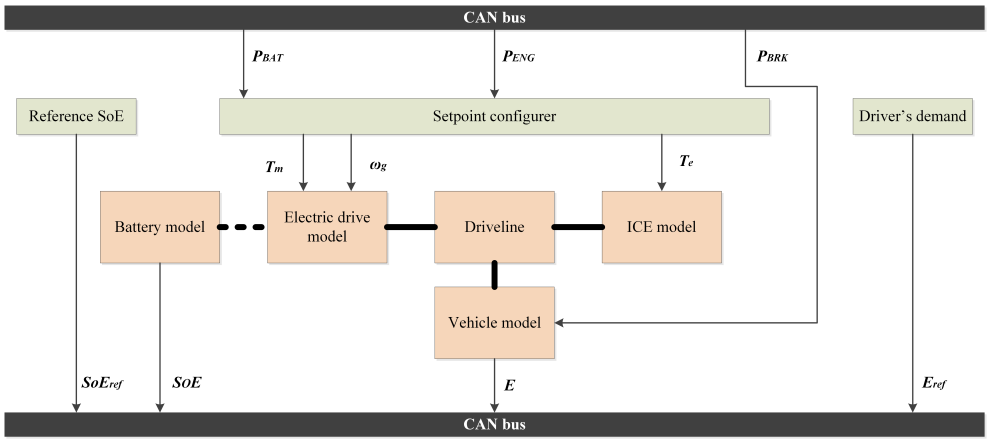

Figure 12: Powertrain model inside the real-time computer

stored in the hardware memory. The search algorithm code is not in-lined, and cannot be compiled to the MotoTron ECU. Unfortunately, by in-lining the algorithm code, the size of code plus eMPC database exceeds the 2 MB flash memory size of the ECU.

To solve this problem, the eMPC power management was modified. The resulting control action surfaces (Figure 5) versus state variables were approximated with some new surfaces. Using this technique, we reduced the size of the controller from 2 MB to 121 kB for the eMPC power management system.

Figure 12 shows different parts of the simulation model inside the real-time computer.

The ECU passes 3 control actions: P_{BAT}, P_{ENG}, and P_{BRK} to the real-time target via CAN bus at every 5 milliseconds. The real-time target runs the simulation model and corresponding low-level controls at every 1 ms. As shown in Figure 12, P_{BAT} and P_{ENG} are fed into the setpoint configurer where the setpoints for the low level controllers are determined. Then, T_m, ω_g, and T_e are transferred to the electric drive and engine low-level controls. By applying the low-level controls commands to each component of the powertrain model, SoC, E, SoE_{ref}, and E_{ref} are measured at each 1 ms and passed to the ECU via

CAN bus.

10. Model-in-the-loop Simulation Results

In this section, two different strategies to deplete the battery along two UDDS drive cycles are used.

10.1. No Knowledge of Trip Information

The first strategy is charge depletion charge sustenance (CDCS). In CDCS, the vehicle is primarily operated in charge depletion mode (CD) by utilizing battery electrical energy. When the SoC reaches a predefined level, the operation mode is switched into the charge sustenance (CS) mode to maintain SoC close to that predefined level. Figure 13.a shows the drivability performance as well as the battery depletion curve. In Figure 13.b the propulsion power and the demanded power at each time step are compared and we can see that eMPC is able to maintain the vehicle drivability. Figure 13.c shows the propulsion power (P_{BAT} and P_{ENG}). As shown, the engine is off for the first part of the trip and takes over when the battery SoC drops to the predefined level. Fuel economy in this strategy is equal to 119 mile per gallon (MPG).

10.2. Known Travelling Distance

For the second strategy, we assume that we have knowledge of travelling distance to the next charging station. If travelling distance was less than the vehicle all electric range, the best strategy would be going in pure electric mode. Otherwise, we follow another strategy; we assume the battery SoC linearly decreases with the distance traveled by the vehicle (linear blended mode [44]). In Figure 14.a, the SoC follows the linear profile and the engine operation is distributed along the entire drive cycle (Figure 14.b), which results in 133 MPG. Therefore, known travelling distance will improve fuel economy by 11.76 %. For demonstrating the performance of the controller along the drive cycle, zoomed view of some part of the plots were added to Figures 13 and 14.

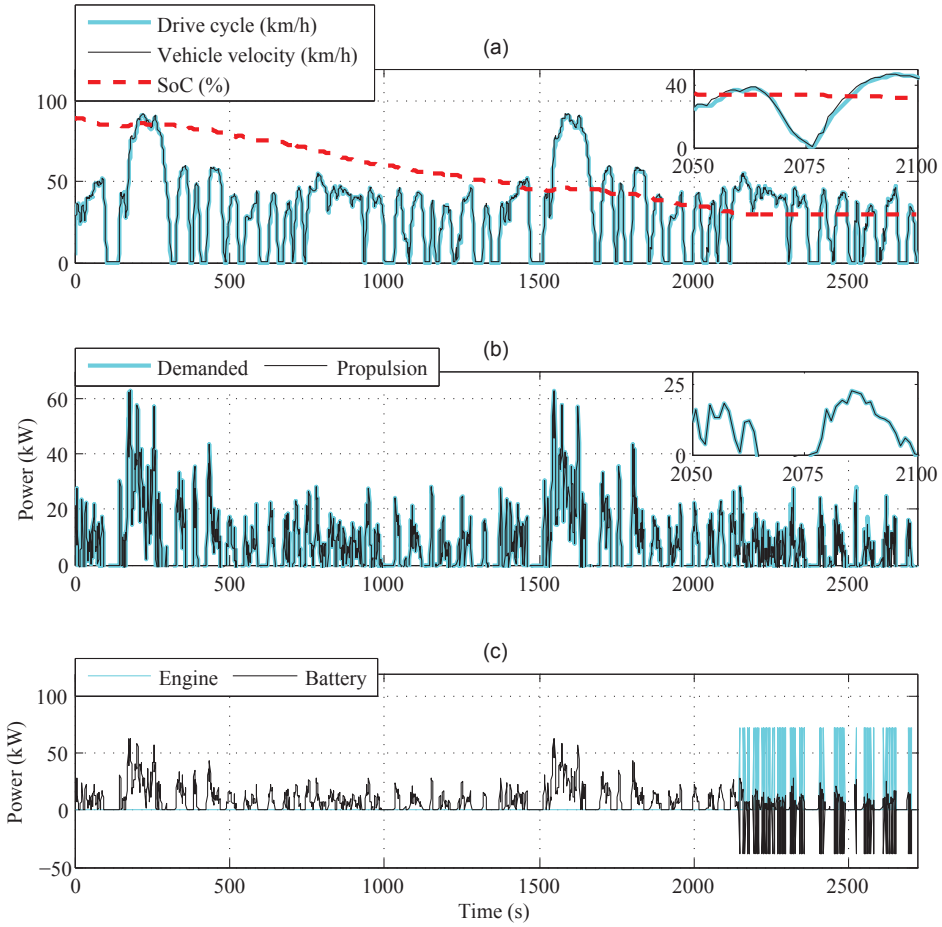

Figure 13: Charge depletion/ charge sustenance strategy in MIL (a) vehicle velocity and battery depletion profile (b) demanded and propulsion power (c) engine and battery power

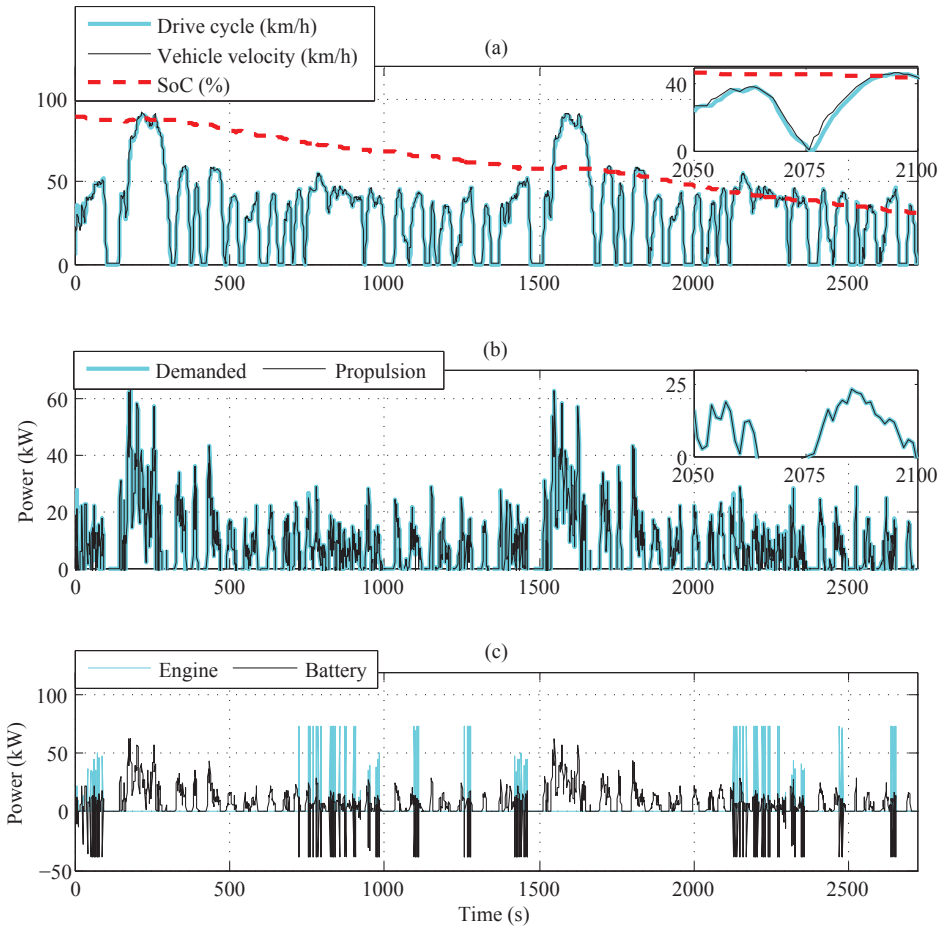

Figure 14: Linear blended strategy in MIL (a) vehicle velocity and battery depletion profile (b) demanded and propulsion power (c) engine and battery power

10.3. Discussion

Fuel consumption for each strategy is listed in Table 3. Compared to MPC [17], the newly-designed eMPC performs better in terms of fuel economy as well as real-time implementation capability. Table 3 shows how much improvement we made by changing the control-oriented model and the cost function. Previously, we designed a non-linear model predictive controller using the plant model as the control-oriented model. In this paper, we developed a simpler control-oriented model, which led to better real-time performance of the controller, i.e the new control-oriented model results in a faster controller with better performance. In the control implementation procedure we directly used the engine optimal operating line, instead of estimating the fuel consumption, and considered it inside the cost function. Instead of minimizing the fuel consumption rate, we showed that minimizing the engine power led to a better fuel efficiency performance for the controller. In this way, we kept the cost function convex and also removed the requirement of estimating the fuel consumption rate, which can introduce errors in the solution. Indeed, the optimal operating line of engine might be alternating during the engine operation due to the engine temperature. In such cases, we can develop an adaptive version of the present controller so that some parameters get updated during the engine operation. In our experience, the performance of the model predictive controller is closely related to the control-oriented model, more than any other factor. Computationally, it took 17.53 and 22.36 s in real time respectively, for CDCS and blended, for 2828 seconds of simulation (for two successive UDDS drive cycles) to be completed. The simulation is conducted on a machine which is powered by a 3.16 GHz dual core CPU and a 4 GB memory. Therefore, the eMPC strategy is 44% faster than MPC on average.

11. Hardware-in-the-loop Simulation Results

In Figure 15.a, the vehicle drivability performance and battery state of charge for CDCS strategy are demonstrated. In Figure 15.b, we can see that the

Table 3: MIL with low-fidelity powertrain model: Fuel economy for different control strategies

Control Strategy	MPC (MPG)	eMPC (MPG)	improvement (%)
Charge Depletion/Charge Sustenance	105	119	13.33
Linear blended mode	112	133	18.75

Table 4: eMPC MIL and HIL test: Fuel economy for different control strategies

Control Strategy	MIL (MPG)	HIL (MPG)
Charge Depletion/Charge Sustenance	119	116
Linear blended mode	133	127

driver demanded power is followed by the propulsion power. This shows that the powertrain is able to provide the required propulsion power, so the vehicle velocity can follow the predefined UDDS schedule. Figure 15.c shows the index of demanded power as well as SoE index. Figure 16 shows these results for the blended mode strategy. Note that the engine operation has reduced the battery SoC depletion slope which results in better fuel economy as compared to CDCS strategy. For demonstrating the performance of the controller along the drive cycle, zoomed view of some part of the plots were added to Figures 15 and 16.

Table 4 shows the MIL and HIL fuel economy for the eMPC power management system.

Note that if we use the same controller and simulation model for MIL and HIL, the simulation results should be the same. The oscillations of the vehicle velocity shown in Figure 13 and Figure 14, as compared to Figure 15 and Figure 16, is due to switching between different polytope sets along the drive cycle. Fuel economy for CDCS and blended mode strategies in HIL testing are worsened by

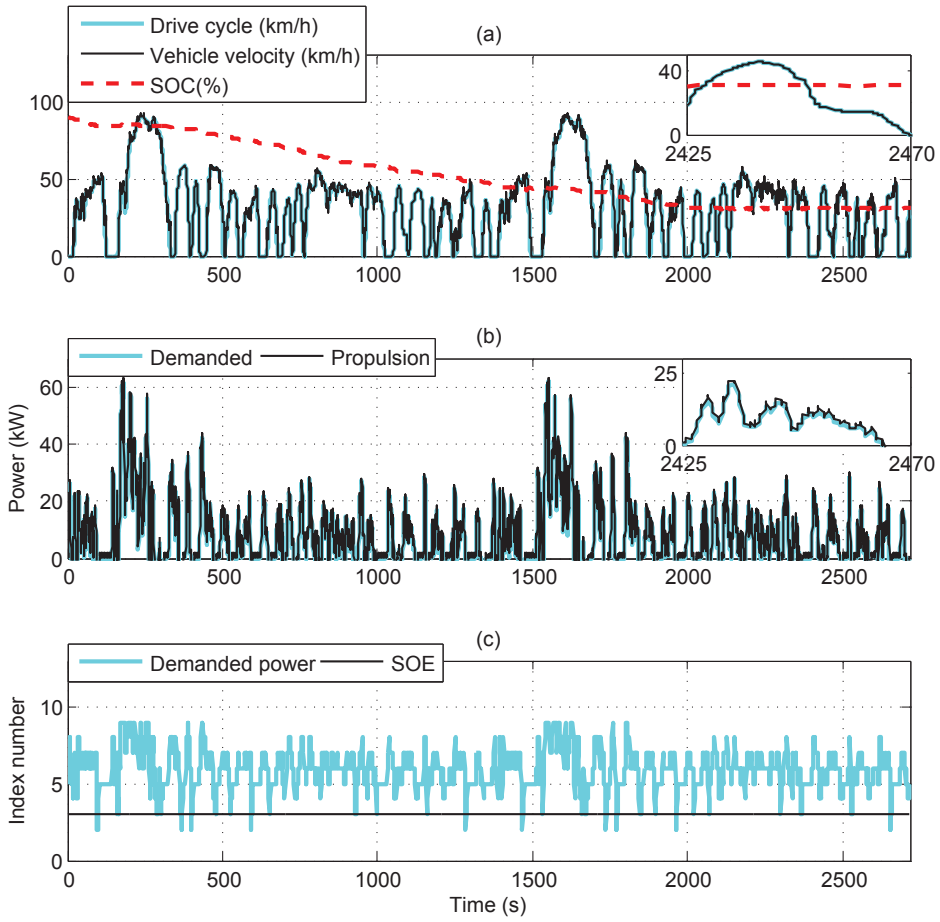

Figure 15: Charge depletion/ charge sustenance strategy in HIL (a) vehicle velocity and battery depletion profile (b) demanded and propulsion power (c) Demanded power and SoE indices

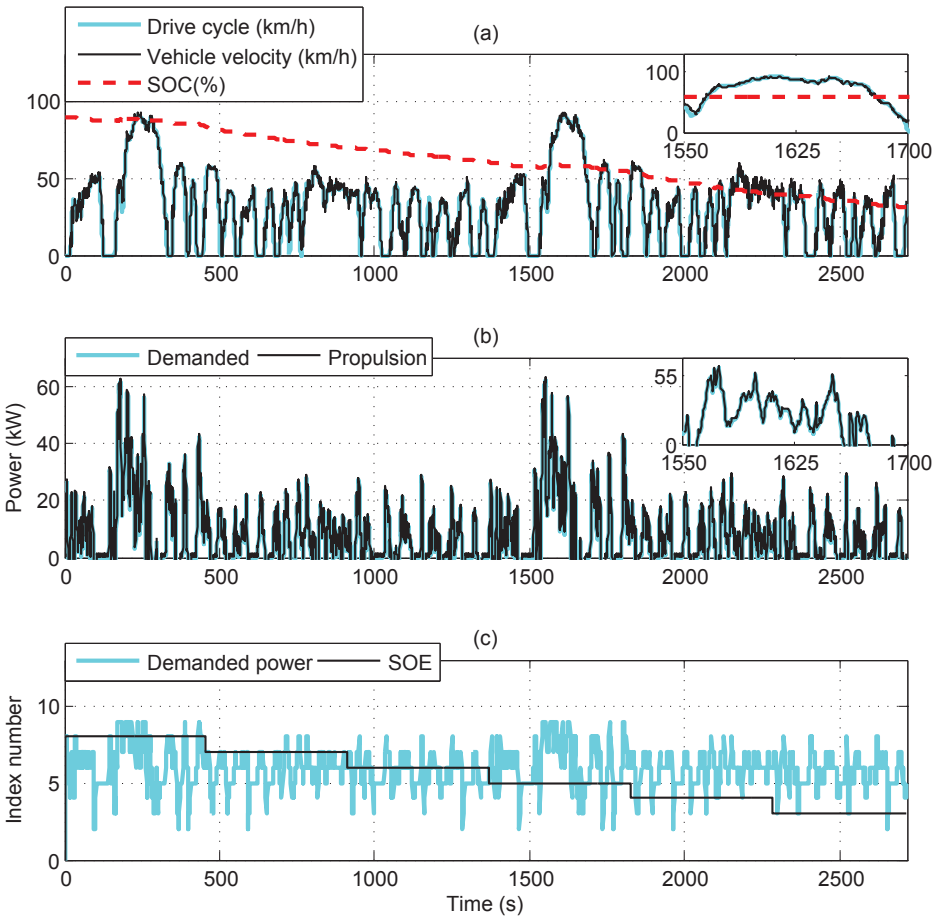

Figure 16: Blended mode strategy in HIL (a) vehicle velocity and battery depletion profile (b) demanded and propulsion power (c) Demanded power and SoE indices

2.5% and 4.5%, as compared to MIL simulation. This error is due to replacing the eMPC data base and its search algorithm with the approximated look-up tables. In brief, the difference between MIL and HIL simulation results is due to the difference between the original and modified CRPE-eMPC power management systems.

12. Conclusions

In this paper, we used the explicit model predictive control approach to design a power management strategy for a plug-in hybrid powertrain. We solved a multi-parametric problem to improve real-time implementation performance over a conventional model predictive control. We developed a new control-oriented model which contains 4 parameters. We implemented the developed controller to a PHEV simulation model and obtained a promising fuel economy as well as real-time implementation performance. We reduced the simulation time by 44% and improved fuel economy by 16% on average, in comparison to MPC. Moreover, the designed power management system performance was validated through hardware-in-the-loop testing. To implement the power management system to the control hardware with limited memory size and computational capability, some modifications were applied to the original control scheme. HIL simulations show that the proposed power management system can be implemented to a commercial hardware in real time. It is noteworthy that we will pursue full vehicle validation once our Green and Intelligent Automotive HEV facility is completed at the University of Waterloo. Then, we will be able to by-pass the current powertrain ECU of our Toyota plug-in Prius and implement the proposed power management strategy along with a calibration procedure.

13. Acknowledgements

The authors gratefully acknowledge the NSERC/Toyota/Maplesoft Industrial Research Chair program for financial support of this research.

14. References

[1] B. Saerens, M. Diehl, J. Swevers, and E. Van den Bulck, "Model predictive control of automotive powertrains - first experimental results," in *47th IEEE Conference on Decision and Control*, 2008, pp.5692–5697.

[2] P. Ortner, P. Langthaler, J. V. G. Ortiz, and L. del Re, "MPC for a diesel engine air path using an explicit approach for constraint systems," in *Proc. IEEE International Conference on Control Applications,Munich, Germany*, 2006, pp.2760-2765.

[3] H. Hur, T. Nagata, and M. Tomizuka, "Model-based optimal gear shift pattern scheduling and smooth gear shifting control," *Steuerung und Regelung von Fahrzeugen und Motoren - AUTOREG 2006, VDIBerichte Nr. 1931, 2006. VDI Wissensforum*, 2006, pp.303-312.

[4] S. Di Cairano, W. Liang, I.V. Kolmanovsky, M.L. Kuang, and A.M. Phillips, "Power smoothing energy management strategy for a series hybrid electric vehicle," *IEEE Trans. on Control Systems Technology*, vol. 21, no. 6, pp 2101-2106 , 2013.

[5] S.J. Moura SJ, H.K. Fathy, D.S. Callaway and J.L. Stein , "A Stochastic Optimal Control Approach for Power Management in Plug-in Hybrid Electric Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp 545-555, 2011.

[6] C. Musardo, G. Rizzoni, and B. Staccia, "A-ECMS: An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management," *European Journal of Control*, vol. 11, no. 4, pp. 509524, 2005.

[7] P. Tulpule, V. Marano, and G. Rizzoni, "Effects of different PHEV control strategies on vehicle performance," In *American Control Conference*, pp. 39503955, 2009.

[8] A. Alessio and A. Bemporad, *A Survey on Explicit Model Predictive Control*: Lecture Notes in Control and Information Sciences: Springer, vol. 384, pp 345-369, 2009.

[9] S. Di Cairano, D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "An MPC design ow for automotive control and applications to idle speed regulation," in *Proc. 47th IEEE Conf. on Dec. and Control*, pp.56865691, 2008.

[10] P. Falcone, F. Borrelli, J. Asgari, H. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Contr. Systems Technology*, vol. 15, no. 3, pp 566580, 2007.

[11] S. Di Cairano and H. Tseng, "Driver-assist steering by active front steering and differential braking: Design, implementation and experimental evaluation of a switched model predictive control approach," in *Proc. 49th IEEE Conf. on Dec. and Control*, pp.28862891, 2010.

[12] P. Ortner, P. and L. Del Re, "Predictive Control of a Diesel Engine Air Path," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp 449-456, 2007.

[13] S. Trimboli , S. Di Cairano , A. Bemporad and I. Kolmanovsky, (2009) "Model predictive control for automotive time-delay processes: An application to air-to-fuel ratio," in *Proc. 8th IFAC Workshop Time-Delay Syst.*, pp 1-6, 2009.

[14] R. Amari , M. Alamir and P. Tona, "Unified MPC strategy for idle-speed control, vehicle start-up and gearing applied to an automated manual transmission," in *Proc. 17th IFAC World Congr.*, pp 7079 -7085, 2008.

[15] G. Ripaccioli, A. Bemporad, F. Assadian, C. Dextreit, S. Di Cairano, and I. Kolmanovsky, *Hybrid Modeling, Identication, and Predictive Control: An Application to Hybrid Electric Vehicle Energy Management*: Hy-

brid Systems: Computation and Control, ser. Lec. Not. in Computer Science: Springer, vol. 5469, pp 321335, 2009.

[16] H. Borhan, A. Vahidi, A.M. Phillips, and I.V. Kolmanovsky, "Predictive Energy Management of a Power-Split Hybrid Electric Vehicle'" in *American Control Conference* , pp. 39703976, 2009.

[17] A. Taghavipour, M. Vajedi, N.L. Azad, and J. McPhee, "Predictive Power Management Strategy for a PHEV Based on Different Levels of Trip Information," in *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling*, vol. 3, no. 1, pp 326-333, 2012.

[18] A. Taghavipour, N.L. Azad, and J. McPhee, "An optimal power management strategy for power split plugin hybrid electric vehicles," *International Journal of Vehicle Design*, vol. 60, no. 3, pp 286-304, 2012

[19] K.I. Kouramasa, C. Panosa, N.P. Faiscab, and E.N. Pistikopoulos, "An algorithm for robust explicit/multi-parametric model predictive control," *Automatica*, vol. 49, no. 2, pp 381389, 2013.

[20] M. Vajedi, A. Taghavipour, N.L. Azad, and J. McPhee, "A comparative analysis of route-based power management strategies for real-time application in plug-in hybrid electric vehicles," In *American Control Conference 2014*.

[21] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp 3-20, 2002.

[22] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat, "A hybrid approach to traction control," *Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001*, vol. 2034, no. 5, pp 162-174, 2001.

[23] R. Conti, E. Meli, A. Ridolfi, and A. Rindi, "An innovative hardware in the loop architecture for the analysis of railway braking under degraded

adhesion conditions through roller-rigs," *Mechatronics*, vol. 24, no. 2, pp 139–150, 2014.

[24] D. Ramaswamy, R. McGee, S. Sivashankar, A. Deshpande, and et al., "A Case Study in Hardware-In-the-Loop Testing: Development of an ECU for a Hybrid Electric Vehicle," *SAE Technical Paper 2004-01-0303*, 2004.

[25] R. Trigui, B. Jeanneret, B. Malaquin, F. Badin, and C. Plasse, "Hardware In the Loop Simulation of a Diesel Parallel Mild-Hybrid Electric Vehicle," *IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2007.

[26] M. Petersheim, S. Brennan, "Scaling of hybrid-electric vehicle powertrain components for Hardware-in-the-loop simulation," *Mechatronics*, vol. 19, no. 7, pp 1078-1090, 2009.

[27] H.K. Fathy and Z.S. Filipi and J. Hagena and J.L. Stein, "Review of hardware-in-the-loop simulation and its prospects in the automotive area," *Proc. SPIE 6228, Modeling and Simulation for Military Applications*, 2006.

[28] W. Lee, S. Park, and M. Sunwoo, "Towards a Seamless Development Process for Automotive Engine-Control System," *Control Engineering Practice*, vol. 12, no. 1, pp 977–986, 2004.

[29] K. Muta, M. Yamazaki, and J. Tokieda, "Development of new-generation hybrid system ths ii drastic improvement of power performance and fuel economy", *SAE technical paper 2004-01-0064*, 2004.

[30] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, "Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, vol. 38, no. 1, pp. 3-20, 2002.

[31] F. Borrelli, *Constrained Optimal Control Of Linear And Hybrid Systems*: Lecture Notes in Control and Information Sciences: Springer, vol. 290, 2003.

[32] P. Tondel, T. A. Johansen, and A. Bemporad, "An algorithm for multiparameteric quadratic programming and explicit MPC solution," *Automatica*, vol. 39, no. 5, pp 489-497, 2003.

[33] K. FUKUDA, "Polyhedral computation FAQ," *http://www.ifor.math.ethz.ch/staff/fukuda.*, 2000.

[34] S.W. Cheng and A. Janadan 1992) "Algotithms for Ray-Shooting and Intersection Searching," *Journal of Algorithms*, vol. 13, no. 3, pp 670-692, 1992.

[35] M. Kvasnica, "Efficient Software Tools for Control and Analysis of Hybrid Systems," *PhD thesis ETH Zurich*, 2008.

[36] M. De Berg, O. Schwarzkopf, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*: Springer, 2nd edition, 2000.

[37] M. Kvasnica, P. Grieder, and M. Baotic, " Multi-Parametric Toolbox (MPT)," *Available from http://control.ee.ethz.ch/ mpt/*, 2004.

[38] A. Zheng, and M. Morari, "Stability of Model Predictive Control with Soft Constraints," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp 1818-1823, 1995.

[39] G. Ferrari-Trecate, F.A. Cuzzola, D. Mignone, and M. Morari, "Analysis of Discrete-time Piecewise Affine and Hybrid Systems ," *Automatica*, vol. 38, no. 12, pp 21392146, 2002.

[40] D. Liberzon, *Switching in Systems and Control*: Birkhauser Boston, 2003.

[41] R. McGee, "Ford Motor Company Hybrid Electric Escape Powertrain Control System Development and Verification Utilizing Hardware-in-the-Loop Technology," *dSPACE User Conference*, 2002.

[42] D. Winkler, and C. Ghmann, "Hardware-in-the-Loop simulation of a hybrid electric vehicle using Modelica/Dymola," *Proceedings of the 22nd International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium (EVS-22), Yokohama, Japan*, 2006.

[43] A. Taghavipour, N.L. Azad, and J. McPhee, "Design and Evaluation of a Predictive Powertrain Control System for a Plug-in Hybrid Electric Vehicle to Improve Fuel Economy and Emissions," *IMechE, Part D: Journal of Automobile Engineering*, Accepted for publication, 2014.

[44] C. Zhang, A. Vahidi, X. Li, and D. Essenmacher, "Role of trip information preview in fuel economy of plugin hybrid vehicles," *ASME Dynamic Systems and Control Conference*, pp. 253-258, 2009.