11ICE-0374

Self-Learning Neural Controller for Hybrid Power Management using Neuro-Dynamic Programming

Rajit Johri and Zoran Filipi
Mechanical Engineering, University of Michigan

ABSTRACT

A supervisory controller strategy for a hybrid vehicle coordinates the operation of the two power sources onboard of a vehicle to maximize objectives like fuel economy. In the past, various control strategies have been developed using heuristics as well as optimal control theory. The Stochastic Dynamic Programming (SDP) has been previously applied to determine implementable optimal control policies for discrete time dynamic systems whose states evolve according to given transition probabilities. However, the approach is constrained by the curse of dimensionality, i.e. an exponential increase in computational effort with increase in system state space, faced by dynamic programming based algorithms. This paper proposes a novel approach capable of overcoming the curse of dimensionality and solving policy optimization for a system with very large design state space. We propose developing a supervisory controller for hybrid vehicles based on the principles of reinforcement learning and neuro-dynamic programming, whereby the cost-to-go function is approximated using a neural network. The controller learns and improves its performance over time. The simulation results obtained for a series hydraulic hybrid vehicle over a driving schedule demonstrate the effectiveness of the proposed technique.

Keywords: Neuro dynamic programming (NDP), reinforcement learning, series hydraulic hybrid, power management, online learning, optimal control, numerical optimization, temporal difference.

INTRODUCTION

Hybrid powertrains are becoming increasingly popular due to regulatory pressures to improve fuel efficiency and growing environmental concerns. Hybrid vehicles have two power sources onboard and can be coordinated to maximize fuel economy and reduce emissions. Efficient management of the secondary source of energy provides additional degree of freedom in operation of engine and the whole powertrain can be designed to improve fuel economy by the possibilities of (i) downsizing the engine, (ii) recovering energy during braking event by regeneration, (iii) optimizing engine operation and, (iv) engine shutdowns. However, the vehicle system becomes more complicated and requires more complex control strategy to maximize the performance.

A number of methodologies have been proposed for development of control strategies, ranging from rule-based, to Equivalent Consumption Minimization Strategy [1], [2], fuzzy logic [3], [4], and horizon optimization [5], [6], [7], [8], [9]. The nature of the series hybrid system, with the engine decoupled from the wheels, allows significant freedom in designing the supervisory control strategy. This creates a special challenge when it comes to application of advanced algorithms, since the typical power-split problem is replaced with a decision about controlling the SOC. Previous studies of series electric systems basically relied on thermostatic (on-off) engine power management according to the State-of-Charge (SOC) in the battery [10], [11]. Similar rules were extended to hydraulic hybrid powertrains by Kim et al. [12] and Filipi et al. [13]. Kim [14] and Johri et al. [15] applied horizon optimization to power management problem for series hydraulic hybrid and designed controller using Stochastic Dynamic Programming (SDP).

Dynamic programming can generate optimal benchmark for nonlinear problem with nonlinear constraints. In addition, it can include multiple objectives during policy optimization. Theoretically, the hybrid power management controller can be designed with multiple objectives in consideration like reduced transient emissions or reduced noise vibration harshness (NVH) along with the original fuel economy objective. However, detailed modeling of multiple phenomena involves increase in system/plant states and the space spanned by the states grows exponentially. This in turn results in exponential growth in computational/memory resources required to calculate optimal solution. This is widely known as *curse of dimensionality* of dynamic programming. The curse of dimensionality is not only restricted to state space but can also arise from action and decision spaces [16]. Figure 1 shows the exponential rise in

computational demand with increase in space spanned by states. Therefore, classical dynamic programming algorithms are applicable only to problems with few thousand state counts with the present computational resources and this effectively limits the number of states to 2 or 3 with discretization level of approximately 20. The goal of the authors is to alleviate the curse of dimensionality and further advance the horizon optimization techniques with machine learning concepts.

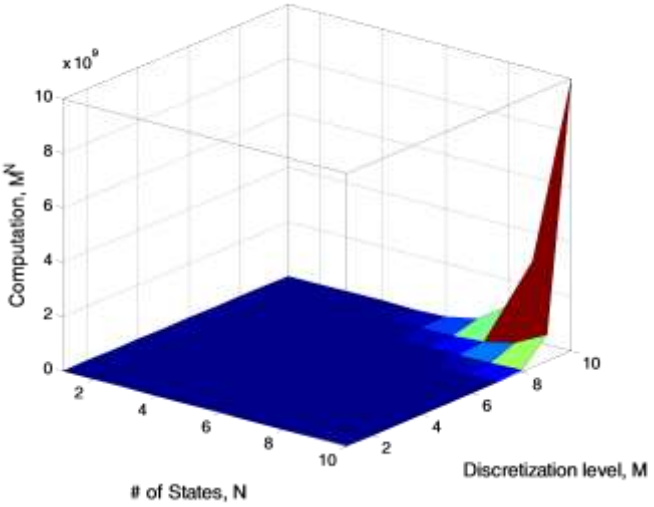

Figure 1: Curse of dimensionality

This paper focuses on development of algorithm that produces near-optimal policy with reasonable amount of computational resources. The idea centers on evaluation and approximation of optimal cost-to-go function through the use of neural networks. At the center of this approach is a self-learning neural network which adapts over time to reflect the optimal cost-to-go function. The approach is hence called *Neuro Dynamic Programming*. The approach is then demonstrated through the development of the supervisory controller for a series hydraulic hybrid vehicle that self learns with time to optimally manage two onboard power sources, namely engine and hydraulic energy stored in accumulator.

The paper is divided into four main sections. Firstly we give a brief background on dynamic programming and reinforcement learning. It is followed by a detailed overview of neuro-dynamic programming and temporal difference learning. Next, we formulate the power management problem for hybrids and apply neuro-dynamic programming to design a self-learning neural network based controller. Finally the findings are discussed. The paper ends with conclusions.

BACKGROUND

Reinforcement learning (RL) is learning by an agent to accomplish a particular task through trial-and-error interactions with environment based on reinforcement signals from the environment [17]. Reinforcement learning is different from supervised learning which is widely used in many fields, such as artificial neural networks, and statistical pattern recognition. Supervised learning involves agent learning to perform a certain task after training from a knowledgeable supervisor. The agent does not learn from its interaction with the environment. In contrast, a general RL model includes an agent (controller) that interacts with environment (system) over a sequence of discrete steps. The agent, based on the state of environment, selects and takes an action, u (controller output) according to a given policy, π and incurs an instantaneous cost, g. The goal of the agent is to minimize (maximize) the cost over time (objective function). A policy's value function gives the expected return if a given agent uses that policy.

Dynamic programming (DP) provides a framework to solve for optimal value function and from which an optimal policy can be derived. The *Bellman's equation* or *Hamiltonian-Jacobi-Bellman equation* is special consistency condition that the optimal value function has to satisfy.

$$J^*(i) = \min_{u \in U} E\left[g(i,u,j) + \alpha J^*(j) \mid i,u \right], \quad \forall i, \quad i, j \in X \tag{1}$$

where J^* is the *optimal value function* or *cost-to-go function*, i is the present state, j is the subsequent state to i, $g(.)$ is the cost incurred to go to state j from under control u, and $E[.|i,u]$ is the expected cost with respect to j, given i and u.

The objective of DP is to numerically calculate the optimal cost-to-go function J^*. The Bellman's equation can be solved using classical dynamic programming techniques; value iteration and policy iteration algorithms. The *Value Iteration Algorithm* is a principal method of calculating the optimal cost-to-go vector J^*. The algorithm starts with some initial J and iterates over following equation till J^k converges.

$$J^{k+1}(i) = \min_{u \in U(i)} \left(g(i,u,j) + E\left[\alpha J^k(j) \mid i,u \right] \right), \ \forall i \tag{2}$$

where k is the iteration number, and j is the next state subsequent to i. However, the value iteration algorithm takes infinite iterations to converge. An alternative is to use *Gauss Seidel iteration* i.e. to iterate one step at a time and incorporating this computation for next subsequent steps.

The *Policy Iteration Algorithm* is an alternative to the value iteration algorithm which is guaranteed to converge within finite steps. The algorithm starts with an initial stationary policy π_0 and generates a sequence of updated policies $\pi_1, \pi_2 \ldots$ with every iteration. The policy iteration algorithm iterates between a policy evaluation step and a policy improvement step until the optimal cost function converges to J^*. In policy evaluation step, given a policy π_k, $J_\pi^k(i)$ is calculated by solving linear set of equations

$$J_\pi^{k+1}(i) = g(i,\pi(i),j) + E\left[\alpha J_\pi^k(j) \mid i,u \right], \ \forall i \tag{3}$$

where k is the iteration number, and j is the next state subsequent to i given the control input u. The policy improvement step is evaluated next and updated policy π_{k+1} is calculated.

$$\pi_{k+1}(i) = \arg\min_{u \in U(x^i)} \left[g(i,u,j) + E\left[\alpha J_\pi^{k+1}(j) \mid i,u \right] \right], \ \forall i \tag{4}$$

where J_π^{k+1} is the approximate cost function obtained from the policy evaluation step.

For problems with large number of states, solving linear set of equations either by inversion or *Gauss elimination methods* is very computationally and time consuming. To alleviate this problem, the linear set of equations can be solved using value iteration. This modified algorithm is known as *Hybrid Policy/Value Iteration Algorithm*. More details about this approach pertaining to hybrids can be found in work by Johri et al. [15].

NEURO-DYNAMIC PROGRAMMING

Neuro-Dynamic Programming (NDP) is also known as Approximate Dynamic Programming (ADP) by many researchers. The term approximate comes from the fact that the method centers on approximation of optimal cost-to-go function. In particular, the optimal cost-to-go function $J^*(.)$ is replaced with suitable approximation $\tilde{J}(.,r)$ where r is a vector of parameters. Hence, the representation can be considered as mapping of higher dimensional cost-to-go vector, $J \in \mathrm{R}^n$ using a lower dimensional parameter vector, $r \in \mathrm{R}^m$ $(m << n)$.

$$\pi^*(i) = \arg\min_u E\left[g(i,u,j) + \alpha \tilde{J}(j,r) \mid i,u \right] \tag{5}$$

The above equation is modified Bellman's equation with J replaced by mapping $\tilde{J} : \mathrm{R}^m \rightarrow \mathrm{R}^n$.

The function $\tilde{J}$ is called the scoring function and the value $\tilde{J}(j,r)$ is called the score of state j [18]. In most problems, optimal cost-to-go is a highly complicated function of states. A compact representation by a scoring function attempts to break this complexity. However, an important issue is the selection of compact representation with a tradeoff between complexity and size. Some of the architectures in literature [18], [19], [20] are:

1. **Feature Mapping:** In a feature based compact representation, each component $\tilde{J}_i$ of the scoring function, $\tilde{J}$ is a function of some feature vector, *f(i)* and parameter vector r but not an explicit function of state i

$$f(i) = \left(f_1(i), \cdots, f_m(i) \right) \quad (6)$$

where *m* is the cardinality of feature space. The features can be constructed heuristically or through optimization.

a. Lookup table: The feature space is represented by lookup table and the parameter vector, *r* contains one component for each possible feature vector. With effective feature extraction, many states can be associated with one feature vector. The feature space will be smaller than total number of states. In an extreme case, each feature vector corresponds to single state and there are as many parameters as states.
b. Linear Architecture: The scoring function is a linear combination of features and the cost approximation is given by

$$\tilde{J}(i, r) = r_0 + \sum_{n=1}^{m} r_n f_n(i) \quad (7)$$

where $r = \{r_0, r_1, \cdots, r_m\}$ is the parameter vector

2. **Multilayer Perceptron Neural Network:** The score of a state, $\tilde{J}(i, r)$ is represented by a multilayer perceptron network with the layer weights being the parameter vector. The feature extraction mapping can either be absent or explicitly included. A neural network is a universal function approximator and has been shown capable of fitting any nonlinear function to an arbitrary degree of precision [21]. This makes neural network an excellent choice for the scoring function.

Neural networks have been used successfully in many fields like pattern recognition, artificial intelligence and nonlinear system identification owing to their universal approximator property. The neural network, in traditional sense, is trained by optimizing in least square sense the error between input-output mapping and the desired nonlinear function. The training is performed by using a training data set comprising of input-output combination, i.e. *{i, F(i)}* which is representative of mapping *F* to be approximated. This paper uses neural networks for approximating the cost-to-go function. It should be noted that in contrast to supervised learning of neural network, there is no data pair of input-output combination for the scoring function to be approximated. A least square optimization with pair *{i, J*(i)}* to approximate $\tilde{J}$ is not possible as the optimal cost-to-go value *J*(i)* for a given state *i* is unknown. The only possibility is to simulate for the cost-to-go estimate, *J(i)* for a given policy (suboptimal usually) and to iteratively improve the policy based on the simulation outcome. The target for the neural network training changes with every iteration as the simulation finds a better cost-to-go estimate, *J(i)*. This creates computational difficulties that do not arise in traditional neural network applications.

The methods for solving modified Bellman's equation are mostly derived from policy iteration algorithm. The algorithm generates a sequence of policies, $\pi = \{\pi_1, \cdots, \pi_k\}$ with every iteration and the corresponding estimate of cost-to-go $J_{\pi k}$ is calculated using compact representation, $\tilde{J}(., r)$. The approximating architecture used in this paper is neural networks based.

TEMPORAL DIFFERENCE

Reinforcement learning has great intuitive appeal and has received considerable interest by robotics and machine learning community. The major breakthrough came with implementation of Temporal Difference (TD) learning method [17]. The most noteworthy result in TD methods is a TD-Gammon program that learned to play Backgammon at grandmaster level [22]. The TD methods have become a powerful choice for Markovian environments like game playing.

The algorithm can be viewed as looking back in time and correcting for previous predictions. Temporal difference, d_k can be considered as prediction error between predicted performance and observed performance in response to action, u_k applied to the system.

$$d_k = g(i_k, u_k, i_{k+1}) + \alpha \tilde{J}(i_{k+1}, r_k) - \tilde{J}(i_k, r_k) \quad (8)$$

For Bellman equation to hold and $\tilde{J}(., r) \approx J^*$, the TD error should be zero. Therefore, for a given control policy, π, the equation $d_k = 0$ can be solved for $\tilde{J}(., r)$ in least square sense. Temporal difference methods are family of algorithms and detailed discussion is given by Bertsekas et al. [18] and Sutton et al. [17].

Consider a series of simulated sequence $i = \{i_1, \cdots, i_n\}$ of states generated by Monte Carlo simulation. At a typical iteration, the system is at state i_k and a control $u_k \in U$ based on current policy is applied. The next state i_{k+1} is generated by simulating transition probability $p_{i_k j}(u)$.

The parameter vector is then updated by running a TD(λ) update [18]

$$r_{k+1} = r_k + \gamma_k d_k \sum_{m=0}^{k} (\alpha\lambda)^{k-m} \nabla \tilde{J}(i_m, r_m) \tag{9}$$

where γ is the step size, λ is the TD parameter and d_k is the temporal difference and gradient $\nabla \tilde{J}(i, r)$ is the vector of partial derivatives with respect to parameter vector r. The equation (9) is an incremental gradient update. This paper uses the Extended Kalman Filter to calculate this update which results in faster convergence.

Define eligibility vector, z_k

$$z_k = \sum_{m=0}^{k} (\alpha\lambda)^{k-m} \nabla \tilde{J}(i_m, r_m) \tag{10}$$

The TD update can be written as

$$r_{k+1} = r_k + \gamma_k d_k z_k \tag{11}$$

where z_k is updated by

$$z_{k+1} = \alpha\lambda z_k + \nabla \tilde{J}(i_{k+1}, r_{k+1}) \tag{12}$$

In the approximate policy iteration, the policy is fixed for every policy iteration step. The approximation $\tilde{J}(\cdot, r)$ of J_π is constructed by least squares problem. The new policy $\bar{\pi}$ is then calculated which is greedy with respect to $\tilde{J}(\cdot, r)$. A greedy policy means that selection of new policy is based solely on immediate value of $\tilde{J}(\cdot, r)$ and does not take into account future. A greedy policy is optimal if $\tilde{J}(\cdot, r)$ is actually optimal J^* [17]. An alternative is to optimistically replace the policy π with new policy $\bar{\pi}$ before the approximate evaluation of J_π converges. An extreme possibility is to replace policy π with new policy $\bar{\pi}$ subsequent to every state transition, i.e. new control u_k is calculated after every iteration

$$u_k = \arg\min_{u \in U(i_k)} \left(g(i_k, u, j) + E\left[\alpha \tilde{J}(j, r_k) \right] \right) \tag{13}$$

where α is the discount factor, r_k is the current parameter vector and j is the possible next states.

The convergence behavior of this algorithm is quite complex and is not fully understood [18]. However, optimistic policy with TD(λ) update is one of the most effective NDP methods and is used in this paper.

The step size, γ in equation (11) plays an important role in performance of incremental gradient descent type algorithm, equation (9). To ensure convergence in stochastic gradient algorithms following rules must be met [16].

$$\begin{aligned} &\gamma_n \geq 0, \qquad n = 1, 2, \ldots \\ &\sum_{n=1}^{\infty} \gamma_{n-1} = \infty, \\ &\sum_{n=1}^{\infty} (\gamma_{n-1})^2 < \infty \end{aligned} \tag{14}$$

The second condition is required to guarantee that step size is not too small and the algorithm stalls prematurely. It ensures that steps are large enough to overcome any initial conditions or random fluctuations. The last condition guarantees that the step size diminishes with iteration and eventually become small enough to assure convergence.

The step size selection is even more important for neuro-dynamic programming due to the moving target value $\tilde{J}$ which at the start of algorithm can be very far from optimal *J**. The *Sompolinsky-Barkai-Seung* algorithm is used for step size calculation [23] in this paper.

$$\gamma_k = \gamma_{k-1} + a\gamma_{k-1}\{b(f(\cdot, r_{k-1}) - F^*) - \gamma_{k-1}\} \quad (15)$$

where γ is the step size, *a* and *b* are positive constants, *r* is the parameter vector, $f(\cdot, r)$ is a differentiable loss function defined by $f(\cdot, r) = \sum\left(J^* - \tilde{J}(\cdot, r)\right)^2$, and $F^* = \min_r E[f(\cdot, r)]$ is minimal loss function. The idea is that when the error is large, the step size γ is large, $\gamma_k \approx b(f(\cdot, r_k) - F^*)$ and when error is small and the estimator is close to optimal value, α approaches 0, $\gamma_k = \gamma_{k-1} - a\gamma_{k-1}^2$.

ACTOR CRITIC MODEL

An interesting and intuitive way to describe the NDP algorithm, presented in this paper, is to view it as an actor-critic system [18], [17]. The general schematic of the NDP is shown in Figure 2.

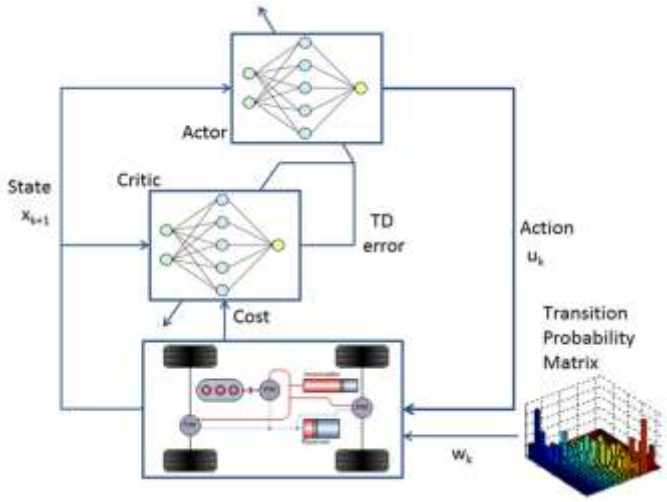

Figure 2: Schematic diagram for implementation of Neuro Dynamic Programming

The NDP structure in Figure 2 includes two networks; actor and critic. The critic network is trained to estimate the approximation of optimal cost-to-go function. The actor network is trained to produce control inputs which are greedy with respect to optimal cost-to-go function. The objective is to optimize the desired performance by learning to choose appropriate control actions through interaction with environment. During the learning process, the actor's actions are criticized by the critic and the actor incorporates the latest evaluation by the critic for next control action. The critic learns about and critiques whatever the policy is being followed by the actor. The reinforcement signal (critique) is the output of the critic network which drives the learning of both the actor and critic networks.

The controller is "naïve" at the starting of the algorithm. Both the actor and critic networks are initialized with random weights. Based on the present system states, the actor network produces control actions and the system moves to a newer state. The critic evaluates the new state and calculates the reinforcement signal to tune the parameters in the actor as well as critic network. With time, the actor learns to produce "favorable" control actions.

The critic network is used to estimate an approximation of the cost-to-go function. The critic network is implemented as a standard multilayer feedforward neural network. A *hyperbolic tangent* function is used as activating transfer function for hidden and output layers. The input to the critic is the states of system and the output is the approximate cost-to-go function $\tilde{J}$. The network is trained by backpropogating the TD error signal and the weights, r_k of the critic network for the k^{th} iteration step are updated using equation (11).

The actor network is similar to the critic network. The output is the control signal to the system. The actor network is required because it requires lot of computation and memory to compute improved policy, $\bar{\pi}$, given in equation (5) online. The algorithm computes $\bar{\pi}(i)$ only at a set $\hat{S}$ of sample states and an approximation architecture $\tilde{\pi}(i,v)$ where v is a vector of tunable parameters. The neural network is trained by minimizing the least square problem

$$\min_{v} \sum_{i \in \hat{S}} \left\| \tilde{\pi}(i,v) - \bar{\pi}(i) \right\|^2 \qquad (16)$$

EXPLORATION VS. EXPLOITATION

Classical SDP algorithms require evaluating the cost-to-go function at every state. This is only possible for problems with small state space. It's computationally impossible to evaluate the cost-to-go function at every state for a problem with infinite state space or very large state space. This is known as exploration of the state space. On the other hand, exploitation involves utilizing the present information about the cost-to-go function and making decisions that are greedy with respect to the present cost-to-go function approximation. An advantage of the exploitation strategy in the context of problems with large state space is computational efficiency. However, the problem with pure exploitation strategy is that algorithm is susceptible to getting stuck in local optimum because of the poor estimate of certain states.

The strategy used in this paper is a mix of exploration and exploitation strategies and is known as *ε-greedy* strategy [17]. The algorithm chooses a greedy policy i.e. exploitation strategy for most of the times but reverts to exploration strategy with small probability ε and selects action at random, independently of the cost-to-go function. An advantage of such a strategy is that on limit, as number of iteration increases, every control action will be sampled infinitely and the control policy π will converge to optimal π^*. Sutton et al. [17] showed the effectiveness of ε-greedy strategy compared to greedy policy.

HYBRID VEHICLE SUPERVISORY CONTROLLER PROBLEM

This paper focuses on 4X4 military vehicle intended for on-road and off-road purposes. The baseline vehicle specifications correspond to High Mobility Multipurpose Wheeled Vehicle (HHMWV). The vehicle is modeled to have series hydraulic hybrid powertrain, shown in Figure 3. The design of HMMWV is similar to that presented by Kim et al. [24] and Filipi et al. [13]. The engine is connected to pump to create a power generation subsystem capable of charging the accumulator. The motors propel the vehicle with hydraulic energy stored in accumulator. There is no mechanical linkage between engine and the wheels giving flexibility in operating engine. The motors can be used for regenerative braking.

Previous work on a series hydraulic hybrid [24] showed the advantage of having one motor per axle over single motor design with transfer case for a 4X4 mid-size truck. It was also shown that sequential operation of the two motors can result in better fuel economy over simultaneous operation. The concept is to operate motors sequentially resulting in higher loads per motor and hence higher efficiency. Rear motor is primarily used for propulsion and front motor augments the torque in extreme cases. While braking, front motor is used for regeneration. The choice between front and rear motor operation was based on weight transfer during acceleration and braking. The results shown in the paper are obtained with sequential operation of motors.

The vehicle and powertrain components are modeled in Simulink and are based on the vehicle simulation platform developed at the University of Michigan [25]. The powertrain system simulation has been validated with vehicle test data from proving ground [25] and subsequently thoroughly updated to represent HMMWV [26]. The hydraulic components were added to the vehicle simulation in context of parallel [27] and series hybrid systems [24].

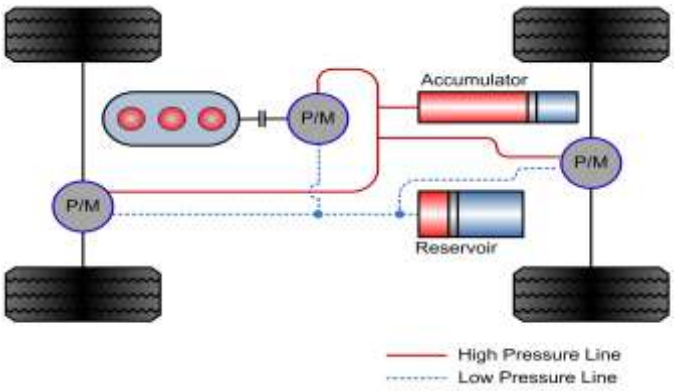

Figure 3: Series Hydraulic Hybrid configuration

Table 1 gives the specification of the vehicle and powertrain. For completeness a brief description of key models for series hydraulic hybrid is given next.

Table 1: Series Hydraulic Hybrid Specifications

Engine	Description	6.4L Navistar
	Max. Power	261 kW @ 3000 RPM
	Max. Torque	881 Nm @ 2000 RPM
Pump	Design	Axial Piston Variable Displacement
	Size	300 cc/rev
	Max Power	700 kW @ 350 bar, 4000 RPM
Motor	Design	Axial Piston Variable Displacement
	Size	180 X 2 cc/rev
	Max Power	420 kW @ 350 bar, 4000 RPM
Accumulator	Capacity (Max. Gas Volume)	98 Liter
	Max Pressure	350 bar
	Min Pressure	120 bar
Vehicle	Type	HMMWV
	Weight	5112 kg
	Coeff. of Drag	0.7
	Frontal Area	3.58 m^2
	Tire Radius	0.4412 m
	Final Drive Ratio	4.086
Transmission	Design	2 speed automatic
	1^{st} Gear Ratio	3 : 1
	2^{nd} Gear Ratio	1 : 1

The **engine** is modeled as lookup table that provides engine torque as a function of engine speed and fueling rate. The engine brake torque map was generated from experiments performed at the University of Michigan. The diesel injection controller calculates the amount of fuel injected per cycle based on present engine speed and throttle command. A carefully calibrated time delay is incorporated to simulate turbo-lag. The engine speed is calculated by

$$\omega_e = \frac{1}{I_e}\int (T_e - T_l)\, dt \qquad (17)$$

where ω_e is the engine speed, I_e is the engine inertia, T_e is the brake engine torque and T_l is the load torque due to pump.

The **hydraulic pump/motor model** is an updated version of Wilson's P/M theory [28]. The P/M is an axial piston variable displacement type. The torque and hydraulic fluid flow are controlled by displacement factor. Details of the model are provided in Filipi et al. [7] and Kim et al. [24].

The **accumulator** is a hydro-pneumatic device and stores energy by compressing the nitrogen gas. A positive fluid flow rate into the accumulator compresses the gas stored in the bladder, thus storing energy. A low pressure reservoir is used to allow transfer of fluid to and from the accumulator during charging/discharging and prevent cavitation of pump and motors. The net pressure difference between accumulator and reservoir pressure is the head pressure on pump and motors. Modeling of the accumulator and the reservoir is based on the application of the energy conservation equation to the gas, as proposed by Pourmovahed and Otis [28]. The accumulator is modeled with elastomeric foam on the gas side to increase the thermal time constant to increase thermal capacity and reduce heat loss [29]. This helps in achieving high conversion efficiencies of mid-nineties in the accumulator.

PROBLEM FORMULATION

Given the vehicle, engine and powertrain configuration, the paper examines the power management problem which can be formulated as

$$\begin{aligned} \text{Minimize:}\quad & J = \lim_{N\to\infty} \underset{w_k}{E}\left\{\sum_{k=0}^{N-1} \alpha^{\kappa} g\left(x_k, u_k, w_k\right)\right\} \\ \text{subject to:}\quad & x_{k+1} = f\left(x_k, u_k, w_k\right) \\ & \mathrm{x} \in \mathrm{X} \\ & \mathrm{u} \in \mathrm{U} \end{aligned} \tag{18}$$

where x is the state vector, u is the control input, w is the disturbance vector, g is the instantaneous cost function and $0<\alpha<1$ is the discount factor. Discounting factor implies cost incurred at present is more important than incurred in future.

The state vector, $x = \{SOC, \omega_v\}$ forms a 2-dimensional finite state space. Both the components are continuous which are discretized as

$$\begin{aligned} SOC &= \{SOC^1, SOC^2, \cdots, SOC^{NS}\} \\ \omega_v &= \{\omega_v^1, \omega_v^2, \cdots, \omega_v^{Nv}\} \end{aligned} \tag{19}$$

where N_s and N_v is the cardinality of SOC and ω_v vector respectively.

The disturbance vector, w is the driver power demand, P_{dem}. The driver power demand is modeled as a discrete Markov process and is used to generate future power demands given present states. The driver power demand is also a continuous vector which is discretized as

$$P_{dem} = \left\{P_{dem}^1, P_{dem}^2, ..., P_{dem}^{N_p}\right\} \tag{20}$$

The disturbance vector, w evolves with probability distribution

$$\begin{aligned} & p_{ij,l} = \Pr\left\{w = P_{dem}^j \mid P_{dem} = P_{dem}^i, \omega_{wh} = \omega_{wh}^l\right\} \\ & i, j = 1, 2, ..., N_p \qquad l = 1, 2, ..., N_v \end{aligned} \tag{21}$$

where N_p is the cardinality of driver power demand. The transition probability matrix is generated by statistically analyzing driving cycles. In this work, driving cycles are naturalistic driving cycles based on actual driving behavior of randomly selected drivers in South East Michigan [30]. Figure 4 shows some of the driving cycles used for this work for generating transition probability matrix, Figure 5.

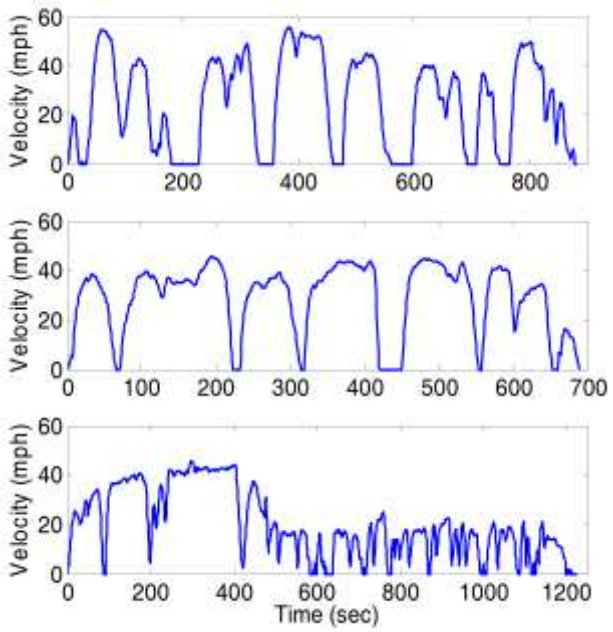

Figure 4 : Naturalistic Driving Cycles recorded during typical commutes in SE Michigan.

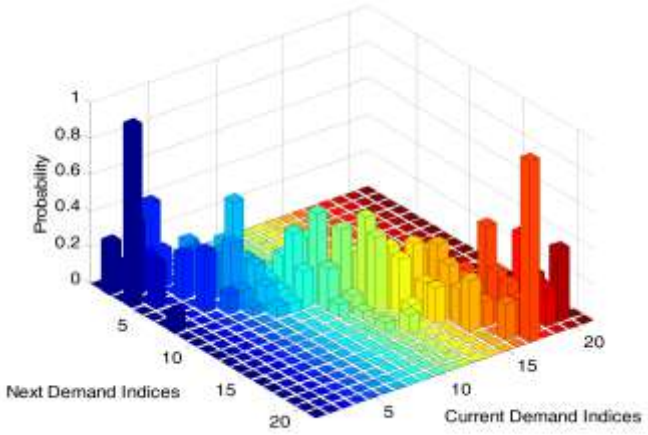

Figure 5: Transition Probability of power demand (ω_v = 54 rad/s) derived from naturalistic driving schedules

The control variable, $u = \{T_e, \omega_e\}$ forms a 2 Dimensional space and is also discretized with cardinality N_T and N_e respectively.

$$T_e = \{T_e^1, T_e^2, \cdots, T_e^{NT}\}$$
$$\omega_e = \{\omega_e^1, \omega_e^2, \cdots, \omega_e^{Ne}\} \quad (22)$$

The instantaneous cost function, g is function of states, control input and driver demand. The cost function in this paper is defined as

$$g = FC(\omega_e, \alpha, SOC) + \mu \cdot (SOC - SOC_{ref})^2 \cdot (SOC < SOC_{ref}) \quad (23)$$

where FC is the fuel consumption by engine at given SOC and engine operating point i.e. engine speed, ω_e and throttle, α. The instantaneous cost is based on overall system efficiency. The power generation unit includes engine and pump subsystem. The pump

efficiency is a function of SOC and hence the overall system efficiency changes with SOC. The latter term penalizes the deviation of SOC below a threshold value. This penalty factor is different from the one used by Lin et al. [9] for HEV. In HEV, a penalty factor was added to the cost function to satisfy charge sustaining constraint and limit the operation of SOC within a narrow window due to battery health and operating characteristics. Hydraulic accumulator does not suffer from similar constraints and SOC can vary over complete range. However, a lower bound on SOC is imposed to maintain vehicle drivable at all conditions. The above penalty function tries to maintain low SOC reference value, e.g. 0.2 in this study, to allow maximum energy regeneration during braking. μ is a constant and is chosen by trial and error.

The power management controller needs to satisfy certain bounds on states and control input. These bounds ensure that vehicle, engine and hydraulic devices do not operate in regimes which are unfavorable from performance standpoint or are detrimental to health of these devices.

$$\begin{aligned}
&\omega_{e,\min} \le \omega_{e,k} \le \omega_{e,\max} \\
&T_{e,\min}(\omega_{e,k}) \le T_{e,k}(\omega_{e,k}) \le T_{e,k}(\omega_{e,k}) \\
&T_{m,\min}(\omega_{m,k}, SOC_k) \le T_{m,k} \le T_{m,\max}(\omega_{m,k}, SOC_k) \\
&T_{p,\min}(\omega_{p,k}, SOC_k) \le T_{p,k} \le T_{p,\max}(\omega_{p,k}, SOC_k) \\
&SOC_{\min} \le SOC_k \le SOC_{\max} \\
&\omega_{wh,k} = \omega_{wh,req}
\end{aligned} \tag{24}$$

where k is the time index. ω_e, T_e, T_m, T_p, SOC and ω_{wh} are engine speed, engine torque, propulsion motor torque, pump torque, state of charge and wheel speed respectively. The subscripts *min* and *max* denote the lower and upper bounds of these variables.

APPROACH

This paper employs near-optimal method that centers around evaluation and approximation of cost-to-go, J with a self-learning neural network. The state space spanned by the problem considered in this paper is not huge. Hence, the power management problem could be solved using classical stochastic dynamic programming (SDP) algorithms [15]. This is done intentionally, to provide an opportunity to solve the problem with SDP and use the results as baseline for evaluating the results of NDP. This effectively allows us to validate the new approach before embarking on future studies of larger problems with even more states.

The self-learning controller has three neural networks, one critic and two actor networks, which are trained simultaneously by interacting with the environment. All the three neural networks are multilayer perceptron networks with *hyperbolic tangent* as the activation function. The hidden layers have 30 neurons. The networks have one hidden layer and take three inputs, namely the states and disturbance vector. The output of critic network is the approximate value of cost-to-go and the two actor networks output desired engine speed and engine torque.

The algorithm starts with an initial random point and follows sample trajectory generated using Monte Carlo simulation. The neural networks are initialized with random weights. At any given state, the control input from action network is applied. The algorithm calculates temporal difference and updates the weights of critic network using equation (11). This updated approximation of cost-to-go function is used to calculate new ε-greedy control policy which in turn updates actor network weights. The system moves to next state based on transition probability and the algorithm steps are repeated till the cost-to-go function approximation converges.

The control action chosen by algorithm and the states visited depend on approximation of cost-to-go function which in turn depends on states visited thus far by algorithm. This can lead to algorithm getting stuck in the local optima where poor approximation of cost-to-go function for certain states can prevent algorithm from visiting those states. To overcome this problem, the algorithm is frequently restarted from initial random states.

RESULTS

In this section we present simulation results for hybrid powertrain with self-learning neural controller over the federal urban driving schedule (FUDS). The results are compared against baseline controller i.e. the thermostatic controller [12]. The thermostatic controller is an intuitive engine-centric approach and resembles a “bang-bang” control. The controller switches the engine on whenever the SOC hits the lower threshold and charges the accumulator. The engine stays on till it crosses the upper threshold. The engine during this period operates at predetermined fixed desired power level. The predetermined power level is obtained by performing a parametric

sweep of different threshold powers similar to previous work done by Kim et al. [12]. The predetermined power level for this powertrain is 60kW which is lower than the "sweet spot" i.e. the eye of the BSFC map with lowest fuel consumption as suggested by Kim et al. [12]. If the power required for propulsion exceeds the threshold level and the SOC falls below the lower limit, the engine power is progressively increased. The increased power demand essentially keeps the system operating in the hydrostatic mode. More details about the thermostatic controller can be found in the previous work done by Kim et al. [12] and Filipi et al [13].

Figure 6 and Figure 7 shows temporal results for SOC, engine power and motor power for thermostatic and NDP based controllers over a section of driving cycle respectively. It can be seen from the Figure 7 that the NDP based controller learns how to actively manage two power sources namely engine and hydraulics and is able to follow driving cycle. Also the controller does a good job in maintaining low SOC throughout the driving cycle (Figure 7a). This allows for maximum energy to be recuperated during braking event. The NDP controller learns to use hydraulic energy at launch when SOC is high (engine power demand is zero). As SOC drops, engine is ramped up and produces enough power to maintain the desired value, 0.2 in this case. However the engine operation by NDP based controller is vastly different from the thermostatic controller. Thermostatic controller either runs engine at the threshold power of 60 kW or keeps the engine at idle (Figure 6b). The engine is forced to go through a step change in load frequently whereas the NDP based controller operates the engine in a milder fashion by slowly ramping up the engine power (Figure 7b) and significantly reducing fluctuations of SOC (Figure 7a). It appears that the engine is almost in a "load following" mode but without sharp changes of load or high frequency fluctuations. At the same time the algorithm selects the best combination of engine speed and torque to produce desired power as discussed in the next paragraph. Overall, NDP identifies the best strategy from the system point of view rather than engine-centric approach.

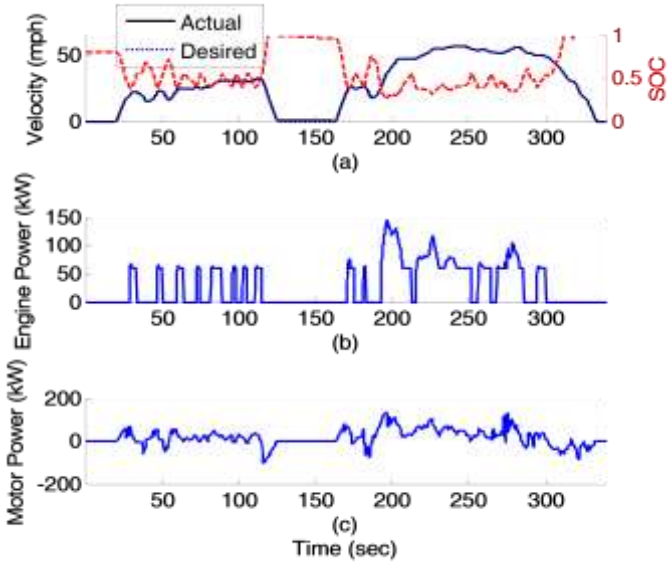

Figure 6: Simulation results over section of FUDS for baseline thermostatic controller

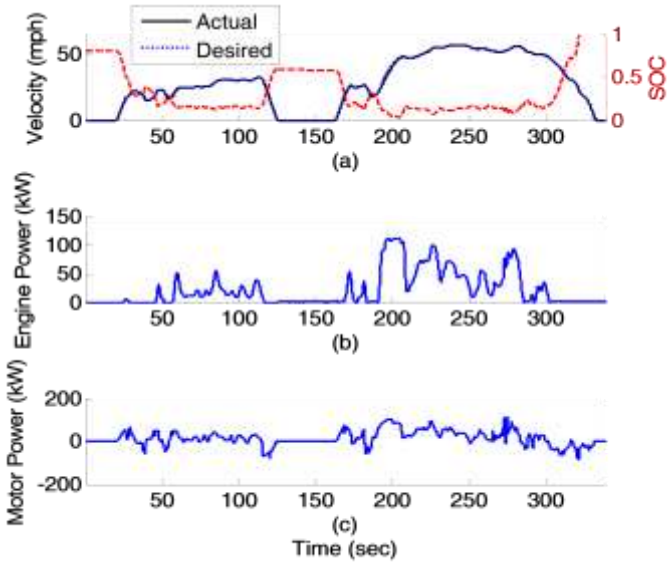

Figure 7: Simulation results over section of FUDS for NDP based controller

The engine operating points in the BSFC map visited during FUDS with thermostatic and NDP based controller are shown in Figure 8a and Figure 8b respectively. The color scale indicates the amount of fuel consumed by the engine at a given operating region over FUDS. Figure 8a shows the engine behavior with thermostatic controller while Figure 8b illustrate the same for NDP based controller. Thermostatic controller keeps the engine at selected threshold power of 60kW and most of the fuel consumption happens at that point as indicated by red spot in Figure 8a. In contrast, NDP controller operates the engine over a wider region. It can be seen that the NDP based controller intelligently controls engine in the low BSFC regions for any power level without any prior information about the best BSFC trajectory. However, the engine operation deviates slightly from best BSFC line. This is attributed to the fact that controller is trying to maximize system efficiency i.e. engine and pump combined subsystem efficiency rather than engine efficiency alone.

Table 2: Fuel economy comparison of different power management strategies over FUDS driving schedule

	MPG	% improvement
4-Speed Conventional	10.59	–
SHHV with thermostatic controller	14.77	39%
SHHV with SDP based controller	17.47	65%
SHHV with NDP based controller	17.84	68%

To allow assessment of the new approach, the baseline conventional vehicle with 4-speed gearbox along with the series hydraulic hybrid vehicle with three different control strategies, namely thermostatic, SDP and NDP, is simulated over FUDS. The SDP controller is developed using the approach described in [15], to subsequently allow validation of NDP. Table 2 gives the results and percentage improvement of series hydraulic hybrid over conventional baseline. The thermostatic based controller improvements over baseline conventional come from engine operation near low BSFC region and energy recuperated from braking. The results are slightly different from the previously reported results by Filipi et al. [13] due to newer generation engine used in this study. Both the SDP based controller and NDP based controller gives another 17% improvement over thermostatic controller by more effective management of system. The engine operation and management of SOC by NDP based controller is very similar to SDP based controller and hence, the fuel economy benefits are very similar. This is a favorable outcome providing evidence of NDP's ability to learn and discover best modes of operation.

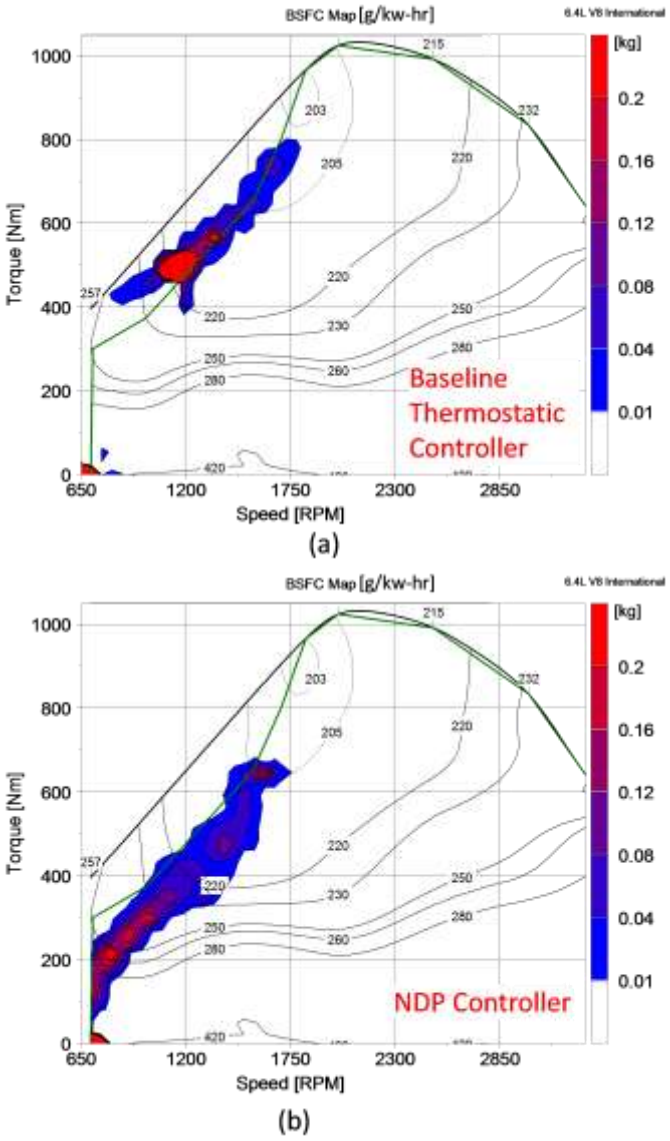

Figure 8: Engine visitation points on the BSFC map with best BSFC trajectory (green line). The color scale indicates the relative amount of fuel consumed in a given zone during FUDS for (a) baseline thermostatic controller (b) self-learning neural controller.

CONCLUSION

A self-learning neural network based power management controller is designed for series hydraulic hybrid. The vehicle power management problem is formulated as a Markov decision process and solved using neuro dynamic programming (NDP) techniques. The NDP based controller is compared against a baseline thermostatic controller and resulted in 17% improvement in fuel economy by more effective management of the hybrid system. The controller not only learned to manage the two power sources, namely engine and hydraulic energy stored in accumulator but learned to do so efficiently. The NDP based controller is also compared with controller designed using stochastic dynamic programming. The fuel economy and system operation by both the controllers are very similar and this effectively validates the NDP based approach for design of supervisory controllers.

The paper demonstrated the effectiveness of application of neuro-dynamic programming technique for policy optimization. The significance lies in the fact that, in contrast to deterministic dynamic programming or stochastic dynamic programming, the computational effort and memory resources in case of NDP increases linearly with parameters in functional representation rather than

exponentially with state space. This makes the approach scalable to more complex problems with larger state and action space. The key idea enabling the NDP development for supervisory controller is to represent the cost-to-go with a functional approximation e.g. neural network and the function is updated recursively using iterative method. This reduces computational and memory requirement and makes the future applications to much larger problems apparent. The NDP approach can be further extended with real-time learning of driver behavior by adapting the driver Markov model in real time [31].

REFERENCES

[1] Paganelli, G. , Delprat, S. , Guerra, T.M. , Rimaux, J. , and Santin, J.J. , "Equivalent consumption minimization strategy for parallel hybrid powertrains," presented at IEEE 55th Vehicular Technology Conference, **4**, pp. 2076-81, 2002.

[2] Paganelli, G. , Guerra, T.M. , Delprat, S. , Santin, J. , Delhom, M. , and Combes, E. , "Simulation and assessment of power control strategies for a parallel hybrid car," *Proceedings of the Institution of Mechanical Engineers, Part D (Journal of Automobile Engineering)*, **214**(D7), pp. 705-17, 2000.

[3] Farrall, S.D. and Jones, R.P. , "Energy management in an automotive electric/heat engine hybrid powertrain using fuzzy decision making," presented at Proceedings of the 1993 IEEE International Symposium on Intelligent Control, pp. 463-468, 1993.

[4] Baumann, B. M., Washington, G. , Glenn, B. C., and Rizzoni, G. , "Mechatronic design and control of hybrid electric vehicles," *IEEE/ASME Transactions on Mechatronics*, **5**(1), pp. 58-72, 2000.

[5] Sciarretta, A. and Guzzella, L. , "Control of hybrid electric vehicles," *IEEE Control Systems Magazine*, **27**(2), pp. 60-70, 2007.

[6] Sciarretta, A. , Guzzella, L. , and Onder, C.H. , "On the power split control of parallel hybrid vehicles: from global optimization towards real-time control," *Automatisierungstechnik*, **51**(5), pp. 195-203, 2003.

[7] Filipi, Z. , Louca, L. , Daran, B. , Lin, C.-C. , Yildir, U. , Wu, B. , Kokkolaras, M. , Assanis, D. , Peng, H. , Papalambros, P. , Stein, J. , Szkubiel, D. , and Chapp, R. , "Combined optimisation of design and power management of the hydraulic hybrid propulsion system for the 6 X 6 medium truck," *International Journal of Heavy Vehicle Systems*, **11**(3-4), pp. 372-402, 2004.

[8] Lin, C. C., Filipi, Z. , Louca, L. , Peng, H. , Assanis, D. , and Stein, J. , "Modelling and control of a medium-duty hybrid electric truck," *International Journal of Heavy Vehicle Systems*, **11**(3-4), pp. 349-371, 2004.

[9] Lin, C. C., Peng, H. , and Grizzle, J.W. , "A stochastic control strategy for hybrid electric vehicles," presented at Proc. of American Control Conference, **5**, pp. 4710-4715, 2004.

[10] Caratozzolo, P. , Serra, M. , and Riera, J. , "Energy management strategies for hybrid electric vehicles," presented at IEEE Electric Machines and Drives Conference, **1**, pp. 241-248, 2003.

[11] Jalil, N. , Kheir, N.A. , and Salman, M. , "A rule-based energy management strategy for a series hybrid vehicle," presented at Proceedings of the American Control Conference, **1**, pp. 689-93, 1997.

[12] Kim, Y. and Filipi, Z. , "Series Hydraulic Hybrid Propulsion for a Light Truck – Optimizing the Thermostatic Power Management," *SAE Transactions, Journal of Engines*, **116-3**(2007-24-0080), pp. 1597-1609, 2007.

[13] Filipi, Z and Kim, Y J, "Hydraulic Hybrid Propulsion for Heavy Vehicles: Combining the Simulation and Engine-In-the-Loop Techniques to Maximize the Fuel Economy and Emission Benefits," *Oil & Gas Science & Technology*, **65**(1), pp. 155-178, 2010.

[14] Kim, Y. , "Integrated modeling and hardware-in-the-loop study for systematic evaluation of hydraulic hybrid propulsion options," PhD thesis Mechanical Engineering, University of Michigan, , 2008.

[15] Johri, R. and Filipi, Z. , "Low-Cost Pathway to Ultra Efficient City Car: Series Hydraulic Hybrid System with Optimized Supervisory Control," *SAE International Journal of Engines*, **2**(2), pp. 505-520, 2010.

[16] Powell, W. B., Approximate dynamic programming: solving the curses of dimensionality, Wiley-Interscience., 2007.

[17] Sutton, R. S. and Barto, A. G., Reinforcement learning an introduction, MIT Press., 1998.

[18] Bertsekas, D. P. and Tsitsiklis, J. N., Neuro-dynamic programming, Athena Scientific., 1996.

[19] Tsitsiklis, J.N. and Roy, B. van, "Feature-based methods for large scale dynamic programming," *Machine Learning*, **22**(1-3), pp. 59-94, 1996.

[20] Bertsekas, D. P., Homer, M. L., Logan, D. A., Patek, S. D., and Sandell, N. R., "Missile defense and interceptor allocation by neuro-dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, **30**(1), pp. 42-51, 2000.

[21] Cybenko, G. , "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, **2**(4), pp. 303-14, 1989.

[22] Tesauro, G. , "Practical issues in temporal difference learning," *Machine Learning*, **8**(3-4), pp. 257-77, 1992.

[23] Saad, D. , On-line learning in neural networks, Cambridge University Press., 1998.

[24] Kim, Y. and Filipi, Z. , "Simulation Study of a Series Hydraulic Hybrid Propulsion System for a Light Truck," *SAE Transactions, Journal of Commercial Vehicles*, **116**(2007-01-4151), pp. 147-161, 2007.

[25] Assanis, D. , Filipi, Z. , Gravante, S. , Grohnke, D. , Gui, X. , Louca, L. , Rideout, G. , Stein, J. , and Wang, Y. , "Validation and Use of SIMULINK Integrated, High Fidelity, Engine-In-Vehicle Simulation of the International Class VI Truck," *SAE Transactions: Journal of Engines*, **109**(2000-01-0288), 2000.

[26] Fathy, H.K. , Filipi, Z.S. , Hagena, J. , and Stein, J.L. , "Review of hardware-in-the-loop simulation and its prospects in the automotive area," *Proceedings of the SPIE - The International Society for Optical Engineering*, **6228**(1), pp. 117-136, 2006.

[27] Wu, B. , Lin, C. C., Filipi, Z. , Peng, H. , and Assanis, D. , "Optimal power management for a hydraulic hybrid delivery truck," *Vehicle System Dynamics*, **42**(1-2), pp. 23-40, 2004.

[28] Pourmovahed, A. , Beachley, N.H. , and Fronczak, F.J. , "Modeling of a hydraulic energy regeneration system. Part I. Analytical treatment," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, **114**(1), pp. 155-159, 1992.

[29] Pourmovahed, A. , Baum, S.A. , Fronczak, F.J. , and Beachley, N.H. , "Experimental Evaluation Of Hydraulic Accumulator Efficiency With And Without Elastomeric Foam.," *Journal of Propulsion and Power*, **4**(2), pp. 185-192, 1988.

[30] Lee, T.-K. and Filipi, Z. S., "Synthesis and validation of representative real-world driving cycles for Plug-In Hybrid vehicles," presented at Vehicle Power and Propulsion Conference, pp. 1-6, 2010.

[31] Bichi, M. , Ripaccioli, G. , Cairano, S. Di, Bernardini, D. , Bemporad, A. , and Kolmanovsky, I.V. , "Stochastic model predictive control with driver behavior learning for improved powertrain control," presented at 49th IEEE Conference on Decision and Control (CDC), pp. 6077-6082, 2010.

CONTACT INFORMATION

Zoran Filipi*
Mechanical Engineering,
University of Michigan
1231 Beal Avenue,
Ann Arbor, Michigan, USA 48109
filipi@umich.edu
* Corresponding Author

ACKNOWLEDGMENTS

The financial support for this research has been provided by the State of Michigan 21st Century Jobs Fund in partnership with the Bosch-Rexroth Corporation, Engineering Hydraulics - North America. In addition, the authors wish to acknowledge technical interactions with Simon Baseley and Ed Greif, both at Bosch-Rexroth.