SP-SDP for Fuel Consumption and Tailpipe Emissions Minimization in an EVT Hybrid

Ed D. Tate, J. W. Grizzle, and Huei Peng

Abstract—Control strategies have been developed for Hybrid Electric Vehicles (HEV) that minimize fuel consumption while satisfying a charge sustaining constraint. Since one of the components of an HEV is typically the ubiquitous internal combustion engine, tailpipe emissions must also be considered. This paper uses Shortest-Path Stochastic Dynamic Programming (SP-SDP) to address the minimization of a weighted sum of fuel consumption and tailpipe emissions for an HEV equipped with a dual mode Electrically Variable Transmission (EVT) and a catalytic converter. The shortest path formulation of SDP is chosen to directly address the charge sustaining requirement. Using simple methods, an SP-SDP solution required more than eight thousand hours. Using linear programming and duality, an SP-SDP problem is solved in about three hours on a desktop PC. The resulting time-invariant feedback controller reduces tailpipe emissions by more than 50% when compared to a popular baseline controller.

Index Terms—Hybrid electric vehicle, dynamic programming, fuel economy, powertrain control

I. INTRODUCTION

The problem of maximizing fuel economy for charge sustaining hybrid electric vehicles (HEVs) has been extensively studied. Many early control policies were based on heuristics [1-5]. In some systems, the heuristics were formalized using fuzzy logic [6]. More recent control laws have used some form of optimization to determine the power split between the engine and the electric motors. These optimization techniques explicitly minimize the fuel consumption subject to a charge sustaining constraint. Deterministic dynamic programming is one optimization technique. It has been used to develop optimal trajectories [7-10] . These trajectories have been used to gain insight and develop heuristic control laws [8]. Recently, the concept of directly designing an HEV control law through infinite horizon, discounted cost, stochastic dynamic programming (SDP) was developed [11, 12]. As noted in [13], the discounted cost criteria is chosen for expediency and is difficult to justify on engineering grounds. Therefore, in this work, a variation on infinite horizon SDP, known as Shortest Path SDP [14] is used to create a charge sustaining HEV energy management policy.

In addition to fuel economy, HEVs must also be certified to some tailpipe emissions level by the EPA or CARB [15-18]. There are multiple categories of emissions. A manufacturer chooses the certification level of a vehicle model to achieve specific goals for their production fleet. This choice usually involves a tradeoff among fleet objectives, vehicle cost objectives, technical feasibility, and fuel economy objectives. During development, the initial goal of a particular certification level may change as tradeoffs become more apparent.

Prior work has focused on minimization of fuel consumption and engine out emissions. If catalysts or other after treatments are used, minimization of engine out emissions does not guarantee minimization of tailpipe emissions. Therefore, there is a need for techniques to design controllers that minimize both fuel economy and tailpipe emissions.

This paper shows how shortest path stochastic dynamic programming (SP-SDP) is used to design controllers that minimize fuel consumption and tailpipe emissions. This application of SP-SDP is illustrated on a dual mode EVT HEV [19] with a thermally transient catalyst. A collection of controllers is created and evaluated. Each controller minimizes a unique weighted combination of fuel consumption and emissions. Together, these controllers are used to estimate the Pareto set [20] . From the Pareto set, the tradeoffs between fuel consumption and tailpipe emissions are determined.

To generate the control laws, an objective to minimize a weighted combination of fuel consumption and tailpipe emissions, a vehicle model, a powertrain model, and a stochastic driver model are combined to form a SP-SDP. This SP-SDP is solved using a collection of techniques which include Linear Programming [21-29], barycentric interpolation [30], and constraint generation [26]. A novel interpretation of LP duality [31] is used to solve the Linear Program (LP). The technique of state censoring is introduced. This technique ignores areas in the state space which are known to be rarely, if ever visited. Finally, the calls to the model during construction of the SP-SDP are reduced by taking advantage of linearity in the model. Combined, these

Manuscript received XXX XX, 2006.

Ed D. Tate is with General Motors, Hybrid Powertrain Engineering, Troy, Michigan 48007 USA (phone: 248-310-4488; fax: 248-680-5119; e-mail: ed.d.tate@gm.com, etate@umich.edu) . I think you should say you are with UofM also.

J. W. Grizzle is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan 48109-2122, USA, (e-mail: grizzle@umich.edu).

Huei Peng is with the Mechanical Engineering Department, University of Michigan, Ann Arbor, Michigan 48109-2133, USA, (e-mail: hpeng@umich.edu).

techniques allow solution of the SP-SDP in less than three hours on a 3.0 GHz, single core Pentium CPU.

II. Plant Model

The controller development uses a plant model that includes the powertrain, the vehicle, and the driver. Figure 1 illustrates the major components in the model and their interactions. These components include the engine, the Dual-Mode Electrically Variable Transmission (DM-EVT), two electric machines, the final drive ratio, the battery pack, the chassis, the body, the tires, and the chassis brakes. The powertrain and the vehicle models are deterministic. The driver model is stochastic.

The transmission is based on the design in [19] and illustrated in Figure 2. It has four shafts which are coupled to the engine, the wheels through the final drive ratio, and the two electric machines. The electric machines are identified as 'A' and 'B'. Unlike a fixed-gear transmission, the engine speed is independent of the vehicle speed. However, the electric machines speeds are determined by the transmission mode, the speed of the engine, and the speed of the vehicle. The engine and vehicle speeds are independent because the engine acceleration and vehicle acceleration can be separately controlled using the torques from the engine and electric machines. The range of acceleration is limited by the engine and motor torque limits which vary depending on individual component speeds. Proper selection of the modes in the transmission avoids these acceleration limits by changing the ratio of motor speeds to engine and vehicle speeds. This allows the motors to operate in regions with less torque restriction.

In the DM-EVT transmission, two no-slip or dog clutches are used. To change modes, the clutches are released allowing motor speeds to be controlled independently of the engine and vehicle speeds. The engine torque and machine torques are used to accelerate the electric machines to a speed where one of the clutches can close without slipping. For modeling, this event is assumed to occur instantaneously with negligible fuel and battery energy usage. The kinematics of the transmission and the vehicle are illustrated in Figure 3.

The driver is modeled as a stochastic process that generates the input a_{now}, which represents the driver's acceleration request, in Figure 1. The driver model specifies the distribution of the next acceleration commands, the dynamics for updating a_{now}, and the probability that the vehicle will continue to operate with the key on. When the controller is evaluated, the state a_{now} is replaced by the acceleration command from the vehicle's driver.

Although the plant is naturally described in continuous time, to facilitate the application of SP-SDP, a discrete time model with a one second time step is used. This is the same rate used for similar work in [11, 12].

A. The Plant Model Summary

The plant model has states associated with the engine speed, ω_e, the vehicle speed, v_{veh}, the battery state of charge, q_{batt}, the catalyst temperature, T_{cat}, and the driver acceleration command, a_{now}. The model states are combined to form the state vector:

$$x \triangleq \begin{bmatrix} v_{veh} & q_{batt} & \omega_e & T_{cat} & a_{now} \end{bmatrix}^T \in \mathcal{X} \subset \mathbb{R}^5 . \quad (1)$$

The transmission mode is allowed to change at any time and the changes are assumed to occur instantaneously, therefore, there is no state associated with transmission clutch configuration. The transmission mode input is

$$M_{trn} \in \mathcal{M}_{\text{trn}} = \{\text{'mode 1','mode 2'}\} . \quad (2)$$

When M_{trn} is equal to 'mode 1' it means that clutch 1 in Figure 2 is closed with clutch 2 open. Likewise, if M_{trn} is equal to 'mode 2', then clutch 2 is closed with clutch 1 open.

When the engine changes mode between 'on' and 'off', the transition from a running speed to motionless and from motionless to an idle speed occurs instantaneously in the model. The input to change the engine mode is

$$M_e \in \mathcal{M}_{\text{e}} = \{\text{'eng on','eng off'}\} . \quad (3)$$

When M_e is equal to 'eng off', the engine is motionless with its brake torque equal to zero. Otherwise, the engine is restricted to operate between a minimum idle speed and a maximum redline speed, with the torque limited by the engine's capabilities.

The engine torque, T_e, electric machine A torque, T_A, electric machine B torque, T_B, and chassis brake torque, T_k, are all independently controlled. These torque inputs are combined with the transmission mode command, M_{trn}, and the engine mode command, M_e, to form the input vector for the plant:

$$u_{plant} \triangleq \begin{bmatrix} M_{trn} & M_e & T_e & T_A & T_B & T_k \end{bmatrix} \in \mathcal{U}_{\text{plant}} \subset \mathbb{Z}^2 \times \mathbb{R}^4 . \quad (4)$$

The discrete-time dynamics of the model are

$$x_{k+1} = f_{plant}\left(x_k, u_{plant,k}, w_k\right) = \begin{bmatrix} f_{v_{veh}}\left(x_k, u_{plant,k}, w_k\right) \\ f_{q_{batt}}\left(x_k, u_{plant,k}, w_k\right) \\ f_{\omega_e}\left(x_k, u_{plant,k}, w_k\right) \\ f_{T_{cat}}\left(x_k, u_{plant,k}, w_k\right) \\ f_{a_{now}}\left(x_k, u_{plant,k}, w_k\right) \end{bmatrix} . \quad (5)$$

In these equations,

$$w_k = a_{next,k} \quad (6)$$

where $a_{next,k}$ is the output of the driver model. The variable a_{next} represents the vehicle acceleration command in the next time step, which is different than a_{now} which represents the current vehicle acceleration command.

The outputs of the plant model are W_f, the fuel consumed per time step, W_e, the normalized tailpipe emissions per time step, and P_{batt}, the battery power. The output functions are

$$\begin{bmatrix} W_{f,k} \\ W_{e,k} \\ P_{batt,k} \end{bmatrix} = h_{plant}\left(x_k, u_{plant,k}, w_k\right) = \begin{bmatrix} h_{W_f}\left(x_k, u_{plant,k}, w_k\right) \\ h_{W_e}\left(x_k, u_{plant,k}, w_k\right) \\ h_{P_{batt}}\left(x_k, u_{plant,k}, w_k\right) \end{bmatrix} . \quad (7)$$

In (7), the variable w represents the stochastic input to the system and is used for the sake of generality. These functions do not use w in determining the output values. However, while the models used in this work assume the engine behaves deterministically, unmodeled dynamics in the engine can cause the fuel consumption and engine-out emissions to appear to have a stochastic component. For example, manifold dynamics, exhaust gas recirculation (EGR) valve position, EGR valve control, and engine temperatures are not modeled. These dynamic behaviors will cause fuel consumption and engine out emissions to have a distribution of values. Therefore, this notation is maintained because of generality and applicability.

The constraints on operation of the individual components are grouped into a vector function such that feasibility is equivalent to all elements in the vector being less than or equal to zero. This constraint vector is

$$g_{plant}\left(x,u_{plant}\right)=\begin{bmatrix} g_{vehicle}\left(x,u_{plant}\right) \\ g_{engine}\left(x,u_{plant}\right) \\ g_{catalyst}\left(x,u_{plant}\right) \\ g_{motors}\left(x,u_{plant}\right) \\ g_{battery}\left(x,u_{plant}\right) \end{bmatrix}. \qquad (8)$$

These constraints ensure that components operate within their limits and the system does not leave a safe operating envelope in the state space. For example, an action choice is not feasible if it results in the engine operating beyond the maximum engine speed. Equation (8) is used to define the set of feasible actions

$$\mathcal{U}_{\text{plant}}\left(x\right)=\left\{u_{plant}\in\mathcal{U}_{\text{plant}}\left|g_{plant}\left(x,u_{plant}\right)\le 0, x\in\mathcal{X}\right.\right\}. \qquad (9)$$

The model and constraint equations are constructed so that a feasible action always exists by insuring that $\mathcal{U}_{\text{plant}}\left(x\right)$ is non-empty for all $x\in\mathcal{X}$.

The detailed model parameters and equations are in [32].

B. Driver Behavior Model

The driver model is part of the plant model. The driver model predicts the distribution of future accelerations and the probability the vehicle will be turned off at the next time step. In [12], torque is used in the stochastic driver model. The use of torque or power tightly couples the driver model to the characteristics of a vehicle. In other words, if the driver follows a velocity trace in a small economy vehicle and follows the same trace in a large truck, then different torque based driving models result. The torque and power required at the wheels varies with vehicle, while the acceleration is identical for the same driving cycle. Therefore, using acceleration, the driver model is constructed and maintained independent of vehicle characteristics.

The driver model consists of two conditional probabilities. One conditional probability is

$$a_{next}\sim\Pr\left(a_{next}\left|v_{veh},a_{now}\right.\right), \qquad (10)$$

where a_{next} is the vehicle acceleration at the next time step, v_{veh} is the velocity at the current time step, and a_{now} is the acceleration at the current time step. This conditional probability specifies the distribution of acceleration commands at the next time step. The other conditional probability defines the probability that the vehicle will continue to operate at the next time step. This model is

$$\begin{aligned}&\Pr_{on}\left(v_{veh},a_{now}\right)= \\ &\Pr\left(\text{next time step vehicle is } 'on'\left|v_{veh},a_{now}\right.\right)\end{aligned}. \qquad (11)$$

Both conditional probability models are constructed using discrete values of v_{veh} and a_{now} selected to match the sample points used in the SP-SDP. The values of a_{next} are limited to a finite set of discrete values,

$$a_{next}=w\in\mathcal{W}=\left\{a_1,a_2,\cdots,a_n\right\}. \qquad (12)$$

For generality, the variable w represents the stochastic variable in the model. The values in (12) correspond to the discrete values of a_{now} used to construct the SP-SDP.

III. Problem Definition

Consider the problem of determining the optimal tradeoffs between expected fuel consumption and expected tailpipe emissions. One way to do this is to find a collection of controllers, where each controller minimizes a specific combination of fuel consumption and emissions, while satisfying plant and system constraints.

A. Control Constraints

In addition to the plant constraints in (8), the system imposes constraints on the action choices. The system constraints include the charge sustaining constraint previously discussed. In addition, the controller must choose actions that minimize the error in matching the driver's acceleration command and minimize chassis (friction) brake torque. The chassis brake torque is constrained to minimize energy loss and to minimize brake wear. These three constraints are the system constraints and further restrict the set of feasible actions.

Since a charge sustaining constraint is difficult to directly address, it is reformulated as an objective to minimize the deviation from an SOC set point at each time step. The relationship between this objective, fuel economy optimality, and satisfaction of the charge sustaining constraint is discussed in Appendix A. This objective cost is defined as

$$c_q\left(x,u_{plant},w\right)=K_q\cdot\left(f_{q_{batt}}\left(x,u_{plant},w\right)-q_0\right)^2. \qquad (13)$$

The values of K_q and q_0 are selected by experimentation to achieve charge sustaining operation over a range of driving conditions.

The remaining constraints on acceleration and chassis braking are satisfied by restricting the set of feasible actions. Let $g_{sys}\left(x,u_{plant}\right)$ be defined so that the constraints on acceleration and braking are satisfied with no acceleration error and no chassis braking torque if

$$g_{sys}\left(x,u_{plant}\right)\le 0. \qquad (14)$$

Since the set of action choices in (9) that satisfy (14) may be empty, a relaxation is introduced using the vector ε_{sys}. The relaxed form of (14) becomes

$$g_{sys}\left(x,u_{plant}\right)-\varepsilon_{sys}\leq 0\,. \tag{15}$$

If the acceleration error is zero and no brake torque is used then equation (15) is satisfied with all elements in ε_{sys} equal to zero. The relaxed system constraints, (15), and the plant constraints, (8), define the set of feasible actions at each state. Let

$$\mathcal{U}_{sys+plant}\left(x\right)=\left\{u_{plant}\in\mathcal{U}_{plant}\left(x\right)\middle|g_{sys}\left(x,u_{plant}\right)-\varepsilon^{*}_{sys}\left(x\right)\leq 0\right\} \tag{16}$$

be the set of feasible control choices available for control. At many values of x, this set will be non-empty with every element in $\varepsilon^{*}_{sys}\left(x\right)$ equal to zero. Otherwise, $\varepsilon^{*}_{sys}\left(x\right)$ is a vector with one or more elements containing the smallest non-zero value that ensures $\mathcal{U}_{sys+plant}\left(x\right)$ is non-empty. There are many ways the value of $\varepsilon^{*}_{sys}\left(x\right)$ can be found. For this work, the element in $\varepsilon^{*}_{sys}\left(x\right)$ that relaxes the chassis braking torque constraint is increased until a feasible action is available or the constraint is no longer active. If relaxing the chassis braking torque constraint does not yield a feasible action, then the element in $\varepsilon^{*}_{sys}\left(x\right)$ corresponding to the acceleration error is increased until a feasible action is found. By construction of the model, relaxation of acceleration error is guaranteed to find at least one feasible action.

B. *Control Objectives*

The objective for the controller is to minimize a weighted combination of total expected fuel consumption and total expected tailpipe emissions subject to the system and plant constraints. The set in (16) defines the actions available to the controller for use in minimizing the objective.

The total expected fuel consumption while operating under the feedback law $\pi\left(x\right)$ is

$$J^{\pi}_{f}\left(x_0\right)=E^{W}\left(\sum_{k=0}^{\infty}c_f\left(x_k,u_{plant,k},w_k\right)\right). \tag{17}$$

The total expected tailpipe emissions under the same control law is

$$J^{\pi}_{e}\left(x_0\right)=E^{W}\left(\sum_{k=0}^{\infty}c_e\left(x_k,u_{plant,k},w_k\right)\right). \tag{18}$$

In these equations, $c_f\left(\cdot\right)$ is the fuel consumption per time step and $c_e\left(\cdot\right)$ is the normalized tailpipe emissions per time step. In calculating these total costs, the evolution of the state is

$$x_k=f_{plant}\left(x_k,u_{plant,k},w_k\right). \tag{19}$$

The stochastic input, w_k, is generated by the probability distribution in (6) and

$$u_{plant,k}=\pi\left(x_k\right), \tag{20}$$

where $\pi\left(\cdot\right)$ is a static, time-invariant, state feedback controller. To satisfy the charge sustaining constraint, a cost for SOC variation,

$$J^{\pi}_{q}\left(x_0\right)=E^{W}\left(\sum_{k=0}^{\infty}c_q\left(x_k,u_{plant,k},w_k\right)\right), \tag{21}$$

is introduced as a minimization objective. This objective uses $c_q\left(\cdot\right)$ in (13).

The optimization problem is to find a controller, π^{*}, that satisfies $\forall x_0\in\mathcal{X}$

$$\pi^{*}\left(x_0\right)\in\underset{\pi\in\Pi_{\infty}}{\arg\inf}\left\{\alpha\cdot J^{\pi}_{f}\left(x_0\right)+\left(1-\alpha\right)\cdot J^{\pi}_{e}\left(x_0\right)+J^{\pi}_{q}\left(x_0\right)\right\}, \tag{22}$$

where Π_{∞} is the set of all possible time-invariant state feedback controllers with $u_{plant,k}\in U_{sys+plant}\left(x_k\right)$ for all k and $\alpha\in\left[0,1\right]$. Without special structure, the optimization problem in (22) is intractable. Fortunately, this optimization problem can be significantly simplified. Under the assumption that perfect, or noise free, state information is available, a controller that minimizes (22) is found using dynamic programming. A controller that minimizes (22) satisfies the dynamic programming equations [33-35],

$$\pi^{*}\left(x\right)=\underset{u_{plant}\in\mathcal{U}_{sys+plant}\left(x\right)}{\arg\min}\left\{E^{W}\left(\begin{array}{l}c_{plant}\left(x,u_{plant},w\right)\\+V\left(f_{plant}\left(x,u_{plant},w\right)\right)\end{array}\right)\right\}, \tag{23}$$

and uses the value function defined as

$$V\left(x\right)=\min_{u_{plant}\in\mathcal{U}_{sys+plant}\left(x\right)}\left\{E^{W}\left(\begin{array}{l}c_{plant}\left(x,u_{plant},w\right)\\+V\left(f_{plant}\left(x,u_{plant},w\right)\right)\end{array}\right)\right\}. \tag{24}$$

In these equations, $c_{plant}\left(\cdot\right)$ is the cost for operation at each time step, defined here as

$$\begin{aligned}c_{plant}\left(x,u_{plant},w\right)=&\left(\alpha\right)\cdot c_f\left(x,u_{plant},w\right)\\&+\left(1-\alpha\right)\cdot c_e\left(x,u_{plant},w\right),\\&+c_q\left(x,u_{plant},w\right)\end{aligned} \tag{25}$$

and formed from the cost functions in (17), (18), and (21). The parameter α is used to select the relative weighting of emissions versus fuel consumption.

IV. Controller Synthesis via Approximate SP-SDP

While an SP-SDP controller could be designed that worked on the full state and action space defined in (1) and (9), the SP-SDP controller design problem is greatly simplified by first creating a low-level controller. In this case, the low-level controller is a static mapping from a set of synthetic high-level inputs, $\mathcal{U}$ to the plant inputs, $\mathcal{U}_{plant}$.

A. *Simplified Action Set via a Low-Level Controller*

To generate the high-level controller, the state and action spaces are spatially discretized; this is referred to as sampling. The sample points are used to generate the value function in (24). It is desirable to have a minimal number of samples since this minimizes the time required to generate the controller. If the constraints on the actions reduce the dimension of the action space, it is best to work in the reduced dimensional action set.

The low-level controller reduces the dimension of the high-level controller's action space by using the constraints in (14).

This dramatically reduces the complexity of the high-level controller and the time required to design it. This method was used rather than two other alternatives considered. The first alternative considered was to sample the action set and remove all points that fail to satisfy (14). This approach does not work well because some of the constraints in (14) are equality constraints. If the sampling is performed without consideration of these equality constraints, the probability of sampling feasible points is very small. The second alternative considered was to construct the sampling so that the constraints in (14) were satisfied by each sample. This approach increased the complexity of the algorithms used to create and solve the stochastic dynamic program because of the complex sampling needed for the action set. The introduction of a low-level control eliminated the problem with infeasible samples and increased complexity in the stochastic dynamic programs.

When the engine is commanded on and the transmission mode is fixed, the system constraints limit the choices in T_e, T_A, T_B, and T_k to satisfy the acceleration command and brake torque restrictions. The system constraints in (15) are always active, therefore they act as equality conditions that the action choices must satisfy. Since there are four variables and two equality conditions, this results in two independent decisions available to the high-level controller. While these independent decisions can be represented in many ways, in this work, the next engine speed and battery power were selected to represent the choices available to the high-level controller when the engine is on and the transmission mode has been selected.

When the engine is commanded off, the engine speed and torque are constrained to equal zero. These two additional constraints limit the high-level control choice to the transmission mode command.

The high-level controller action set $\mathcal{U}$ is

$$\mathcal{U} = \left\{ \begin{array}{c} \left(M_{e,cmd} = \text{'eng off'}, M_{trn,cmd} = \text{'mode 1'}\right), \\ \left(M_{e,cmd} = \text{'eng off'}, M_{trn,cmd} = \text{'mode 2'}\right), \\ \left(M_{e,cmd} = \text{'eng on'}, M_{trn,cmd} = \text{'mode 1'}, \omega_{e,cmd}, P_{batt,cmd}\right), \\ \left(M_{e,cmd} = \text{'eng on'}, M_{trn,cmd} = \text{'mode 2'}, \omega_{e,cmd}, P_{batt,cmd}\right) \end{array} \right\}, \tag{26}$$

where $M_{trn,cmd}$ commands a transmission mode, $M_{e,cmd}$ commands the engine on or off, $\omega_{e,cmd}$ is the engine speed command, and $P_{batt,cmd}$ is the battery power command.

The low-level controller is a static mapping from $\mathcal{U}$ and $\mathcal{X}$ to $\mathcal{U}_{plant}$:

$$P_{ll} : \mathcal{U} \times \mathcal{X} \to \mathcal{U}_{plant}\,. \tag{27}$$

The mapping P_{ll} is composed of six scalar functions which map the high-level actions to the plant actions. This mapping, $P_{ll}(x,u)$, is defined as

$$P_{ll}(x,u) = \begin{bmatrix} P_{ll,M_{trn}}(x,u) \\ P_{ll,M_e}(x,u) \\ P_{ll,T_e}(x,u) \\ P_{ll,T_A}(x,u) \\ P_{ll,T_B}(x,u) \\ P_{ll,T_k}(x,u) \end{bmatrix}. \tag{28}$$

For many high-level actions, u, the low-level controller may not be able to find a feasible plant action for the desired engine speed and battery power. For example, if the driver commands an acceleration of five meters per second squared which requires two hundred kilowatts at the wheels and the high-level controller commands the battery to charge at fifty kilowatts, a two hundred kilowatt engine can not generate sufficient power to satisfy both the driver and the high-level controller commands. Furthermore, some acceleration commands from the driver may require the engine to operate, while the high-level action specifies that the engine is off. This can occur is the acceleration commanded by the driver exceeds the electrical capability of the powertrain. These conflicts are resolved by introducing a set of constraints which are satisfied if a feasible plant action exists that matches the commanded modes, engine speed and battery power. Otherwise, these constraints are relaxed until a feasible plant action is found. Let

$$g_{ll}\left(x,u,u_{plant}\right) \le 0 \tag{29}$$

be the inequality that is satisfied when a high-level command $u \in \mathcal{U}$ is satisfied by an action $u_{plant} \in \mathcal{U}_{plant}$ for a state $x \in \mathcal{X}$. More precisely, Let

$$g_{ll}\left(x,u,u_{plant}\right) = \begin{bmatrix} I_{false}\left(M_{trn,cmd} = M_{trn}\right) \\ I_{false}\left(M_{e,cmd} = M_e\right) \\ \left|\omega_{e,cmd} - f_{\omega_e}\left(x,u_{plant}\right)\right| \cdot I_{true}\left(M_{e,cmd} = \text{'eng on'}\right) \\ \left|P_{batt,cmd} - h_{P_{batt}}\left(x,u_{plant}\right)\right| \cdot I_{true}\left(M_{e,cmd} = \text{'eng on'}\right) \end{bmatrix}, \tag{30}$$

where $I_{false}(\cdot)$ returns a 1 if the argument is false and a zero if it is true and conversely, $I_{true}(\cdot)$ returns 1 is the argument is true and zero if it is false. Similar to the problem encountered with the system constraints in (15), the low-level controller needs to ensure that for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$, the set defined by the intersection of (30) and (16) is non-empty. To resolve this problem, a vector of slack variables similar to (15) is introduced. This vector, ε_{ll}, is used to define the set of feasible control actions available to the low-level controller. The resulting set is

$$U_{ll+sys+plant}(x,u) = \left\{ u \in U_{sys+plant}(x) \middle| g_{ll}\left(x,u,u_{plant}\right) + \varepsilon_{ll}^{*}(x,u) \le 0 \right\}, \tag{31}$$

where the vector $\varepsilon_{ll}^{*}(x,u)$ is selected to minimize the deviation from the commanded values in u, while insuring that for all $x \in \mathcal{X}$, the set $U_{ll+sys+plant}(x,u)$ is non-empty.

Many different schemes can be used to define $\varepsilon_{ll}^{*}(x,u)$. In this work, the elements in $\varepsilon_{ll}^{*}(x,u)$ are selected by sequentially relaxing individual constraints until a non-empty set results. The order of relaxation is: transmission mode, $M_{trn,cmd}$; the engine mode command, $M_{e,cmd}$; the battery power command, $P_{batt,cmd}$; and finally, the engine speed command, $\omega_{e,cmd}$. By allowing these relaxations, for every possible command in $\mathcal{U}$, a feasible plant action in $\mathcal{U}_{plant}$ is found.

Using the set $U_{ll+sys+plant}(x,u)$, the low-level controller selects the feasible plant action that minimizes the same instantaneous cost function used in the high-level controller:

$$P_{ll}(x,u)=\min_{u_{plant}\in U_{ll+sys+plant}(x,u)} E^{W}\left\{c_{plant}\left(x,u_{plant},w\right)\right\}. \quad (32)$$

Figure 4 illustrates the relations between the driver, the high-level controller, the low-level controller, and the plant.

B. The SP-SDP based Controller's View of the System

With the addition of the low-level controller, the entire model is simplified for design using SP-SDP. In summary, the state of the model is defined in (1). The inputs to the low-level controller are defined in (26). The propagation function for the combined plant and low-level controller is

$$f(x,u,w)=f_{plant}\left(x,P_{ll}(x,u),w\right). \quad (33)$$

The value of w, the stochastic input from the driver, is defined in (10). The high-level objective function, $c(x,u,w)$, is obtained by substituting (32) into (25) , resulting in

$$c(x,u,w)=c_{plant}\left(x,P_{ll}(x,u),w\right). \quad (34)$$

The value function used to design the controller is

$$V(x)=\min_{u\in\mathcal{U}}\left\{E^{W}\left(c(x,u,w)+V\left(f(x,u,w)\right)\right)\right\}, \quad (35)$$

with the optimal control law found using

$$\pi^{*}(x)=\arg\min_{u\in\mathcal{U}}\left\{E^{W}\left(c(x,u,w)+V\left(f(x,u,w)\right)\right)\right\}. \quad (36)$$

C. SDP Formulation and Solution

To synthesize the high-level controller, a SP-SDP problem is posed and solved. An approximate solution to the SP-SDP equation in (35) is found using linear programming. Briefly, the method followed is to

1) select a linear basis for the value function, in this case barycentric interpolation;
2) create a set for sampling the state, action, and noise spaces;
3) censor the sampling set to exclude those points known beforehand to not be visited;
4) sample the model to generate the information for construction of the full, primal linear program;
5) solve the linear program; and
6) create the feedback controller from the approximate value function.

These six steps are described in detail in the following.

1) Linear Basis Selection: Barycentric Interpolation

The difficulty of using discretization to solve a continuous state and action space SDP problem has been extensively studied [36, 37]. The approach taken in this work is to use linear programming [25, 38]. The value function is approximated using a linear basis:

$$V(x)\approx\Phi(x)\cdot r. \quad (37)$$

The basis function, $\Phi(x)$, maps every point in $\mathcal{X}$ to a row vector which in general has a higher dimension than $\mathcal{X}$. The elements in the column vector r scale each basis function in $\Phi(x)$ to approximate $V(x)$.

There are many possible linear bases. A few bases studied for approximate linear programming include multi-linear interpolation [12], barycentric interpolation [30], splines [39-41] , radial bases [42-45], polynomials [46], wavelets [47, 48], and coarse coding [37, 49, 50]. Experiments were performed using each of these bases. A barycentric basis is used in this work because it was found to provide the best tradeoff between implementation complexity and performance.

A barycentric basis is formed by selecting a set $\mathcal{T}=\{t_1,t_2,\cdots,t_M\}$ of M points in $\mathcal{X}$. The notation, t_i, is used because of similarities between the behavior of the points used to form the simplices and knots in a spline. The points t_i should not be confused with time. The M points are used to partition a convex region into simplices which have mutually exclusive interiors. For each point in $\mathcal{T}$, a basis function is created from the product of an indicator function and a function which resolves the barycentric coordinates. The basis is

$$\Phi(x)=\begin{bmatrix} I_1(x)\cdot f_1(x) \\ I_2(x)\cdot f_2(x) \\ \vdots \\ I_M(x)\cdot f_M(x) \end{bmatrix}. \quad (38)$$

The function I_i is an indicator function that is equal to one if x is on the interior or boundary of any simplex that has the i^{th} point in $\mathcal{T}$ at one of its vertices. Otherwise, $I_i(x)$ is equal to zero. The function f_i returns p, the barycentric coordinate of x, with respect to the i^{th} point in $\mathcal{T}$. For a space with N dimensions, the barycentric coordinates will be a vector with $N+1$ dimensions. If a point x_0 is in the interior or on the boundary of a simplex with vertices at the points t_{i_1} through $t_{i_{N+1}}$, then the barycentric coordinates are found by solving

$$\begin{bmatrix} 1 \\ \hline x_0 \end{bmatrix}=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \hline t_{i_1} & t_{i_2} & \cdots & t_{i_{N+1}} \end{bmatrix}\cdot[p]. \quad (39)$$

To formally define the functions, f_i, let

$$T(x,t)=\begin{cases} 1 & ,\text{if } t \text{ is a corner in the simplex containing } x \\ 0 & ,\text{otherwise} \end{cases}, \quad (40)$$

be an indicator function. Let I_i be defined as

$$I_i(x)=T(x,t_i). \quad (41)$$

Let

$$\mathcal{S}(x)=\left\{t\in\mathcal{T}\,\middle|\,T(x,t)=1\right\} \tag{42}$$

be the set of vertices for the simplex containing x. The function f_i, is defined as

$$f_i(x)=\left\{p_1\left|\begin{array}{l}\begin{bmatrix}p_1\\p_2\\\vdots\\p_{N+1}\end{bmatrix}=\begin{bmatrix}1 & 1 & \cdots & 1\\\hline t_i & s_1 & \cdots & s_N\end{bmatrix}^{-1}\cdot\begin{bmatrix}1\\\hline x\end{bmatrix}\\ s_j\in\mathcal{S}(x),s_j\neq t_i,\forall j\in\{1,2,\cdots,N\}\\ s_j\neq s_k,\forall j\neq k\end{array}\right.\right\}. \tag{43}$$

Substituting (41) and (43) into (38) creates the barycentric basis.

While MATLAB provides routines for barycentric interpolation, in practice those routines were unacceptably slow. For this work, custom codes were generated that executed approximately ten times faster. These codes achieved this speed improvement by precomputing a lookup table, or hash table, to identify small sets of potentially active simplices.

2) State, Action, and Noise Sample Selection

To reduce this problem to an approximate linear program, the state space, $\mathcal{X}$, is sampled. Let the set,

$$\mathcal{X}_{samples,full}=\{x_1,x_2,\cdots,x_M\}\subset\mathcal{X}, \tag{44}$$

be the set of all sample points to be considered. These points are chosen on a regular grid such that the state constraints implied by (8) are satisfied. These state constraints include battery charge, engines speed, and vehicle speed limits.

The action sample set is more complex than the state space sample set because the dimension of the actions changes. In the high-level action set $\mathcal{U}$, defined in (26), there are four discrete choices in actions formed from the combination of engine mode and transmission mode commands. In the cases where the engine is commanded on, the battery power and engine speeds are sampled on a regular grid.

Finally, the sampling for the noise space, $\mathcal{W}_{samples}=\mathcal{W}$, is identical to the sampling used to generate the driver model.

3) State Censoring

When building the driver model, there are typically combinations of vehicle speed and acceleration that are never encountered. Figure 5 illustrates the velocity and acceleration pairs that are visited for the UDDS and HWFET cycles. In this figure, the grey region shows where velocity and acceleration pairs occur. The white space is the region where no combination of velocity and acceleration is observed. Since some combinations are never seen, there is no need to design the controller to optimize in these regions. State censoring uses this concept to reduce the number of states used when constructing the controller. Let

$$\mathcal{R}=\left\{x\in\mathcal{X}\,\middle|\,\exists a_{next}\text{ with }\Pr\left(a_{next}\,\middle|\,v_{veh},a_{now}\right)>0\right\} \tag{45}$$

be the set of all points in the state space where the combination of velocity and acceleration have a nonzero probability of occurring.

Typically, the intersection of the convex hull of $\mathcal{R}$ with $\mathcal{X}_{samples,full}$ will be a strict subset of $\mathcal{X}_{samples,full}$. In other words, the hypercube bounded by possible vehicle speeds and possible accelerations will contain combinations which the driver will never request or the vehicle is not capable of realizing. If this is the case, then the size of the sample set can be reduced by selectively removing the sample points with zero probability of occurring. This act of going through the list of samples and removing those with zero probability of occurring is termed censoring. More formally, let

$$\mathcal{X}_{samples}=\mathcal{X}_{samples,full}\cap\operatorname{co}(\mathcal{R}). \tag{46}$$

In practice, the number of samples in $\mathcal{X}_{samples}$ is up to 40% smaller than the number of samples in $\mathcal{X}_{samples,full}$. For these numerical examples, censoring reduced the time to construct and solve the SP-SDP equations by more than 50%.

4) Model Call Reduction

The model is sampled based on the points identified in the sets $\mathcal{X}_{samples}$, $\mathcal{U}_{samples}$, and $\mathcal{W}_{samples}$. A simple sampling scheme would use each element in the set formed from the Cartesian product of

$$\mathcal{G}_{samples}\triangleq\mathcal{X}_{samples}\times\mathcal{U}_{samples}\times\mathcal{W}_{samples}. \tag{47}$$

Another significant improvement in execution time was achieved by taking advantage of linearity in the propagation function, (5), with respect to battery charge and the drivers stochastic input, w. This can be seen be creating a linearized form of this equations as

$$f_{plant}(x_k,u_k,w_k)=\begin{bmatrix}f_{v_{veh}}(x_k,u_k,w_k)\\q_{batt}(x_k)\\f_{\omega_e}(x_k,u_k,w_k)\\f_{T_{cat}}(x_k,u_k,w_k)\\0\end{bmatrix}+\begin{bmatrix}0\\\Delta_{q_{batt}}(x_k,u_k,w_k)\\0\\0\\w_k\end{bmatrix}. \tag{48}$$

In this case, the value of a_{now} at the next time step is independent of the all states and actions. Its value is wholly determined by the stochastic input, w. Therefore, instead of calling the full model to evaluate the next value of a_{now}, a simple linear model is used in its place. The battery state of charge in this model has a similar property. The change in battery charge, $\Delta_{q_{batt}}(x_k,u_k,w_k)$, is independent of the battery charge. This allows the model to be called once for any samples in (47) that differ only in the value of q_{batt}.

Therefore, for each collection of elements in $\mathcal{G}_{samples}$ where all values are identical except q_{batt} and w, the propagation function, (33), is only called for a single point and linearity is used to determine the next state at all other points.

Recall that in (7), the variable w is retained for generality but is not used here for the calculation of the outputs. Because of this, the cost function in (25) can be sampled in an economical manner by only sampling elements in (47) that are unique in all terms except w. Similarly, the transition probabilities, (10) and (11), are only sampled once for each unique combination of a_{next} , a_{now} , and v_{veh} in (47).

This selective sampling and reconstruction of model functions using linearity increased the speed between ten and one hundred times.

5) Primal LP Construction

The first step in solving an approximate SP-SDP problem is to formulate the value function using a linear basis. Substituting (37) into (35) results in a value function in terms of the linear basis:

$$\Phi(x)\cdot r=\min_{u\in\mathcal{U}}\left\{E^{W}\left(c(x,u,w)+\Phi\left(f(x,u,w)\right)\cdot r\right)\right\}. \quad (49)$$

The next step is to expand the expectation in (49) with respect to the sampling of w . Recall from (1) and (12) the definitions of x and w respectively, let

$$\Pr_0(x,w)=\Pr\left(a_{next}\,|v_{veh},a_{now}\right)\cdot\Pr_{on}\left(v_{veh},a_{now}\right) \quad (50)$$

be the probability that the driver transitions to a_{next} and continues to drive given the current speed, v_{veh} , and acceleration, a_{now} . For notational convenience, let

$$c_0(x,u,w,r)=c(x,u,w)+\Phi\left(f(x,u,w)\right)\cdot r\,. \quad (51)$$

Substituting (51) into the approximate value function in (49) and expanding the expectation into a sum over the samples in $\mathcal{W}_{samples}$ gives

$$\Phi(x)\cdot r=\min_{u\in\mathcal{U}}\left\{\sum_{w\in\mathcal{W}_{samples}}\Pr_0(x,w)\cdot c_0(x,u,w,r)\right\}. \quad (52)$$

In (52), the optimization is over the entire set defined by $\mathcal{U}$. To convert (52) into a form that can be solved using linear programming, u and x are restricted to a finite set of values. Only considering u at the points in $\mathcal{U}_{samples}$ and x at the points defined in $\mathcal{X}_{samples}$, allows (52) to be replaced by a set of inequalities,

$$\Phi(x)\cdot r\le\sum_{w\in\mathcal{W}_{samples}}\Pr_0(x,w)\cdot c_0(x,u,w,r), \quad (53)$$

with one equation for each unique pair of x and u . The equation in (53) is algebraically manipulated to collect the term, r , resulting in the inequality

$$\begin{aligned}&\left(\Phi(x)-\left(\sum_{w\in\mathcal{W}}\Pr_0(x,w)\cdot\Phi\left(f(x,u,w)\right)\right)\right)\cdot r\\&\le\sum_{w\in\mathcal{W}_{samples}}\Pr_0(x,w)\cdot c(x,u,w)\end{aligned}. \quad (54)$$

To create the linear program (LP), let

$$A(x,u)=\Phi(x)-\sum_{w\in\mathcal{W}_{samples}}\Pr_0(x,w)\cdot\Phi\left(f(x,u,w)\right), \quad (55)$$

and let

$$b(x,u)=\sum_{w\in\mathcal{W}_{samples}}\Pr_0(x,w)\cdot\left(c(x,u,w)\right). \quad (56)$$

Equations (55) and (56) define a row of linear constraints in the LP that correspond to a single state and action. Let u^i be the i^{th} element of $\mathcal{U}_{samples}$ and define

$$A_i(x)=A\left(x,u^i\right) \quad (57)$$

and let

$$b_i(x)=b\left(x,u^i\right). \quad (58)$$

Equations (57) and (58) define a row of linear constraints in the LP that correspond to $u^i\in\mathcal{U}_{samples}$ and an arbitrary state, x . Let x^j be the j^{th} element of $\mathcal{X}_{samples}$ and define

$$A_{ij}=A_i\left(x^j\right) \quad (59)$$

and

$$b_{ij}=b_i\left(x^j\right). \quad (60)$$

Equations (59) and (60) define a row of linear constraints in the LP that correspond to the ith element of the M_u elements in $\mathcal{U}_{samples}$ and the jth element of the M_x elements in $\mathcal{X}_{samples}$. Using this notation, the primal LP is

$$\left\{\begin{array}{l}r^*\in\arg\max\sum r\\s.t.\\A\cdot r\le b\end{array}\right\}, \quad (61)$$

where

$$A=\left[A_{11}^T\ \cdots\ A_{M_u1}^T\ \ A_{12}^T\ \cdots\ A_{M_u2}^T\ \ldots\ A_{M_uM_x}^T\right]^T \quad (62)$$

and

$$b=\left[b_{11}^T\ \cdots\ b_{M_u1}^T\ \ b_{12}^T\ \cdots\ b_{M_u2}^T\ \ldots\ b_{M_uM_x}^T\right]^T. \quad (63)$$

This above procedure used to develop the LP is similar to the methods discussed in [25, 34, 35, 38]. For a discussion of the meaning of the objective function in (61) with respect to the optimality properties of the approximate value function, see [25, 38].

6) Solution of the Primal LP

As formed, the primal LP has thousands of variables and hundreds of thousands of constraints, requiring a significant amount of time and memory to solve. While this problem is numerically tractable, an interior point algorithm repeatedly stalled or required significant tuning in our numerical experiments to find a solution. To decrease the amount of time and memory needed to solve the LP, two approaches are used: the constraint generation algorithm from [26] and the solution of the resulting primal LP using a dual from [31, 51]. Furthermore, the dual solution approach from [31] is refined by using knowledge of the actions and states along with the Lagrangian from the solution of the dual LP to select the best action for each state.

Constraint generation relies on the insight that the vast majority of constraints in an LP are not active. Furthermore, in general, solution time for an LP scales super linearly with problem size. Doubling the number of constraints in an LP more than doubles the solution time. Therefore, if the problem size can be reduced by omitting inactive constraints, a solution can be found faster. Constraint generation involves iteratively solving a sequence of LP's. The first LP is constructed with enough constraints to ensure the problem is bounded. On each iteration, the previous solution is used to identify the most violated constraint associated with the actions for each state. These constraints are added to the existing LP to form a new LP. This process is repeated until a solution is found without any constraint violations. This algorithm is guaranteed to converge. At worst case, the algorithm iterates until all constraints in the original LP are used. Experimentally, this algorithm can be several times faster than solution of the original LP [27].

To solve the LP at each iteration in the constraint generation algorithm, the dual of the LP is solved and the primal solution constructed from that solution. Recall that the primal problem being solved is

$$\begin{Bmatrix} \max \kappa^T \cdot r \\ s.t. \\ A \cdot r \le b \end{Bmatrix}, \tag{64}$$

where $\kappa = \underline{1}$. The symbol κ is used rather than the conventional notation of c to avoid confusion with (34). The dual of this LP is

$$\begin{Bmatrix} \min b^T \cdot \lambda \\ s.t. \\ A^T \cdot \lambda = \kappa \\ \lambda \ge 0 \end{Bmatrix}. \tag{65}$$

In conventional linear programming, where $r \in \mathbb{R}^M$, the M largest values of λ are used to identify the rows of A and b that are active constraints. These rows are collected into a simpler matrix and vector forming A_λ and b_λ respectively. The value of r is found by solving

$$A_\lambda \cdot r = b_\lambda . \tag{66}$$

However, because of the structure in this problem, an alternative approach is used. The values of λ associated with each state in $\mathcal{X}_{samples}$ are considered. For each state, a single action is selected as the active action. This active action is the largest value of λ. The rows in the LP associated with the active actions are used to form a system of linear equations. These equations are used to find r. This approach resolved problems that occurred where the M largest values of λ resulted in polices with no actions in some states and multiple action in other states.

The vector r^* is an argument that satisfies (64) and is obtained on the last iteration of the constraint generation algorithm. The algorithm terminates with the selection of r^*.

D. *The SP-SDP feedback controller*

The high-level feedback controller uses the approximate value function found by solving (61). The resulting controller, valid for all $x \in \mathcal{X}$, is

$$\pi^*(x) = \min_{u \in \mathcal{U}} E^W \left(c(x,u,w) + \Phi\left(f(x,u,w) \right) \cdot r^* \right). \tag{67}$$

As opposed to using the discrete set, $\mathcal{U}_{samples}$, as is the case where solving for the approximate value function, the optimal action choice is found by searching the entire action space. In (67), neither monotonicity [20, 52] nor convexity properties [51] can be guaranteed, therefore a global optimization method is used to find u. The direct algorithm [53] was found to provide a robust solution.

V. Value Function Solution Discussion

To formulate a specific SP-SDP based controller, the function c in (67) must be defined. This function is formed by applying the low-level controller to the cost function for the plant as shown in (34). The plant cost function is formed from the linear combination of the fuel consumption, the tailpipe emissions, and the cost function for SOC deviations as shown in (25). The SOC cost function has two tuning variables, K_q and q_0. For this work, the value of K_q was set to 25 and q_0 was set to 0.55. These values were selected by experimentation on a fuel minimizing controller to provide charge sustaining operation while keeping the SOC less than q_0. By keeping the SOC less than q_0 during a drive cycle, the cost of battery energy is always penalized to reflect the equivalent cost of future energy replacement. Furthermore, problem solution requires definition of the samples in (47). For this work, the set of samples, prior to the use of state censoring, is obtained by using a regular grid. The vehicle speed is sampled at

$$v_{veh} \in \{0, 5.7, 11.4.17.1, 22.9, 28.6, 34.3, 40\} \tag{68}$$

in units of meters per second. The battery SOC is sampled at

$$q_{batt} \in \{0.4, 0.433, 0.467, 0.5, 0.533, 0.567, 0.6\}. \tag{69}$$

The driver's acceleration command is sampled at

$$a_{now} \in \{-5, -3.75, -2.5, -1.25, 0, 1.25, 2.5, 3.75, 5\} \tag{70}$$

in units of meters per second squared. The engine speed is sampled at

$$\omega_e \in \{0, 100, 200, 300, 400, 500, 600\} \tag{71}$$

in units of radians per second. Finally, the catalyst brick temperature is sampled at

$$T_{cat} \in \{38, 132, 221, 315, 410, 499, 593\} \tag{72}$$

in units of degrees Celsius.

The stochastic input w is sampled identically to a_{now}. The action space for the high-level controller is sampled with battery power commands of

$$P_{batt,cmd} \in \{-40000, -24375, -8750, 0, 6875, 22500\} \tag{73}$$

in units of watts. The engine speed command is sampled with values of

$$\omega_{e,cmd} \in \{100, 200, \cdots, 600\} \tag{74}$$

in units of radians per second. To form the high-level actions, the samples in (73) and (74) are used to select discrete elements in (26). This results in a total of 62 actions per state sample. Using the procedure outlined in section 3.4.3, the approximate value function was found.

Inspection of the value function showed there was no appreciable change with respect to the engine speed. This insensitivity stems from the fact that in the model, the engine speed can be rapidly changed with minimal energy. Based on this observation, the engine speed was omitted from the feedback to the high-level controller, thus reducing the dimension of the SP-SDP and decreasing solution time. Note that, while engine speed has been removed from the feedback for the high-level controller, it is still used by the low-level controller in (32).

After removal of the engine speed from the value function, the initial state sampling included 3528 state values. After state censoring, this sample set was reduced to 1568 state values, a 55% reduction. Without state censoring and using a barycentric basis with a node at each state sample, this problem generates an LP with approximately 220,000 constraints and 3528 variables. An LP this size was not directly solvable using a desktop machine. By a combination of state censoring and constraint generation, the largest LP solved had 3396 constraints and 1569 variables. The constraint generation algorithm converged to a solution in 5 iterations.

The initial combination of state samples, action samples, and noise samples required a total of approximately 1.97 million model calls. After state censoring, this was reduced to 874,944 model calls. This was further reduced to less than 20,000 model calls through the use of model linearity.

To solve the SP-SDP, a total execution time of just over 3 hours was required on a single CPU 3.0 GHz Pentium using MATLAB. Initialization of the hash table for barycentric interpolation required 40 minutes. The time to sample the model was 65 minutes. The construction of the rows in the LP from the model samples took a total of 40 minutes. The solution of the LP using constraint generation and solution via duality required less than 35 minutes. The peak memory utilization by the MATLABimage on a Windows PC was 174 Mb.

These techniques decreased the time to find a solution to the SP-SDP from an initial estimate of greater than eight thousand hours to less than three hours.

VI. The Baseline Controller

To understand the value of the SP-SDP approach to optimizing emissions and fuel consumption, a baseline controller is constructed from (24) where the instantaneous cost, $c(\cdot)$, is (25). The value of alpha in (25) is evaluated at various values to determine the tradeoffs between fuel consumption and emissions provided by this controller. The value of K_q is 50,000 and q_0 is set to 0.60. Both values were selected through experimentation to achieve charge sustaining operation over a range of driving conditions. The value function is replaced by zero. This controller is referred to as the 'baseline' controller:

$$\pi_{baseline}(x) = \underset{u_{plant} \in \mathcal{U}_{sys+plant}(x)}{\arg\min} \left\{ E^W \left(c\left(x, u_{plant}, w\right)\right)\right\}. \tag{75}$$

VII. Fuel and Emissions Results

Since the control goals are minimization of fuel and emissions in a certification testing scenario, the controllers are evaluated against a fixed drive cycle: the UDDS cycle. The Pareto set for the SP-SDP problem is generated by sweeping α in equation (25) and generating a unique controller for each point in the sweep. Similarly, the baseline controller is evaluated by sweeping alpha.

One of the objectives for this work was to understand the impact of catalyst heat loss, so Pareto sets were generated for several catalyst designs. The first catalyst design included no heat transfer directly from the catalyst to the environment. This catalyst is referred to as the 'isolated catalyst.' All heat generated in the catalyst is rejected into the exhaust gases. The evaluation started with the catalyst at 93 C. To equalize the impact of changes in battery charge on fuel consumption and emissions, the SOC was searched to find an initial value that resulted in charge sustaining operation. Figure 6 shows the fuel consumption and tailpipe emissions for the baseline controller and the SP-SDP controller. The utopia point [54] is plotted to highlight the best single objective optimization results obtained.

The best baseline controller for both emissions and fuel economy occurs when tuned for fuel economy operation. Tuning this baseline controller for any emissions minimization degraded both fuel economy and emissions performance. This happens because the control law is greedy [37]: it optimizes instantaneous decisions without consideration of future costs. Specifically, as tailpipe emissions were considered, the instantaneous cost prevented the engine from being used, this discharged the battery. Eventually, to achieve charge sustaining operation, the battery forces the engine on in a manner that increases fuel use and emissions.

The SP-SDP controller reduced the normalized tailpipe emissions from a peak of 44.5 grams to a minimum of 26.1 grams, more than a 41% reduction. The tradeoff between fuel economy and emissions is clearly visible. For example, these plots illustrate that for the 'isolated catalyst,' the tailpipe emissions can be reduced more than 20% for about a 3% increase in fuel consumption.

To assess the impact of heat loss on the tailpipe emissions, a catalyst with a thermal time constant of 1725 seconds for heat transfer to the ambient was evaluated. The initial conditions used were the same as those for the 'isolated catalyst'. The results for the controllers designed for this catalyst are shown in Figure 7.

For the SP-SDP controller, the heat loss from the catalyst to the ambient does not affect the minimal emissions achieved by this system, however almost 20% more fuel is required to achieve the minimum emissions level than with the 'isolated catalyst'.

Because of the heat loss, the baseline controller is not able to match the emissions level for the isolated catalyst. For both controllers, the minimal fuel consumption is the same as for the isolated catalyst, as would be expected.

Finally, a catalyst with a 400 second time constant for heat transfer to the ambient was evaluated. This evaluated the impact of a heat transfer much higher then discussed in the literature [55, 56] and is used to establish bounds on the performance of the controllers. Figure 8 illustrates this Pareto set.

The SP-SDP controller is able to beat the minimum emissions from the baseline controller more than 60%. Interestingly, in this case, the baseline controller demonstrates ability to tradeoff emissions and fuel economy. Across the

Pareto set, the SP-SDP controller shows significant performance advantages. As expected, for the case where only fuel economy is considered, both controllers have equivalent performance.

One effect that occurred in all three studies was a slight increase in emissions from the minimum as the controllers were increasingly weighted for emissions. This was investigated by examining higher resolution sampling of the state space. As also shown in Figure 8, increased resolution in sampling the catalyst temperature and battery charge decreased this effect. The root cause of this problem appears to be related to the controller attempting to drive the catalyst to the edge of the state space and encountering numerical issues.

VIII. Conclusions

Because automobiles are always operated for a finite period of time, SP-SDP is a natural formulation for developing controllers that minimize total undiscounted costs. This formulation is important when considering control objectives like fuel consumption and emissions. The SP-SDP formulation is able to minimize the expected costs over an infinite horizon because the cost is kept finite by the fact that the vehicle will be turned off in finite time.

Compared to a baseline controller, a SP-SDP controller was able to provide significantly better performance and tradeoffs between emissions and fuel consumption. The improved performance is obtained through better control of the catalyst as it warms up and transitions from a low conversion efficiency operating point to a high efficiency operating point. Since the heat transfer between the catalyst and the environment is not well documented, the ability to improve performance for a range of catalysts was explored. For each case, a unique collection of SP-SDP controllers were developed. The SP-SDP based controller was able to decrease tailpipe emissions and provide better tradeoffs with fuel consumption than a baseline controller. This capability was demonstrated over a broad range of thermal characteristics in catalysts.

SP-SDP methods were successfully used to find a controller that uses four of five states in feedback, with the other state managed by a low level controller. A low level controller was synthesized that reduced the complexity in designing the high level controller using SP-SDP. The SP-SDP design technique synthesized a controller in three hours on a desktop PC. The combination of dynamic programming, approximate linear programming, constraint generation, solution of the LP via its dual, reduced sampling, state censoring, and barycentric interpolation reduced the amount of time required to find the value function for the SP-SDP by orders of magnitude from initial efforts. This short design time enabled the generation of large sets of controllers for generating Pareto sets against a range of component characteristics.

Appendix A
Optimality of a Quadratic SOC Penalty

As implied by [57, 58], if a hard charge sustaining constraint is replaced by a linear cost that is applied when the vehicle is turned off,

$$c_{term} = K_{term} \cdot \left(q_{batt}\left(T_{off}\right) - q_{batt}\left(T_{start}\right) \right), \tag{76}$$

then the terminal cost (76) can be replaced by a summed instantaneous cost, per

$$\begin{aligned} &K_{term} \cdot \left(q_{batt}\left(T_{off}\right) - q_{batt}\left(T_{start}\right) \right) = \\ &K_{term} \cdot \left(\sum_{t=T_{start}}^{T_{off}-\Delta t} q_{batt}\left(t+\Delta t\right) - q_{batt}\left(t\right) \right). \end{aligned} \tag{77}$$

This substitution changes the terminal constraint into an instantaneous cost,

$$c_{q,0}\left(x,u,w\right) = K_{term} \cdot \left(q_{batt,next}\left(x,u,w\right) - q_{batt}\left(x\right) \right), \tag{78}$$

where $q_{batt,next}\left(\cdot\right)$ is the battery charge at the next time step and K_{term} is a gain selected to achieve the termination objective. A drawback of this approach is that the value of K_{term} required in (76) to achieve charge sustaining operation is cycle dependent. The approach of using a linear cost to solve the charge sustaining constraint has been termed an Energy Consumption Minimization Strategy (ECMS) [9, 59]. An alternative criterion that approaches ECMS optimality is to use a quadratic cost function and use changes in battery SOC to find a local slope approximately equal to a cycle specific K_{term}. Consider the instantaneous cost function

$$c_q\left(x,u,w\right) = K_q \cdot \left(q_{next}\left(x,u,w\right) - q_0 \right)^2, \tag{79}$$

where q_0 is a target battery charge where no penalty is applied. Let

$$\Delta q_{batt}\left(x,u,w\right) = q_{batt,next}\left(x,u,w\right) - q_{batt}\left(x\right) \tag{80}$$

be a function that returns the change in battery charge at a given state, x, when the action u is applied and the disturbance w occurs. By substitution of (80) into (78), the cost function becomes,

$$c_q\left(x,u,w\right) = K_q \cdot \left(\Delta q_{batt}\left(x,u,w\right) + q_{batt}\left(x\right) - q_0 \right)^2. \tag{81}$$

Through expansion of the quadratic term, (81) becomes

$$\begin{aligned} &c_q\left(x,u,w\right) = K \cdot \left(q_{batt}\left(x\right)^2 - 2 \cdot q_{batt}\left(x\right) \cdot q_0 + q_0^2 \right) \\ &+K \cdot \left(\Delta q_{batt}\left(x,u,w\right)^2 - 2 \cdot \Delta q_{batt}\left(x,u,w\right) \cdot \left(q_{batt}\left(x\right) - q_0 \right) \right). \end{aligned} \tag{82}$$

In operation, the change in battery charge between any two instants, $\Delta q_{batt}\left(x,u,w\right)$, is much smaller than $q_{batt}\left(x\right)$ or q_0. This allows the term $\Delta q\left(x,u,w\right)^2$ to be dropped from (82), resulting in

$$\begin{aligned} &c_q\left(x,u,w\right) \approx c_{\tilde{q}}\left(x,u,w\right) = \\ &\qquad K \cdot \left(q_{batt}\left(x\right)^2 - 2 \cdot q_{batt}\left(x\right) \cdot q_0 + q_0^2 \right) \\ &\qquad -2 \cdot K \cdot \left(\Delta q_{batt}\left(x,u,w\right) \cdot \left(q_{batt}\left(x\right) - q_0 \right) \right). \end{aligned} \tag{83}$$

Without loss of generality assume u is scalar. Consider the partial derivative of $c_{\tilde{q}}$ in (83) with respect to u after expanding Δq_{batt}:

$$\frac{\partial c_{\tilde{q}}}{\partial u} = -2 \cdot K \cdot \left(q_{batt}\left(x\right) - q_0\right) \cdot \frac{\partial q_{batt,next}\left(x,u,w\right)}{\partial u}. \tag{84}$$

Consider the partial derivative $c_{q,0}$ in (78) with respect to u:

$$\frac{\partial c_{q,0}}{\partial u} = K_{term} \cdot \frac{\partial q_{batt,next}\left(x,u,w\right)}{\partial u}. \tag{85}$$

If $\left(-2 \cdot K \cdot \left(q_{batt}\left(x\right) - q_0\right)\right)$ is equal to K_{term}, then

$$\arg\min_u c_{q,o}\left(x,u,w\right) = \arg\min_u c_{\tilde{q}}\left(x,u,w\right). \tag{86}$$

In other words, if $\left(-2 \cdot K \cdot \left(q_{batt}\left(x\right) - q_0\right)\right)$ equals K_{term}, then a value of u that minimizes (78) also minimizes (83)

If the peak to peak change in $q_{batt}\left(x\right)$ during operation is small and $\left(-2 \cdot K \cdot \left(q_{batt}\left(x\right) - q_0\right)\right)$ is approximately equal to K_{term}, then (83) can be used in an optimization to select u and get results that are close to the results from using (78). Furthermore, the average value of $q_{batt}\left(x\right)$ acts as an adaptation. For q_0 greater than $q_{batt}\left(x\right)$, decreases in $q_{batt}\left(x\right)$ increase the penalty for battery usage. Conversely, increases in $q_{batt}\left(x\right)$ decrease the penalty for battery usage. This change in penalty provides ensures charge sustaining operation over different drive cycles.

References

[1] J. Adcock, B. Allen, R. Cleary, C. Dobbins, L. Grooms, G. Hayzen, S. Hutcheson, M. Johnson, B. Johnston, B. Jordan, C. Kukla, A. Lehner, C. Lela, B. McConkey, P. Perry, B. Quinton, R. Reece, S. Sheriff, Tony Spezia, J. Thompson, S. Tibbals, J. Welsh, J. Freeman, W. Hamel, and J. Hodgson, "Design and Construction of the University of Tennessee, Knoxville FutureTruck 2000/2001 Parallel Hybrid Vehicle," *SAE Paper No. 2002-01-1213,* 2002.

[2] S. Aylor, M. Parten, T. Maxwell, and J. Jones, "An Electrically Assisted, Hybrid Vehicle," in *Proceedings of the 1998 48th IEEE Vehicular Technology Conference*, Ottawa, Canada, 1998, pp. 2089.1-2089.6.

[3] P. Bowles, H. Peng, and X. Bang, "Energy Management in a Parallel Hybrid Electric Vehicle With a Continuously Variable Transmission," in *Proceedings of the American Control Conference*, Chicago, Illinois, 2000, pp. 55-59.

[4] V. H. Johnson, K. B. Wipke, and D. J. Rausen, "HEV Control Strategy for Real-Time Optimization of Fuel Economy and Emissions," *SAE Paper No. 2000-01-1543,* 2000.

[5] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, "A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management," *European Journal of Control,* vol. 11, pp. 509-524, 2005.

[6] A. Brahma, B. Glenn, Y. Guezennec, T. Miller, G. Rizzoni, and G. Washington, "Modeling, Performance Analysis and Control Design of a Hybrid Sport-Utility Vehicle," in *Proceedings of the IEEE International Conference on Control Applications*, Kohala Coast-Island of Hawaii, Hawaii, 1999, pp. 448-453.

[7] E. Tate, "Towards Finding the Ultimate Limits of Performance for Hybrid Electric Vehicles," *SAE Paper No. 2000-01-3099,* 2000.

[8] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang, "Power Management Strategy for a Parallel Hybrid Electric Truck," *IEEE Transactions on Control Systems Technology,* vol. 11, pp. 839-849, November 2003.

[9] A. Brahma, Y. Guezennec, and G. Rizzoni, "Optimal Energy Management in Series Hybrid Electric Vehicles," in *Proceedings of the American Control Conference*, Chicago, Illinois, 2000, pp. 60-64.

[10] M. Anatone, R. Cipollone, A. Donati, and A. Sciarretta, "Control-Oriented Modeling and Fuel Optimal Control of a Series Hybrid Bus," *SAE Paper No. 05P-256.*

[11] C.-C. Lin,"Modeling and Control Strategy Development for Hybrid Vehicles," Ph.D. dissertation, Dept. of Mechanical Engineering, University of Michigan, Ann Arbor, Michigan, 2004.

[12] C.-C. Lin, H. Peng, and J. W. Grizzle, "A Stochastic Control Strategy for Hybrid Electric Vehicles," in *Proceedings of the 2004 American Control Conference*, 2004, pp. 4710-4715 vol. 5.

[13] J. A. Cook, J. Sun, J. Buckland, I. V. Kolmanovsky, H. Peng, and J. W. Grizzle, "Automotive Powertrain Control - A Survey," *Asian Journal of Control,* vol. 8, pp. 237-260, September 2006.

[14] E. Tate, J. Grizzle, and H. Peng, "Shortest Path Stochastic Control for Hybrid Electric Vehicles," *Internation Journal of Robust and Nonlinear Control,* 2006.

[15] "California Code of Regulations, Zero Emission Vehicle Standards for 2003 and Subsequent Model Passenger Cars, Light Duty Trucks and Medium Duty Vehicles." vol. Title 13

[16] "California Code of Regulations Title 13," Rule Number 1962.

[17] "California Exhaust Emission Standards and Test Procedures for 2003 and Subsequent Model Zero Emission Vehicles and 2001 and Subsequent Model Hybrid Electric Vehicles in the Passenger Car, Light Duty Truck and Medium Duty Truck Classes," EPA and CARB, Eds., 1999.

[18] "California Exhaust Emission Standard and Test Procedures for 2005 and Subsequent Model Zero-Emission Vehicles, 2001 and Subsequent Model Hybrid Electric Vehicles, in the Passenger Car, Light-Duty Truck and Medium-Duty Vehicle Classes," EPA and CARB, Eds., 2003.

[19] A. Holmes, D. Klemen, and M. R. Schmidt, "Electrically Variable Transmission with Selective Input, Compound Split, Neutral and Reverse Modes," U.S. Patent 6 527 658,Mar 4, 2003.

[20] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design: Models and Computation*, 2 ed. New York, New York: Cambridge University Press, 2000.

[21] J. N. Tsitsiklis and B. Van Roy, "Feature Based Methods for Large Scale Dynamic Programming," *Machine Learning,* vol. 22, pp. 59-94, 1996.

[22] J. N. Tsitsiklis and B. Van Roy, "Feature Based Methods for Large Scale Dynamic Programming," in *34th Conference on Decision and Control*, New Orleans, LA, 1995, pp. 565-567.

[23] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis, "A Neuro-Dynamic Programming Approach to Retailer Inventory Management," in *36th Conference on Decision & Control*, San Diego, California, 1997, pp. 4052-4057.

[24] W. B. Powell and B. Van Roy, "Approximate Dynamic Programming for High-Dimensional Resource Allocation Problems," January 23 2003.

[25] D. P. de Farias and B. Van Roy, "The Linear Programming Approach to Approximate Dynamic Programming," *Operations Research,* vol. 51, pp. 850-865, November-December 2003.

[26] M. A. Trick and S. E. Zin, "A Linear Programming Approach to Solving Stochastic Dynamic Programs," Carnegie Mellon University 1993.

[27] M. A. Trick and S. E. Zin, "Spline Approximations to Value Functions: A Linear Programming Approach," *Macroeconomic Dynamics,* pp. 255-277, 1997.

[28] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-Dynamic Programming: An Overview," in *34th Conference on Decision and Control*, New Orleans, LA, 1995, pp. 560-564.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming.* Belmont, Mass: Athena Scientific, 1996.

[30] R. Munos and A. Moore, "Barycentric Interpolators for Continuous Space and Time Reinforcement Learning," *Advances in Neural Information Processing Systems,* vol. 11, pp. 1024-1030, 1998.

[31] V. F. Farias and B. Van Roy, "Tetris: Experiments with the LP Approach to Approximate DP," 2004.

[32] Jessy W. Grizzle's Publications, http://www.eecs.umich.edu/~grizzle/papers/auto.html.

[33] P. R. Kumar and P. Variaya, *Stochastic Systems: Estimation, Identification and Adaption*. Englewood Cliffs, New Jersey: Prentice Hall, 1986.

[34] D. Bertsekas, *Dynamic Programming and Optimal Control: Vol 2*. Belmont, Mass: Athena Scientific, 1995.

[35] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Vol 1*. Belmont, Mass: Athena Scientific, 1995.

[36] S. M. Lavelle, *Planning Algorithms*: Cambridge University Press, 2006.

[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Mass: MIT Press, 1999.

[38] D. P. de Farias,"The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application," Ph.D. Dissertation, Department of Management Science and Engineering, Stanford University, Palo Alto, Ca, 2002.

[39] V. C. P. Chen, D. Ruppert, and C. A. Shoemaker, "Applying Experimental Design and Regression Splines to High Dimensional Continuous State Stochastic Dynamic Programming," *Operations Research,* vol. 47, pp. 38-53, January-February 1999.

[40] V. C.-P. Chen,"Applying Experimental Design and Regression Splines to High Dimensional Continuous State Stochastic Dynamic Programs," Ph.D. Dissertation, Cornell University, Ithaca, NY, 1993.

[41] J. W. Daniel, "Splines and Efficiency in Dynamic Programming," *Journal of Mathematical Analysis and Applications,* vol. 54, pp. 402-407, 1976.

[42] T. Parsini and R. Zoppoli, "Radial Basis Functions and Multilayer Feedforward Neural Networks for Optimal Control of Nonlinear Stochastic Systems," pp. 1853-1858, 1993.

[43] M. Baglietto, C. Cervellera, T. Parisini, M. Sanguineti, and R. Zoppoli, "Approximating Networks, Dynamic Programming and Stochastic Approximation," in *Proceedings of the American Control Conference*, Chicago, Illinois, 2000, pp. 3304-3308.

[44] G. Lendaris, C. Cox, and R. Saeks, "A Radial Basis Function Implementation of the Adaptive Dynamic Programming Algorithm," in *Proceedings of the 45th Midwest Symposium on Circuits and Systems*, 2002, pp. II-338-341 vol 2.

[45] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive Dynamic Programming," *IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews,* vol. 32, pp. 140-153, 2002.

[46] P. J. Schweitzer, "Generalized Polynomial Approximations in Markovian Decision Processes," *Journal of Mathematical Analysis and Applications,* vol. 110, pp. 568-582, 1985.

[47] M. Maggioni and S. Mahadevan, "A Multiscale Framework for Markov Decision Processes using Diffusion Wavelets," University of Massachusetts, Department of Computer Science Technical Report TR-2006-35, 2006.

[48] S. Mahadevan and M. Maggioni, "Value Function Approximation with Diffusion Wavelets and Laplacian Eigenfunctions," in *Neural Information Processing Systems (NIPS)*: MIT Press, 2006.

[49] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, pp. 832-844, 1998.

[50] E. Kussul, D. Rachkovskij, and D. Wunsch, "The Random Subspace Coarse Coding Scheme for Real-valued Vectors," in *Proceeding of the International Joint Conference on Neural Networks*, 1999, pp. 450-455.

[51] S. Boyd and L. Vendenberghe, *Convex Optimization*. New York, N.Y.: Cambridge University Press, 2004.

[52] P. Y. Papalambros,"Monotonicity Analysis in Engineering Design Optimization," Ph.D. Dissertation, Dept. of Mechanical Engineering, Stanford University, Palo Alto, California, 1979.

[53] D. R. Jones, C. D. Peritunen, and B. E. Stuckman, "Lipschitzian Optimization without the Lipschitz Constant," *Journal of Optimization Theory and Applications,* vol. 79, pp. 157-181, 1993.

[54] R. M. Smaling and O. L. d. Weck, "Fuzzy Pareto Frontiers in Multidisciplinary System Architecture Analysis," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* Albany, New York, 2004.

[55] H. Li, G. E. Andrews, G. Zhu, B. Daham, M. Bell, J. Tate, and K. Ropkins, "Impact of Ambient Temperatures on Exhaust Thermal Characteristics During Cold Start for Real-World SI Car Urban Driving Test," *SAE Paper No. 2005-01-3896,* 2005.

[56] D. Klein and W. K. Cheng, "Spark Ignition Engine Hydrocarbon Emissions Behaviors in Stopping and Restarting," *SAE Paper No. 2002-01-2804,* 2002.

[57] S. Delprat, T. M. Guerra, G. Pagnelli, J. Lauber, and M. Delhom, "Control strategy optimization for an hybrid parallel powertrain," in *Proceedings of the American Controls Conference*, Arlington, Va, 2001, pp. 1315-1320.

[58] A. Sciarretta, M. Back, and L. Guzzella, "Optimal Control of Parallel Hybrid Electric Vehicles," *IEEE Transactions on Control Systems Technology,* vol. 12, pp. 352-363, May 2004.

[59] G. Pagnelli, Y. Guezennec, and G. Rizzoni, "Optimizing Control Strategy for Hybrid Fuel Cell Vehicle," *SAE Paper No. 2002-01-0102,* 2002.

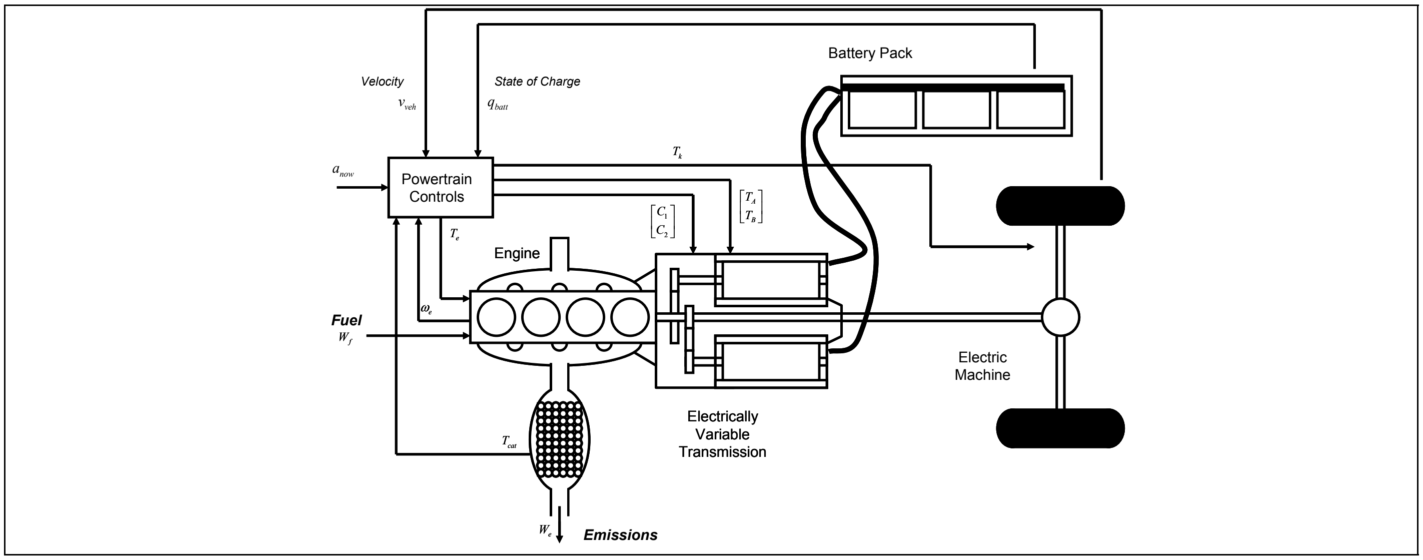


Figure 1 - Illustration of Major Components in Dual-Mode EVT HEV

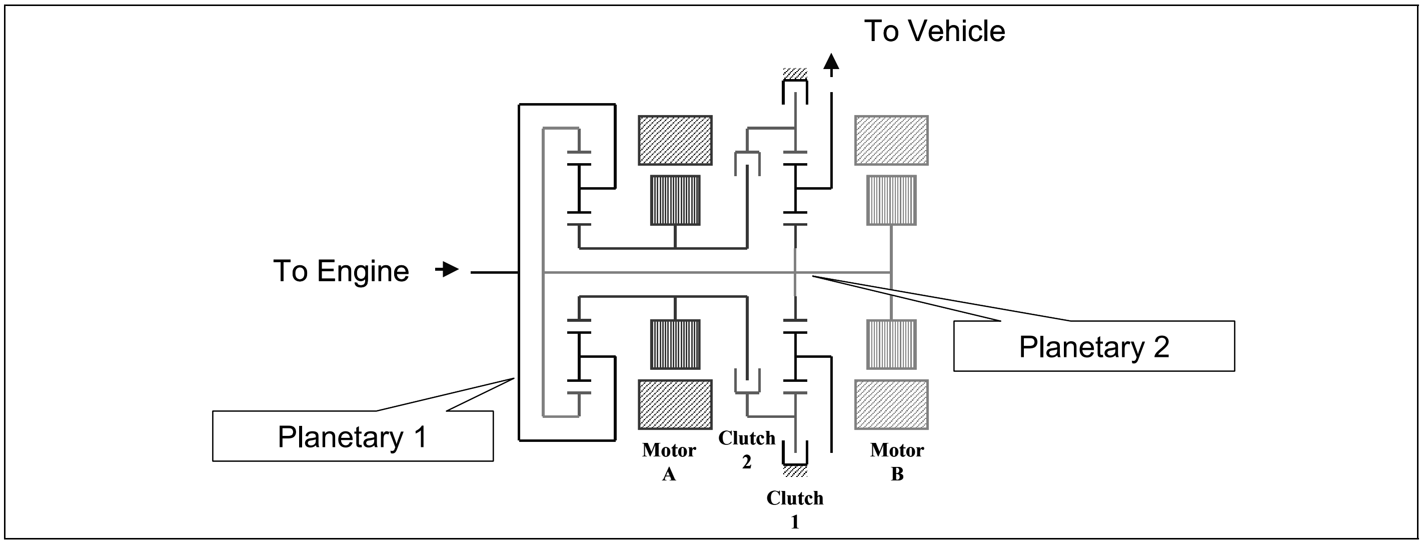


Figure 2 - Stick Diagram of gearing, clutches, and electric machines in DM-EVT

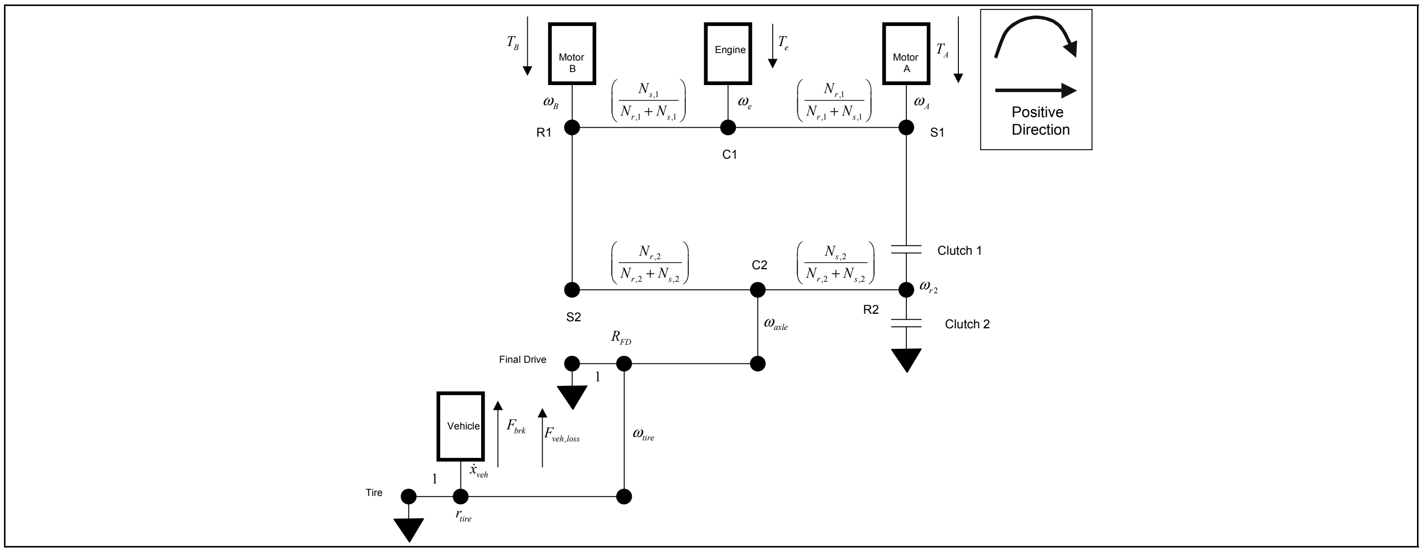
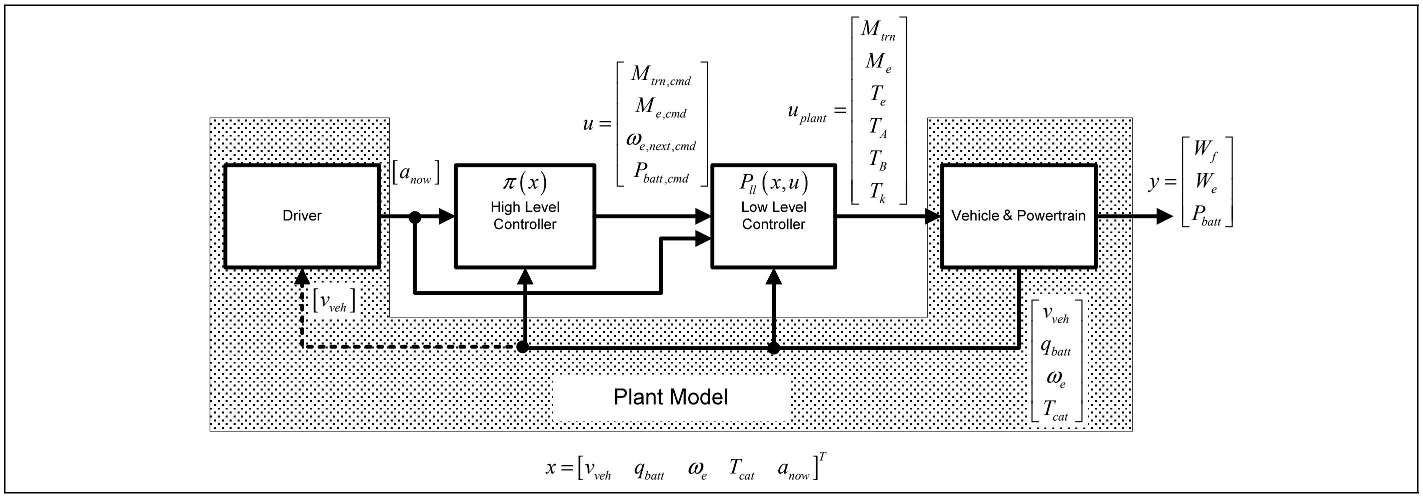


Figure 3 - Lever Diagram of Kinematics in the DM-EVT HEV Model



Driver
$[a_{now}]$
$\pi(x)$ High Level Controller
$u = \begin{bmatrix} M_{trn,cmd} \\ M_{e,cmd} \\ \omega_{e,next,cmd} \\ P_{batt,cmd} \end{bmatrix}$
$P_{ll}(x,u)$ Low Level Controller
$u_{plant} = \begin{bmatrix} M_{trn} \\ M_e \\ T_e \\ T_A \\ T_B \\ T_k \end{bmatrix}$
Vehicle & Powertrain
$y = \begin{bmatrix} W_f \\ W_e \\ P_{batt} \end{bmatrix}$
$[v_{veh}]$
$\begin{bmatrix} v_{veh} \\ q_{batt} \\ \omega_e \\ T_{cat} \end{bmatrix}$
Plant Model
$x = [v_{veh} \quad q_{batt} \quad \omega_e \quad T_{cat} \quad a_{now}]^T$

Figure 4 - Signal Flow Diagram for Control and Feedback Signals

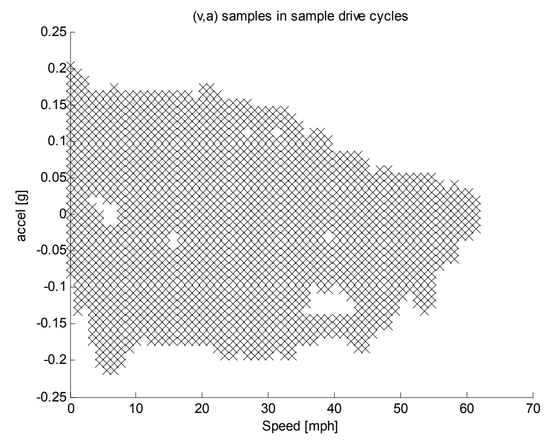


Figure 5 - Distribution of Visited Speed, Acceleration Pairs

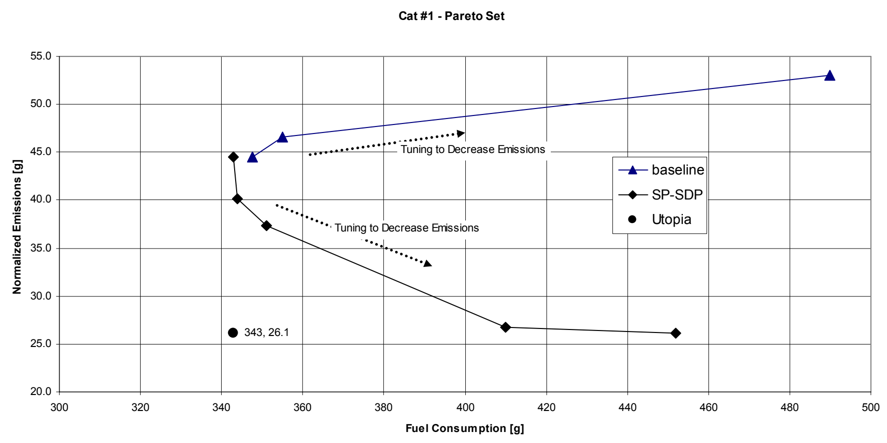


Figure 6 – Isolated Catalyst Pareto Set for Fuel and Emissions

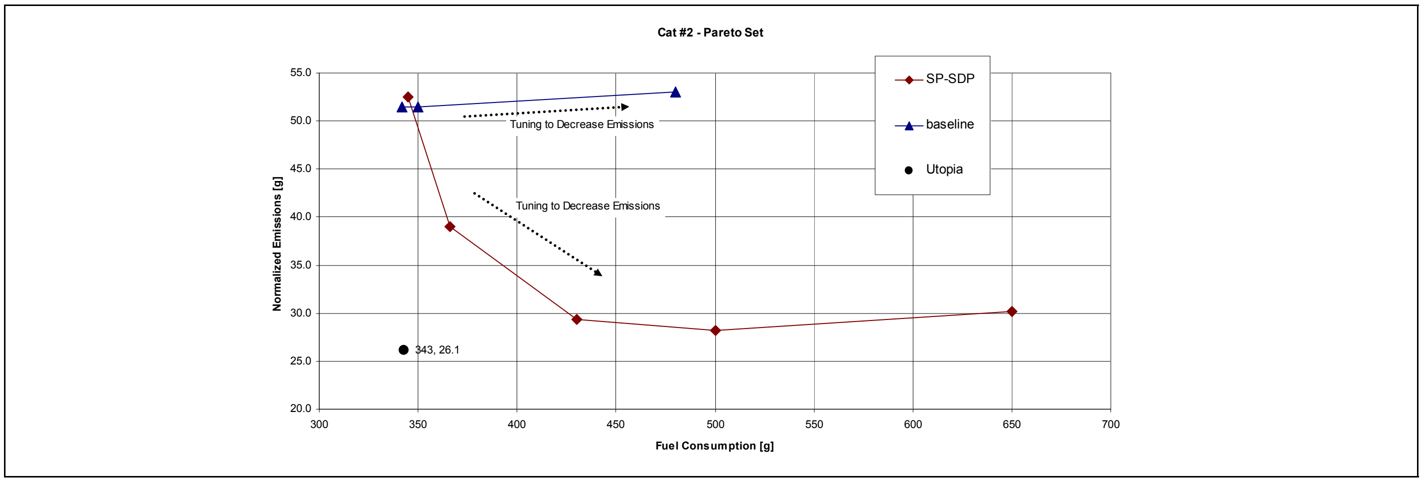


Figure 7 - Low Heat Loss Catalyst Pareto Set for Fuel and Emissions

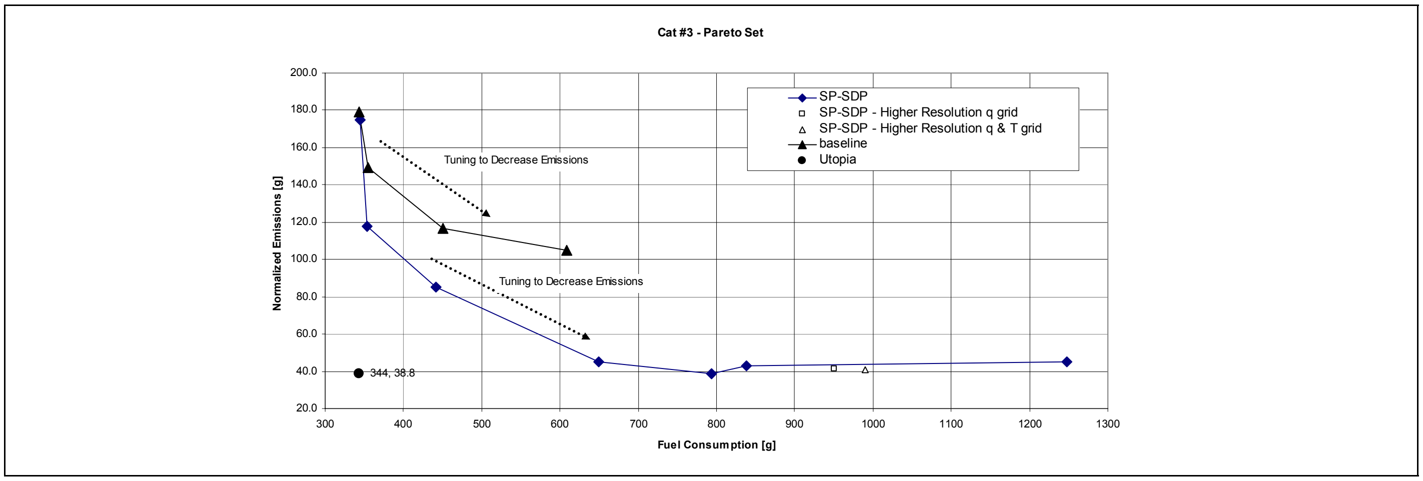


Figure 8 - High Heat Loss Catalyst Pareto Set for Fuel and Emissions