A Data Driven Real Time Control Strategy for Power Management of Plug-in Hybrid Electric Vehicles

Jafar Abbaszadeh Chekan

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Engineering Mechanics

Saied Taheri, Chair

Mehdi Ahmadian

Shima Shahab

2/16/2018

Blacksburg, Virginia

Keywords:

Plug-in Hybrid Electric Vehicle, Optimal Control,

Power Management, Dynamic Programming, Real Time Control

A Data Driven Real Time Control Strategy for Power Management of Plug-in Hybrid Electric Vehicles

Jafar Abbaszadeh Chekan

ABSTRACT (academic)

This thesis presents a real-time control scheme to improve the fuel economy of plug-in hybrid electric vehicles (PHEVs) by accounting for the instantaneous states of the system as well as the future trip information. To design the mentioned parametric real-time power management policies, we use dynamic programming (DP). The state and decision variables in the DP algorithm are selected in a way that provides the best tradeoff between the computational time and accuracy which is the first contribution of this research effort. The obtained DP trajectories are then used to train a set of linear maps for the powertrain control variables such as the engine and electric motor/generator torque inputs, via linear regression method. The DP results indicate that the trip length is a key factor in determining the optimal control decisions. To account for this factor, an additional input variable pertaining to the remaining length of the trip is considered during the training of the real-time control policies. The proposed controller receives the demanded propulsion force and the powertrain variables as inputs, and generates the torque commands for the engine and the electric drivetrain system. Numerical simulations indicate that the proposed control policy is able to approximate the optimal trajectories with a good accuracy using the real-time information for the same drive cycles as trained and also drive cycles out of training set. To maintain the battery state-of-charge (SOC) above a certain lower bound, two logics have been devised for soc constraint management in real time.

A Data Driven Real Time Control Strategy for Power Management of Plug-in Hybrid Electric Vehicles

Jafar Abbaszadeh Chekan

GENERAL AUDIENCE ABSTRACT

During the past two decades desperate need for energy-efficient vehicles which has less emission have led to a great attention to and development of electrified vehicles like pure electric, Hybrid Electric Vehicle (HEV) and Plug-in Hybrid Electric Vehicles (PHEVs). Resultantly, a great amount of research efforts have been dedicated to development of control strategies for this type of vehicles including PHEV which is the case study in this thesis.

This thesis presents a real-time control scheme to improve the fuel economy of plug-in hybrid electric vehicles (PHEVs) by accounting for the instantaneous states of the system as well as the future trip information. To design the mentioned parametric real-time power management policies, we use dynamic programming (DP). First, a representative power-split PHEV powertrain model is introduced, followed by a DP formulation for obtaining the optimal powertrain trajectories from the energy cost point of view for a given drive cycle. The state and decision variables in the DP algorithm are selected in a way that provides the best tradeoff between the computational time and accuracy which is the first contribution of this research effort. These trajectories are then used to train a set of linear maps for the powertrain control variables such as the engine and electric motor/generator torque inputs, through a least-squares optimization process. The DP results indicate that the trip length (distance from the start of the trip to the next charging station) is a key factor in determining the optimal control decisions. To account for this

factor, an additional input variable pertaining to the remaining length of the trip is considered during the training of the real-time control policies. The proposed controller receives the demanded propulsion force and the powertrain variables as inputs, and generates the torque commands for the engine and the electric drivetrain system. Numerical simulations indicate that the proposed control policy is able to approximate the optimal trajectories with a good accuracy using the real-time information for the same drive cycles as trained and drive cycle out of training set. To maintain the battery state-of-charge (SOC) above a certain lower bound, two logics have been introduced a switching logic is implemented to transition to a conservative control policy when the battery SOC drops below a certain threshold. Simulation results indicate the effectiveness of the proposed approach in achieving near-optimal performance while maintaining the SOC within the desired range.

To MY PARENTS

Acknowledgment

First and Foremost, I would like to express my sincere thanks to my advisor Prof. Saied Taheri for his continuous support in my MSc studies and his patience and motivation and immense knowledge in Vehicle Dynamics and Control. His support helped me in all the time of research and writing of this thesis.

I would like to thank Prof. Saied Bashash at San Jose State University (SJSU) for his collaboration and guidance on this thesis while I was in research internship position at SJSU. Saeid has been an "unofficial", yet dedicated co-advisor, who has instilled in me.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Mehdi Ahmadian and Prof. Shima Shahab.

Last but not the least, I would like to thank my parents for being supportive spiritually throughout my life.

Table of Contents

Table of Figures

List of Tables

Chapter 1: INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Nowadays, automobiles have been the essential and non-separable part of our daily life. By ever increasing population and subsequently demand for more vehicle, the associated challenges of economical energy required to drive these vehicles considering the increasing fuel prices and tailpipe emission reduction have been raised up. Electrification of the automobiles have received a tremendous attention within the past two decades as a solution way to address the so called problems. Electrified vehicles due to having lower overall energy cost, lower emission and tax credit and carpool pass have been the popular vehicle choices.

There are three types of electrified vehicles, namely electric vehicles, hybrid electric vehicles and plug in hybrid electric vehicles (PHEV).

Hybrid electric vehicles (HEVs) consist of an engine, a battery and two motor/generators (MGs). The battery can be charged by both MG1 and MG2. Plug-in version of HEVs, plug-in HEV (PHEV) has the same components with this difference that the battery pack has higher storage capacity and can be plugged in to electric grid. In other words in PHEV powertrain the battery can be charged by MGs and external power grid. Battery as an additional power source enables the engine to work near energy efficient points while the demand power is being delivered. In other words, the philosophy of HEVs is adding a degree of freedom to power transmission system and decoupling the vehicle speed from engine speed. In the PHEV (or HEV) engine can operate always near energy efficient points. In addition to the mentioned advantage of both HEV and PHEV over the conventional vehicles, downsized engine which means less weight and regeneration of the electrochemical energy during the breaking or low power demand instances are the second and third

advantage of these type of vehicles [1]. A right-size engine has better energy efficiency and smaller heat loss[2]. Extendable to PHEV, the architecture of the HEV can be described in three categories of parallel, series and power split (series/parallel) [3]. In Parallel hybrid there are two separate power paths from engine and battery, in which deliver the demand power individually or collaboratively. In the series hybrid, the engine is decouples from the vehicles which enables the engine to work in near its optimal operation point. The power split configuration which is the architecture used in this thesis is combination of series and parallel models and has the advantages of both of them [2].

1.2 Power Split Hybrid (Series Parallel) architecture

In this thesis our PHEV model is the so-called Power Split Hybrid which is similar architecture to most of HEV and PHEV vehicles. Figure 1 shows a power-split PHEV powertrain configuration similar to a Toyota Prius PHEV, adopted from [2, 4]. The model consists of an engine and two motor/generators (MGs), MG1 and MG2. The PHEV battery can be charged by both MG1 and MG2 as well as by directly being plugged in to grid. This architecture uses a planetary gear set as the device to split power between the engine and the MGs, which enables decoupling the engine from the wheels, and provides the opportunity for optimal power management. This power split device encompasses a ring gear, a sun gear, a carrier, and three pinion gears. The engine is connected to planet carrier, and MG1 and MG2 are connected to the sun and the ring gears, respectively. There is also a torque coupler between MG2 and the ring gear.

1.3 Objectives of Power management of Plug-in Hybrid Electric Vehicles

There are different objectives to be minimized when a supervisory control is designed to split the demand power between the battery and engine. Sometimes the goal is finding optimal powertrain system trajectories for energy cost which is the cost of fuel for HEVs and combined fuel and electricity cost for PHEVs considering the price ratio of electricity to fuel [2, 4-8]. This objective is our concern in this thesis.

Far apart from this objective, the environmental concerns is the other motivation in designing power split control policy for HEV and PHEV. In this case tailpipe emissions reduction is the crucial objective for control engineers [9-11].

Battery longevity is the other criteria that sometimes is the focus of optimal power split control policy to be designed [12, 13]. One way to address this objective is constraining the state of charge (SOC) of the battery for a specific interval, e.g. $0.3<SOC<0.8$ which is usually considered in the mention emission reduction and energy cost minimization objectives too. However, this constraint is a little conservative and even though helps to avoid battery degradation, it causes not to use the capacity of the battery and leads to need to bigger battery pack. Hence, the optimization problem for battery degradation objective will be finding an optimal trajectory considering this tradeoff [1].

1.4 Control Algorithms to address the power split management for PHEV

There are different approaches and algorithms to address the power management for either HEV or PHEVs which have been developed to address the previously mentioned objectives; energy cost, emission, and battery longevity. In general, the design approaches for the PHEV power management problem can be divided in two categories, the rule-based and the trajectory-based strategies [1]. The rule-based strategy is an instantaneous optimization approach that computes the control decision based on the instantaneous values of the vehicle's state variables and the demanded

power. The Equivalent Consumption Minimization Strategy (ECMS) is an example of a rule-based method that derived from the Pontryagin's minimum principle [14-16]. On the other hand, trajectory-based algorithms such as deterministic dynamic programming (DDP) yield the global optimal solution [17-21], compared to the rule-based methods that merely reach a sub-optimal solution. For the first time Bellamn [22] introduced dynamic programming. DP is a useful tool that is used in addressing optimal control for different problems like automotive engineering [23-25], robotics [26, 27], economics [28], and chaos control [29], to name but a few. Despite of some approach to make DDP running faster [30], it still needs an excessive computation effort which makes it unsuitable for online power management and is mostly regarded as a benchmark to evaluate the optimality of the other approaches.

Recently, in the shadow of the advancements of intelligent transportation technologies enabled by the global positioning systems (GPS) and geographical information systems (GIS), gaining access to the real-time traffic, road information, and vehicle location data is not far-fetched [17]. Such data can be brought into the vehicle power management system for further adaptation and optimization based on the near future events and conditions. In [17], a GPS-based PHEV power management strategy was studied. An optimal charge-depletion pattern was introduced for two case studies, the local roads and the freeways. In [31], an optimal charge depletion strategy is addressed using online traffic data. First, a supervisory level algorithm computes the global SOC trajectory using the traffic data [32]. Then, the SOC trajectory is fed as the final SOC constraint to a lower level model predictive controller for handling the real-time power management decisions.

It is worthy to mention that model predictive control as a useful tool which is widely used in conventional automotive systems [33, 34], has received attention in energy management of both HEVs and PHEVs [35-37]. The computation burden of MPC-based approach in power split

management of PHEVs, makes it difficult for online implementation. The online policy provided in this thesis, in the case of being well-trained can be very useful for real time and on-board implementation. Since battery is an important element of the PHEVs, it must be packed for control design aim. In [38] impact of the battery sizing on the power management has been studied and battery parameters, voltage, resistance, and capacity has been formulated based on cell-level battery parameters.

1.5 Goal of this Thesis

This thesis investigates the power management problem in plug-in hybrid electric vehicles (PHEVs), and proposes a novel real-time control strategy for the near-optimal power management of power-split PHEVs. HEVs and PHEVs encompass two energy sources, namely, fuel and electricity, to provide propulsion power to the wheels through an internal combustion engine and a set of electric machines. The conventional HEV configuration contributes to the vehicle energy management problem by decoupling the vehicle's speed from that of the internal combustion engine, thereby enabling the engine to operate at a higher efficiency mode. The PHEV configuration has further advantages over the HEV, which include longer all-electric driving range and lower tailpipe emissions. Compared to the all-electric vehicles, which suffer from a side-effect called range anxiety, PHEVs can provide unlimited travel range through the onboard engine when the battery charge is depleted. As a result, PHEVs have invaluable advantages over both HEVs and all-electric vehicles, making them a popular vehicle type in the market.

1.6 Contributions of this thesis

In This thesis, we adopt the widely-used power-split powertrain configuration as a representative PHEV model. First, we formulate and apply the dynamic programming (DP) method to obtain the optimal PHEV powertrain trajectories. This thesis contributes to the literature two folds. First, new

insights are provided to facilitate the computation and the accuracy of the DP for the adopted powertrain model. We reformulates the dynamic programing for power split model of PHEV so that to make it effective in dealing with constraint leakage by reducing the interpolation dimension from 2D to 1D. Second, the obtained optimal trajectories are used to develop and train a new class of real-time control policies that approximate the DP results with a linear polynomial model in the least-squares sense. The model receives the demanded propulsion force as the driver (or drive cycle) input, and the powertrain system's states to make near-optimal decisions on the instantaneous engine and electric drivetrain torque values. The results described in the chapter 2, 3, and 4 in this thesis are based on the author's published paper [25] and submitted paper [39].

1.7 Nomenclature

Variable	*Description*
ω_s	Sun gear speed
ω_c	Carrier speed
ω_r	Ring gear speed
ω_E	Engine speed
ω_{MG1}	MG1 speed
ω_{MG2}	MG2 speed
v	Vehicle speed
R	Radius of ring gear
S	Radius of sun gear
F	Internal reaction force of sun and planet gears
T_e	Engine torque
T_{MG1}	MG1 torque
T_{MG2}	MG2 torque
I_e	Engine rotational inertias
I_{MG1}	MG1 rotational inertias
I_{MG2}	MG2 rotational inertias
I_ω	Wheel inertia
r	Tire radius
K	Final drive ratio
F_{road}	Road load acting on vehicle
f_{damp}	Damping force of wheels/axle

f_{roll}	Rolling resistance of the tires
f_{drag}	Aerodynamic drag force
b_{ω}	Damping coefficient
μ	Rolling resistance coefficient
m	Vehicle mass
g	Gravitational acceleration
ρ	Air density
A_{fr}	Vehicle frontal area
C_d	Aerodynamic drag coefficient
Soc	Battery state-of-charge
I_{batt}	Battery current
Q_{batt}	Battery charge capacity
V_{oc}	Battery open-circuit voltage
R_{batt}	Battery Internal resistance,
V_{batt}	Battery terminal voltage
R_{cell}	Cell-level battery internal resistance
Q_{cell}	Cell-level battery charge capacity
$V_{oc-cell}$	Cell-level battery open-circuit voltage
n_s	Number of battery cells (in series) per string
n_p	Number of parallel strings
P_{batt}	Battery power
P_{elec}	Electric Power
η_{MG1}	MG1 electromechanical efficiency
η_{MG2}	MG2 electromechanical efficiency
J	Accumulated cost function
L	Instantaneous transition cost
N	Final time step
β	Relative price of gasoline to electricity
α_{fuel}	Equivalent energy of 1 gram of gasoline (MJ)
α_{elec}	Equivalent energy of 1 watt of electricity (MJ)
W_{fuel}	Instantaneous fuel consumption rate in gr/s
TL	Trip length
RTL	Remaining trip length
m_{eq}	Equivalent vehicle mass
F_{dem}	Driver force demand

Chapter 2: POWER-SPLIT PHEV POWERTRAIN MATHEMATICAL MODEL

2.1 Introduction

In this section the mechanical, electrical, and electromechanical model for the power split PHEV are introduced and the dynamic equations of the PHEV model are derived.

A power-split PHEV powertrain configuration is adopted and examined in this paper from [2, 4]. The PHEV model configuration and dynamics are similar to those of Toyota Prius. The model consists of an internal combustion engine, a battery pack, and two motor/generators (MGs), MG1 and MG2. The battery can be charged by directly being plugged into the electric grid, or through the engine and one or both of the MGs on the road. Figure 1 shows the schematic diagram of the PHEV.

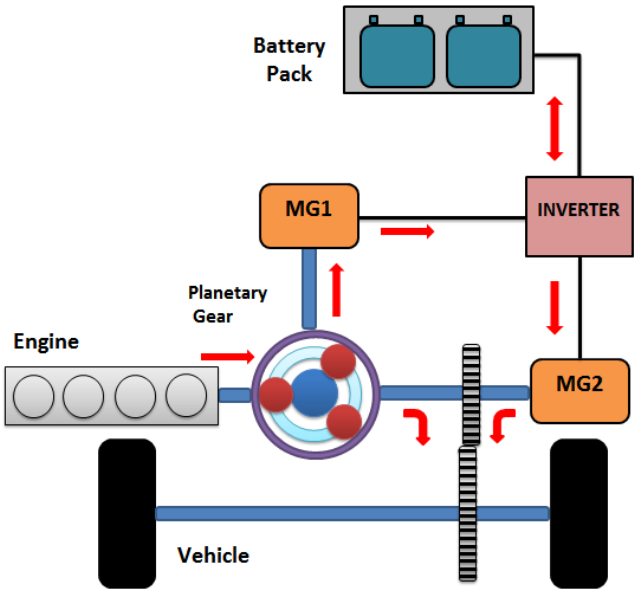


Figure 1 Power-split PHEV Architecture.

2.2 Mechanical Sub-System Modeling

A planetary gear set is used to split the propulsion power between the engine and the MGs. This device encompasses a ring gear, a sun gear, a carrier, and three planet gears. The engine is connected to the planet carrier, and MG1 and MG2 are connected to the sun and the ring gears, respectively.

$$\omega_c(R+S) = \omega_s S + \omega_r R \tag{1}$$

where R and S represent the radii of the ring gear and the sun gear, respectively. Since the engine, MG1, and MG2 are connected to planet carrier, the sun gear, and the ring gear, respectively, they have same angular speed as their corresponding gears, i.e., ω_e, ω_{MG1}, and ω_{MG2}.

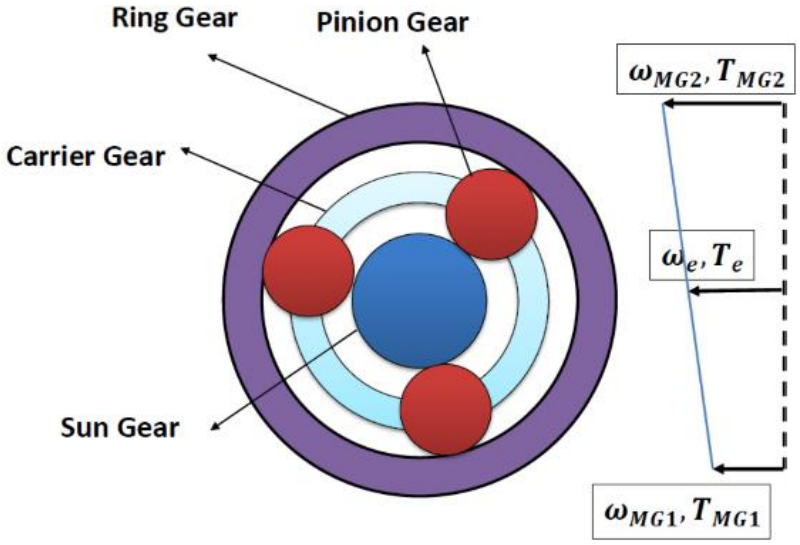


Figure 2 Planetary gear set and lever diagram

By neglecting the flexibility of the shafts, the inertia of the planet gears, and the dynamics of the vehicle in all directions except the longitudinal one, the dynamics of system can be summarized as [4]:

$$\begin{bmatrix} I_e & 0 & 0 & (R+S) \\ 0 & I_{MG1} & 0 & -S \\ 0 & 0 & I'_{MG2} & -R \\ -(R+S) & S & R & 0 \end{bmatrix} \begin{bmatrix} \dot{\omega}_e \\ \dot{\omega}_{MG1} \\ \dot{\omega}_{MG2} \\ F \end{bmatrix} = \begin{bmatrix} T_e \\ T_{MG1} \\ T'_{MG2} \\ 0 \end{bmatrix} \tag{2}$$

where F denotes the internal reaction force between the sun gear and the planet gears. T_e and T_{MG1} are the engine and the MG1 torques, with I_e and I_{MG1} representing their rotational inertias, respectively. Finally, the effective torque and inertia of MG2 (which is connected to the wheels) are represented by T'_{MG2} and I'_{MG2}, and defined as follow:

$$T'_{MG2} = T_{MG2} - F_{road} r/K \qquad (3)$$
$$I'_{MG2} = I_{MG2} + (I_\omega + mr^2)/K^2$$

in which, K is the final drive ratio, m represents the vehicle's mass, and I_ω and r are the wheel inertia and tire radius, respectively. The exerted torque by and inertia of MG2 are represented by T_{MG2} and I_{MG2}, respectively. Finally, the road load acting on vehicle is denoted by F_{road}, defined as:

$$F_{road} = f_{damp} + f_{roll} + f_{drag} \qquad (4)$$

In (4), Fdamp, Froll, and Fdrag represent the damping force of wheels/axle, rolling resistance of the tires, and the aerodynamic drag force acting on vehicle, respectively, as follows:

$$f_{damp} = \frac{b_\omega v}{r}, \quad f_{roll} = \mu mg, \quad f_{drag} = 0.5\rho A_{fr} C_d v^2 \qquad (5)$$

where b_ω is damping coefficient, and v is vehicle speed. Rolling resistance coefficient is denoted by μ, g represents the gravitational acceleration, ρ is the air density, and A_{fr} and C_d represents the PHEV's frontal area and aerodynamic drag coefficient, respectively.

2.3 Electrical Sub-System Modeling

The complete PHEV model includes an additional state variable that is the State-of-Charge of the battery denoted as SOC. To model the SOC dynamics, an equivalent circuit model is used (Figure 3), comprising a voltage source in series with a resistor representing the internal resistance of the battery [4]:

$$\begin{cases} \dot{Soc}(t) = -\frac{1}{Q_{batt}} I_{batt}(t) \\ V_{batt}(t) = V_{oc}(Soc) - R_{batt}(Soc) I_{batt}(t) \end{cases} \qquad (6)$$

where I_{batt}, V_{batt}, V_{oc}, Q_{batt}, R_{batt} are the battery pack's current, terminal voltage, open-circuit voltage, charge capacity, and internal resistance, respectively. In (6), the current is assumed to be positive during battery discharging, and negative during charging.

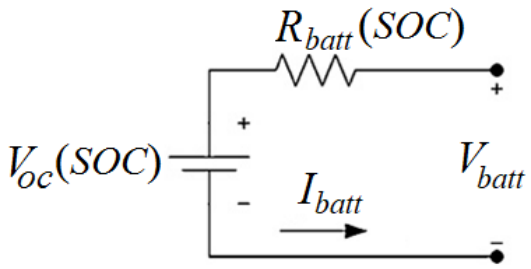


Figure 3 Equivalent Circuit Battery Model

For a battery pack with n_p parallel strings and n_s individual cells (in series) per string, V_{oc}, Q_{batt}, R_{batt} can be computed from the individual cell parameters as [38]:

$$V_{oc} = n_s V_{oc-cell}, \quad Q_{batt} = n_p Q_{cell}, \quad R_{batt} = \frac{n_s}{n_p} R_{cell} \tag{7}$$

where $V_{oc\text{-}cell}$, Q_{cell}, R_{cell} are the cell-level battery parameters, obtained from [40]. Figure 4 shows a schematic of a battery pack. In this thesis, we assume that the open-circuit voltage and the internal resistance of the battery cells are only functions of SOC, and their dependency on temperature is neglected.

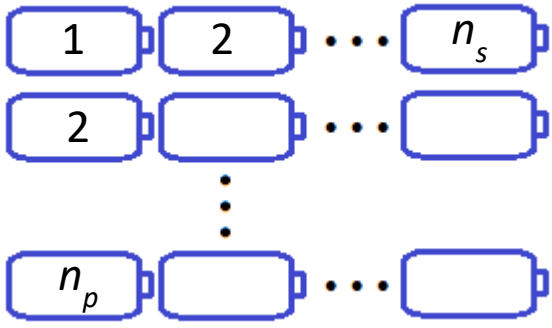


Figure 4 Battery cell to pack scaling

The electrical battery power for the equivalent circuit battery model is obtained from multiplying the battery current and terminal voltage given by (6), as follows:

$$P_{batt} = V_{oc} I_{batt} - R_{batt} I_{batt}^2 \tag{8}$$

Solving (8) for I_{batt} and substituting it in (6) yields:

$$\dot{Soc} = -\frac{V_{oc} - \sqrt{{V_{oc}}^2 - 4 P_{batt} R_{batt}}}{2 Q_{batt} R_{batt}} \tag{9}$$

The energy conversion relationship between the electrical battery power and the mechanical powers of MG1 and MG2 can be written as:

$$P_{batt} = \eta_{MG1}^{k_{MG1}}(T_{MG1}, \omega_{MG1}) T_{MG1} \omega_{MG1} + \eta_{MG2}^{k_{MG2}}(T_{MG2}, \omega_{MG2}) T_{MG2} \omega_{MG2} \tag{10}$$

Where

$$k_i = \begin{cases} -1 & T_i \omega_i > 0 \\ 1 & T_i \omega_i \leq 0 \end{cases} \quad for\ i = \{MG1, MG2\} \tag{11}$$

In (10), ηMG1 and ηMG2 are the electromechanical efficiencies of MG1 and MG2, respectively. In practice these efficiencies are nonlinear functions of the corresponding MG torque and speed. The efficiency map for the MG2 has been plotted and shown in Figure adopted from [40].

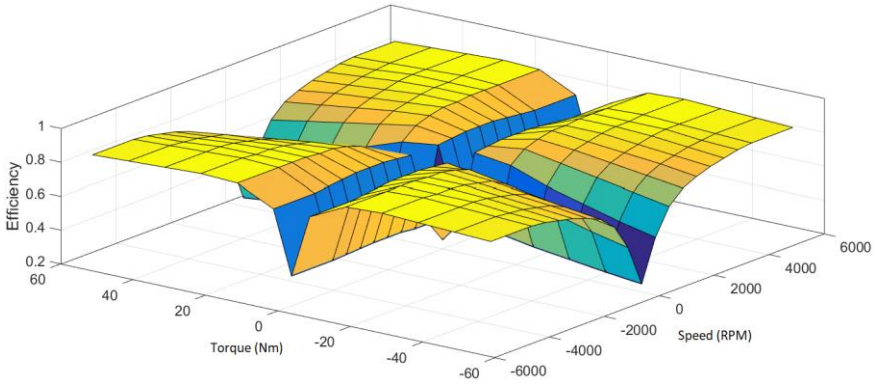


Figure 5 Efficiency of MG2

Chapter 3: DYNAMIC OPTIMIZATION PROBLEM FOR POWER SPLIT

3.1 Introduction

In this chapter, we develop a novel Dynamic Programming formulation to address combined cost of fuel and electricity consumption minimization of PHEVs for a given drive cycle. Dynamic programming is a trajectory based optimization algorithm which is widely used as a benchmark to evaluate the Optimality or as a tabulated controller. The schematic of DP has been shown in Figure 6. In this chapter, the dynamics of the vehicle presented in chapter two are discretized using the finite difference approach to carry out the numerical optimization.

3.2 Dynamic Programming Formulation

DP is a multistep optimization solving algorithm, which uses the Belman's optimality principle to find the global optimal solution. This approach is computationally exhaustive, and is not generally used for real-time implementation. However, it provides useful benchmarks for evaluating the optimality of alternative approaches. Besides, the optimal trajectories obtained from DP can be used to train control policies for real-time applications, as will be discussed later in Chapter four.

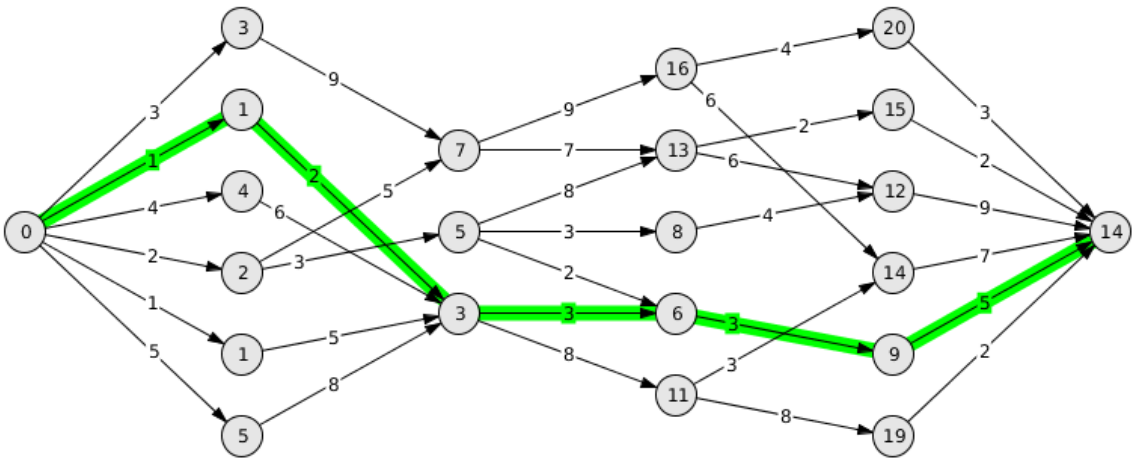


Figure 6 Dynamic programming (DP) schematic [41]

The goal of the optimization problem in this section is to find the sequence of decision variables (control inputs) and the corresponding state variables that minimize the accumulated cost of fuel and electricity over a given drive cycle, as follows:

$$\underset{u}{\text{minimize}}\ J = \sum_{k=1}^{N-1} L\big(x(k+1), x(k), u(k)\big) \tag{12}$$

subject to

$$\begin{cases} x(k+1) = f\big(x(k), u(k)\big) \\ x_{min} \le x \le x_{max} \\ u_{min} \le u \le u_{max} \end{cases} \tag{13}$$

where $x = \{\omega_e, \omega_{MG1}, \omega_{MG2}, SOC\}$ and $u = \{T_e, T_{MG1}, T_{MG2}\}$ are the system's state and control vectors, respectively, with x_{min}, u_{min} and x_{max}, u_{max} representing their lower and upper bounds; k is the time step with N being its final value corresponding to the last point in the drive cycle; f represents the collective dynamics of the PHEV powertrain system, enforced through an equality constraint; and L is the instantaneous cost of transitioning from state $x(k)$ to $x(k+1)$ under control input $u(k)$ as follows:

$$L\big(x(k+1), x(k), u(k)\big) = \beta\, \alpha_{fuel} \bar{W}_{fuel}(k) + \frac{1}{\eta_{grid}} \alpha_{elec} \bar{P}_{elec}(k) \tag{14}$$

where β is the relative price of one Joule energy obtained from gasoline to that obtained from electricity. Parameters α_{fuel} and α_{elec} convert the fuel consumption and electric energy usage within one time step to MJ (Mega Joule). The charging efficiency from the grid is denoted as η_{grid}. Variables $\bar{W}_{fuel}(k)$ and $\bar{P}_{elec}(k)$ are the average energy consumption rates for fuel and electricity during time-step k. For fuel price of $3.5 per gallon and electricity cost of $0.12 per kWh, the value of β is calculated as about 0.8 (considered in the simulations of this study). The parameters α_{fuel} and α_{elec} convert the fuel consumption and electric energy usage within one time step to MJ (Mega Joule). The charging efficiency from the grid is denoted as η_{grid} and set to be 0.98.

In this paper, DP is used to obtain and examine the optimal PHEV powertrain trajectories. DP is an optimization algorithm based on the Belman's optimality principle which is widely used to address global trajectory optimization problems. For numerical optimization, the dynamic equations of system represented in section 2 are discretized using the finite differences approach.

For the accurate calculation of fuel consumption and electric energy within each time step, we calculate the fuel consumption rate and electric power at the beginning and at the end of each time step and use the average value. Given that the engine and MG speeds can change significantly within one time step (in the order of 1s) for a constant engine or MG torque, the amount of fuel consumption and electric power can vary dramatically over the course of the time step. For example, the battery can switch from discharging to charging over the course of one the time step, resulting in a small net electric power. Calculating the average rates comes at the expense of increased computational time, but it is inevitable to for the sake of obtaining smooth and accurate optimal trajectories, and therefore, is strongly recommended here.

Assuming engine torque remain constant, while the speed changes over the course of one time-step, the average fuel consumption rate at time step k is calculated as:

$$\bar{W}_{fuel}(k) = \frac{W_{fuel}\big(T_e(k), \omega_e(k)\big) + W_{fuel}\big(T_e(k), \omega_e(k+1)\big)}{2} \tag{15}$$

where W_{fuel} is the instantaneous fuel consumption rate in gr/s, which is a function of engine speed and torque [22].

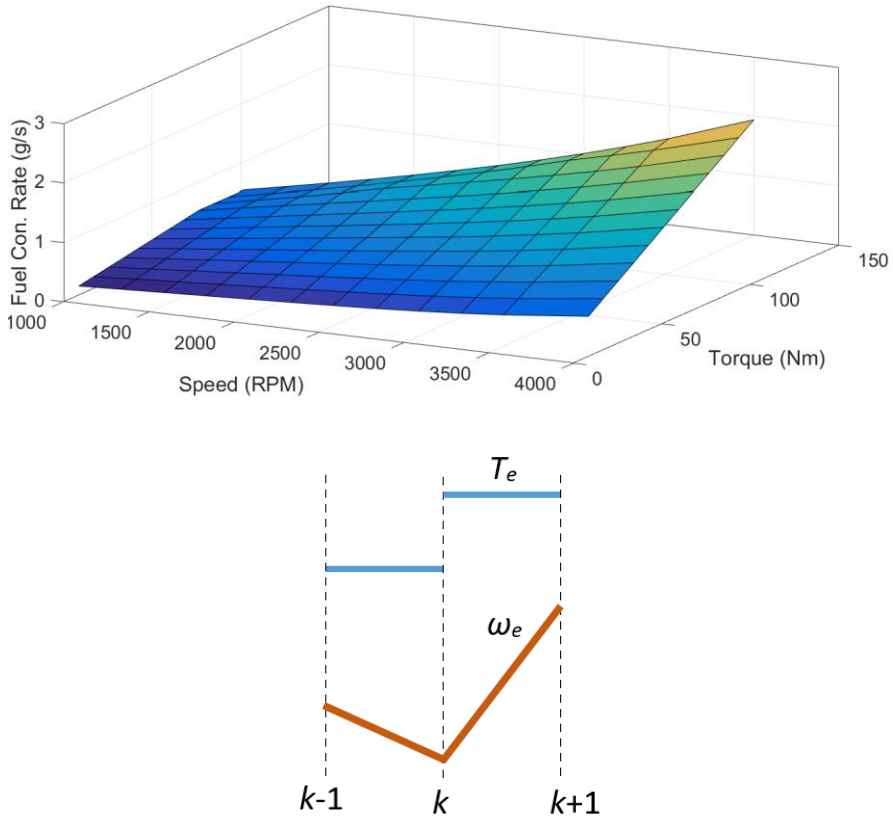


Figure 7 Calculation of fuel consumption

Moreover, the net electric power imposed on the battery during time step k is calculated as:

$$\bar{P}_{elec}(k) = -V_{oc}\, Q_{batt}\overline{\dot{Soc}}(k) \tag{16}$$

where $\overline{\dot{Soc}}(k)$ is obtained from (9) after replacing Pbatt, with the average battery power calculated for time step k as follows:

$$\bar{P}_{batt}(k) = \frac{1}{2}\left[P_{batt}\left(T_{MG1}(k), \omega_{MG1}(k), T_{MG2}(k), \omega_{MG2}(k)\right) + P_{batt}\left(T_{MG1}(k), \omega_{MG1}(k+1), T_{MG2}(k), \omega_{MG2}(k+1)\right)\right] \quad (17)$$

To complete the DP problem formulation, the admissible ranges for the states and inputs are imposed in the form of inequality constraints, as follows:

$$T_{e,min} < T_e < T_{e,max} \quad (18)$$

$$T_{MG1,min} < T_{MG1} < T_{MG1,max}$$

$$T_{MG2,min} < T_{MG2} < T_{MG2,max}$$

$$\omega_{e,min} < \omega_e < \omega_{e,max}$$

$$\omega_{MG1,min} < \omega_{MG1} < \omega_{MG1,max}$$

$$\omega_{MG2,min} < \omega_{MG2} < \omega_{MG2,max}$$

$$Soc_{min} < Soc < Soc_{max}$$

$$P_{batt,min} < P_{batt} < P_{batt,max}$$

To solve the DP problem, we start from the final time step N with a given cost value $G_N(x(N))$. By solving a finite number of optimization sub-problems backward from the final condition, the optimal control policy can be obtained based on the Bellman's optimality principle. To compute the optimal control policy for step N-1, we solve the following minimization problem:

$$V^*[x(N-1)] = \min_{u(N-1)} \{G_N\left(x(N)\right) + L(x(N), x(N-1), u(N-1))\} \quad (19)$$

By propagating backward for the steps $1 \leq k \leq N-2$, the optimal control policy and the cost-to-go function, $V^*[x(k)]$, can be obtained by solving:

$$V^*[x(k)] = \min_{u(k)}\{L\big(x(k+1), x(k), u(k)\big) + V^*[x(k+1)]\} \quad (20)$$

As can be realized, different optimal paths are obtained for the different initial values of the states. Therefore, the optimal path belonging to the intended initial state is selected as the ultimate optimal solution.

3.3 Selection of State and Decision Variables for DP

The PHEV model in this study consists of four state variables, i.e., ω_e, ω_{MG1}, ω_{MG2}, and Soc, and three control inputs, T_e, T_{MG1}, and T_{MG2}. Note that the vehicle speed, v, is directly related to ω_{MG2}. Since vehicle speed v is known resultantly ω_{MG2} is known and considering the kinematic constraints in eq.(1), we have just ω_e and Soc as the independent state variables. As far as the decision varables is concerned, because of the known demand power (known drive cycle), just two of the decision variables are independent. The decision of which variables and inputs to be discretized is an important step in DP. In [5] and [6], ω_e and Soc have been chosen as the independent state variables for forming the DP grid, and T_e and T_{MG1} are selected as the decision variables. The other variables can be calculated from the inverse powertrain dynamics and the selected state and decision variables. For a given set of decision variables at time step k, since the next time step states, ω_e(k+1) and Soc(k+1), might not land on a grid point, a 2-D interpolation is performed on the cost-to-go function based on the adjacent grid points. This interpolation might result in additional computational burden and accuracy degradation and constraint leakage if the DP grid size is not sufficiently small. In our algorithm, we penalize violation of the boundary constraints with hard penalization, hence linear interpolation may cause problem by mixing the constraint violation cost with the actual transition cost. As the consequence, this constraint leakage can propagate constraint violation costs through the state and time steps and eliminate the feasible paths that are being appeared as the infeasible paths [1].

To improve the accuracy, computational time of DP, and reduce the sensitivity of in this thesis, we propose to choose ω_e and Soc as the main state variables for the DP grid similar to the other references. However, we propose to choose T_e and ωek+1 as the decision variables. This way, ω_e is guaranteed to be locked on the DP grid throughout the DP computations, and only 1-D interpolation is needed for the cost-to-go calculation based on where the next step Soc lands on the grid. Ideally, it is desirable to choose ω_e (k+1) as one of the decision variables instead of T_e, to completely eliminate the need for the interpolation. However, due to the nonlinearity of the MG efficiency maps, the inverse model cannot be easily attained for the Soc dynamics. Therefore, the proposed combination of the state and input variables is one of the most effective ways to apply DP the PHEV power management optimization problem. Next, we discuss how to calculate all the required variables for the transition cost computation at each time step using the proposed set of DP variables. Figure 8 shows the schematic of the proposed DP algorithms. As it can be seen, in all time ω_e will be landed on the grids and we need just to do interpolation on Soc.

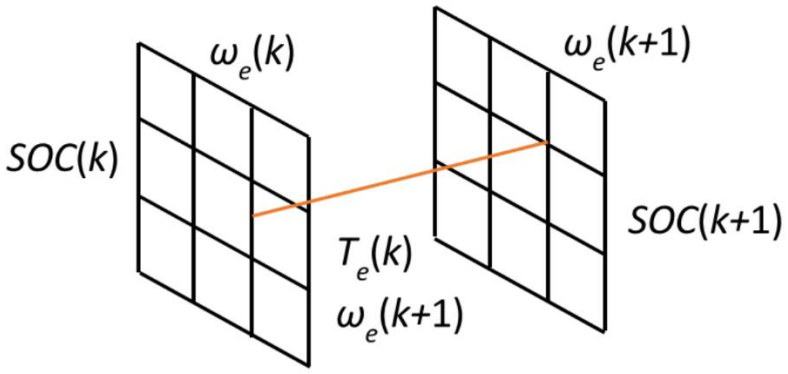


Figure 8 Schematic of proposed DP in one iteration

Assuming there is no tire slip, the speed of MG2 becomes proportional to the vehicle speed, and can be calculated from the drive cycle at each time step, as follows:

$$\omega_{MG2}(k) = \frac{K}{r}\, v(k), \;\; \omega_{MG2}(k+1) = \frac{K}{r}\, v(k+1) \tag{21}$$

With $\omega_e(\mathrm{k})$ and $\omega_e(\mathrm{k}+1)$ known from the DP grid, and $\omega_{MG2}(k)$ and $\omega_{MG2}(k+1)$ known a priori from the drive cycle, we can compute the speed of MG1 at time steps k and k+1 using the planetary gearbox kinematics relation presented in (2):

$$\omega_{MG1}(k) = \frac{(R+S)\omega_e(\mathrm{k}) - R\omega_{MG2}(k)}{S}, \tag{22}$$

$$\omega_{MG1}(k+1) = \frac{(R+S)\omega_e(\mathrm{k}+1) - R\omega_{MG2}(k+1)}{S}$$

Moreover, by knowing the value of T_e as a DP decision variable, and using the system dynamics from (2), we can obtain the planetary gear's internal force, F, as:

$$F(k) \cong \frac{T_e(k) - I_e\,(\omega_e(\mathrm{k}+1) - \omega_e(\mathrm{k}))/\Delta t}{(R+S)} \tag{23}$$

where Δt is the discretization time step, noting that the approximation error would diminish as $\Delta t \rightarrow 0$. The remaining control variables, i.e., T_{MG1} and T_{MG2} can be calculated from (2) and (3), as follows:

$$T_{MG1}(k) \cong I_{MG1}\left(\frac{\omega_{MG1}(k+1) - \omega_{MG1}(k)}{\Delta t}\right) - SF(k), \tag{24}$$

$$T_{MG2}(k) \cong I'_{MG2}\left(\frac{\omega_{MG2}(k+1) - \omega_{MG2}(k)}{\Delta t}\right) - RF(k) + F_{road}r/K.$$

The last unknown state, i.e., Soc(k+1), can be calculated from Sock on the DP grid, and using (9) as:

$$Soc(k+1) \cong \tag{25}$$

$$Soc(k) - \frac{V_{oc} - \sqrt{{V_{oc}}^2 - 4\bar{P}_{batt}(k)R_{batt}}}{2Q_{batt}R_{batt}}\Delta t.$$

where $\bar{P}_{batt}^{k}$ is calculated from (17). The obtained state values at time step k and k+1, and the control inputs at time step k, can be used to calculate the transition cost functions needed to carry out the DP. Next, we present a numerical simulation demonstrating the optimal PHEV powertrain trajectories obtained from the DP algorithm.

3.4 DP Simulation

To evaluate the proposed optimization algorithm, a full PHEV model is developed, with the engine fuel consumption rate, W_{fuel}, the MG efficiency maps, η_{MG1} and η_{MG2}, and the vehicle parameters adopted from the National Renewable Energy Laboratory's Advanced Vehicle Simulator (ADVISOR) data [4]. Table 1 lists some of the main PHEV powertrain specifications and parameter values.

The optimal powertrain trajectories are obtained by simulating the PHEV model in the forward direction based on the calculated optimal decision variables from the DP algorithm. For each grid point (ω_e, Soc) at time step k, the optimal engine torque, T_e*, and the next step engine speed, ω_e^{k+1}*, are obtained from the DP's stored information. For each time step k, the control input and state values for time step k as well as the state values for time step k+1 are calculated (from (21)-(25)). Since the resulting ω_e^{k+1}, Soc^{k+1} might not land on a grid point, a 2-D interpolation is performed to obtain the approximate optimal engine torque and the next step engine speed values for time step k+1. The process is continued until arriving at the final time step, N.

The EPA's FTP-75 drive cycle is adopted for the simulations of this study. This drive cycle is composed of the UDDS drive cycle followed by its first 505 seconds, as shown in Fig. 9 (a). The

obtained optimal Soc trajectories corresponding to this drive cycle and different starting Soc values are shown in Fig. 9 (b). The Soc range is set to vary between 0.3-0.8 to comply with the practical battery health-related constraints, with DP grid resolution of 0.005. As expected, the utilization of the battery energy is distributed throughout the drive cycle, with Soc trajectories ending near the lowest limit. The regenerative braking at the end of the drive cycle pushes the final Soc value to be slightly above the minimum limit.

Table I PHEV main specifications

Powertrain	**Parameter**	**Value**
Vehicle	EPA Classification	Midsize Sedan
	Base Curb Weight	1400 Kg
ICE	Displacement& Model	1.5L Prius (Atkinson cycle) engine
	Maximum Power	43kW @4000rpm
	Peak Torque	102 N.m @ 4000 rpm
Motor Generators	Type	Permanent Magnet AC
	Maximum MG1 Power	15-kW
	Maximum MG2 Power	30-kW
Battery Pack	Chemistry	NiMH
	Nominal Voltage	1.2 V per cell
	Nominal Capacity	6.0 A-h per cell
	number of cells in series per parallel string, n_s	240
	number of parallel strings, n_p	2
	Pack Energy Capacity	3.7 kWh

The trajectories that start from lower initial Soc values show intermittent battery charging cycles via the engine to provide energy cost saving at future high demand periods.

The optimal engine speed trajectories remain largely near the minimum speed for this drive cycle, while the engine torque varies broadly to deliver the demanded power.

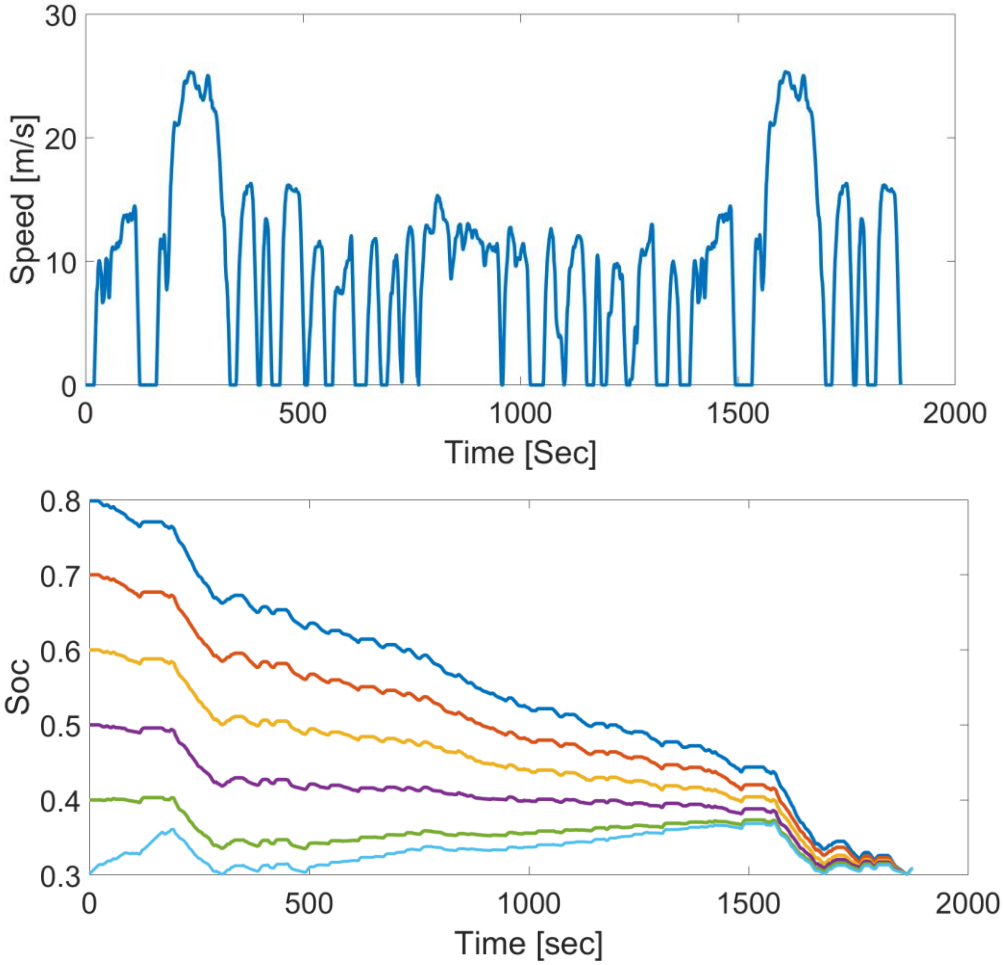


Figure 9. (a) The FTP-75 Drive Cycle, (b) Optimal Soc trajectories for different initial conditions and admissable Soc range of 0.3-0.8.

To show the overall effects of the powertrain speed and torque variations, we provide the power trajectories of the engine and MGs for a limited time period as shown in Fig. 10, for the starting Soc of 0.8. It is seen from this figure that, MG2 connected to the weels takes the most responsibility for meeting the power demand.

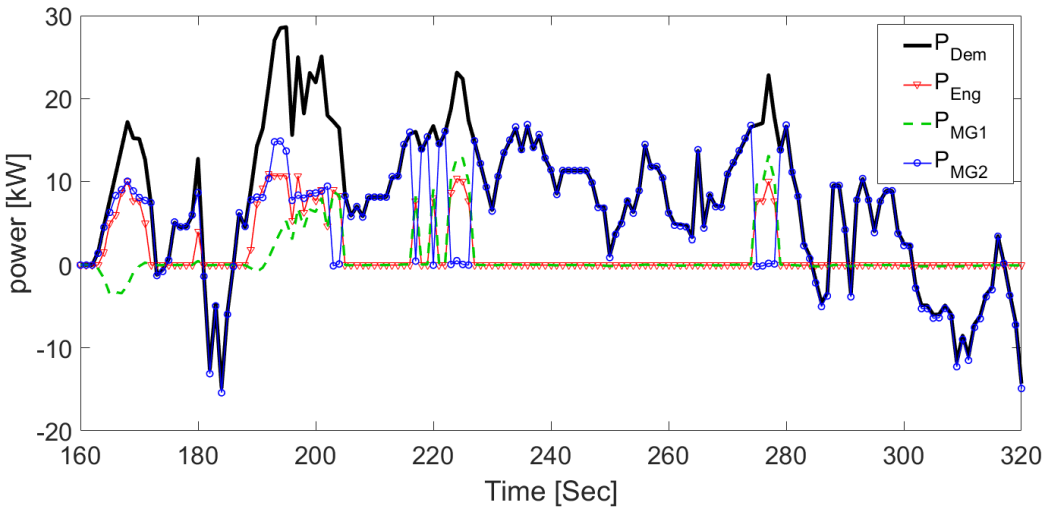
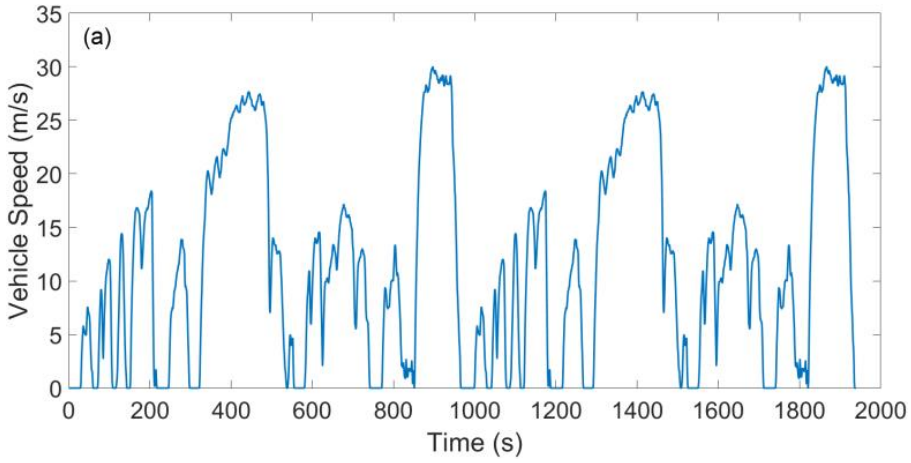


Figure 10 Contributions of the engine, MG1, and MG2 power to the total power demand obtained from DP

The EPA's "short" LA92 drive cycle is also selected for the simulation purpose. This drive cycle contains the first 969 seconds of the LA92 Unified Cycle Driving Schedule (UCDS). To extend the trip range to account for daily commute, we concatenate two back-to-back cycles as shown in Fig. 11a. For different initial SOC values, the optimal SOC trajectories and their corresponding equivalent fuel consumption trajectories are plotted and shown in Fig. 11b and 11c.



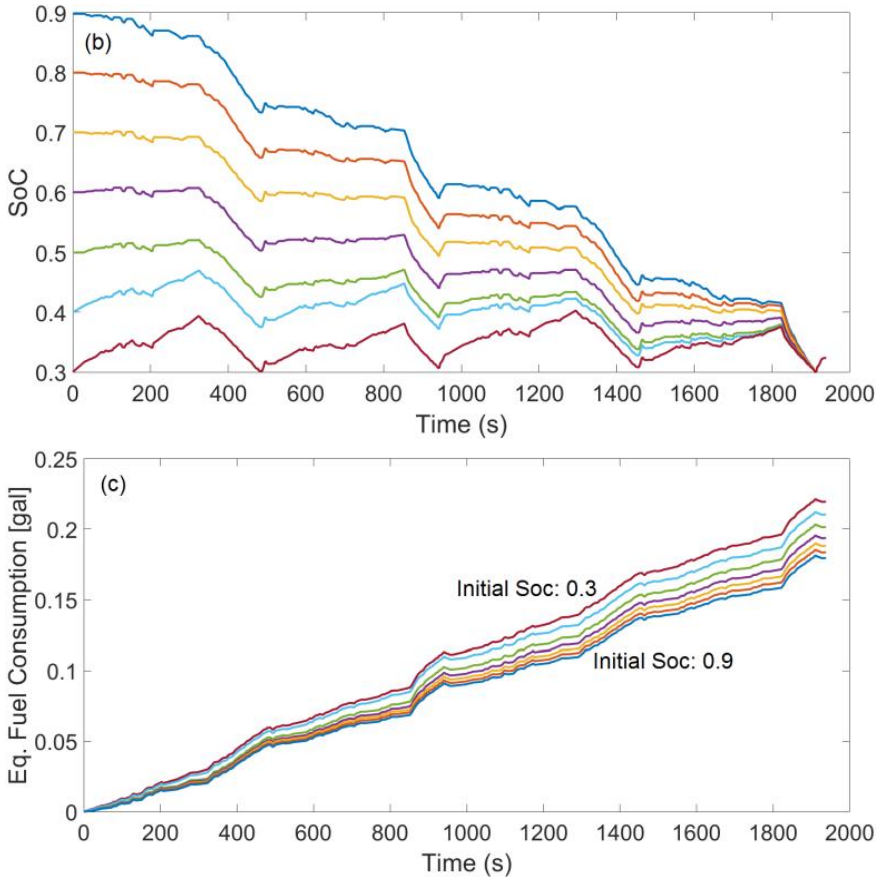


Figure 11 (a) Back-to-back short LA92 drive cycle, (b) Optimal SOC trajectories for different initial conditions and admissable SOC range of 0.3-0.9. (c) Equivalent fuel consumption trajectories.

The admissible SOC range for the simulation is considered to be within 0.3-0.9. As expected, the extraction of battery energy is distributed throughout the drive cycle with the optimal SOC trajectories ending near the lower SOC limit. The trajectories that start from lower initial SOC values undergo intermittent battery charging cycles via the engine to provide cost saving during high demand periods in the future.

Although DP is a powerful algorithm for obtaining the optimal powertrain trajectories of the nonlinear PHEV system studied here, it does not provide a straightforward solution for real-time control due to complexity, computational time, and the requirement of future information. In the next section, we present a novel real-time control method for the PHEV power management problem based on a set of linear polynomial functions approximating the optimal DP decisions.

Chapter 4: REAL TIME CONTROL POLICY DEVELOPMENT FOR PHEV

4.1 Training real time-control policy without trip length inclusion

In this section, we develop a set of real-time control policies that receive the instantaneous force demand (from the drive cycle and F_{road}) and the powertrain system states, and generate the control decisions for the powertrain system inputs, i.e., engine and MG torques. The control policies for Te, TMG1, and TMG2 are chosen to be linear functions of the demanded propulsion force and the system states as follows:

$$T_l(F_{dem}, v, \omega_e, \omega_{MG1}, Soc) = \lambda_{0,l} + \lambda_{1,l}F_{dem} + \lambda_{2,l}v + \lambda_{3,l}\omega_e + \lambda_{4,l}\omega_{MG1} + \lambda_{5,l}Soc \quad (26)$$

$$\forall\, l \in \{e, MG1, MG2\}$$

where λ_i, l's are the parameters of the control policy. Note that three separate policies must be formed for each power source in the system.

The demanded propulsion force, F_{dem}, is calculated from:

$$F_{dem} = m_{eq}\dot{v} + F_{road} \quad (27)$$

where m_{eq} is the equivalent mass of the vehicle obtained from the inverse dynamics of the PHEV powertrain as:

$$m_{eq} = m + \frac{(I_{MG2}\,K^2 + I_\omega)}{r^2} + \frac{R^2 I_e I_{MG1}}{(R+S)^2 I_{MG1} + S^2 I_e}\left(\frac{K}{r}\right)^2 \quad (28)$$

To obtain the optimal parameter values for the control policies, we train them using the obtained optimal DP trajectories. Therefore, we minimize the difference between the DP decisions and those formed by the control policies, through a least-square minimization problem formulated as follows:

$$\boldsymbol{\Lambda}_l^* = \underset{\boldsymbol{\Lambda}_l}{\operatorname{argmin}}\ (\mathbf{A}^*\boldsymbol{\Lambda}_l - \boldsymbol{T}_l^*)^T\,(\mathbf{A}^*\boldsymbol{\Lambda}_l - \boldsymbol{T}_l^*) \quad (29)$$

$$\forall\, l \in \{e, MG1, MG2\}$$

where

$$\mathbf{A}^* = \begin{bmatrix} 1 & F_{dem}(1) & v(1) & \omega_e^*(1) & \omega_{MG1}^*(1) & Soc^*(1) \\ 1 & F_{dem}(2) & v(2) & \omega_e^*(2) & \omega_{MG1}^*(2) & Soc^*(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & F_{dem}(N-1) & v(N-1) & \omega_e^*(N-1) & \omega_{MG1}^*(N-1) & Soc^*(N) \end{bmatrix} \quad (30)$$

$$\boldsymbol{T_l^*} = [T_l^*(1) \quad T_l^*(2) \quad \dots \quad T_l^*(N-1)]^T$$

$$\boldsymbol{\Lambda_l} = [\lambda_{0,l} \quad \lambda_{1,l} \quad \lambda_{2,l} \quad \lambda_{3,l} \quad \lambda_{4,l} \quad \lambda_{5,l}]^T$$

In (29) and (30), the optimal DP trajectories are stacked to form matrix A* and vector T_l*, as the input and output data for training the control policies. The optimality of the trained control policies need to be evaluated through numerical simulations. It is expected that by adding more terms to the control policy and including trip forecast information, the optimality of the real-time policy will be improved. In the next section, we evaluate the performance of the proposed control policies in approximating the optimal PHEV powertrain trajectories.

Figure 12 shows the schematic signal and power flow using the proposed supervisory control policy. In the first simulation, we form a set of linear maps for the supervisory engine and MG torque controllers based on DP data belonging to the initial Soc of 0.8. Figure 13 compares the optimal Soc and fuel consumption trajectories with those obtained from the real-time controllers. We can see that the trajectories obtained from the real-time policy follow the optimal trajectories very closely. The final fuel consumption value is 2.9% higher than the optimal value for this simulation.

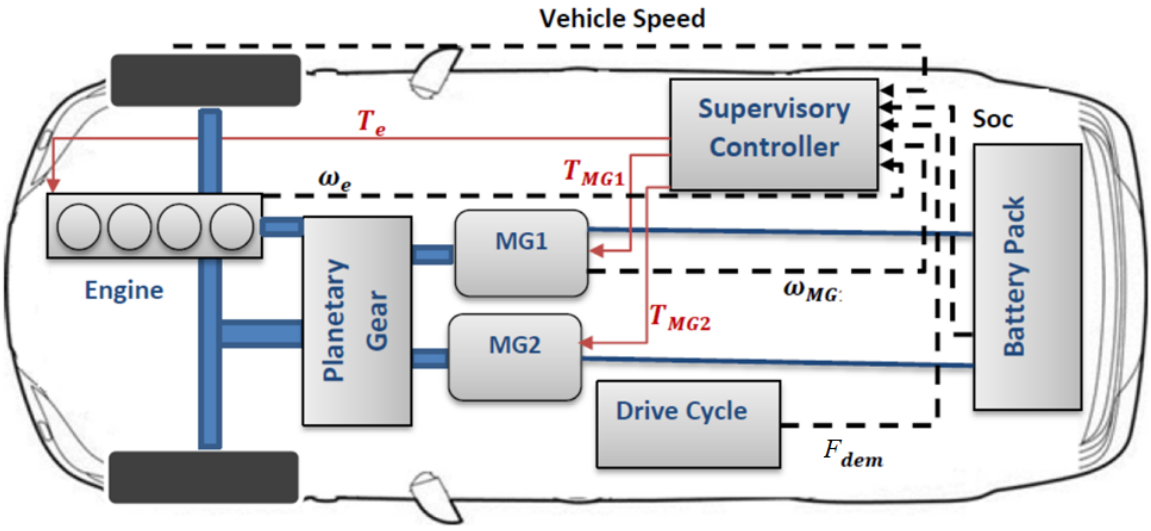


Figure 12 PHEV power and signal flow architecture under the real-time

One of the main drawbacks of the approximated control policy is its lack of control over the constraints. For example, when the torque controllers are trained for the starting Soc of 0.8, but used for a different starting Soc in practice, the Soc may drop significantly below the limit as shown in Figure 14.

To provide a remedy for this problem, we develop a switching controller, which transitions from one policy to another based on the states of the system. The simplest switching controller to mitigate the Soc constraint violation is the one that is trained based on at least two sets of optimal data, one belonging to the starting Soc of 0.8 (i.e., max Soc) and the other belonging to the starting Soc of 0.3 (i.e., min Soc). This way, we can maximize the likelihood of maintaining the Soc within its lower and upper limits.

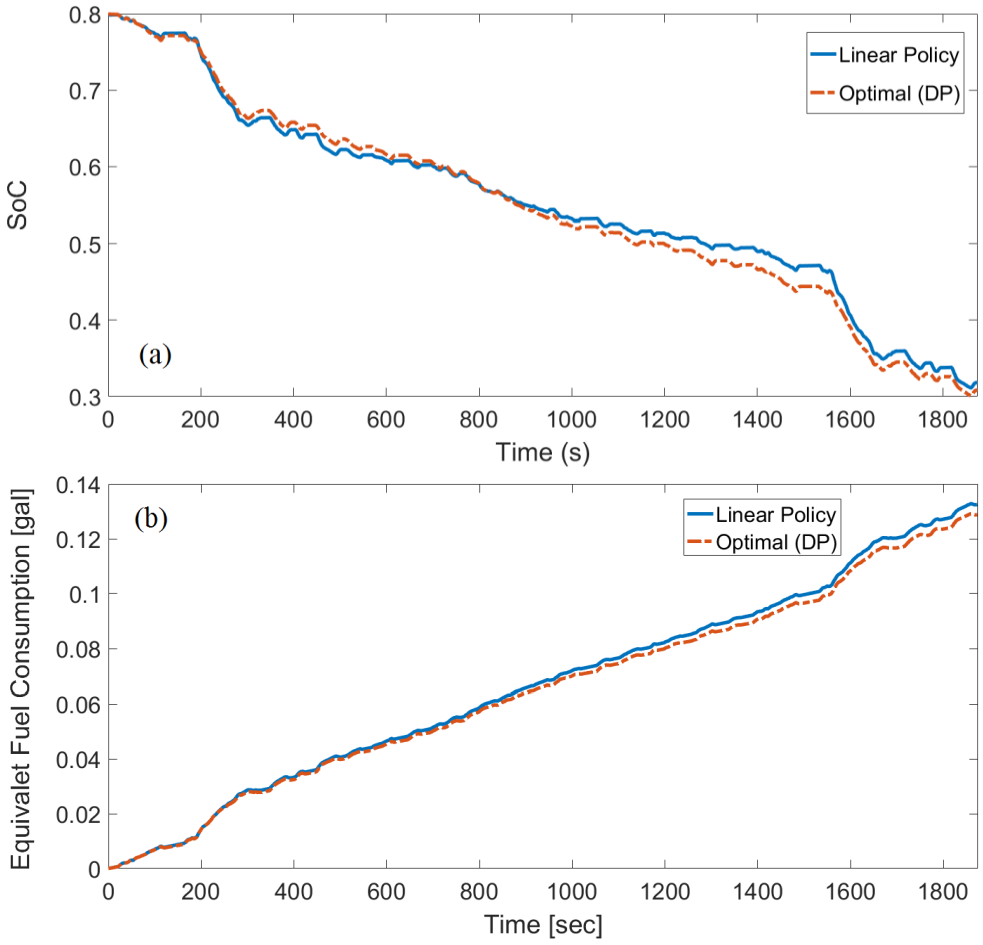


Figure 13 Comparison of the optimal the approximate linear policy trajectories: (a) Soc, and (b) Fuel consumption.

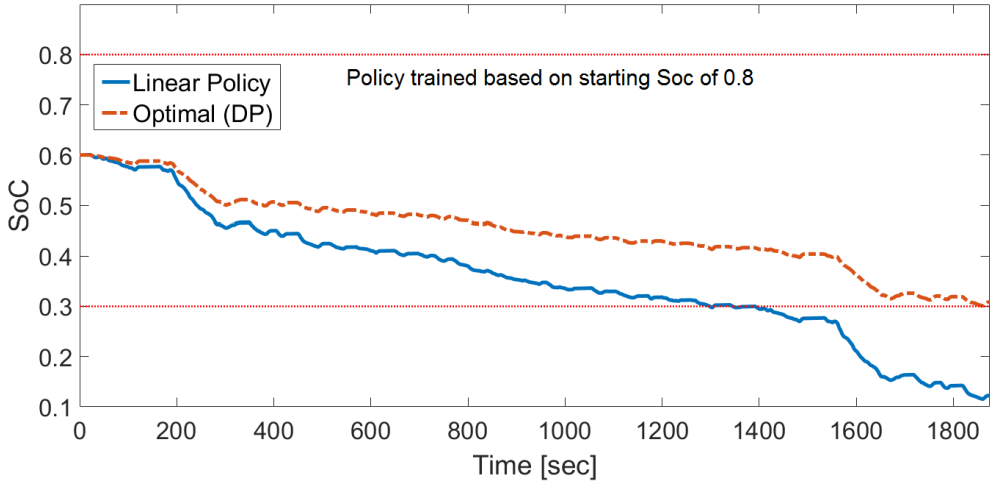
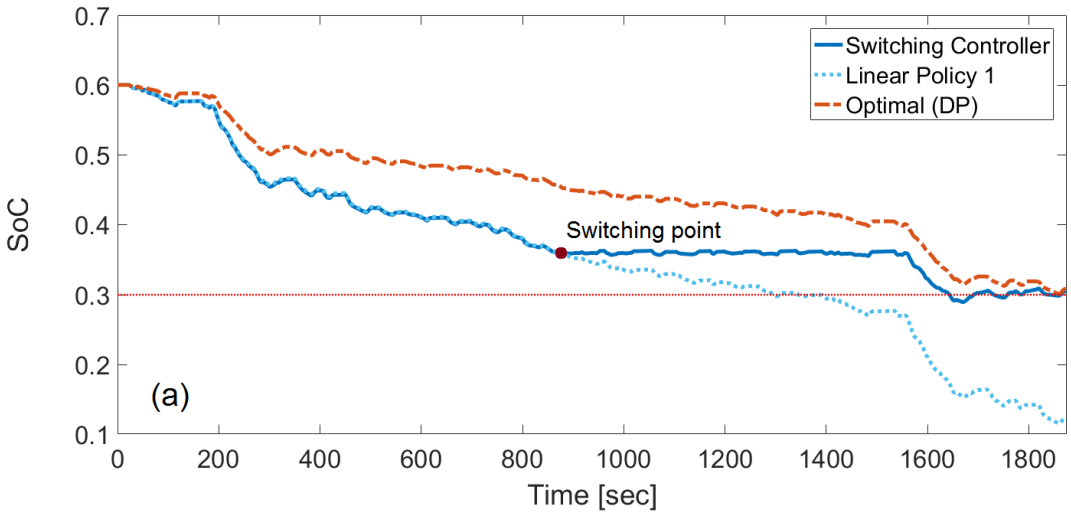


Figure 14 Constraint violation of the linear control policy in the presence of mismatch between the training and test conditions (e.g., initial Soc).

To include a margin for the Soc constraint management method, we set the controller to switch from Policy 1 to Policy 2 when the Soc drops below 0.36 (instead of 0.3). The simulation results

for the switching controller are shown in Fig. 15. The starting Soc is different from the ones used for training the policies. Therefore, the Soc trajectory deviates from the optimal trajectory obtained from the DP. However, once Soc drop below a certain threshold, the controller switches to the second policy which is inclined toward maintaining the Soc above the lower limit. The constraint violation of the resulting Soc trajectory is therefore significantly improved. Comparison of the final equivalent fuel consumption values indicate 4.5% gap between the switching controller and the optimal policy.

The methods and simulations presented in this paper demonstrate the potential of the proposed approach for developing highly effective sub-optimal real-time control policies for PHEVs. Additional research is needed to further improve the structure, robustness, and optimality of the method.



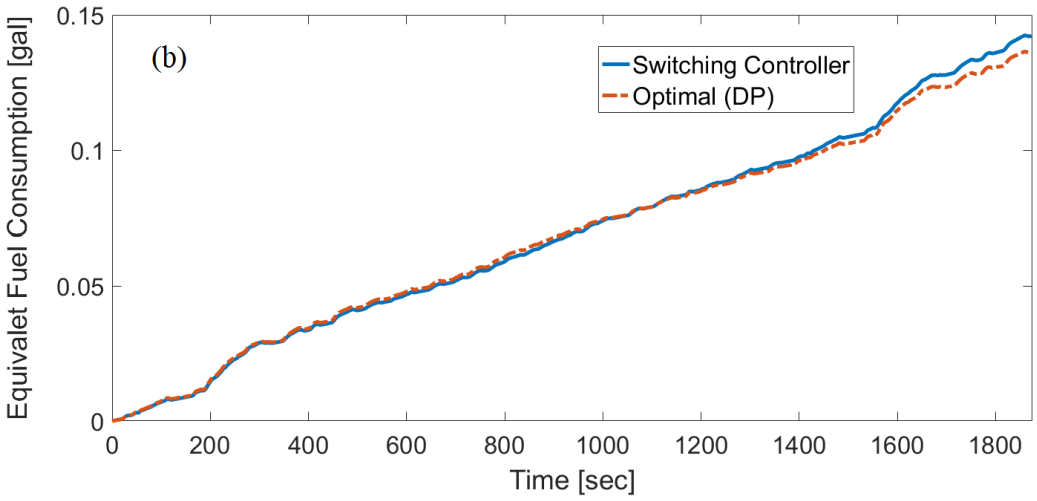


Figure 15 Comparison of the switching controller with the optimal and non-switching policy: (a) Soc, and (b) fuel cost.

4.2 Training real time-control policy with Trip Length Inclusion

In this section, we develop a policy-based data-driven controller for the real-time power management of the PHEV system. We extend our previous work, [22], by including a predictive element related to trip length in the control policies. We also improve the robustness of the control policies by training them over multiple drive cycles and different initial SOC conditions.

The proposed controller comprises three individual control policies that receive the instantaneous values of vehicle speed, demanded force (F_d), and powertrain system states, i.e., SOC and ω_e, and compute the required torque levels for the engine and MGs, as shown in Fig. 16. Note that the angular speeds of the MGs are dependent on, and can be computed from, v and ω_e through the kinematic constraint imposed by the planetary gear set. Therefore, they are omitted from the control policies to avoid redundancy.

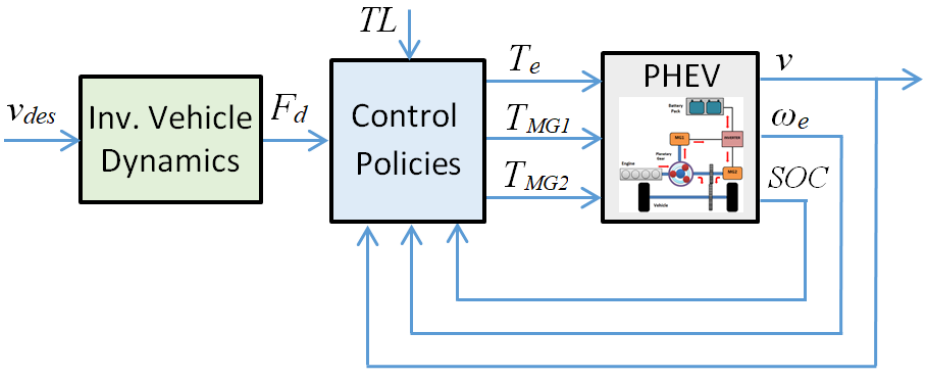


Figure 16 Proposed PHEV control system architecture.

Since the optimal usage of battery energy is dependent on the overall trip demand, we introduce another input variable to the control policy, i.e., the trip length (TL). This input can be set by the driver at the beginning of the trip, or estimated via an onboard embedded technology, based on the previous trip statistics and real-time GPS information. Note that the proposed method in this paper does not consider trip length information as a requirement for the control system, but rather as an auxiliary input with potential benefits if made available to the controller.

As the car drives on the road, the remaining trip length is updated by subtracting the distance traveled from its initial value as follows:

$$RTL(k) = TL - \sum_{n=1}^{k} v(n)\Delta t \tag{31}$$

where RTL stands for remaining trip length. This variable can be re-adjusted by the driver or the onboard estimator when there is a change of trip route or destination.

To develop the control policies, we choose T_e, T_{MG1} and T_{MG2} to be linear functions of the input signals:

$$T_j(F_d, v, \omega_e, SOC, RTL) = \tag{32}$$
$$\lambda_{0,j} + \lambda_{1,j}F_d + \lambda_{2,j}v + \lambda_{3,j}\omega_e + \lambda_{4,j}SOC + \lambda_{5,j}\,RTL$$
$$\forall\, j \in \{Eng., MG1, MG2\}$$

where $\lambda_{i,j}$'s are the coefficients of the control policies to be identified through a statistical learning process. Note that three separate policies must be formed for the three power sources in the system.

To obtain the optimal values of the control policy coefficients, we minimize the difference between the optimal decisions obtained from DP and those formed by the control policies through a least-squares minimization problem formulated as follows:

$$\boldsymbol{\Lambda}_j^* = \underset{\boldsymbol{\Lambda}_j}{\operatorname{argmin}} \left(\mathbf{A}^* \boldsymbol{\Lambda}_j - \boldsymbol{T}_j^*\right)^T \left(\mathbf{A}^* \boldsymbol{\Lambda}_j - \boldsymbol{T}_j^*\right) \quad (33)$$

$$\forall\, j \in \{Eng., MG1, MG2\}$$

where

$$\boldsymbol{\Lambda}_j = [\lambda_{0,j} \quad \lambda_{1,j} \quad \lambda_{2,j} \quad \lambda_{3,j} \quad \lambda_{4,j} \quad \lambda_{5,j}]^T \quad (34)$$

$$\boldsymbol{T}_j^* = [T_j^*(1) \quad T_j^*(2) \quad \dots \quad T_j^*(M)]^T$$

$$\mathbf{A}^* = \begin{bmatrix} 1 & F_d(1) & v(1) & \omega_e^*(1) & SOC^*(1) & RTL(1) \\ 1 & F_d(2) & v(2) & \omega_e^*(2) & SOC^*(2) & RTL(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & F_d(M) & v(M) & \omega_e^*(M) & SOC^*(M) & RTL(M) \end{bmatrix}$$

In Eq. (16), the optimal DP trajectories are stacked to form matrix $\mathbf{A}^*$ and vector $\boldsymbol{T}_j^*$, as the input-output training data for the control policies. The number of rows of $\mathbf{A}^*$ and $\boldsymbol{T}_j^*$ (i.e., M) would depend on the number of drive cycles in the training set, the size of each drive cycle, and the number of the initial conditions include in the training data.

In the next section, we evaluate the performance of the proposed control strategy in approximating the optimal PHEV powertrain decisions.

To evaluate the proposed control framework, a number of urban and highway drive cycles are adopted here for training the control policies. Each drive cycle is separately optimized using DP for

different initial SOC values. The resulting data are stacked based on (16) to form the input and output data matrices, $\mathbf{A}^*$ and $\boldsymbol{T}_j^*$. The control policies are developed by solving the linear regression problem formulated in (15).

Simulations presented in [22] show that if the training and test drive cycles and initial SOC values are identical, the proposed control policy provides a close approximation to the DP decisions. In this paper, we consider a more practical scenario in which the control policies are trained for a completely different set of drive cycles from the one used for the validation. The drive cycles used for training the control policies include HWFET, FTP-75, SC03, and LA92, optimized for initial SOC values of 0.3, 0.4, … , 0.9. The proposed controller is evaluated on the repeated short LA92 drive cycle shown previously in Fig. 11.

Figure 17 shows the SOC trajectories resulted from simulating the controllers trained based on the described training scenario. Two different controllers are developed, one blind to the trip length information (Fig. 17(a)), and the other one aware of the trip length input (Fig. 17(b)). Since the drive cycles in the training data are more aggressive on average compared to the LA92 drive cycle, the SOC depletion rate is faster than optimal in both control scenarios. The SOC trajectories drop below the specified lower limit of 0.3. The controller with RTL input, however, provides a much closer approximation to the optimal DP results by adjusting the battery depletion rate according to the remaining trip demand.

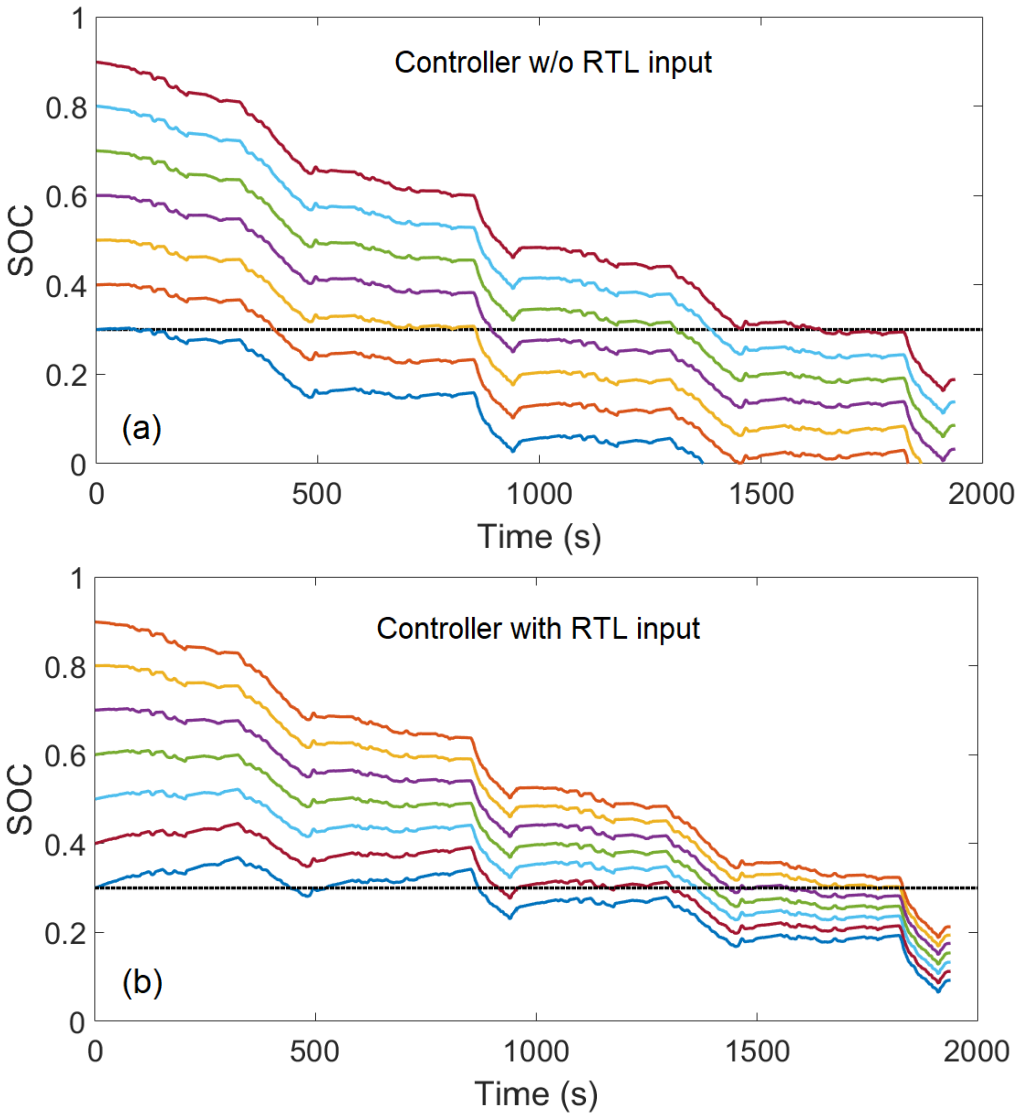


Figure 17 SOC trajectories resulted from the real-time control policy: (a) without and (b) with RTL input.

To address the SOC constraint violation problem using the proposed control method, a constraint management algorithm is added to the control system. This algorithm transfers the electrical drivetrain load to the engine when the SOC drops below the minimum threshold. The balance of power supply and demand is used to compute the adjusted values of the engine and MG torques during the active constraint management periods to ensure that the overall trip demand is met by the hybrid powertrain system.

Table II SOC constraint management algorithm

- Tmg_Switch=.8;
- if SOC(k) > SOC_min
- Use the designed online controller
- End
- if SOC(k) < SOC_min
- if TorqMg1(k)*SpMotoMg1(k)> 0 && TorqMg2 (k)*SpMotoMg2(k) > 0
- TorqMg1P=TorqMg1(k)*TmgSwitch;
- TorqMg2P=TorqMg2(k)*TmgSwitch;
- Compute the TorqEng(k)by balancing the demand and supply powers using TorqMg1P and TorqMg2P
- TorqMg1 (k) = TorqMg1P;
- TorqMg2 (k) = TorqMg2P;
- elseif TorqMg1 (k)*SpMotoMg1(k) > 0
- TorqMg1P=TorqMg111(k)*TmgSwitch;
- Compute the TorqEng(k)by balancing the demand and supply powers using TorqMg1P
- TorqMg1 (k) = TorqMg1P;
- elseif TorqMg222(k)*SpMotoMg2(k) > 0
- TorqMg2P=TorqMg222(k)*TmgSwitch;
- Compute the TorqEng(k)by balancing the demand and supply powers using TorqMg2P
- TorqMg2 (k) = TorqMg2P;
- end
- End

Figure 18 shows an example of real-time PHEV control under the SOC constraint management algorithm. As can be seen, in both control schemes (with and without RTL inputs) the SOC is maintained above the specified lower limit. The drive cycle tracking error remains below 1% of the maximum vehicle speed, which shows the constraint management algorithm does not impact the vehicle driving performance. The energy cost trajectories in Fig. 18(b) indicate that the proposed control scheme is able to keep the gap between the real-time and the optimal fuel consumption

trajectories in a narrow range, particularly, when the trip length information is available to the controller.

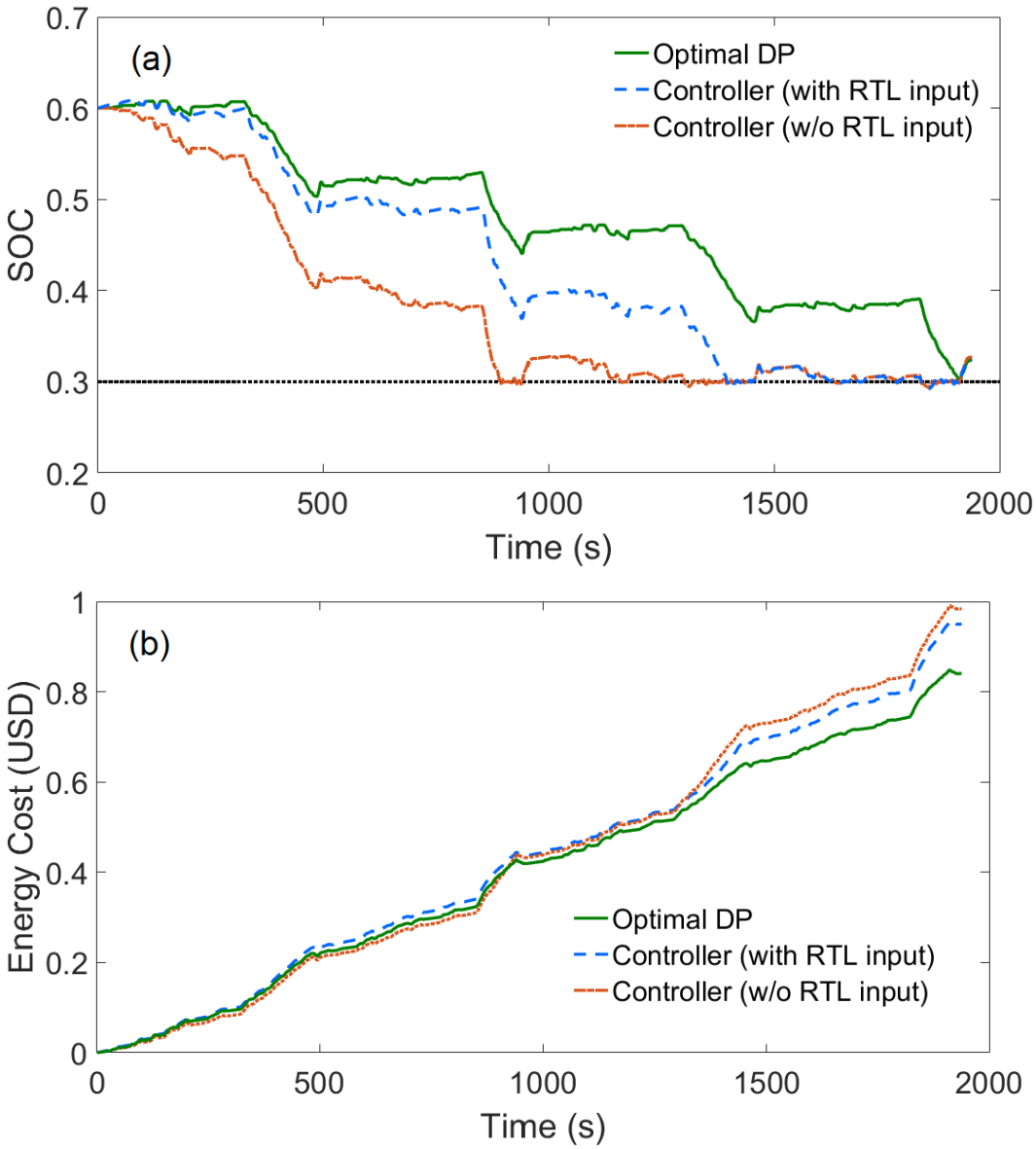


Figure 18. Constraint management for battery SOC: (a) SOC trajectories and (b) cumulative energy cost trajectories.

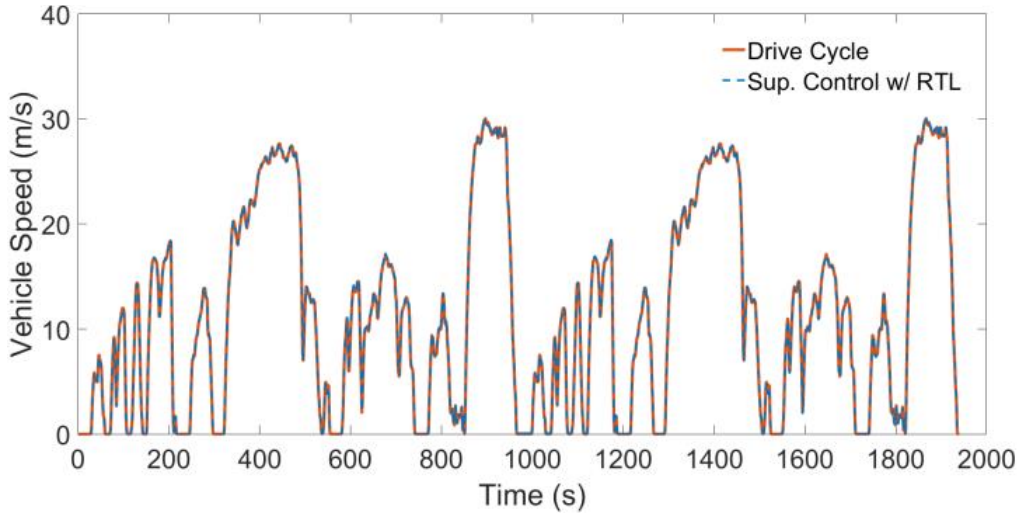


Figure 19. Drive cycle tracking using the supervisory control policy with RTL input.

To evaluate the speed control performance, Figure 19 shows the drive cycle tracking using the supervisory controller with RTL input. As can be seen, the speed control objective is accomplished with significant accuracy, with the error being bounded within ±0.1 m/s.

To extend our perspective on the proposed control strategy, we present another simulation for the case of using a drive cycle from the training data for evaluating the controller. Figure 20 shows the comparison of the optimal and real-time SOC trajectories for the FTP-75 drive cycle, obtained from DP and the proposed control strategy, respectively. As can be seen, the controller is less aggressive for this drive cycle, thereby resulting in a lower overall battery depletion at the end of the drive cycle. Nevertheless, the SOC constraint management algorithm is activated during some time intervals to maintain the SOC within the allowable limit. The similarity between the optimal and the real-time SOC trajectories is more pronounced in this example.

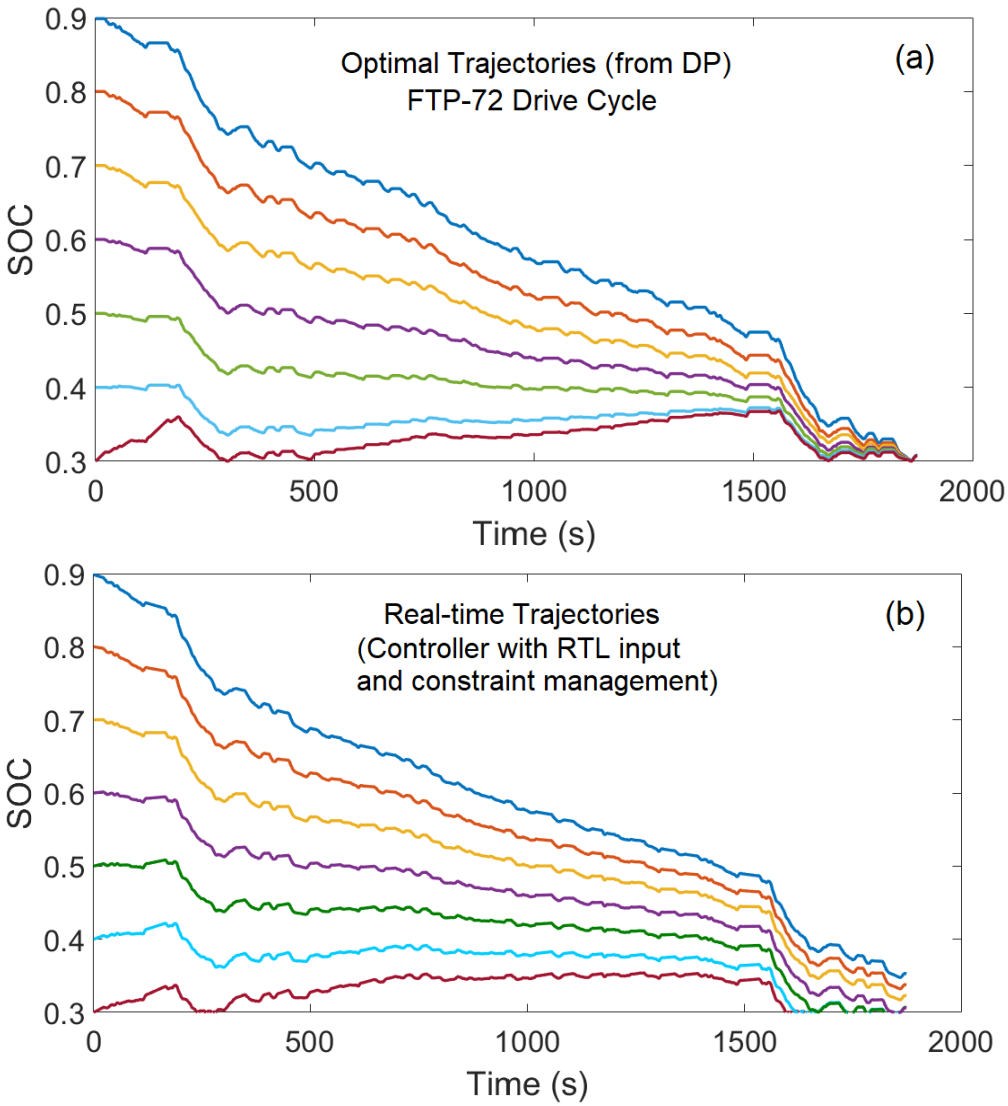


Figure 20. Comparison of the optimal and real-time SOC trajectories for a drive cycle inside the training set.

4.3 Quantitative Optimality comparison

Table III tabulates the cost of energy incurred on top of the optimal cost, for the different drive cycles studied in this thesis. The first three drive cycles are included in the training set, whereas the last drive cycle is used only for validations. It is concluded from the table that the inclusion of trip length input in the control policies can improve fuel economy of the PHEV by about 1.1% for the drive cycles inside the training set, and by 2.9% for the drive cycle outside the training set. Although

these numbers are specific to the simulation settings of this study, they provide important insights into the PHEV power management problem in practice.

Table III average energy cost addition to the optimal cost for different drive cycles inside and outside training set.

Drive Cycle	Control Policy w/o RTL input	Control Policy with RTL input
HWFET	9.7%	8.4%
FTP75	12.2%	11.3%
3× SC03	13.0%	11.5%
2× LA92 Short	16.9%	14.0%

CONCLUSION

In this thesis, we investigated the application of Dynamic Programming in developing a simple, yet effective class of real-time control policies for PHEVs. Using a power-split PHEV model, a DP problem was formulated that used a combination of two states and two decision variables to find the optimal powertrain trajectories in an accurate and computationally efficient manner. To further improve the accuracy and robustness of the optimization, we suggested using the average fuel consumption and battery power values for calculating the transition cost within each time step. The DP simulations results indicated that optimal strategy is to allocate the utilization of battery energy over the course of the drive cycle, as anticipated.

After obtaining the optimal power train trajectories using the method of dynamic programming applied to a power split PHEV model, the optimal trajectories were then applied in the form of input/output data to train a set of near-optimal control policies for the PHEV engine and motor/generators (MGs). The input data comprise (i) the powertrain-level information such as the angular velocities of the engine and the MGs, as well as the battery Soc, (ii) driver-level force demand information, and (iii) trip-level remaining trip length (RTL) data. Comparison with the optimal DP indicates that the proposed control method reaches a near optimal performance. However, the system state constraints might be violated due to the blindness of the controller with respect to them. Constraint violation can be mitigated using a switching controller, or improving he structure of the control policy and training methods. Further research is required to develop fully robust near-optimal real-time control policies for the PHEVs.

FUTURE WORKS

In future work, the following is recommended:

1. Modeling engine on-off in the dynamics of the PHEV and DDP formulation
2. Developing an effective learning method for the On and Off command which is a map from continuous to discrete space.
3. Considering the information like acceleration, deceleration, stop time, etc to train more precise online policy.
4. Considering higher orders terms and nonlinearities in the learning process to come up with more precise online policy
5. Applying the real time control policy for PHEVs' power management in the connected vehicles fuel economy problems

REFERENCES

[1] T. R. Montgomery, *Computationally Efficient Deterministic Dynamic Programming for Optimal Supervisory Power Management of Power-split PHEVs*. Pennsylvania State University, 2013.

[2] J. Liu and H. Peng, "Modeling and control of a power-split hybrid vehicle," *IEEE transactions on control systems technology*, vol. 16, no. 6, pp. 1242-1251, 2008.

[3] A. Sciarretta and L. Guzzella, "Control of hybrid electric vehicles," *IEEE Control systems*, vol. 27, no. 2, pp. 60-70, 2007.

[4] S. J. Moura, 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.

[5] S. J. Moura, "Plug-In Hybrid Electric Vehicle Power Management: Optimal Control and Battery Sizing," The University of Michigan, 2008.

[6] R. M. Patil, J. C. Kelly, Z. Filipi, and H. K. Fathy, "A framework for the integrated optimization of charging and power management in plug-in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 6, pp. 2402-2412, 2013.

[7] Y. He, M. Chowdhury, P. Pisu, and Y. Ma, "An energy optimization strategy for power-split drivetrain plug-in hybrid electric vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 22, pp. 29-41, 2012.

[8] J. Zhang and T. Shen, "Real-Time Fuel Economy Optimization With Nonlinear MPC for PHEVs," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 6, pp. 2167-2175, 2016.

[9] D. Kum, H. Peng, and N. K. Bucknor, "Supervisory control of parallel hybrid electric vehicles for fuel and emission reduction," *Journal of dynamic systems, measurement, and control*, vol. 133, no. 6, p. 061010, 2011.

[10] S. Kitayama, M. Saikyo, Y. Nishio, and K. Tsutsumi, "Torque control strategy incorporating charge torque and optimization for fuel consumption and emissions reduction in parallel hybrid electric vehicles," *Structural and Multidisciplinary Optimization*, vol. 54, no. 1, pp. 177-191, 2016.

[11] D. Kum, H. Peng, and N. K. Bucknor, "Optimal energy and catalyst temperature management of plug-in hybrid electric vehicles for minimum fuel consumption and tail-pipe emissions," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 1, pp. 14-26, 2013.

[12] X. Hu, C. M. Martinez, and Y. Yang, "Charging, power management, and battery degradation mitigation in plug-in hybrid electric vehicles: a unified cost-optimal approach," *Mechanical Systems and Signal Processing*, vol. 87, pp. 4-16, 2017.

[13] F. Martel, Y. Dubé, S. Kelouwani, J. Jaguemont, and K. Agbossou, "Long-term assessment of economic plug-in hybrid electric vehicle battery lifetime degradation management through near optimal fuel cell load sharing," *Journal of Power Sources*, vol. 318, pp. 270-282, 2016.

[14] L. Serrao, S. Onori, and G. Rizzoni, "ECMS as a realization of Pontryagin's minimum principle for HEV control," in *American Control Conference, 2009. ACC'09.*, 2009, pp. 3964-3969: IEEE.

[15] B. Geng, J. K. Mills, and D. Sun, "Energy management control of microturbine-powered plug-in hybrid electric vehicles using the telemetry equivalent consumption minimization strategy," *IEEE transactions on Vehicular Technology*, vol. 60, no. 9, pp. 4238-4248, 2011.

[16] P. Tulpule, V. Marano, and G. Rizzoni, "Energy management for plug-in hybrid electric vehicles using equivalent consumption minimisation strategy," *International Journal of Electric and Hybrid Vehicles*, vol. 2, no. 4, pp. 329-350, 2010.

[17] Q. Gong, Y. Li, and Z.-R. Peng, "Trip-based optimal power management of plug-in hybrid electric vehicles," *IEEE Transactions on vehicular technology*, vol. 57, no. 6, pp. 3393-3401, 2008.

[18] A. Brahma, Y. Guezennec, and G. Rizzoni, "Optimal energy management in series hybrid electric vehicles," in *American Control Conference, 2000. Proceedings of the 2000*, 2000, vol. 1, no. 6, pp. 60-64: IEEE.

[19] M. P. O'Keefe and T. Markel, "Dynamic programming applied to investigate energy management strategies for a plug-in HEV," National Renewable Energy Laboratory Golden, Colorado, USA2006.

[20] X. Wang, H. He, F. Sun, and J. Zhang, "Application study on the dynamic programming algorithm for energy management of plug-in hybrid electric vehicles," *Energies*, vol. 8, no. 4, pp. 3225-3244, 2015.

[21] L. Guzzella and A. Sciarretta, *Vehicle propulsion systems*. Springer, 2007.

[22] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503-515, 1954.

[23] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European Journal of Operational Research*, vol. 196, no. 2, pp. 509-515, 2009.
[24] L. Johannesson, M. Asbogard, and B. Egardt, "Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 71-83, 2007.
[25] J. A. Chekan and S. Bashash, "Dynamic programming-based approximate real-time control policies for plug in hybrid electric vehicles," in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, 2017, pp. 205-210: IEEE.
[26] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491-500, 1986.
[27] D. G. Wilson, R. D. Robinett, and G. R. Eisler, "Discrete dynamic programming for optimized path planning of flexible robots," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, vol. 3, pp. 2918-2923: IEEE.
[28] K. Merat, J. A. Chekan, H. Salarieh, and A. Alasty, "Control of Discrete Time Chaotic Systems via Combination of Linear and Nonlinear Dynamic Programming," *Journal of Computational and Nonlinear Dynamics*, vol. 10, no. 1, p. 011008, 2015.
[29] J. A. Chekan, M. Ali Nojoumian, K. Merat, and H. Salarieh, "Chaos control in lateral oscillations of spinning disk via linear optimal control of discrete systems," *Journal of Vibration and Control*, vol. 23, no. 1, pp. 103-110, 2017.
[30] A. De Madrid, S. Dormido, and F. Morilla, "Reduction of the dimensionality of dynamic programming: A case study," in *American Control Conference, 1999. Proceedings of the 1999*, 1999, vol. 4, pp. 2852-2856: IEEE.
[31] C. Sun, S. J. Moura, X. Hu, J. K. Hedrick, and F. Sun, "Dynamic traffic feedback data enabled energy management in plug-in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1075-1086, 2015.
[32] C. Sun, F. Sun, X. Hu, J. K. Hedrick, and S. Moura, "Integrating traffic velocity data into predictive energy management of plug-in hybrid electric vehicles," in *American Control Conference (ACC), 2015*, 2015, pp. 3267-3272: IEEE.
[33] K. Merat, J. A. Chekan, H. Salarieh, and A. Alasty, "Online Hybrid Model Predictive Controller Design for Cruise Control of Automobiles," in *ASME 2017 Dynamic Systems and Control Conference*, 2017, pp. V001T44A003-V001T44A003: American Society of Mechanical Engineers.
[34] S. D. Cairano, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat, "Model predictive control of magnetically actuated mass spring dampers for automotive applications," *International Journal of Control*, vol. 80, no. 11, pp. 1701-1716, 2007.
[35] F. Yan, J. Wang, and K. Huang, "Hybrid electric vehicle model predictive control torque-split strategy incorporating engine transient characteristics," *IEEE transactions on vehicular technology*, vol. 61, no. 6, pp. 2458-2467, 2012.
[36] H. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, I. V. Kolmanovsky, and S. Di Cairano, "MPC-based energy management of a power-split hybrid electric vehicle," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 593-603, 2012.
[37] G. Ripaccioli, D. Bernardini, S. Di Cairano, A. Bemporad, and I. Kolmanovsky, "A stochastic model predictive control approach for series hybrid electric vehicle power management," in *American Control Conference (ACC), 2010*, 2010, pp. 5844-5849: IEEE.
[38] S. J. Moura, D. S. Callaway, H. K. Fathy, and J. L. Stein, "Impact of battery sizing on stochastic optimal power management in plug-in hybrid electric vehicles," in *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, 2008, pp. 96-102: IEEE.
[39] J. A. Chekan, S. Taheri, S. Bashash, " A Data-driven Control Strategy for Trip Length-Conscious Power Management of Plug-in Hybrid Electric Vehicles, Submitted
[40] Advanced Vehicle Simulator, "National Renewable Energy Laboratory," Golden, CO, 2005 [Online]. Available: https://sourceforge.net/projects/adv-vehicle-sim/
[41] Online available in https://www.cs.rit.edu CSCI-351.

APPENDIX A

A.1 DYNAMIC PROGRAMMING CODE

```
clear all
clc
% tic

%% DP Parameters
load('1874.txt')
% load('drivecycle28.mat')
iter = 0.0005;
% Ssoc = 0.3:iter:0.8;
Ssoc = 0.27:iter:0.9;
Se = linspace(1000*(2*pi/60),2500*(2*pi/60),40);
dt=1;
Beta = 0.8;
NgridMG1 = 300;
NgridMG2 = 200;

%% Vehicle Parameters
Q_bat = 6.0*2;

up_P_batt=2.7e+04;
lo_P_batt=-100e+04;

I_e = 0.18;
I_mg1 = 0.023;
I_mg2 = 0.023;
R = 78;
S = 30;
r = 0.287;
m = 1254;
I_w = 181/2.205*r^2/2;
K = 3.93;
b_w = 0.015;
C_d = 0.3;
Rou = 1.2;
A_fr = 1.746;
mu = 0.02;
ete_grid = 0.98;
alfa_fuel = 22/453.592;
alfa_elec = 10^-6;

Ip_mg2 = I_mg2+(I_w+m*r^2)/K^2;
Iner = [I_e 0 0 (S+R);0 I_mg1 0 -S;0 0 Ip_mg2 -R;-(R+S) S R 0];
B = inv(Iner);
% Vel=[Drive_Cycle27(:,2)]';
% V1=[0; Drive_Cycle27(:,2)];
% time_size=length(Drive_Cycle27(:,1));
Vel = 0.447*[X1874(:,2)]';
V1 = 0.447*[0; X1874(:,2)];
time_size = length(X1874(:,1));
mm = time_size;

R_batP_vec  = 0.5*Ssoc.^2-0.551*Ssoc+0.687;
R_batN_vec  = 0.1298*Ssoc.^2-0.1387*Ssoc+0.43;
valV_oc_vec = 0.7238*Ssoc.^2+19.8181*Ssoc+297.107;

Sp = Vel;
dtt = 1;
```

```
for i=1:mm-1
    a(i)=(Sp(i+1)-Sp(i))/dtt;
    Av_Vel(i)=(Vel(i)+Vel(i+1))/2;
    P_dem(i)=m*a(i)*Sp(i)+0.5*Rou*A_fr*C_d*Sp(i)^3+mu*m*Sp(i)*9.81+b_w*(Sp(i)^2)/r;
end

Vel=Sp;
% MG1
gc_map_trq_mg1=[-55 -45 -35 -25 -15 -5 0 5 15 25 35 45 55];
% (rad/s), speed vector corresponding to rows of efficiency & loss maps
gc_map_spd_mg1=[-5500 -4000 -3500 -3000 -2500 -2000 -1500 -1000 -500 0 500 1000 1500
2000 2500 3000 3500 4000 5500]*(2*pi)/60;
% data reported was from 500 rpm to 4000 rpm, values for 0 and 5500 rpm are identical
% to nearest neighbors. Map was mirrored for negative values
gc_max_trq=[26 36 41 48 55 55 55 55 55 55 55 55 55 55 55 48 41 36 26];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LOSSES AND EFFICIENCIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%multiply everything by 0.95 for power electronics efficiency
%data in brackets is electric machine only as tested by UQM.

gc_eff_map_mg1=0.95*[...
0.88    0.89    0.9    0.91   0.9    0.79   0.79/2  0.79    0.9     0.91     0.9
0.89    0.88
0.88    0.89    0.9    0.91   0.9    0.79   0.79/2  0.79    0.9     0.91     0.9
0.89    0.88
0.87    0.88    0.9    0.9    0.9    0.8    0.8/2   0.8     0.9     0.9      0.9
0.88    0.87
0.85    0.87    0.89   0.9    0.9    0.81   0.81/2  0.81    0.9     0.9     0.89
0.87    0.85
0.83    0.85    0.87   0.89   0.89   0.82   0.82/2  0.82    0.89    0.89    0.87
0.85    0.83
0.8     0.83    0.85   0.87   0.89   0.82   0.82/2  0.82    0.89    0.87    0.85
0.83    0.8
0.76    0.79    0.82   0.85   0.87   0.82   0.82/2  0.82    0.87    0.85    0.82
0.79    0.76
0.68    0.72    0.76   0.8    0.84   0.81   0.8/2   0.81    0.84    0.8     0.76
0.72    0.68
0.52    0.57    0.63   0.69   0.77   0.8    0.8/2   0.8     0.77    0.69    0.63
0.57    0.52
0.52/2  0.57/2  0.63/2 0.69/2 0.77/2 0.8/2  0.8/3   0.8/2   0.77/2  0.69/2  0.63/2
0.57/2  0.52/2
0.52    0.57    0.63   0.69   0.77   0.8    0.8/2   0.8     0.77    0.69    0.63
0.57    0.52
0.68    0.72    0.76   0.8    0.84   0.81   0.8/2   0.81    0.84    0.8     0.76
0.72    0.68
0.76    0.79    0.82   0.85   0.87   0.82   0.82/2  0.82    0.87    0.85    0.82
0.79    0.76
0.8     0.83    0.85   0.87   0.89   0.82   0.82/2  0.82    0.89    0.87    0.85
0.83    0.8
0.83    0.85    0.87   0.89   0.89   0.82   0.82/2  0.82    0.89    0.89    0.87
0.85    0.83
0.85    0.87    0.89   0.9    0.9    0.81   0.81/2  0.81    0.9     0.9     0.89
0.87    0.85
0.87    0.88    0.9    0.9    0.9    0.8    0.8/2   0.8     0.9     0.9     0.9
0.88    0.87
0.88    0.89    0.9    0.91   0.9    0.79   0.79/2  0.79    0.9     0.91    0.9
0.89    0.88
0.88    0.89    0.9    0.91   0.9    0.79   0.79/2  0.79    0.9     0.91    0.9
0.89    0.88]';

[spd_mg1,trq_mg1]=meshgrid(gc_map_spd_mg1,gc_map_trq_mg1);
```

```
%%%%% ADVISOR data file: MG2
mc_map_trq_mg2=[-305 -275 -245 -215 -185 -155 -125 -95 -65 -35 -5 0 5   35  65  95
125 155 185 215 245 275 305];
mc_map_spd_mg2=[0 500 1000 1500 2000 2500 3000 3500 4000 4500  6000]*(2*pi)/60;
%pushed data out from 4500 rpm to 6000 rpm
mc_max_trq_data=[305.0 305.0 305.0 305.0 305.0 244.0 203.3 174.3 152.5 135.6 122.0
110.9 101.7 93.8 87.1 81.3 76.3 71.8 67.8 47.7];
mc_spd_data=[0 235 470 705 940 1175 1410 1645 1880 2115 2350 2585 2820 3055 3290 3525
3760 3995 4230 6000]*(2*pi)/60;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LOSSES AND EFFICIENCIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% multiplied by 0.95 because data was for motor only, .95 accounts for
inverter/controller efficiencies
mc_eff_map_mg2=0.95*[...
0.56/2  0.59/2  0.62/2  0.65/2  0.68/2  0.72/2  0.76/2  0.8/2   0.85/2  0.9/2   0.87/2
0.87/3  0.87/2  0.9/2   0.85/2  0.8/2   0.76/2  0.72/2  0.68/2  0.65/2  0.62/2  0.59/2
0.56/2
0.56    0.59    0.62    0.65    0.68    0.72    0.76    0.8     0.85    0.9     0.87
0.87/2  0.87    0.9     0.85    0.8     0.76    0.72    0.68    0.65    0.62    0.59
0.56
0.72    0.74    0.76    0.78    0.81    0.83    0.86    0.89    0.91    0.94    0.85
0.85/2  0.85    0.94    0.91    0.89    0.86    0.83    0.81    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.9     0.92    0.93    0.94    0.83
0.83/2  0.83    0.94    0.93    0.92    0.9     0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.93    0.95    0.95    0.82
0.82/2  0.82    0.95    0.95    0.93    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.94    0.95    0.95    0.81
0.81/2  0.81    0.95    0.95    0.94    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.95    0.96    0.95    0.81
0.81/2  0.81    0.95    0.96    0.95    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.95    0.96    0.95    0.8
0.8/2   0.8     0.95    0.96    0.95    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.95    0.95    0.95    0.8
0.8/2   0.8     0.95    0.95    0.95    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.95    0.95    0.95    0.79
0.79/2  0.79    0.95    0.95    0.95    0.92    0.88    0.86    0.78    0.76    0.74
0.72
0.72    0.74    0.76    0.78    0.86    0.88    0.92    0.95    0.95    0.95    0.79
0.79/2  0.79    0.95    0.95    0.95    0.92    0.88    0.86    0.78    0.76    0.74
0.72]';

[spd_mg2,trq_mg2]=meshgrid(mc_map_spd_mg2,mc_map_trq_mg2);

% Engin
% ADVISOR Data file:  FC_PRIUS_JPN.m
%
% Data source: Fuel consumption (g/s) and emissions from :
% Feng An's model (ANL) using real test data for calibrating it.

%
% Data confidence level:
%
% Notes:
```

```
% This file (FC_PRIUS_JPN.m) contains fuel use data which has been
% generated by a computer model developed by Feng An but calibrated using
% test data from dyno.
% The model is simulating a
% 1.5L Prius_jpn (Atkinson cycle)engine
% Maximum Power 43kW @4000rpm
% Peak Torque  75 lb-ft @ 4000 rpm.
%
%
% Created on:  6/15/99
% By:  SS, NREL
%
% Revision history at end of file.
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FILE ID INFO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fc_description='Prius_jpn 1.5L (43kW) from FA model and ANL test data';
% % one line descriptor identifying the engine
% fc_version=2003; % version of ADVISOR for which the file was generated
% fc_proprietary=0; % 0=> non-proprietary, 1=> proprietary, do not distribute
% fc_validation=1; % 1=> no validation, 1=> data agrees with source data,
% % 2=> data matches source data and data collection methods have been verified
% fc_fuel_type='Gasoline';
% fc_disp=1.5;  % (L), engine displacement
% fc_emis=1; % boolean 0=no emis data; 1=emis data
% fc_cold=0;      % boolean 0=no cold data; 1=cold data exists
% disp(['Data loaded: FC_PRIUS_JPN - ',fc_description]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SPEED & TORQUE RANGES over which data is defined
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (rad/s), speed range of the engine
fc_map_spd_eng=[1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 4000]*2*pi/60;

lbft2Nm =1.356; %conversion from lbft to Nm
% (N*m), torque range of the engine
fc_map_trq_eng=[6.3 12.5 18.8 25.1 31.3 37.6 43.9 50.1 56.4 62.7 68.9 75.2]*lbft2Nm;

fc_max_trq=[57 60.5 62.5 64 65.9 67.2 68.5 69.8 71.1 72.4 73.7 75.2]*lbft2Nm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUEL USE AND EMISSIONS MAPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fuel_idle (g/s)=  0.5693
%CO2_idle (g/s)=  1.806
%CO_idle (g/s)=  0.0068
%HC_idle (g/s)=  0.0073
%NO_idle (g/s)=  0.0000

% (g/s), fuel use map indexed vertically by fc_map_spd and
% horizontally by fc_map_trq
% fuel use from Feng An's model calibrated with actual data for Prius_jpn (Atkinson
cycle) engine
fc_fuel_map = [
 0.1513  0.1984  0.2455  0.2925  0.3396  0.3867  0.4338  0.4808  0.5279  0.5279
0.5279  0.5279
 0.1834  0.2423  0.3011  0.3599  0.4188  0.4776  0.5365  0.5953  0.6541  0.6689
0.6689  0.6689
 0.2145  0.2851  0.3557  0.4263  0.4969  0.5675  0.6381  0.7087  0.7793  0.8146
0.8146  0.8146
```

```
 0.2451  0.3274  0.4098  0.4922  0.5746  0.6570  0.7393  0.8217  0.9041  0.9659
0.9659  0.9659
 0.2759  0.3700  0.4642  0.5583  0.6525  0.7466  0.8408  0.9349  1.0291  1.1232
1.1232  1.1232
 0.3076  0.4135  0.5194  0.6253  0.7312  0.8371  0.9430  1.0490  1.1549  1.2608
1.2873  1.2873
 0.3407  0.4584  0.5761  0.6937  0.8114  0.9291  1.0468  1.1645  1.2822  1.3998
1.4587  1.4587
 0.3773  0.5068  0.6362  0.7657  0.8951  1.0246  1.1540  1.2835  1.4129  1.5424
1.6395  1.6395
 0.4200  0.5612  0.7024  0.8436  0.9849  1.1261  1.2673  1.4085  1.5497  1.6910
1.8322  1.8322
 0.4701  0.6231  0.7761  0.9290  1.0820  1.2350  1.3880  1.5410  1.6940  1.8470
1.9999  2.0382
 0.5290  0.6938  0.8585  1.0233  1.1880  1.3528  1.5175  1.6823  1.8470  2.0118
2.1766  2.2589
 0.6789  0.8672  1.0555  1.2438  1.4321  1.6204  1.8087  1.9970  2.1852  2.3735
2.5618  2.7501 ]';
% omitting the fabricated data for first column, we reach to:
% fc_fuel_map=fc_fuel_map(2:end,:);

[Spd_eng,Trq_eng]=meshgrid(fc_map_spd_eng,fc_map_trq_eng);

% Battery
xess2_soc=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];

% module's resistance to being discharged, indexed by ess_soc and ess_tmp
ess2_r_dis = 20*[0.0377 0.0338  0.0300  0.0280  0.0275  0.0268  0.0269  0.0273  0.0283
0.0298  0.0312];
% module's resistance to being charged, indexed by ess_soc and ess_tmp
ess2_r_chg = 20*[0.0235 0.0220  0.0205  0.0198  0.0198  0.0196  0.0198  0.0197  0.0203
0.0204  0.0204];
% module's open-circuit (a.k.a. no-load) voltage, indexed by ess_soc and ess_tmp
ess2_voc = 40*[7.2370   7.4047  7.5106  7.5873  7.6459  7.6909  7.7294  7.7666  7.8078
7.9143  8.3645];

P_dis_up = ess2_voc.^2./(4*ess2_r_dis);
P_ch_lo = -ess2_voc.^2./(4*ess2_r_chg);

% Interpolation for battery parameters
V_oc = interp1(xess2_soc,ess2_voc,Ssoc);
R_batt_dis = interp1(xess2_soc,ess2_r_dis,Ssoc);
R_bat_chg = interp1(xess2_soc,ess2_r_chg,Ssoc);
P_batt_up = interp1(xess2_soc,P_dis_up,Ssoc);
P_batt_lo = interp1(xess2_soc,P_ch_lo,Ssoc);


% Interpolation for engine fuel conumption
eng_torq_cn = [6.3 12.5 18.8 25.1 31.3 37.6 43.9 50.1 56.4 62.7 68.9 75.2]*1.356;
eng_vel_st = [1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 4000]*2*pi/60;
Torqeng = [linspace(0,eng_torq_cn(end),400)];
Spdeng = linspace(eng_vel_st(1),eng_vel_st(end),500);
w_fuel=interp2(Spd_eng,Trq_eng,fc_fuel_map,Spdeng',Torqeng,'spline');

% w_fuel(find(Torqeng>=eng_torq_cn(12),1),find(Spdeng>=eng_vel_st(7),1));
Cee_max=interp1(eng_vel_st,fc_max_trq,Spdeng);

% Interpolation for engine MG1 Efficiency
gc_map_trq_mg1=[-55 -45 -35 -25 -15 -5 0 5 15 25 35 45 55];
gc_map_spd_mg1=[-5500 -4000 -3500 -3000 -2500 -2000 -1500 -1000 -500 0 500 1000 1500
2000 2500 3000 3500 4000 5500]*(2*pi)/60;
Torqmg1=linspace(gc_map_trq_mg1(1),gc_map_trq_mg1(end),NgridMG1);
SpdMg1=linspace(gc_map_spd_mg1(1),gc_map_spd_mg1(end),NgridMG1);
```

```
eta_mg1=interp2(spd_mg1,trq_mg1,gc_eff_map_mg1,SpdMg1',Torqmg1);
Cmg1mg1_max=interp1(gc_map_spd_mg1,gc_max_trq,SpdMg1);

% Interpolation for engine MG2 Efficiency
mc_map_trq_mg2=[-305 -275 -245 -215 -185 -155 -125 -95 -65 -35 -5 0 5   35  65  95
125 155 185 215 245 275 305];
mc_map_spd_mg2=[0 500 1000 1500 2000 2500 3000 3500 4000 4500  6000]*(2*pi)/60;
Torqmg2=linspace(mc_map_trq_mg2(1),mc_map_trq_mg2(end),NgridMG2);
SpdMg2=linspace(mc_map_spd_mg2(1),mc_map_spd_mg2(end),NgridMG2);
eta_mg2=interp2(spd_mg2,trq_mg2,mc_eff_map_mg2,SpdMg2',Torqmg2);

Cmg2mg2_max=interp1(mc_spd_data,mc_max_trq_data,SpdMg2);

N=mm;
eng_vel_st=[1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 4000]*2*pi/60;
eng_vel_st_mesh=[750 1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500
4000]*2*pi/60;
mg2_vel_st=[0 500 1000 1500 2000 2500 3000 3500 4000 4500  6000]*(2*pi)/60;

%%%%%%%%%%..............................  ..........................................
mesh of engien speed
% Se =
[linspace(eng_vel_st(1),eng_vel_st(10),30),linspace(eng_vel_st(10)+10,eng_vel_st(end),
4)];
Se_min=Se(1);
Se_max=Se(end);
Se_minset=Se(5);
Se_maxset=Se(end-4);
Se_dif=Se(end)-Se(1);

%%%%%%%.................................................. SOC mesh
Ssoc_min=Ssoc(1);
Ssoc_max=Ssoc(end);
% Ssoc_minst=Ssoc(3);
Ssoc_minst=0.3;
Ssoc_maxst=Ssoc(end-2);
Ssoc_dif=Ssoc(end)-Ssoc(1);
Ssoc_N = length(Ssoc);

mg1_torq_cn=[-55 -45 -35 -25 -15 -5 0 5 15 25 35 45 55];

%%%%%%%.....................................................................
.Engine Torque Mesh
eng_torq_cn=[6.3 12.5 18.8 25.1 31.3 37.6 43.9 50.1 56.4 62.7 68.9 75.2]*1.356;
eng_torq_cn_mesh=[3.15 6.3 12.5 18.8 25.1 31.3 37.6 43.9 50.1 56.4 62.7 68.9 75.2
78]*1.356;

% Ce = [0 linspace(eng_torq_cn(1),eng_torq_cn(end),20)];
Ce = [0 linspace(eng_torq_cn(1),eng_torq_cn(end),30)];
Ce_min=Ce(1);
Ce_max=Ce(end);
Ce_maxset=Ce(end-3);
Ce_dif=Ce(end)-Ce(1);
Cmg1=[-55 -45 -35:5:35, 45 55];
% [-55 -45 -35 -25 -15 -5 0 5 15 25 35 45 55]; control range, MG1 torque
% Cmg2= [linspace(mc_map_trq_mg2(1),mc_map_trq_mg2(10),20),-25,-15,-
5,0,5,15,25,linspace(mc_map_trq_mg2(15),mc_map_trq_mg2(end),20)];
% Cmg2= [linspace(mc_map_trq_mg2(1),-1,100),0,linspace(1,mc_map_trq_mg2(end),100)];

Smg1_min=gc_map_spd_mg1(1);
Smg1_max=gc_map_spd_mg1(end);
Smg1_minset=gc_map_spd_mg1(2);
```

```
Smg1_maxset=gc_map_spd_mg1(end-1);
Cmg1_min=gc_map_trq_mg1(1);
Cmg1_max=gc_map_trq_mg1(end);
Smg1_dif=Smg1_max-Smg1_min;
Cmg2_min=mc_map_trq_mg2(1);
Cmg2_max=mc_map_trq_mg2(end);
Cmg1_dif=Cmg1_max-Cmg1_min;
Cmg2_dif=Cmg2_max-Cmg2_min;
% Ce_min=Ce(1);
% Ce_max=Ce(end);


Se_dif=eng_vel_st(end)-eng_vel_st(1);
L2= zeros(length(Se),length(Ssoc));
% L2{N}(:,1:400)=0;

[se_gr,soc_gr]=meshgrid(Se,Ssoc);
eng_sp_in=[linspace(eng_vel_st(1),eng_vel_st(7),500)];
% L22{N}=interp2(se_gr,soc_gr,L2{N}',eng_sp_in',Ssoc);
% L22=interp2(se_gr,soc_gr,L2{N}',eng_sp_in',Ssoc);
F_roll=mu*m*9.81;
low_Ce=0;
drag_c= 0.5*Rou*A_fr*C_d;
alfa=(R+S)*I_mg1/(S*I_e);
alfa2=-R*I_mg1/S;
I_pf=I_mg1*(R+S)^2/(I_e*S)+S;

%% DP Code
for k = N-1:-1:1
   k
   v1= Vel(k);
   Smg2_1=K*v1/r;
   v2= Vel(k+1);
   Smg2_2=K*v2/r;
   Smg2_dot=(Smg2_2-Smg2_1)/dt;
   matr=ones(length(Se),length(Ssoc),length(Ce),length(Se))*1e20;
   F_roll=mu*m*9.81*sign(v1);
   F_damp=b_w*v1/r;
   F_drag=drag_c*v1^2;
   F_road=F_roll+F_drag+F_damp;
   fc_fuel_coef = [1.5463e-002, 8.5156e-004, -1.7334e-003, 5.8427e-005];
   for Se_i = 1 : length(Se)
      Se_i;
      Smg1_1=((R+S)*Se(Se_i)-R*Smg2_1)/S;
%
      if (Smg1_1>Smg1_max || Smg1_1<Smg1_min)
          matr(Se_i,:,:,:)=1e20;
      else
          for Se_ii = 1 : length(Se)
              Se_dot=(Se(Se_ii)-Se(Se_i))/dt;
              Smg1_dot=((R+S)*Se_dot-R*Smg2_dot)/S;
              Smg1_2=Smg1_dot*dt+Smg1_1;
              if (Smg1_2>Smg1_max || Smg1_2<Smg1_min)
                  matr(Se_i,:,:,Se_ii)=1e20;
              else
                  for Ce_i = 1 : length(Ce)
                      F=(Ce(Ce_i)-Se_dot*I_e)/(R+S);
                      T_mg1=I_mg1*Smg1_dot-F*S;
                      T_mg2=Ip_mg2*Smg2_dot-F*R+F_road*r/K;
                      if (T_mg1>Cmg1_max ||T_mg1<Cmg1_min || T_mg2>Cmg2_max )
                          matr(Se_i,:,Ce_i,Se_ii)=1e20;
                      else
                          if  T_mg2<Cmg2_min
```

```
                                T_mg2=Cmg2_min;
                            end

                            valP1_mg1=T_mg1*Smg1_1;
                            valP2_mg1=T_mg1*Smg1_2;

                            valP1_mg2=T_mg2*Smg2_1;
                            valP2_mg2=T_mg2*Smg2_2;

                            k11 = -sign(valP1_mg1);
                            k12 = -sign(valP2_mg1);

                            k21 = -sign(valP1_mg2);
                            k22 = -sign(valP2_mg2);


valeta1_mg1=eta_mg1(find(Torqmg1>=T_mg1,1),find(SpdMg1>=Smg1_1,1));

valeta2_mg1=eta_mg1(find(Torqmg1>=T_mg1,1),find(SpdMg1>=Smg1_2,1));


valeta1_mg2=eta_mg2(find(Torqmg2>=T_mg2,1),find(SpdMg2>=Smg2_1,1));

valeta2_mg2=eta_mg2(find(Torqmg2>=T_mg2,1),find(SpdMg2>=Smg2_2,1));

                            P_bat= (valP1_mg1*valeta1_mg1^k11 +
valP2_mg1*valeta2_mg1^k12)/2 + ...
                                (valP1_mg2*valeta1_mg2^k21 +
valP2_mg2*valeta2_mg2^k22)/2;

                            if (P_bat<lo_P_batt ||  P_bat>up_P_batt)
                                matr(Se_i,:,Ce_i,Se_ii)=1e20;
                            else

valw_fuel1=fc_fuel_coef(1)+fc_fuel_coef(2)*Se(Se_i)+fc_fuel_coef(3)*Ce(Ce_i)+fc_fuel_c
oef(4)*Se(Se_i)*Ce(Ce_i);

valw_fuel2=fc_fuel_coef(1)+fc_fuel_coef(2)*Se(Se_ii)+fc_fuel_coef(3)*Ce(Ce_i)+fc_fuel_
coef(4)*Se(Se_ii)*Ce(Ce_i);

                                valw_fuel=(valw_fuel1+valw_fuel2)/2;
                                for Ssoc_i=1: length(Ssoc)
                                  if P_bat>0
                                      R_bat = R_batP_vec(Ssoc_i);
                                  else
                                      R_bat = R_batN_vec(Ssoc_i);
                                  end
                                  valV_oc = valV_oc_vec(Ssoc_i);

                                  SOC_dot=-(valV_oc-sqrt(valV_oc^2-
4*P_bat*R_bat))/(7200*Q_bat*R_bat);
                                  fSoc=SOC_dot*dt+Ssoc(Ssoc_i);

                                  if(fSoc>Ssoc_max || fSoc<Ssoc_min)
                                      matr(Se_i,Ssoc_i,Ce_i,Se_ii)=1e20;
                                  else
                                      P_elect=-valV_oc*3600*Q_bat*SOC_dot;
                                      %%%%.......................... Penalty of SOC
                                      if Ssoc(Ssoc_i)<Ssoc_minst
                                          delSoc_lo=(Ssoc(Ssoc_i)-Ssoc_minst)/Ssoc_dif;
                                      else
                                          delSoc_lo=0;
                                      end
```

```
                                    pen_soc1=1e8*delSoc_lo^2;

                                    if Ssoc(Ssoc_i)>Ssoc_maxst
                                        delSoc_up=(Ssoc(Ssoc_i)-Ssoc_maxst)/Ssoc_dif;
                                    else
                                        delSoc_up=0;
                                    end
                                    pen_soc2=1e8*delSoc_up^2;
                                    pen_soc=pen_soc1+pen_soc2;
                                    %%%%.......................... Penalty of fSOC

                                    h_x =
(Beta*alfa_fuel*valw_fuel+alfa_elec*P_elect/ete_grid)*dt+pen_soc;

                                    % up_s = find(Ssoc>=fSoc,1);

                                    up_s = (fSoc-Ssoc_min)/(Ssoc_max-
Ssoc_min)*(Ssoc_N-1)+1;
                                    up_s_floor = floor(up_s);
                                    if up_s_floor ~= up_s
                                        up_s = up_s_floor + 1;
                                    end

                                    l0w_s = up_s-1;
                                    if l0w_s == 0
                                        l0w_s = 1;
                                    end

                                    LL2_1 = L2(Se_ii,l0w_s);
                                    LL2_2 = L2(Se_ii,up_s);
                                    cost_toGo = LL2_1+(fSoc-Ssoc(l0w_s))*(LL2_2-
LL2_1)/iter;
                                    matr(Se_i,Ssoc_i,Ce_i,Se_ii) = cost_toGo + h_x;
                                end
                            end
                        end
                    end
                end
            end
        end
    end

    [L1,nxtSeDP{k}] = min(min(matr,[],3),[],4);
    [L2,torqEngDP{k}] = min(min(matr,[],4),[],3);

End
% toc
```

A.2 DYNAMIC PROGRAMMING IMPLEMENTATION CODE

```
%% Implementation
clear x
load('LAB2BDPnoOnOff.mat')
x(1,1) = eng_vel_st(1);
% x(3,1)=Ssoc(end-3);
x(3,1)=0.4;
[S1,S3]=meshgrid(Se,Ssoc);
[S11,S33,C1]=meshgrid(Ssoc,Se,Ce);
lpw=0;
h_xx=zeros(1,N-1);
fuel_cons=zeros(1,N-1);
for k = 1: N-1
 k


C_Ce=Ce(torqEngDP{k});
Se_next=Se(nxtSeDP{k});

if k<1
eng_torq_imp=C_Ce(find(Se>=x(1,k),1),find(Ssoc>=x(3,k),1));
x(1,k+1)=Se_next(find(Se>=x(1,k),1),find(Ssoc>=x(3,k),1));
else
    if (x(3,k)>=0.29995 && x(3,k)<=0.3)
    x(3,k)=.3;
    end
    eng_torq_imp = interp2(S1,S3,C_Ce',x(1,k),x(3,k));
    x(1,k+1) = interp2(S1,S3,Se_next',x(1,k),x(3,k));
end


v1= Vel(k);
v2=Vel(k+1);
%%%%%%%%%%%%%%.....................................Compute the road force F_r
   F_rolling=mu*m*9.81*sign(v1);
   F_damping=b_w*v1/r;
   F_draging=drag_c*v1^2;
   F_r=F_rolling+F_draging+F_damping;

 %%%%%%%%%%%%%%.....................................Compute the MG2 acceleration "WMG2_dot"
and "WMG1_dot" using vehicle speed
  Wmg2_1=v1*K/r;
  Wmg2_2=v2*K/r;
  Wmg2_dot=(Wmg2_2-Wmg2_1)/dt;


  We_dot=(x(1,k+1)-x(1,k))/dt;
  Wmg1_dot=((R+S)*We_dot-R*Wmg2_dot)/S;

  %%%%%%%%%%%%%.....................................Compute the MG1 and MG2 speeds
  x(4,k)=((R+S)*x(1,k)-R*Wmg2_1)/S;
  x(2,k)=Wmg2_1;
  x(4,k+1)=Wmg1_dot*dt+x(4,k);

  %%%%%%%%%%..................... Compute the MG1 and MG2 torques
  Torq_eng(k)=eng_torq_imp;

                       FF=(Torq_eng(k)-We_dot*I_e)/(R+S);
                        Torq_mg1(k)=I_mg1*Wmg1_dot-FF*S;
                        Torq_mg2(k)=Ip_mg2*Wmg2_dot-FF*R+F_r*r/K;
```

```
        P_engine(k)=(Torq_eng(k)*x(1,k)+Torq_eng(k)*x(1,k+1))/2;

 %%%%%%%%%.........Compute power of MG1 and MG2
 x(2,k)=Wmg2_1;

Pow_mg1_1(k)=Torq_mg1(k)*x(4,k);
   Pow_mg2_1(k)=Torq_mg2(k)*Wmg2_1;



                    if Pow_mg1_1(k)>0
                        k1=-1;
                    else
                        k1=1;
                    end
                    if Pow_mg2_1(k)>0
                        k2=-1;
                    else
                        k2=1;
                    end


eta_mg1_Val_1=eta_mg1(find(Torqmg1>=Torq_mg1(k),1),find(SpdMg1>=x(4,k),1));


eta_mg2_Val_1=eta_mg2(find(Torqmg2>=Torq_mg2(k),1),find(SpdMg2>=Wmg2_1,1));


                     t11_1(k)=Pow_mg1_1(k)*eta_mg1_Val_1.^k1;
                     t22_1(k)=Pow_mg2_1(k)*eta_mg2_Val_1.^k2;
                    Pow_bat_1(k)=t11_1(k)+t22_1(k);

   Pow_mg1_2(k)=Torq_mg1(k)*x(4,k+1);
   Pow_mg2_2(k)=Torq_mg2(k)*Wmg2_2;



                    if Pow_mg1_2(k)>0
                        k1=-1;
                    else
                        k1=1;
                    end
                    if Pow_mg2_2(k)>0
                        k2=-1;
                    else
                        k2=1;
                    end


eta_mg1_Val_2=eta_mg1(find(Torqmg1>=Torq_mg1(k),1),find(SpdMg1>=x(4,k+1),1));


eta_mg2_Val_2=eta_mg2(find(Torqmg2>=Torq_mg2(k),1),find(SpdMg2>=Wmg2_2,1));
                       etaMG1(k)= (eta_mg1_Val_2+ eta_mg1_Val_2)/2;
                       etaMG2(k)= (eta_mg2_Val_2+ eta_mg2_Val_2)/2;

                     t11_2(k)=Pow_mg1_2(k)*eta_mg1_Val_2.^k1;
                     t22_2(k)=Pow_mg2_2(k)*eta_mg2_Val_2.^k2;
                    Pow_bat_2(k)=t11_2(k)+t22_2(k);
                    Pow_bat(k)=(Pow_bat_1(k)+Pow_bat_2(k))/2;
```

```
                    if Pow_bat(k)>0
                        Res_bat=0.5*x(3,k).^2-0.551*x(3,k)+0.687;
%                       Res_bat=interp1(xess2_soc,ess2_r_dis,x(3,k));
                    else
                        Res_bat=0.1298*x(3,k).^2-0.1387*x(3,k)+0.43;
%                      Res_bat=interp1(xess2_soc,ess2_r_chg,x(3,k));
                    end
                     Vol_oc=0.7238*x(3,k).^2+19.8181*x(3,k)+297.107;
%                    Vol_oc= interp1(xess2_soc,ess2_voc,x(3,k));

%

                DSOC(k)=-(Vol_oc-sqrt(Vol_oc^2-
4*Pow_bat(k)*Res_bat))/(7200*Q_bat*Res_bat);
                x(3,k+1)=dt*DSOC(k)+ x(3,k);

valw_fuel1=fc_fuel_coef(1)+fc_fuel_coef(2)*x(1,k)+fc_fuel_coef(3)*Torq_eng(k)+fc_fuel_
coef(4)*x(1,k)*Torq_eng(k);
valw_fuel2=fc_fuel_coef(1)+fc_fuel_coef(2)*x(1,k+1)+fc_fuel_coef(3)*Torq_eng(k)+fc_fue
l_coef(4)*x(1,k+1)*Torq_eng(k);
valw_fuel(k)=(valw_fuel1+valw_fuel2)/2;
% valw_fuel(k)=valw_fuel1;
P_engDP=(x(1,k+1)*Torq_eng(k)+x(1,k)*Torq_eng(k))/2;
EngIndexDP(k)=P_engDP/valw_fuel(k);

Pow_elect(k)=-valV_oc*3600*Q_bat*DSOC(k);


h_xx(k) = (Beta*alfa_fuel*valw_fuel(k)+alfa_elec*Pow_elect(k)/ete_grid)+lpw;

lpw=  h_xx(k);
fuel_cons(k)=h_xx(k);

end
close all
figure
plot (1:N-1, x(3,1:N-1));
title('SoC')
figure
plot (1:N-1, Torq_eng(1:N-1));
title('TorqEng')
figure
plot (1:N-1, Torq_mg1(1:N-1));
title('TorqMg1')
figure
plot (1:N-1, Torq_mg2(1:N-1));
title('TorqMg2')
figure
plot (1:N-1, x(1,1:N-1));
title('SpdEngine')
figure
plot (1:N-1, x(4,1:N-1));
title('SpdMg1')

figure
plot (0:N-1, x(2,1:N)*r/K);
title('SpdVehicle')

figure
 plot(0:N-2,fuel_cons(1:N-1)*0.0333)
```

APPENDIX B

B.1 LINEAR REGRESSION CODE FOR LEARNING

```
EngSpeedDC=[];
MG1SpeedDC=[];
MG1TorqDC=[];
MG2TorqDC=[];
EngTorqDC=[];
VelDC=[];
S_ChDC=[];
RemTRIP_Length=[];
% for ii=1:length(Initial_Soc_Vec)
%     EngSpeedDC=[EngSpeedDC EngSpd1(ii,1:end-1) EngSpd2(ii,1:end-1) EngSpd3(ii,1:end-
1) EngSpd4(ii,1:end-1)];
%     MG1SpeedDC=[MG1SpeedDC Mg1Spd1(ii,1:end-1) Mg1Spd2(ii,1:end-1) Mg1Spd3(ii,1:end-
1) Mg1Spd4(ii,1:end-1)];
%     S_ChDC=[S_ChDC  S_Ch1(ii,1:end-1) S_Ch2(ii,1:end-1) S_Ch3(ii,1:end-1)
S_Ch4(ii,1:end-1)];
%     EngTorqDC=[EngTorqDC Torq_eng_1(ii,:) Torq_eng_2(ii,:) Torq_eng_3(ii,:)
Torq_eng_4(ii,:)];
%     MG1TorqDC=[MG1TorqDC  Torq_mg1_1(ii,:) Torq_mg1_2(ii,:) Torq_mg1_3(ii,:)
Torq_mg1_4(ii,:)];
%     MG2TorqDC=[MG2TorqDC Torq_mg2_1(ii,:) Torq_mg2_2(ii,:) Torq_mg2_3(ii,:)
Torq_mg2_4(ii,:)];
%     VelDC=[VelDC Vel1 Vel2 Vel3 Vel4];
%     RemTRIP_Length=[RemTRIP_Length Rem_trp1 Rem_trp2 Rem_trp3 Rem_trp4];
% end

for ii=1:length(Initial_Soc_Vec)
    EngSpeedDC=[EngSpeedDC EngSpd1(ii,1:end-1) EngSpd2(ii,1:end-1) EngSpd3(ii,1:end-
1)];
    MG1SpeedDC=[MG1SpeedDC Mg1Spd1(ii,1:end-1) Mg1Spd2(ii,1:end-1) Mg1Spd3(ii,1:end-
1)];
    S_ChDC=[S_ChDC  S_Ch1(ii,1:end-1) S_Ch2(ii,1:end-1) S_Ch3(ii,1:end-1)];
    EngTorqDC=[EngTorqDC Torq_eng_1(ii,:) Torq_eng_2(ii,:) Torq_eng_3(ii,:)];
    MG1TorqDC=[MG1TorqDC  Torq_mg1_1(ii,:) Torq_mg1_2(ii,:) Torq_mg1_3(ii,:)];
    MG2TorqDC=[MG2TorqDC Torq_mg2_1(ii,:) Torq_mg2_2(ii,:) Torq_mg2_3(ii,:)];
    VelDC=[VelDC Vel1 Vel2 Vel3];
    RemTRIP_Length=[RemTRIP_Length Rem_trp1 Rem_trp2 Rem_trp3];
end
S_ChDC_2=S_ChDC.*S_ChDC;
VelDC_2=VelDC.*VelDC;
VelS_ChDC=S_ChDC.*VelDC;


% Compute on and off in time based on DP Data
% horizonL=length(EngSpeedDC);
for ii=1:length(Initial_Soc_Vec)
for i=1:N1-1
    if EngSpd1(ii,i+1)==0
        onOff_indx1(ii,i)=0;
    else
      onOff_indx1(ii,i)=1;
    end
end
end
for ii=1:length(Initial_Soc_Vec)
for i=1:N2-1
    if EngSpd2(ii,i+1)==0
        onOff_indx2(ii,i)=0;
```

```
    else
      onOff_indx2(ii,i)=1;
    end
end
end
for ii=1:length(Initial_Soc_Vec)
for i=1:N3-1
    if EngSpd3(ii,i+1)==0
        onOff_indx3(ii,i)=0;
    else
      onOff_indx3(ii,i)=1;
    end
end
end

for ii=1:length(Initial_Soc_Vec)
for i=1:N4-1
    if EngSpd4(ii,i+1)==0
        onOff_indx4(ii,i)=0;
    else
      onOff_indx4(ii,i)=1;
    end
end
end
onOff_indx=[];
for ii=1:length(Initial_Soc_Vec)
%     onOff_indx=[onOff_indx onOff_indx1(ii,:) onOff_indx2(ii,:) onOff_indx3(ii,:)
onOff_indx4(ii,:)];
    onOff_indx=[onOff_indx onOff_indx1(ii,:) onOff_indx2(ii,:) onOff_indx3(ii,:)];
end
HorizonL=length( onOff_indx);
% Compute demand power in time based on DP data
for k=1:HorizonL
    if onOff_indx(k)==1
T_dem_sim(k)=0.0644*EngTorqDC(k)-0.1401* MG1TorqDC(k)+0.1431*MG2TorqDC(k);
    else

T_dem_sim(k)=-0.3670*MG1TorqDC(k)+0.1411*MG2TorqDC(k);
    end
end
T_equ=0.0644*EngTorqDC-0.1401* MG1TorqDC+0.1431*MG2TorqDC;

set2=find(onOff_indx~=0);
EngSpeedDC_on=EngSpeedDC(set2)';
MG1SpeedDC_on=MG1SpeedDC(set2)';
VelDC_on=VelDC(set2)';
RemTRIP_Length_on=RemTRIP_Length(set2)';
S_ChDC_on=S_ChDC(set2)';
EngTorqDC_on=EngTorqDC(set2)';
MG1TorqDC_on=MG1TorqDC(set2)';
MG2TorqDC_on=MG2TorqDC(set2)';
T_equ_on=T_equ(set2)';
dim_on=length(EngSpeedDC_on);
Oness_on=ones(dim_on,1);


EngSpeedDC_on_pow2=EngSpeedDC_on.*EngSpeedDC_on;
MG1SpeedDC_on_pow2=MG1SpeedDC_on.*MG1SpeedDC_on;
VelDC_on_pow2=VelDC_on.*VelDC_on;
S_ChDC_on_pow2=S_ChDC_on.*S_ChDC_on;
EngMG1SpeedDC_on=EngSpeedDC_on.*MG1SpeedDC_on;
EngVelDC_on=VelDC_on.*EngSpeedDC_on;
VelMG1SpeedDC_on=VelDC_on.*MG1SpeedDC_on;
```

```
Eng_SCHDC_on=S_ChDC_on.*EngSpeedDC_on;
SCHDCMG1SpeedDC_on=S_ChDC_on.*MG1SpeedDC_on;
Vel_SCHDC_on=VelDC_on.*S_ChDC_on;

Oness_on ;
% Coef_Mat_on_torgue=[Oness_on T_equ_on S_ChDC_on MG1SpeedDC_on VelDC_on
MG1SpeedDC_on_pow2 VelDC_on_pow2 S_ChDC_on_pow2 SCHDCMG1SpeedDC_on VelMG1SpeedDC_on
Vel_SCHDC_on];
Coef_Mat_on_torgue=[Oness_on T_equ_on S_ChDC_on MG1SpeedDC_on VelDC_on
RemTRIP_Length_on];
%  RemTRIP_Length_on

Constants_TorqueEng_on=pinv(Coef_Mat_on_torgue)*EngTorqDC_on;
Constants_TorqueMG1_on=pinv(Coef_Mat_on_torgue)*MG1TorqDC_on;
Constants_TorqueMG2_on=pinv(Coef_Mat_on_torgue)*MG2TorqDC_on;
% Constants_TorqueEng_on_NoRTL=Constants_TorqueEng_on;
% Constants_TorqueMG1_on_NoRTL=Constants_TorqueMG1_on;
% Constants_TorqueMG2_on_NoRTL=Constants_TorqueMG2_on;
Constants_TorqueEng_on_WithRTL=Constants_TorqueEng_on;
Constants_TorqueMG1_on_WithRTL=Constants_TorqueMG1_on;
Constants_TorqueMG2_on_WithTL=Constants_TorqueMG2_on;
```

B.2 REAL TIME CONTROL IMPLEMENTATION CODE

```
N=N3;
VelDCt=VelDC3;
Rem_trpt=Rem_trp3;
lpw=0;
 h_xx=zeros(1,N-1);
%fuel_cons=zeros(1,N-1);

% x(1,:):engine speed, x(2,:):mg2 speed, x(3,:):SOC and x(4,:): mg1 speed.  inp(1) is
Ce, and
% inp(2) is Cmg1 torgue
SpEngine(1) = eng_vel_st(1);
SpMotoMg2(1)=Vel(1);
StateCharg(1)=Ssoc(end-3);
% x(3,1)=Ssoc(end-3);
% StateCharg(1)=.6;
SpMotoMg1(1)=((R+S)*SpEngine(1)-R*SpMotoMg2(1))/S;
[S1,S3]=meshgrid(Se,Ssoc);
% [S11,S33,C1]=meshgrid(Se,Ssoc,Ce);
[S11,S33,C1]=meshgrid(Ssoc,Se,Ce);
% [S11,S33,C1]=meshgrid(Ssoc,Se,Ce);
Vp=zeros(1,N);
Vp(1)=0;

P_demand(1)=0;

Speed=zeros(1,N-1);
Speed(1)=0;
P_demand=zeros(1,N-1);
acc=zeros(1,N-1);
T_eng=zeros(1,N-1);
T_mg1=zeros(1,N-1);
T_mg2=zeros(1,N-1);
Speed(1)=0;
l(2,1)=0;
lll(1)=0;
le=0;
lmg1=0;
```

```
lmg2=0;
lpww=0;
elec_it=0;
gas_it=0;
Iner = [I_e 0 0 (S+R);0 I_mg1 0 -S;0 0 Ip_mg2 -R;-(R+S) S R 0];
B = inv(Iner);
Iner2=Iner(2:end,2:end);
B2=inv(Iner2);
% st_stp=[
EquTorque=T_equ(1:N-1);
% T_equ_spc=T_equ(N1:N1+N2-2);
% T_equ_spc=T_equ(N1+N2-1:N1+N2+N3-3);
% T_equ_spc=T_equ(N1+N2+N3-2:N1+N2+N3+N4-4);
% NNN=length(T_equ_spc);
SpdVeh(1)=0;
TmgSwitch=.8;
switINDX=0;
for k = 1: N-1
%     rem_trp_lngth(k)=L_trip_tot-(lll(k)*dt+Trip_paved1(k))
       lll(k)=VelDCt(k);

 k
                    F_drag=0.5*Rou*A_fr*C_d*lll(k)^2;
                    F_damp=b_w*lll(k)/r;
                       F_roll=mu*m*9.81*sign(lll(k));
                    F_road=F_roll+F_drag+F_damp;
                    T_r=F_road*r/K;

%     T_equvalent(k)=(Vel(k+1)-lll(k))*K/r+0.1431*T_r;
%     T_equvalent(k)
%     T_equvalent(k)=EquTorque(k);
    lll(k)=VelDCt(k);
%     if lll(k)==0
%        onOrOff(k)=0;
%     else
%     coefff_onOrOff=[lll(k) StateCharg(k) lll(k)^2 StateCharg(k)^2
lll(k)*StateCharg(k)];
%    onOrOff(k)=coefff_onOrOff*Constants_OnOff;
%     end
%     onOrOff=[lll(k)]*Constants_on_ff;
%     ONOFFSWITCH(k)=onOrOff;
onOrOff(k)=1;

% onOrOff=onOff_indx(k);

%          T_equvalent(k)=0.0644*EngTorqDC(k)-0.1401*
MG1TorqDC(k)+0.1431*MG2TorqDC(k);
         %%%%%%%On Scenario
         T_demand_on(k)=(VelDCt(k+1)-VelDCt(k))*K/r+B(3,3)*T_r;
%          [Oness_on T_equ_on S_ChDC_on MG1SpeedDC_on VelDC_on MG1SpeedDC_on_pow2
VelDC_on_pow2 S_ChDC_on_pow2 SCHDCMG1SpeedDC_on VelMG1SpeedDC_on Vel_SCHDC_on];
%          Coef_Mat_on_torgue=[1 T_equvalent(k) StateCharg(k) SpMotoMg1(k) lll(k)
SpMotoMg1(k)^2 lll(k)^2 StateCharg(k)^2 StateCharg(k)*SpMotoMg1(k) lll(k)*SpMotoMg1(k)
lll(k)*StateCharg(k)];
         Coef_Mat_on_torgue=[1 T_demand_on(k) StateCharg(k) SpMotoMg1(k) SpdVeh(k)
Rem_trpt(k)];
 Rem_trpt(k);
%     TorqEng1(k)=Coef_Mat_on_torgue*Constants_TorqueEng_on_NoRTL;
%
%      TorqMg111(k)=Coef_Mat_on_torgue*Constants_TorqueMG1_on_NoRTL;
%      TorqMg222(k)=Coef_Mat_on_torgue*Constants_TorqueMG2_on_NoRTL;
    TorqEng1(k)=Coef_Mat_on_torgue*Constants_TorqueEng_on_WithRTL;
```

```
     TorqMg111(k)=Coef_Mat_on_torgue*Constants_TorqueMG1_on_WithRTL;
     TorqMg222(k)=Coef_Mat_on_torgue*Constants_TorqueMG2_on_WithTL;
     tork=TorqMg222(k);


if switINDX==1 && StateCharg(k) > Ssoc_minst+0
    switINDX=0;
end
if switINDX==0 && StateCharg(k) < Ssoc_minst
    switINDX=1;
end

if switINDX==1

   if TorqMg111(k)*SpMotoMg1(k) > 0 && TorqMg222(k)*SpMotoMg2(k) > 0
%        tyjkhh
%        TorqMg1P=-sign(SpMotoMg1(k))*TmgSwitch;
       TorqMg1P=TorqMg111(k)*TmgSwitch;
       TorqMg2P=TorqMg222(k)*TmgSwitch;

TorqEng1(k)=1/B(3,1)*(B(3,2)*TorqMg111(k)+B(3,1)*TorqEng1(k)+B(3,3)*TorqMg222(k)-
B(3,2)*TorqMg1P-B(3,3)*TorqMg2P);
       TorqMg111(k) = TorqMg1P;
       TorqMg222(k) = TorqMg2P;

   elseif TorqMg111(k)*SpMotoMg1(k) > 0
% tpg
%         TorqMg1P=-sign(SpMotoMg1(k))*TmgSwitch;
        TorqMg1P=TorqMg111(k)*TmgSwitch;
        TorqEng1(k)=1/B(3,1)*(B(3,2)*TorqMg111(k)+B(3,1)*TorqEng1(k)-B(3,2)*TorqMg1P);
        TorqMg111(k) = TorqMg1P;

   elseif TorqMg222(k)*SpMotoMg2(k) > 0
% tyi
        TorqMg2P=TorqMg222(k)*TmgSwitch;
        TorqEng1(k)=1/B(3,1)*(B(3,3)*TorqMg222(k)+B(3,1)*TorqEng1(k)-B(3,3)*TorqMg2P);
        TorqMg222(k) = TorqMg2P;
   end
end




%      T_equvalent(k)=0.0644*TorqEng1(k)-0.1401* TorqMg111(k)+0.1431*TorqMg222(k);

                        if   TorqMg222(k)<Cmg2_min
                             TorqMg222(k)=Cmg2_min;
                        end
                        if   TorqMg111(k)<Cmg1_min
                             TorqMg111(k)=Cmg1_min;
                        end


            F_drag=0.5*Rou*A_fr*C_d*lll(k)^2;
                   F_damp=b_w*lll(k)/r;
                      F_roll=mu*m*9.81*sign(lll(k));
                   F_road=F_roll+F_drag+F_damp;
                Tppp_mg2=TorqMg222(k)-F_road*r/K;

                Tkkk=B*[TorqEng1(k);TorqMg111(k);Tppp_mg2;0];


%
```

```
                    TkPp=Tkkk(1:3,1)*dt+[SpEngine(k);SpMotoMg1(k);SpMotoMg2(k)];

                     SpEngine(k+1)=TkPp(1);
                     SpMotoMg1(k+1)=TkPp(2);
                     SpMotoMg2(k+1)=TkPp(3);
  P_eng(k)=(TorqEng1(k)*SpEngine(k)+TorqEng1(k)*SpEngine(k+1))/2;
SpdVeh(k+1)=SpMotoMg2(k+1)*r/K;
if SpdVeh(k+1)<0
    SpdVeh(k+1)=0;
end


        if TorqEng1(k)<0
            TorqEng1(k)=0;
        end

      Pow_mg1_1(k)=TorqMg111(k)*SpMotoMg1(k);
   Pow_mg2_1(k)=TorqMg222(k)*SpMotoMg2(k);



                     if Pow_mg1_1(k)>0
                         k1=-1;
                     else
                         k1=1;
                     end
                     if Pow_mg2_1(k)>0
                         k2=-1;
                     else
                         k2=1;
                     end
                     tpfg1=SpMotoMg1(k);
                     if tpfg1>SpdMg1(end)
                         tpfg1=SpdMg1(end);
                     end


eta_mg1_Val_1=eta_mg1(find(Torqmg1>=TorqMg111(k),1),find(SpdMg1>=tpfg1,1));


eta_mg2_Val_1=eta_mg2(find(Torqmg2>=TorqMg222(k),1),find(SpdMg2>=SpMotoMg2(k),1));


                      t11_1(k)=Pow_mg1_1(k)*eta_mg1_Val_1.^k1;
                      t22_1(k)=Pow_mg2_1(k)*eta_mg2_Val_1.^k2;
                     Pow_bat_1(k)=t11_1(k)+t22_1(k);

   Pow_mg1_2(k)=TorqMg111(k)*SpMotoMg1(k+1);
   Pow_mg2_2(k)=TorqMg222(k)*SpMotoMg2(k+1);



                     if Pow_mg1_2(k)>0
                         k1=-1;
                     else
                         k1=1;
                     end
                     if Pow_mg2_2(k)>0
                         k2=-1;
                     else
```

```
                        k2=1;
                    end
                    tpfg=SpMotoMg1(k+1);
                    if tpfg>SpdMg1(end)
                        tpfg=SpdMg1(end);
                    end

eta_mg1_Val_2=eta_mg1(find(Torqmg1>=TorqMg111(k),1),find(SpdMg1>=tpfg,1));


eta_mg2_Val_2=eta_mg2(find(Torqmg2>=TorqMg222(k),1),find(SpdMg2>=SpMotoMg2(k+1),1));
%                         etaMG1(k)= (eta_mg1_Val_2+ eta_mg1_Val_2)/2;
%                         etaMG2(k)= (eta_mg2_Val_2+ eta_mg2_Val_2)/2;

                     t11_2(k)=Pow_mg1_2(k)*eta_mg1_Val_2.^k1;
                     t22_2(k)=Pow_mg2_2(k)*eta_mg2_Val_2.^k2;
                    Pow_bat_2(k)=t11_2(k)+t22_2(k);
                    Pow_batt(k)=(Pow_bat_1(k)+Pow_bat_2(k))/2;

                    if Pow_batt(k)>0
                        Res_bat=0.5*StateCharg(k).^2-0.551*StateCharg(k)+0.687;
%                       Res_bat=interp1(xess2_soc,ess2_r_dis,x(3,k));
                    else
                        Res_bat=0.1298*StateCharg(k).^2-0.1387*StateCharg(k)+0.43;
%                      Res_bat=interp1(xess2_soc,ess2_r_chg,x(3,k));
                    end
                     Vol_oc=0.7238*StateCharg(k).^2+19.8181*StateCharg(k)+297.107;
%                    Vol_oc= interp1(xess2_soc,ess2_voc,x(3,k));

%

%                  x(3,k+1)=dt*DSOC(k)+ x(3,k);

valw_fuel1=fc_fuel_coef(1)+fc_fuel_coef(2)*SpEngine(k)+fc_fuel_coef(3)*TorqEng1(k)+fc_
fuel_coef(4)*SpEngine(k)*TorqEng1(k);
valw_fuel2=fc_fuel_coef(1)+fc_fuel_coef(2)*SpEngine(k+1)+fc_fuel_coef(3)*TorqEng1(k)+f
c_fuel_coef(4)*SpEngine(k+1)*TorqEng1(k);
valw_fuel(k)=(valw_fuel1+valw_fuel2)/2;
                     radikall(k)=Vol_oc^2-4*Pow_batt(k)*Res_bat;
                     if radikall(k)<0
                      radikall(k)=0;
                     end
                  D_S_OC(k)=-(Vol_oc-sqrt(radikall(k)))/(7200*Q_bat*Res_bat);
                 StateCharg(k+1)=dt* D_S_OC(k)+ StateCharg(k);



                 Pow_elect(k)=-valV_oc*3600*Q_bat*D_S_OC(k);

%  saeidB=
h_xxx(k) = (Beta*alfa_fuel*valw_fuel(k)+alfa_elec*Pow_elect(k)/ete_grid)+lpw;
 Elec_MJ(k)=alfa_elec*Pow_elect(k)/ete_grid+elec_it;
Gas_MJ(k)  =alfa_fuel*valw_fuel(k)+gas_it;

lpw=  h_xxx(k);
gas_it=Gas_MJ(k);
elec_it=Elec_MJ(k);
fuel_cons_app(k)=h_xxx(k);



end
% N=N3;
```

```
% figure(1)
% plot(0:N-1,SpdVeh(1:N))
% hold on
% plot(0:N-1,VelDCt(1:N))
% hold off
% xlabel(['Time (s)'])
% ylabel(['Vehicle Speed (m/s)'])
% legend('Approximated(Linear)','real drive Cycle')

figure(2)
plot(0:N-1,StateCharg(1:N))
hold on
% plot(0:N-1,x(3,1:N))
% hold off
xlabel(['Time (s)'])
ylabel(['SoC'])
legend('Approximated(Linear)','From DP')

figure(3)
plot(0:N-2,fuel_cons_app(1:N-1)*0.0333)
hold on
% hold on
% plot(0:N-2,fuel_cons(1:N-1)*0.0333)
% plot(0:N-2,fuel_cons(1:N-1)/130)
% hold off
xlabel(['Time (s)'])
ylabel(['Equivalet Fuel Consumption [USD]'])
legend('Approximated(Linear)','From DP')
```