Integrated Optimization of Powertrain Energy Management and Vehicle Motion Control for Autonomous Hybrid Electric Vehicles

Mohammadali Kargar, Chen Zhang and Xingyong Song

***Abstract*—Hybrid Electric Vehicles (HEVs) and autonomous vehicles have been widely studied recently for on-road transportation. In the study of autonomous HEVs, the control of vehicle's external dynamics and powertrain dynamics are often treated separately. Optimizing these two problems together can significantly improve fuel economy. In this paper, a customized control strategy based on Approximate Dynamic Programming (ADP) is explored to optimize these dynamics together. At last, a case study shows that the examined control strategy outperforms the one with the separated optimization method by an additional 15% improvement in fuel consumption.**

***Index Terms*—Autonomous vehicles; Hybrid electric vehicles; Energy Management.**

I. Introduction

More than 65 percent of U.S. transportation oil consumption is for personal vehicles [1]. This figure reveals the importance of applying new technologies to improve fuel efficiency in conventional vehicles. Hybridization and autonomy are the top two technological trends to achieve enhanced fuel consumption in recent years.

An HEV is a vehicle powered by both an Internal Combustion Engine (ICE) and a battery pack through electric motors. By having an extra degree of freedom in the vehicle's powertrain system enabled by the alternative power source (battery), HEVs can significantly reduce emissions and improve fuel economy. Since multiple power sources exist in an HEV, a question arises intuitively that is how to optimally manage the power split between the power sources which is referred to as powertrain energy management [2].

Meanwhile, research on autonomous vehicles has gained momentum recently. An autonomous HEV combines autonomy and powertrain hybridization together and is considered to further enhance the fuel economy through this synergy. However, most existing research studies autonomy and power hybridization separately [3, 4]. Studies have shown augmenting these two optimization problems can offer a significant fuel-saving potential that cannot be achieved by powertrain optimization alone [5]. Recently, a few works have been published integrating the external vehicle dynamics and powertrain dynamics together [5]. However, the proposed methodology (forward DP) depends on the system's initial condition, and also is computationally intensive.

This study explores a sub-optimal powertrain energy management strategy to optimize both the powertrain dynamics and the external vehicle dynamics in an integrated fashion under the Approximate Dynamic Programming (ADP) framework for the first time. Solving this problem using ADP is independent of the initial condition of the system and requires much less memory storage capacity compared to the conventional DP. First, a customization to the ADP method is proposed to enable its application to a non-quadratic cost function with non-affine dynamics and nonlinear constraints on control inputs. Second, the concept of the reachable set [6] is adopted when implementing ADP. To the best of the authors' knowledge, this concept has not been implemented in any ADP method before.

The outline of this paper is as follows. In Section II, the HEV modeling is discussed. Section III presents the integrated optimization problem. Reachable sets and ADP framework are introduced in Section IV, followed by a numerical example in Section V. At last, concluding remarks are given in Section VI.

II. HEV Modeling

A. Powertrain Modeling

In this study, a power-split hybrid powertrain is considered, as shown in Fig. 1. This mechanism comprises an ICE, a battery pack, a planetary gear set, a coupler gear set, an inverter, and two electric machines. The planetary gear set is used as the power-split device consisting of three main elements: the sun, the carrier, and the ring. Neglecting the inertia of the moving parts in the powertrain, the power balance at the inverter yields:

$$P_{batt}(t) = \mu_m^{k_m} T_m(t)\omega_m(t) + \mu_g^{k_g} T_g(t)\omega_g(t) \qquad (1)$$

where P_{batt}, ω_g, and ω_m denote the power of the battery, and the angular velocity of the generator and the motor, respectively. Likewise, T_g, and T_m represent the torque of the generator and the motor, respectively. Also, μ_g and μ_m depict the efficiency coefficients of the generator and the motor when they are acting as generators, respectively. Thus, k_m & k_g are equal to 1 if their corresponding electrical machines operate as generators and -1 otherwise.

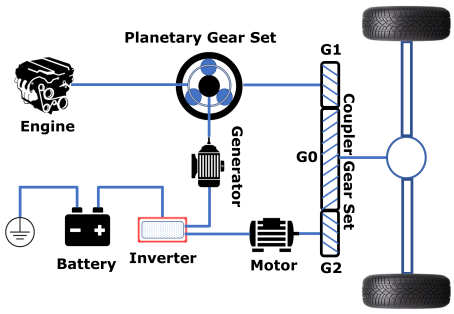

Fig. 1: Powertrain Schematic.

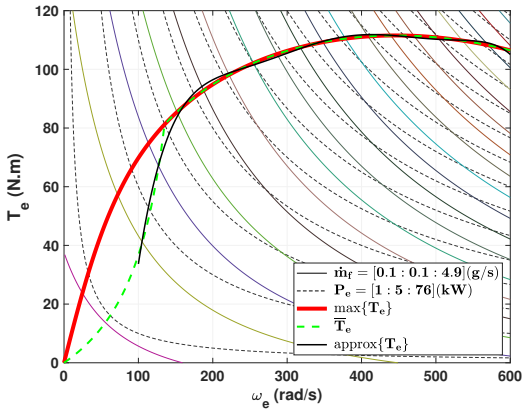

Fig. 2: Engine Map [5].

B. Battery Modeling

The battery is modeled as an equivalent circuit model given below [5]:

$$\dot{SoC}(t) = -\frac{V_{batt} - \sqrt{V_{batt}^2 - 4R_{batt}P_{batt}(t)}}{2R_{batt}Q_{batt}}, \qquad SoC(0) = SoC_0 \tag{2}$$

where SoC is the state of the charge of the battery. Also, V_{batt}, R_{batt}, and Q_{batt} represent the battery's open-circuit voltage, internal resistance, and capacitance, respectively.

C. Engine Modeling

An engine map (Fig. 2) generated from the experimental data is used in this study to model the engine and to calculate the fuel consumption $\dot{m}_f$. The fuel consumption is generally calculated by having engine's angular velocity, ω_e, and engine's torque, T_e, as below:

$$\dot{m}_{fuel}(t) = \gamma(\omega_e(t), T_e(t)) \tag{3}$$

where $\gamma : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$. $\mathbb{R}_+$ is the set of nonnegative real numbers. Given the two degrees of freedom feature in the power-split HEV's powertrain, one can assume that the solution pair $(T_e(t), \omega_e(t))$ for any engine power $P_e(t) \triangleq T_e(t)\omega_e(t)$ will be such that the engine power lies on its most efficient point on the engine map[5]. Therefore, to reduce the dimension of the input space, $T_e(t)$ can be written as a function of $\omega_e(t)$:

$$T_e(t) \propto \omega_e(t) \to T_e(t) = h_1(\omega_e(t)) \tag{4}$$

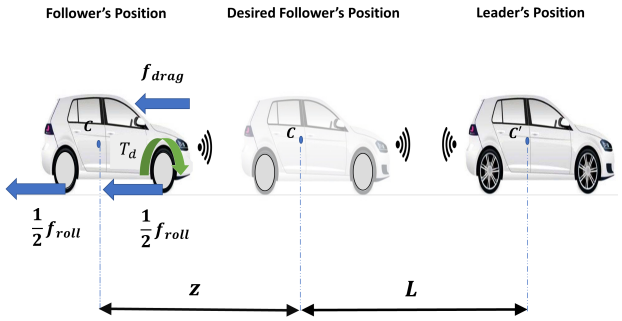

Fig. 3: Free Body Diagram of the Follower and the Interaction Topology.

where $h_1 : \mathbb{R}_+ \to \mathbb{R}_+$.

D. Vehicle Modeling

Fig. 3 shows a vehicle following a leader in a straight path in an inertial frame. Let x be the horizontal position of the center of mass of the follower C, and x_{leader} be the horizontal position of the center of mass of the leader C'. Also, let $x_{desired}$ be the desired position of the follower with respect to the leader defined as below:

$$x_{desired}(t) = x_{leader}(t) - L \tag{5}$$

where L is the desired constant distance between $C\&C'$. The follower's longitudinal position error z is defined as:

$$z(t) = x(t) - x_{desired}(t) \tag{6}$$

The kinematics of the follower can be written as below:

$$\dot{x}(t) = v(t), \qquad x(0) = x_0 \tag{7}$$

$$\dot{z}(t) = v(t) - v_{leader}(t), \qquad z(0) = z_0 \tag{8}$$

where v_{leader} is the velocity of the leader. In addition, the external dynamics of the follower is as follows:

$$\dot{v}(t) = \frac{1}{m}\left[-f_{drag} - f_{friction} + \frac{1}{r}T_d(t)\right]$$
$$f_{drag}(t) = \frac{1}{2}\rho C_{drag} A_f v(t)^2$$
$$f_{roll} = \mu_{roll} mg \tag{9}$$

where m, A_f, f_{drag}, f_{roll}, and μ_{roll} are the follower's mass, frontal area, air drag force, rolling resistance force, and rolling resistance coefficient, respectively. Besides, ρ and C_{drag} denote the air density and the coefficient of drag, respectively.

III. Problem Formulation

Integrated optimization requires the state vector to augment both external vehicle level and powertrain level dynamics. To keep track of the follower's position error and velocity and the SoC of the battery, the following state vector is defined:

$$\mathbf{X}(t) \triangleq [z(t), v(t), SoC(t)]^T. \tag{10}$$

The augmented input vector is also defined as:

$$\mathbf{U}(t) \triangleq [\omega_e(t), T_d(t)]^T \tag{11}$$

Considering Eqs. (1), (2), (8), and (9), the system dynamics can be summarized as:

$$\dot{\mathbf{X}}(t) = \mathscr{F}(\mathbf{X}(t), \mathbf{U}(t)), \qquad \mathbf{X}(0) = \mathbf{X}_0 \tag{12}$$

where $\mathscr{F}$ is defined as below:

$$\mathscr{F} \triangleq \begin{bmatrix} v(t) - v_{leader}(t) \\ \frac{1}{m}\left[-\frac{1}{2}\rho C_{drag} A_f v(t)^2 - \mu_{roll} mg + \frac{1}{r} T_d(t)\right] \\ -\frac{V_{batt} - \sqrt{V_{batt}^2 - 4R_{batt}P_{batt}(t)}}{2R_{batt}Q_{batt}}. \end{bmatrix} \tag{13}$$

Given the initial condition $\mathbf{X}(0) = \mathbf{X}_0$, the cost function J_c in its most general form is defined as below:

$$J_c = \int_0^{t_f} \lambda(\mathbf{X}(t), \mathbf{U}(t))\ dt + \psi(\mathbf{X}(t_f)) \tag{14}$$

where t_f is the final time, and $\lambda(\mathbf{X}(t), \mathbf{U}(t))$ is the cost of intermediate states and inputs. Also, $\psi : \mathbb{R}^3 \to \mathbb{R}_+$ puts a penalty on the terminal state vector to ensure the system will reach the desired terminal point $\mathbf{X}_{des}(t_f)$. In this study, to minimize the fuel consumption during the drive cycle and to keep the distance between the follower and the leader within the desired range, λ is set to be:

$$\lambda(\mathbf{X}(t), \mathbf{U}(t)) = \beta \dot{m}_{fuel}(t) + \alpha z(t)^2. \tag{15}$$

where α and β are two positive weights. Let δt be a small enough sampling time, one can discretize Eq. (12) by using the Euler method:

$$\mathbf{X}(k+1) = \mathbf{X}(k) + \delta t \mathscr{F}(\mathbf{X}(k), \mathbf{U}(k)), k = 0, 1, 2, ..., N-1 \tag{16}$$

Similarly, Eq. (14) can be discretized as:

$$\begin{aligned} J &= \sum_{k=0}^{N-1} \delta t\ (\lambda(\mathbf{X}(k), \mathbf{U}(k))) + \psi(\mathbf{X}(N)) \\ &= \sum_{k=0}^{N-1} \delta t \left(\beta \dot{m}_{fuel}(k) + \alpha z(k)^2\right) + \psi(\mathbf{X}(N)) \end{aligned} \tag{17}$$

where $N \triangleq \frac{t_f}{\delta t}$.

The cost-to-go $J(\mathbf{X}(k))$ is defined as the cost from the system state $\mathbf{X}(k)$ at time step k to the end of the time horizon and is equal to:

$$\begin{aligned} J(\mathbf{X}(k)) &= \sum_{\tau=k}^{N-1} \delta t \left(\beta \dot{m}_{fuel}(\tau) + \alpha z(\tau)^2\right) + \psi(\mathbf{X}(N)) \\ &= \delta t \left(\beta \dot{m}_{fuel}(k) + \alpha z(k)^2\right) + J(\mathbf{X}(k+1)), \\ &\qquad k = 0, 1, 2, ..., N-1. \end{aligned} \tag{18}$$

Also, note that:

$$J(\mathbf{X}(N)) = \psi(\mathbf{X}(N)). \tag{19}$$

The optimal cost-to-go $J^*(\mathbf{X}(k))$ is defined as the minimum cost if the system starts at the state $\mathbf{X}(k)$ in time step k. Based on Bellman's principle of optimality, $J^*(\mathbf{X}(k))$ is defined in a recursive manner:

$$J^*(\mathbf{X}(N)) = \psi(\mathbf{X}(N)) \tag{20}$$

$$J^*(\mathbf{X}(k)) = \min_{\mathbf{U}(k) \in \mathbb{U}(k)} \left(\delta t \left(\beta \dot{m}_{fuel}(k) + \alpha z(k)^2\right) + J^*(\mathbf{X}(k+1))\right)$$

$$\text{subject to } \begin{cases} \mathbf{X}(k) \in \mathbb{X}(k) \\ P_{batt}(k) \leqslant \dfrac{V_{batt}^2}{4R_{batt}} \end{cases} \tag{21}$$

where $\mathbb{X}(k)$ and $\mathbb{U}(k)$ are the time-variant state grid and input grid, respectively:

$$\mathbb{X}(k) = \{\mathbf{X}(k) \mid \mathbf{X}_{min}(k) \leqslant \mathbf{X}(k) \leqslant \mathbf{X}_{max}(k)\} \tag{22}$$

$$\mathbb{U}(k) = \{\mathbf{U}(k) \mid \mathbf{U}_{min}(k) \leqslant \mathbf{U}(k) \leqslant \mathbf{U}_{max}(k)\} \tag{23}$$

$\mathbf{X}_{min}(k)$ and $\mathbf{X}_{max}(k)$ denote the minimum and the maximum range for $\mathbf{X}(k)$, respectively. Similarly, $\mathbf{U}_{min}(k)$ and $\mathbf{U}_{max}(k)$ denote the minimum and the maximum range for $\mathbf{U}(k)$, respectively. Besides, the control input by which $J^*(\mathbf{X}(k))$ is attained is called the optimal control input $\mathbf{U}^*(\mathbf{X}(k))$. Note that the last constraint in Eq. (21) is introduced to ensure the SoC remains a real number.

IV. Solving Integrated Optimization Using a Customized ADP

DP is an optimization method that provides globally optimal solutions to an optimization problem. Based on Eq. (21), known as Hamilton–Jacobi–Bellman equation, the necessary and sufficient condition for optimality is that each period's decision is made by acknowledging that all future decisions will be optimally made. Therefore, in DP, the problem is solved backward from the last time step toward the first time step. In each time step, the optimal input $\mathbf{U}^*(\mathbf{X}(k))$ and the corresponding optimal cost-to-go $J^*(\mathbf{X}(k))$ for every point in the state grid are cached in tables. Storing $\mathbf{U}^*(\mathbf{X}(k))$ & $J^*(\mathbf{X}(k))$ for every point in the state grid requires a vast amount of memory, which is one of the significant drawbacks of DP.

Consider a general control-affine discrete dynamical system shown in Eq. (24) with the corresponding cost function I defined in Eq. (25).

$$\mathbf{x}(k+1) = f(\mathbf{x}(k)) + g(\mathbf{x}(k))\mathbf{u}(k), \qquad k = 0, 1, 2, ..., N-1 \tag{24}$$

$$I = \frac{1}{2} \sum_{k=0}^{N-1} \mathbf{x}(k)^T Q\ \mathbf{x}(k) + \mathbf{u}(k)^T R\ \mathbf{u}(k) + \theta(\mathbf{x}(N)) \tag{25}$$

where f and g represent the dynamics of the system. The positive semi-definite matrix Q and the positive definite matrix R put penalties on the intermediate states $\mathbf{x}(k)$ and inputs $\mathbf{u}(k)$. $\theta(\mathbf{x}(N))$ also makes sure the system reaches the desired final state. Note that lowercase $\mathbf{x}(k)$ and $\mathbf{u}(k)$ defined here are not confused with $\mathbf{X}(k)$ and $\mathbf{U}(k)$ defined before.

The optimal control input $\mathbf{u}^*(\mathbf{x}(k))$ satisfies the Bellman optimality condition $\frac{\partial I(\mathbf{x}(k))}{\partial \mathbf{u}^*(\mathbf{x}(k))} = 0$, and can be calculated as [7]:

$$\mathbf{u}^*(k) = -R^{-1} g(\mathbf{x}_k)^T \frac{\partial I(\mathbf{x}(k+1))}{\partial \mathbf{x}(k+1)} \tag{26}$$

To solve the equation above, mainly two approaches exist in ADP: heuristic dynamic programming, and dual heuristic programming. However, the inputs found by these methods are not constrained as they are proportionally dependent on the magnitude of $\frac{\partial I(\mathbf{x}(k+1))}{\partial \mathbf{x}(k+1)}$, and they can be unreasonably high. Besides, to formulate the problem under the available ADP algorithms, the cost function needs to be a quadratic function of the states and the inputs. However, the fuel consumption is not a quadratic function of the states and the inputs.

In this study, a customized ADP algorithm is proposed which 1) can be applied to both control-affine and non-affine systems, 2) can take care of complex constraints of inputs, and 3) handles non-quadratic nonlinear cost function of the states and the inputs. Specifically, this algorithm approximates the optimal cost-to-go at each time step using a deep neural network:

$$\bar{J}^*(\mathbf{X}(k)) = \phi_k(\mathbf{X}(k)) \tag{27}$$

where the deep neural network $\phi_k(.)$ takes the current state $\mathbf{X}(k)$ as the input and outputs the approximated optimal cost-to-go $\bar{J}^*(\mathbf{X}(k))$. Fig. 4 shows the structure of $\phi_k(.)$. The method then proceeds backward in time. We first define the desired terminal set $\mathcal{R}(N)$ that encompasses all of the desired terminal points $\mathbf{X}_{des}(N)$ at the time step N:

$$\begin{aligned} \mathcal{R}(N) = \{\mathbf{X}_{des}(N) \in \mathbb{R}^3 \mid \\ \mathbf{X}_{des,min}(N) \leqslant \mathbf{X}_{des}(N) \leqslant \mathbf{X}_{des,max}(N)\} \end{aligned} \tag{28}$$

where $\mathbf{X}_{des,min}(N)$ and $\mathbf{X}_{des,max}(N)$ denote the minimum and the maximum range for $\mathbf{X}_{des}(N)$, respectively. To start the learning procedure at $k = N$, one can define:

$$\bar{J}^*(\mathbf{X}(N)) = \begin{cases} \psi(\mathbf{X}(N), & for \quad \mathbf{X}(N) \in \mathcal{R}(N) \\ \infty, & elsewhere \end{cases} \tag{29}$$

and train the network accordingly. The points that are outside of the desired set are not favorable, and therefore, have been assigned the infinite cost. However, this definition will result in numerical issues in the learning process since high gradients will be derived between the points in the vicinity of the boundary of the desired set. Replacing ∞ to a large enough number can help mitigate this issue, but the effectiveness is limited [6]. To solve this numerical issue, the network $\phi_N(.)$ is only trained through the points inside the desired set, and thus the achieved approximate cost-to-go is only optimized within the desired set through the trained network. Without considering the points out of the desired region, the numerical issue mentioned can be avoided.

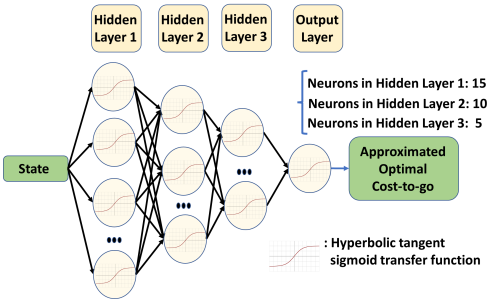

Fig. 4: Structure of the Deep Neural Network.

Following the same idea, the desirable reachable sets should be found for other steps too (from step $N-1$ to step 1) to find the desirable states in any step k that can be projected to the admissible states in step $k+1$, given the constraints on control inputs [6] and the constraints set by the system dynamics. The reachable set $\mathcal{R}(k)$ at step k can be mathematically defined as:

$$\begin{aligned} \mathcal{R}(k) = \{\mathbf{X}(k) \mid \exists\ \boldsymbol{\pi}(k) \\ \text{such that} \begin{cases} \mathbf{X}(k+1) = \mathbf{X}(k) + \delta t \mathscr{F}(\mathbf{X}(k), \boldsymbol{\pi}(k)) \\ \mathbf{X}(k+1) \in \mathcal{R}\ (k+1) \\ \mathbf{X}(k) \in \mathbb{X}(k) \\ \boldsymbol{\pi}(k) \in \mathbb{U}(k) \\ P_{batt}(k) \leqslant \dfrac{V_{batt}^2}{4R_{batt}} \end{cases} \\ \} \end{aligned} \tag{30}$$

Obtaining the reachable set allows us to only train the network using data within the reachable sets, which eliminates the potential numerical issue mentioned above. The reachable sets are also found backward in time.

The network first learns the approximated optimal cost-to-go at the final step using Eq. (29) for the points inside $\mathcal{R}(N)$. Then, it iterates backward in time. At each step k, random samples for the states $\mathbf{X}(k)$ inside the reachable set $\mathcal{R}(k)$ are generated as shown in the Table of Algorithm 1, the algorithm finds the optimal control input $\mathbf{U}^*(\mathbf{X}(k))$ and the minimum approximated cost-to-go $\bar{J}^*(\mathbf{X}(k))$ for each sampling states following steps shown in the Table of Algorithm 1. To find the optimal control $\mathbf{U}^*(\mathbf{X}(k))$, the standard ADP method needs to analytically solve Eq. (25) to reach Eq. (26). This requires the cost function to be quadratic and the dynamics equation to be affine. To resolve this issue, instead of analytically solving for the control input $\mathbf{U}^*(\mathbf{X}(k))$, we numerically solve this step by comparing the cost values of different inputs within its range (Steps 4 and 5 in the Table of Algorithm 1). This modification on the algorithm is not a computation-heavy step in ADP, but it brings significant practical value to enable the

ADP application to this complex energy co-optimization problem. Once the optimal control inputs $\mathbf{U}^*(\mathbf{X}(k))$ and approximated cost to go $\bar{J}^*(\mathbf{X}(k))$ are obtained for the sampling state points, the neural network $\phi_k(.)$ will be trained to fit the manifold of $\bar{J}^*(\mathbf{X}(k))$ in the state space.

$$\bar{J}^*(\mathbf{X}(k)) = \phi_k(\mathbf{X}(k)) = \begin{cases} \psi(\mathbf{X}(N), \quad for \quad k = N \quad \& \quad \mathbf{X}(N) \in \mathcal{R}\ (N) \\ \min\limits_{\mathbf{U}(k)\in\mathbb{U}(k)} \delta t \left(\beta \dot{m}_{fuel}(k) + \alpha z(k)^2\right) + \phi_{k+1}(\mathbf{X}(k+1)) \\ \qquad for \quad k \neq N \quad \& \quad \mathbf{X}(k) \in \mathcal{R}\ (k). \end{cases} \tag{31}$$

The training algorithm is detailed in Algorithm 1. Once the learning procedure is done, it can be implemented in the forward simulation to find the sub-optimal bounded control sequence given the initial condition $\mathbf{X}(0)$. The implementation algorithm is explained in Algorithm 2.

Algorithm 1: Training Neural Network

Select a small positive number ζ, and a big enough integer $IterMax$;
for $k = N$: -1 : 0 **do**
  Choose h different random training samples $\mathbf{X}^p(k)$ in the reachable set $\mathcal{R}\ (k)$ where $p \in \{1, 2, ..., h\}$;
  Discretize the input grid into m points such that $\mathbf{U}_{min}(k) \leqslant \mathbf{U}^j(k) \leqslant \mathbf{U}_{max}(k)$ for $j \in \{1, 2, ..., m\}$;
  For each training sample $\mathbf{X}^p(k)$ find $\bar{J}^*(\mathbf{X}^p(k))$ using Eq. (31) ;
  Initialize $\phi_k^0(.)$ with random parameters;
  for i = 1 : $IterMax$ **do**
    Update the neural network $\phi_k^i(.)$ and find the parameters to approximate $\bar{J}^*(\mathbf{X}^p(k))$ using backpropagation on the entire training samples;
    if $\|\phi_k^i(.) - \phi_k^{i-1}(.)\| \leq \zeta$ **then**
      Break;
    end
  end
  $\phi_k(.) \leftarrow \phi_k^i(.)$;
end

Algorithm 2: Implementation

for $k = 0 : N - 1$ **do**
  $\mathbf{U}^*(\mathbf{X}(k)) = \underset{\mathbf{U}(k)\in\mathbb{U}(k)}{\operatorname{argmin}} \ \delta t(\beta \dot{m}_{fuel}(k) + \alpha z(k)^2) + \phi_{k+1}(\mathbf{X}(k+1))$
  $\mathbf{X}(k+1) = \mathbf{X}(k) + \delta t \mathscr{F}(\mathbf{X}(k), \mathbf{U}^*(\mathbf{X}(k)))$
end

V. Simulation Results

In this section, a motivating case study in which an autonomous HEV is following a leader is considered. Vehicles are connected together through V2V communications with negligible delay. Specifications of the environment and the follower are presented in Table I. One of the most commonly used drive cycles is selected as the leader's velocity profile: FTP 75 Urban Drive Cycle (UDC). Also, the time horizon is set to be 100 seconds.

Parameter	Value	Parameter	Value	Parameter	Value
m (kg)	1350	r (m)	0.28	μ_{roll}	0.007
ρ (kgs/m^3)	1.225	C_{drag}	0.3	A_f (m^2)	2.2
k_c	3.9	r_s (m)	0.030	r_r (m)	0.078
μ_g	0.9	μ_m	0.9	V_{batt} (V)	202
Q_{batt} (A.s)	23400	R_{batt} (Ω)	0.45	L (m)	15
δt (s)	1	ζ	0.05	$IterMax$	10

TABLE I: Environment & Vehicle System Parameters

The cost function in Eq. (17) consists of two parts. The first part considers the cost of the intermediate states and the corresponding inputs. A relatively large α will make the follower to move closely at the desired distance from the leader most of the time at the expense of potential increase in the fuel cost. However, a relatively large β will urge the follower to minimize the fuel consumption while the follower might be following the leader in a more relaxed fashion. In both cases, the second part of the cost function ensures that the follower will reach the desired terminal point, finish the drive cycle in a safe distance with respect to the leader, and satisfy the charge sustaining condition.

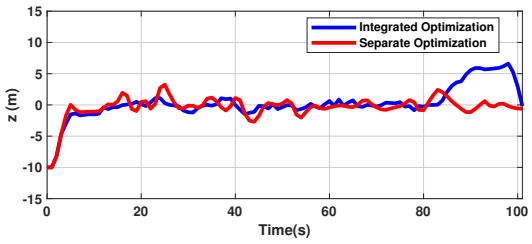

Fig. 5: Longitudinal Position Error History of the Follower.

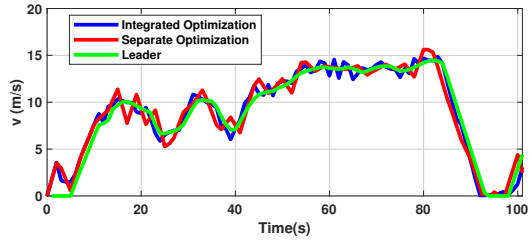

Fig. 6: Velocity History of the Leader and Follower.

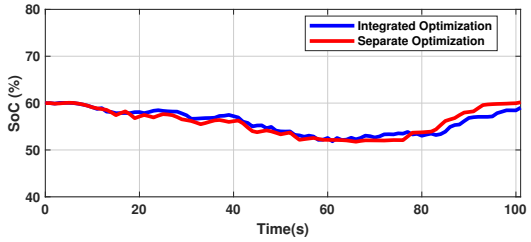

Fig. 7: Battery State of Charge History of the Follower.

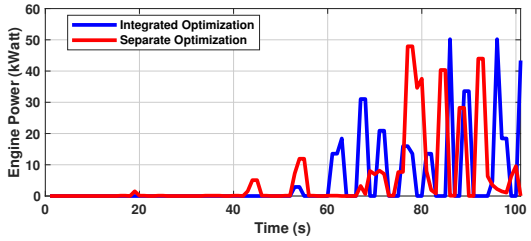

Fig. 8: Engine Power History of the Follower.

The control performance of the system using the customized ADP method for $\mathbf{X}(0) = [-10(m), 0(m/s), 60\%]^T$ is shown in Figs. (5) - (8). To show the efficacy of the method, the performance of the system under the baseline approach is also presented for comparison in Figs. (5) - (8). In the baseline approach, the follower's vehicle level and powertrain level dynamics are optimized separately. First, the vehicle level dynamics are optimized where the states are $[z\ v]^T$ and the input is $[T_d]$. Secondly, the powertrain level is optimized by strictly following the power demand from the vehicle level control in which $[SoC]$ is the state and $[\omega_e]$ is the input. As seen in Fig. (5), the proposed integrated optimization based controller has a looser regulation to follow the leader to achieve a better fuel economy. However, in the baseline approach, the controller has less degree of freedom to solve for the powertrain dynamics to minimize fuel consumption. This is evident in fuel consumption where an additional 15% fuel economy improvement was achieved by the proposed algorithm as compared with the value achieved by the baseline strategy (Table II).

Method	α	β	Terminal SoC(%)	Terminal z (m)	Fuel (g)
ADP	0.01	1	59.03	-0.20	30.46
Separate	–	–	60.17	-0.62	35.82

TABLE II: Summary of Results for FTP 75 UDC

VI. Conclusion

Conventionally, in the study of hybrid electric vehicles, vehicle coordination optimization and powertrain energy management are studied separately. This paper explores the integrated optimization of vehicle coordination and powertrain energy management which can result in considerable fuel economy improvement. Then, the optimization problem is formulated under the framework of a customized ADP. Finally, to examine the efficacy of the method, a numerical example is studied, which shows a 15% improvement in fuel consumption compared to the separated optimization method.

Acknowledgment

This research was partially supported by the National Science Foundation under Grant No. 1826410.

References

[1] X. Qu, Y. Yu, M. Zhou, C.-T. Lin, and X. Wang, "Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: a reinforcement learning based approach," *Applied Energy*, vol. 257, p. 114030, 2020.

[2] H. A. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, and I. V. Kolmanovsky, "Predictive energy management of a power-split hybrid electric vehicle," in *2009 American control conference*. IEEE, 2009, pp. 3970–3976.

[3] H. Kim and D. Kum, "Comprehensive design methodology of input-and output-split hybrid electric vehicles: In search of optimal configuration," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 6, pp. 2912–2923, 2016.

[4] H. Zhang, T. Feng, G.-H. Yang, and H. Liang, "Distributed cooperative optimal control for multiagent systems on directed graphs: An inverse optimal approach," *IEEE Transactions on Cybernetics*, vol. 45, no. 7, pp. 1315–1326, 2014.

[5] G. Ma, M. Ghasemi, and X. Song, "Integrated powertrain energy management and vehicle coordination for multiple connected hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 2893–2899, 2017.

[6] P. Elbert, S. Ebbesen, and L. Guzzella, "Implementation of dynamic programming for n-dimensional optimal control problems with final state constraints," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 924–931, 2012.

[7] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.