5-2018

Real-Time Stochastic Predictive Control for Hybrid Vehicle Energy Management.

Kyle R. Williams
Purdue University

REAL-TIME STOCHASTIC PREDICTIVE CONTROL FOR HYBRID VEHICLE ENERGY MANAGEMENT

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kyle R. Williams

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Monika M. Ivantysynova, Chair
School of Mechanical Engineering

Dr. Gregory M. Shaver
School of Mechanical Engineering

Dr. Kartik B. Ariyur
School of Mechanical Engineering

Dr. Andrea Vacca
School of Mechanical Engineering

Approved by:

Dr. Jay P. Gore
Head of the School Graduate Program

For Sara and India.

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Prof. Monika Ivantysynova. The concepts developed in this work are the result of her encouragement and direction provided during my time at the Maha Fluid Power Research Center. Monika's outstanding guidance has made a profound impact on my career, I feel very fortunate to have been part of her research group. I would also like to thank my advisory committee members Profs. Shaver, Vacca and Ariyur for their invaluable input, particularly during my preliminary defense.

I am truly indebted to Ryan, Anthony, Leo and Mateus for their tremendous support during the experimental phase of this work. Ryan setup the entire test rig, and together with Leo and Mateus, worked through every issue the rig encountered. I cannot emphasize enough how much I appreciated Anthony's unparalleled commitment to the lab. From start to finish Anthony was always there moving this test rig forward. Without the help of these individuals the experiments would not have happened. I also wanted to thank my fellow Maha researchers, past and present, for providing many insightful discussions and questions along the way. I am grateful for the help I have received from the staff within the Maha Lab and the ME grad office. Specifically, Susan Gauger, Julayne Moser, Connie McMindes, and Cathy Elwell have helped me so very much over the past five years.

Finally, I need to thank my wife, Sara, and my daughter, India. Working full time for a large corporation while independently pursuing a PhD over the past five years was difficult. I never would have finished without Sara's strength and constant support. Thank you, India, for your patience. I started this four months before you were born and I am so happy to be done before you get a moment older. Moving forward, I will never again have to miss a zoo trip with you to go work on my PhD.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Williams, Kyle R. Ph.D., Purdue University, May 2018. Real-Time Stochastic Predictive Control for Hybrid Vehicle Energy Management. Major Professor: Monika Ivantysynova, School of Mechanical Engineering.

This work presents three computational methods for real time energy management in a hybrid hydraulic vehicle (HHV) when driver behavior and vehicle route are not known in advance. These methods, implemented in a receding horizon control (aka model predictive control) framework, are rather general and can be applied to systems with nonlinear dynamics subject to a Markov disturbance. State and input constraints are considered in each method. A mechanism based on the steady state distribution of the underlying Markov chain is developed for planning beyond a finite horizon in the HHV energy management problem. Road elevation information is forecasted along the horizon and then merged with the statistical model of driver behavior to increase accuracy of the horizon optimization. The characteristics of each strategy are compared and the benefit of learning driver behavior is analyzed through simulation on three drive cycles, including one real world drive cycle. A simulation is designed to explicitly demonstrate the benefit of adapting the Markov chain to real time driver behavior. Experimental results demonstrate the real time potential of the primary algorithm when implemented on a processor with limited computational resources.

1. INTRODUCTION AND STATE OF THE ART

1.1 Introduction

The hybrid vehicle offers a solution for personal, public and commercial transportation vehicles which can significantly reduce fuel consumption and engine emissions output in comparison to conventional vehicle solutions. Figure 1.1 shows fuel consumption vs. vehicle size in square feet for conventional and hybrid vehicles. Typ-

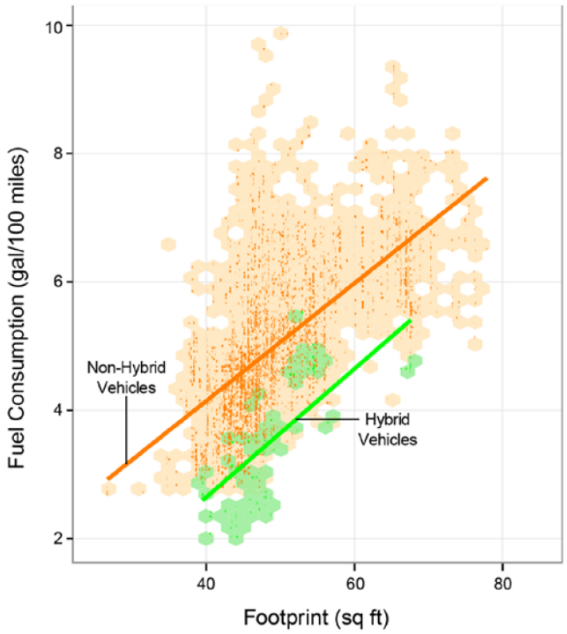

Fig. 1.1. Source: U.S. Department Of Energy, 2014.

ically, for the same size vehicle the hybrid solution offers significantly reduced fuel consumption. By incorporating a reversible energy storage device on-board the hybrid vehicle, kinetic energy conventionally dissipated as heat during braking can be recovered during a process known as regenerative braking. As a secondary benefit, the hybrid vehicle offers greater flexibility in engine management than a conventional

vehicle. The uncertain nature of driver behavior and driving environment presents one of the biggest challenges in hybrid vehicle control. In both hybrid electric vehicles (HEVs) and hydraulic hybrid vehicles (HHVs), the control system must ensure proper charge of the reversible energy storage to ensure future driver demands can be satisfied while also observing system constraints and maximizing overall system efficiency. As such, the challenge of optimally managing the engine and reversible energy sources has been an area of active research over the past two decades. This challenge has focused on the development of control strategies which minimize an objective function based on fuel consumption and/or engine emissions while maintaining vehicle drivability and satisfying system constraints. The development of these strategies has included, but is not limited to, modeling driver behavior, modeling changes in the driving environment, creating an objective function to reflect the optimization goal, incorporating real time telematics information, and developing control methods which incorporate all mentioned models and information to optimize the given objective.

Hydraulic hybrid vehicles can be competitive with and even outperform HEVs in terms of fuel savings at a reduced cost [1]. Figure 1.2 compares fuel economy of a series HHV compared to a series HEV in city driving when the power to weight ratio of the vehicle is low. The series HHV has an advantage of the series HEV in urban

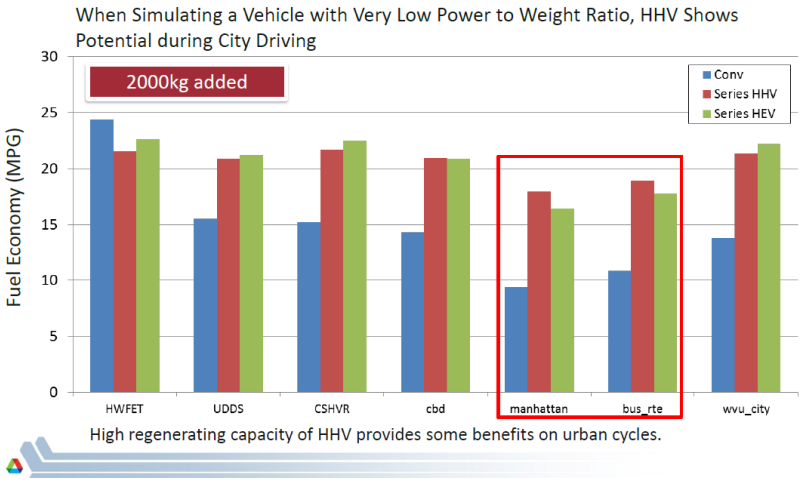

Fig. 1.2. Series HHV vs HEV. Source: U.S. Department Of Energy, 2012.

routes when rapid energy transfer to and from the energy storage device is required. The benefit of the HHV can be explained with a plot of energy density vs. power density as shown in Fig. 1.3. Although batteries typically have greater energy density than hydraulic accumulators, the greater power density of a hydraulic accumulator means the HHV can potentially store and reuse energy much more quickly.

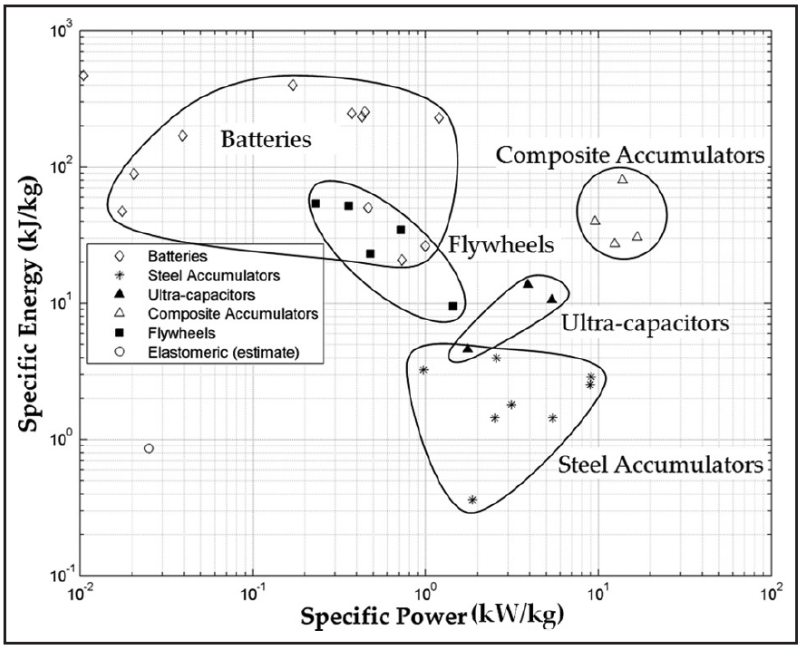

Fig. 1.3. Specific energy versus specific power of various energy storage devices [2].

1.2 State of the Art

The state of the art in hybrid vehicle energy management is reviewed. No significant differentiation between control strategies for HEV vs. HHV is made, since any given strategy can typically be applied to either HEV or HHV with straight-forward adjustment.

1.2.1 Heuristic Policies and Instantaneous Optimization

Energy management for hybrid vehicles (both HEVs and HHVs) is an old problem. Early solutions involved finite horizon dynamic programming (DP) simulations for predefined drive cycles. By creating a time-varying value function $V(\mathbf{x}, t)$, dynamic programming can determine a globally optimal open loop control trajectory for a given drive cycle. A major drawback is the resulting open loop control trajectories are only valid for the specific drive cycle under investigation. To generate an implementable controller, heuristic feedback policies were extracted from the DP results in an attempt to replicate the properties of optimal open loop control trajectories [3, 4]. A downside of heuristic strategies is they must optimistically hope the cycle being driven resembles the training cycle (that is, the cycle(s) on which the heuristic rules were formed). Instantaneous optimization strategies were developed to alleviate the need for human-formed rules. These methods perform real time optimization, producing control inputs which instantaneously minimize fuel consumption or emissions in response to the present operating condition of the vehicle [5–7]. An interesting connection between a type of instantaneous optimization called equivalent consumption minimization strategy (ECMS) [8, 9] and Pontraygin's Minimum Principle (PMP) is presented in [10]. A method for real time energy management based explicitly on PMP is developed in [11]. Here, the authors fix a co-state value associated with the real time solution of PMP which influences fuel consumption results. The challenge with this approach is pairing the best co-state value for the cycle being driven in order to minimize fuel.

1.2.2 Stochastic Methods

A completely different solution category for energy management is developed when a statistical model of driver behavior is incorporated into the solution strategy. A statistical model known as a Markov chain has proven an effective approach for capturing driver behavior [12, 13]. Stochastic dynamic programming (SDP) methods [14] work

directly with the Markov chain to formulate globally optimal time-varying control policies $\mathbf{u} = \boldsymbol{\mu}(\mathbf{x}, t)$ which consider driver statistics, minimizing the expected or average running cost of the objective function over a time horizon. In [1, 15, 16], energy management strategies based on SDP in an infinite horizon setting are developed. Infinite horizon SDP mathematically formulates a time-invariant value function $V(\mathbf{x})$ based on statistics of several drive cycles, from which globally optimal state-feedback control policy $\mathbf{u} = \boldsymbol{\mu}(\mathbf{x})$ can be constructed. A major advantage of the infinite horizon SDP approach is the state-feedback control policy can be implemented in a lookup table manner for real time vehicle control. In the relatively recent work of [17], experiments are carried out on a modified Volvo S-80 HEV using a state-feedback control policy based on SDP. An interesting comparison between finite horizon DP and infinite horizon SDP as applied to a hydraulic hybrid vehicle is discussed in [18].

Like its deterministic counterpart, SDP scales poorly to problems involving large state spaces and becomes computationally intractable for very large problems. Neuro-Dynamic Programming (NDP) [19–21] alleviates the scaling issue through the use of neural networks. In NDP, the value function is represented as a parameterized neural network, $\hat{V}(\mathbf{x}, \boldsymbol{\theta})$, and then tuned by adjusting parameters $\boldsymbol{\theta}$ in order to satisfy the associated Bellman equations. As a result, the value of many states can be adjusted at once by adjusting a single parameter. Using neural networks in this way allows NDP to efficiently handle significantly larger state spaces than SDP since not every state must be visited during construction of the value function. Neuro-Dynamic Programming is employed in [22] to minimize an impressively complex objective function comprising fuel consumption and engine emissions in a HHV.

A shortcoming of computationally intensive stochastic methods such as SDP and NDP is that the resulting control policies are based on models of driver behavior which are typically not adapted in real time. The findings in [23] suggest that stochastically robust methods such as SDP may not provide optimal fuel economy in hybrid vehicles when cycle mispredictions exist. Such mispredictions can be caused, for example, when the Markov chain model used in the SDP formulation is not representative of

the actual drive cycle, emphasizing the need for adaptation of the statistical model if stochastic methods are to be employed.

1.2.3 Model Predictive Control Methods

Model predictive control (MPC) [24] is fundamentally characterized by the *fast* computation of a finite horizon optimization at every time step. The underlying solver can be based on DP, PMP, SDP, quadratic programming (QP), or other general nonlinear programming type methods [25]. At each timestep, MPC generates an open loop control trajectory $\{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}\}$. The first control input $\mathbf{u}_0$ is applied to the system and then the finite horizon optimization re-starts with up-to-date system information. One of the biggest advantages to the MPC method is that real time information can be incorporated to make immediate changes to the problem formulation, resulting in an control trajectory that is more closely tuned to present driving conditions. In [26], model predictive control is used for energy management of an HEV with driver torque demand modeled as an exponentially decreasing process along the horizon according to $\tau_{n+1} = \alpha\tau_n$ with $0 < \alpha < 1$. In [27], MPC is used for energy management of a HHV with driver demand assumed constant along the horizon. The finite horizon optimization is solved using Newton's method with logarithmic barrier functions [28].

Model predictive control can incorporate forecasted information provided by onboard telematics such as a global positioning system. The authors of [29] use path forecasting in the form of previewed vehicle speed and road grade in a hybrid electric vehicle. In a similar approach, road grade is previewed along a horizon assuming constant vehicle speed in a conventional vehicle in [30]. Since the state and action spaces are low in [29] and [30], dynamic programming is used to perform the finite horizon optimization.

1.2.4 Predictive Methods Under Uncertainty

1.2.4.1 Stochastic Model Predictive Control

Stochastic model predictive control (SMPC) methods [31, 32] combine the statistical decision making associated with SDP and NDP with the real time computation of MPC. A unique challenge to SMPC is the development of computationally efficient solvers which can handle the computational burden associated with stochastic optimization. A stochastic QP solver for Markov Jump Linear Systems with transition probability estimation is presented in [33]. Here, driver behavior is represented as a Markov chain and Monte Carlo sampling is used to generate several driver demand paths with relatively high likelihood. To reduce computational burden sample paths with low likelihood are not considered in the problem formulation. A key feature of the method is that the Markov transition probabilities are adapted in real time to the actual drive cycle. The developed strategy performs nearly as well as a benchmark strategy which has full access to the drive cycle and significantly outperforms a strategy incorporating no learning mechanism, indicating that significant benefit can be achieved when the Markov chain is adapted in real time. A method for predicting road grade is incorporated in the framework of SMPC in [34]. In addition to driver behavior, road grade is modeled as a Markov chain and the subsequent stochastic optimization is performed with finite horizon SDP with reported execution times between 10 and 100 seconds.

1.2.4.2 Neural Network Predictors

Neural networks (NN) are used to predict driver acceleration demand and vehicle velocity along a finite horizon in [35]. An MPC formulation based on [36] is used to carry out the finite horizon optimization. A major finding in [35] is that an MPC strategy based on NN-based velocity predictions outperforms the same strategy incorporating Markov chain-based velocity predictors. The NN and Markov chain

were both trained on a large data set and evaluated on a separate data set. It is worth noting the authors of [35] explicitly state no learning mechanisms were used to estimate the parameters of the Markov chain in real time, possibly eliminating one of the most flexible and useful attributes of the Markov chain driver modeling approach for hybrid vehicle energy management.

1.3 Research Goals and Contributions

The primary goal of this research is to develop a control algorithm for hybrid vehicle energy management, with the ultimate goal of maximizing fuel economy. Since driver actions are largely uncertain, the algorithm should be able to consider consequences of possible future driver actions during planning of the state and control trajectories. The algorithm needs to be flexible enough to adapt in real time to driver behavior, and additionally, incorporate real time telematics information in order to reduce uncertainty during planning. A secondary goal of this research is to determine the degree to which learning driver behavior and incorporating real time telematics information can improve fuel economy. A third and final goal of this research is to experimentally demonstrate the algorithm is capable of controlling a hybrid powertrain using a resource limited processor.

1.3.1 Contributions

The primary contributions of this work are:

- Three novel computational methods for real time energy management in a HHV when driver behavior and vehicle route are not known in advance are developed in Chapter 5. These methods, implemented in a receding horizon control (aka model predictive control) framework, are rather general and can be applied to systems with nonlinear dynamics subject to a Markov disturbance. State and control constraints are considered in each method.

- A novel mechanism for planning beyond a finite horizon in the HHV energy management problem is investigated. This mechanism is based on the steady state distribution of the underlying Markov chain model describing driver behavior. The method is initially discussed in Section 4.3 and incorporated into HHV energy management in Section 5.3.

- Road elevation information is forecasted along the horizon and for the first time is merged with the statistical model of driver behavior to increase accuracy of the horizon optimization. The method of incorporating road grade information is developed in Section 5.2.

- The impact of incorrect statistical information, and the required time to adapt to correct statistical information, is for the first time investigated in Section 6.3.2.

- Real time potential of the novel computational methods is assessed for the first time through an experimental setup discussed in Chapter 7.

1.4 Organization of Chapters

The next chapter summarizes several of the underlying concepts and methods of optimal control and reinforcement learning which have been widely used in vehicle control applications. Several of these concepts lead to the development of the algorithms in Chapter 5. Chapter 3 presents an overview of hybrid vehicles and hybrid vehicle dynamics.

In chapter 4, a statistical model of driver behavior based on a Markov chain is presented. The Markov chain is adapted in real time to the drive cycle according to a simple filtering process described in [33]. The Markov multi-step transition probability matrix is analyzed as a mechanism to model driver actions along a horizon. Driver behavior from three drive cycles, including one cycle obtained from real-world

driving measurements, is analyzed. The steady state distribution of the Markov chain model is presented as a way to plan beyond a finite horizon.

Chapter 5 presents three novel methods for real time energy management of an HHV when driver behavior and vehicle route are not known in advance. A simplified discrete-time model of the system dynamics is explained. Two benchmark methods are also created, one is a theoretically best-achievable controller and the second is a simplified strategy based on instantaneous optimization.

In Chapter 6, simulations are carried out. The characteristics of each strategy are compared and the benefit of learning driver behavior is analyzed. A simulation is designed to explicitly demonstrate the benefit of adapting the Markov chain to real time driver behavior. The statistical driver model is initialized on incorrect cycle statistics, then allowed to adapt to the driven cycle. Learning typically converges in 2-3 runs of the given cycle, corresponding to 20 to 60 minutes.

An experiment is performed on a series HHV test rig setup in Chapter 7. The purpose of the experiment is to (1) demonstrate that the computationally intensive algorithms developed in Chapter 5 can run in real time on a processor with limited computational resources and (2) demonstrate the algorithm can successfully control a series hybrid using a simplified control-oriented model of the real physics.

1.5 Notation

symbol	meaning
$\mathbf{x}$	vector
$\mathbf{x}(t)$	vector at time t
$\mathbf{x}_n$	vector at timestep n
$\mathbf{x}_n^\mathsf{T}$	vector at timestep n transposed
$x_{i,n}$	the i^{th} element of a vector $\mathbf{x}$ at timestep n
$\vec{\mathbf{x}}$	$= \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}\} = (\mathbf{x}_n)_{n=0}^{N-1}$ a sequence of vectors
$\vec{\mathbf{x}}^{[k]}$	the k^{th} iteration of vector sequence $\vec{\mathbf{x}}$
$J(\mathbf{x}, \mathbf{u})$	a function evaluated at $\mathbf{x}, \mathbf{u}$
$J^{(x)}(\mathbf{x}_n, \mathbf{u}_n)$	partial of J wrt argument $\mathbf{x}$, evaluated at $\mathbf{x}_n, \mathbf{u}_n$

2. BACKGROUND

This chapter summarizes several of the underlying concepts and methods of optimal control and reinforcement learning which have been widely used in vehicle control applications.

2.1 Deterministic Optimal Control

Consider the discrete time dynamic system described by

$$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n), \quad n = 0, 1, \ldots \tag{2.1}$$

where $\mathbf{x}_n \in \mathbb{R}^{dimX}$ and $\mathbf{u}_n \in \mathbb{R}^{dimU}$ are the system state and control input vectors, respectively, and $\mathbf{x}_0$ is given. The dynamics described by Equation (2.1) can represent a large class of systems, including the discrete time evolution of an inherently continuous time process[1] $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t)$ according to

$$F_n(\mathbf{x}_n, \mathbf{u}_n) = \mathbf{x}_n + \int_{n\Delta t}^{(n+1)\Delta t} f(\mathbf{x}(\tau), \mathbf{u}(\tau))d\tau$$

where $t \geq 0$ and $\mathbf{x}(0)$ is given. The horizon cost

$$J_0 = h(\mathbf{x}_N) + \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{u}_n) \tag{2.2}$$

is the sum of a terminal cost $h(\mathbf{x}_N)$ and a time-varying running cost $g_n(\mathbf{x}_n, \mathbf{u}_n)$ which is affected by the state and control input at each stage in the horizon. The goal of optimal control is to design an appropriate control sequence $\vec{\mathbf{u}} = (\mathbf{u}_n)_{n=0}^{N-1}$ which

[1]The notation $\dot{\mathbf{x}}$ represents $\frac{d\mathbf{x}(t)}{dt}$

minimizes the receding horizon cost J_0 when the system starts from initial state $\mathbf{x}_0$ at time $n = 0$ and is subjected to the control sequence $\vec{\mathbf{u}}$ along the horizon. Additionally, control and state constraints must be satisfied at all points along the horizon. The minimization problem is formally stated as

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \left\{ h(\mathbf{x}_N) + \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{u}_n) \right\} \tag{2.3a}$$

$$\text{subject to} \quad \mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n) \tag{2.3b}$$

$$\mathbf{x}_n \in \mathbf{X} \tag{2.3c}$$

$$\mathbf{u}_n \in \mathbf{U} \tag{2.3d}$$

$$n = 0, 1, \ldots, N-1 \tag{2.3e}$$

where $\mathbf{X}$ and $\mathbf{U}$ are the constrained state and control sets, respectively.

2.1.1 Nonlinear Programming

Perhaps the most straightforward and popular approach for solving Equation (2.3) is by transforming the problem into a nonlinear program [25]. Nonlinear programming refers to the general process of solving an optimization problem subject to equality and inequality constraints in the decision variables. The most common nonlinear programming method used by far in optimal control is *quadratic programming* (QP). A typical QP problem is formulated as

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^\top Q\mathbf{z} + q^\top \mathbf{z} \\ \text{subject to} \quad & A\mathbf{z} \leq \mathbf{b} \\ & D\mathbf{z} = \mathbf{c} \end{aligned}$$

The finite horizon optimal control problem Equation (2.3) can be transformed into a QP problem by approximating the horizon cost with a quadratic function and linearizing the system dynamics about some nominal trajectory $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1}$. Neglecting

for simplicity the terminal cost $h(\mathbf{x}_N)$, the horizon cost Equation (2.2) can be approximated with the quadratic function

$$J_0 \approx \sum_{n=0}^{N-1} \frac{1}{2}\mathbf{z}_n^\mathsf{T} Q_n \mathbf{z_n} + q_n^\mathsf{T}\mathbf{z_n} \tag{2.4a}$$

where

$$Q_n = \begin{bmatrix} g_n^{(xx)} & g_n^{(xu)} \\ g_n^{(ux)} & g_n^{(uu)} \end{bmatrix}_{(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)} \qquad q_n = \begin{bmatrix} g_n^{(x)} \\ g_n^{(u)} \end{bmatrix}_{(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)} \tag{2.4b}$$

$$\mathbf{z_n} = \begin{bmatrix} \delta\mathbf{x}_n \\ \delta\mathbf{u}_n \end{bmatrix} \tag{2.4c}$$

The system dynamics can be linearized according to

$$\delta\mathbf{x}_{n+1} = A_n\delta\mathbf{x}_n + B_n\delta\mathbf{u}_n \tag{2.5a}$$

$$A_n = F_n^{(x)}(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n) \tag{2.5b}$$

$$B_n = F_n^{(u)}(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n) \tag{2.5c}$$

where $(\delta\mathbf{x}_n, \delta\mathbf{u}_n)_{n=0}^{N-1}$ is a small perturbation from the nominal trajectory. The equivalent QP problem can then be described by

$$\min_{\mathbf{z}} \ \frac{1}{2}\mathbf{z}^\mathsf{T} Q \mathbf{z} + q^\mathsf{T}\mathbf{z} \tag{2.6a}$$

$$Q = \begin{bmatrix} Q_0 & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_{N-1} \end{bmatrix} \tag{2.6b}$$

$$q^\mathsf{T} = \begin{bmatrix} q_0 & q_1 & \dots & q_{N-1} \end{bmatrix} \tag{2.6c}$$

$$\mathbf{z}^\mathsf{T} = \begin{bmatrix} \delta\mathbf{x}_0 & \delta\mathbf{u}_0 & \delta\mathbf{x}_1 & \delta\mathbf{u}_1 \dots \delta\mathbf{x}_{N-1} & \delta\mathbf{u}_{N-1} \end{bmatrix} \tag{2.6d}$$

subject to

$$\delta \mathbf{x}_{n+1} - A_n \delta \mathbf{x}_n - B_n \delta \mathbf{u}_n = \mathbf{0} \tag{2.6e}$$

$$\mathbf{a} - \hat{\mathbf{x}}_n \leq \delta \mathbf{x}_n \leq \mathbf{b} - \hat{\mathbf{x}}_n \tag{2.6f}$$

$$\mathbf{c} - \hat{\mathbf{u}}_n \leq \delta \mathbf{u}_n \leq \mathbf{d} - \hat{\mathbf{u}}_n \tag{2.6g}$$

Once the QP problem Equation (2.6) has been solved, the nominal trajectory is updated according to $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1} \leftarrow (\hat{\mathbf{x}}_n + \delta \mathbf{x}_n, \hat{\mathbf{u}}_n + \delta \mathbf{u}_n)_{n=0}^{N-1}$ and the process is restarted. The broad use of the quadratic programming approach for solving the finite horizon optimal control problem can perhaps be attributed to the availability of powerful tools which can efficiently solve Equation (2.6) by exploiting the underlying sparsity of the equivalent problem due to matrix Equation (2.6b) [25].

2.1.2 The Minimum Principle

Unlike the nonlinear programming approach, the minimum principle solves Equation (2.3) using a variational approach. For fixed $\mathbf{x}_0$, let the finite horizon cost of control sequence $\vec{\mathbf{u}}$ be given by

$$J_0(\vec{\mathbf{u}}, \mathbf{x}_0) = h(\mathbf{x}_N) + \sum_{n=0}^{N-1} \Big(g_n(\mathbf{x}_n, \mathbf{u}_n) \tag{2.7}$$

Define the *Hamiltonian*

$$H(\mathbf{x}, \mathbf{u}, \lambda) = g(\mathbf{x}, \mathbf{u}) + \lambda^{\mathsf{T}} F(\mathbf{x}, \mathbf{u}) \tag{2.8}$$

where λ serves as a dynamic Lagrange multiplier ensuring the system dynamics constraint $\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n)$ is satisfied. The horizon cost becomes

$$J_0(\vec{\mathbf{u}}, \mathbf{x}_0) = h(\mathbf{x}_N) + \sum_{n=0}^{N-1} \Big[H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1}) - \lambda_{n+1}^{\mathsf{T}} \mathbf{x}_{n+1} \Big] \Big($$

$$= h(\mathbf{x}_N) - \lambda_N^{\mathsf{T}} \mathbf{x}_N + \lambda_0^{\mathsf{T}} \mathbf{x}_0 + \sum_{n=0}^{N-1} \Big[H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1}) - \lambda_n^{\mathsf{T}} \mathbf{x}_n \Big] \Big($$

The variation in $J_0(\vec{\mathbf{u}}, \mathbf{x}_0)$ due to small variations $\delta \mathbf{u}_n$ about the nominal control sequence $\vec{\mathbf{u}}$ is

$$\delta J_0(\vec{\mathbf{u}}, \mathbf{x}_0) = \left[\frac{\partial h(\mathbf{x}_N)}{\partial \mathbf{x}_N} - \lambda_N \right]^{\mathsf{T}} \delta \mathbf{x}_N + \lambda_0^{\mathsf{T}} \delta \mathbf{x}_0 + \sum_{n=0}^{N-1} \left\{ \left[\frac{\partial H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1})}{\partial \mathbf{x}_n} - \lambda_n^{\mathsf{T}} \right] \delta \mathbf{x}_n + \frac{\partial H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1})}{\partial \mathbf{u}_n} \delta \mathbf{u}_n \right\} \quad (2.9)$$

To enforce the condition $\frac{\partial J_n}{\partial \mathbf{x}_n} = 0$ along an optimal trajectory, the Lagrange multipliers are chosen to satisfy

$$\lambda_n = \frac{\partial H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1})}{\partial \mathbf{x}_n} \qquad \lambda_N = \frac{\partial h(\mathbf{x}_N)}{\partial \mathbf{x}_N} \quad (2.10)$$

Noticing that $\delta \mathbf{x}_0 = 0$ (since the initial state is fixed) and substituting Equation (2.10) into Equation (2.9)

$$\delta J_0(\vec{\mathbf{u}}, \mathbf{x}_0) = \sum_{n=0}^{N-1} \frac{\partial H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1})}{\partial \mathbf{u}_n} \delta \mathbf{u}_n$$

Assuming any control constraints $\mathbf{U}_n$ are convex, the following necessary condition for local optimality of $\vec{\mathbf{u}}^*$ is established [37]

$$\frac{\partial H(\mathbf{x}_n, \mathbf{u}_n^*, \lambda_{n+1})}{\partial \mathbf{u}_n} (\mathbf{u}_n - \mathbf{u}_n^*) \geq 0 \ , \quad n = 0, ..., N-1, \forall \mathbf{u}_n \in \mathbf{U}_n \quad (2.11)$$

2.1.2.1 Global Optimality

If the system dynamics $F(\mathbf{x}, \mathbf{u})$ are linear in $\mathbf{u}$ and the running cost g_n is convex in $\mathbf{u}$, the Hamiltonian $H(\mathbf{x}, \mathbf{u}, \lambda) = g(\mathbf{x}, \mathbf{u}) + \lambda^\mathsf{T} F(\mathbf{x}, \mathbf{u})$ is convex in $\mathbf{u}_n$. In this case, local necessary condition Equation (2.11) is equivalent to the stronger necessary and sufficient condition

$$\mathbf{u}_n^* = \arg\min_{\mathbf{u}_n} \{H(\mathbf{x}_n, \mathbf{u}_n, \lambda_{n+1})\} \tag{2.12}$$

Convexity of H ensures any local optimum is a global optimum, but can only be established with the restrictions on $F(\mathbf{x}, \mathbf{u})$ and $g(\mathbf{x}, \mathbf{u})$ mentioned above.

2.1.2.2 Constraints

Incorporating state and control constraints in the framework of the minimum principle is more challenging than in the nonlinear programming approach. See [38] for a treatment of applying inequality constraints within the framework of the minimum principle.

2.1.3 Dynamic Programming

The minimum principle finds a locally optimal control sequence which minimizes the finite horizon cost Equation (2.2), and under certain restrictions, this control sequence is globally optimal. Dynamic programming (DP), alternatively, always finds a globally optimal state-feedback control *policy*, a mapping from the states and time to control inputs $\pi : \mathbf{X} \times T \to \mathbf{U}$, regardless of restrictions on F and g. Dynamic programming exploits Bellman's *principle of optimality*, which states that if a given state-action sequence is optimal, and we remove the first state and action, the remaining sequence is also optimal (with the second state of the original sequence now acting as the initial state). Under this principle, the problem of minimizing Equation (2.2) is broken down into many smaller problems in a stage-wise manner. Dynamic pro-

gramming constructs a *state value function* $V_n(\mathbf{x})$, a record of the optimal cost-to-go from any state $\mathbf{x}$ at any time n to the end of the horizon

$$\begin{aligned} V_n(\mathbf{x}) &= \min_{\mathbf{u}_n,\ldots,\mathbf{u}_{N-1}} J_n \ \mathbf{x}_n = \mathbf{x} \\ &= \min_{\mathbf{u}_n,\ldots,\mathbf{u}_{N-1}} \left[h(\mathbf{x}_N) + \sum_{k=n}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k) \ \mathbf{x}_n = \mathbf{x} \right] \end{aligned} \tag{2.13}$$

The state value function can be described recursively as

$$\begin{aligned} V_n(\mathbf{x}) &= \min_{\mathbf{u}_n \in \mathbf{U}} \left\{ g_n(\mathbf{x}, \mathbf{u}_n) + \underbrace{\min_{\mathbf{u}_{n+1},\ldots,\mathbf{u}_{N-1}} \left[h(\mathbf{x}_N) + \sum_{k=n+1}^{N-1} g_k \ \mathbf{x}_{n+1} = F_n(\mathbf{x}, \mathbf{u}_n) \right]}_{V_{n+1}(F_n(\mathbf{x},\mathbf{u}_n))} \right\} \\ &= \min_{\mathbf{u}_n \in \mathbf{U}} \left[g_n(\mathbf{x}, \mathbf{u}_n) + V_{n+1}(F_n(\mathbf{x}, \mathbf{u}_n)) \right] \end{aligned} \tag{2.14}$$

with boundary condition

$$V_N(\mathbf{x}) = h(\mathbf{x}) \tag{2.15}$$

The optimal state-feedback control policy can be inferred directly from the state value function through

$$\pi_n^*(\mathbf{x}) = \arg\min_{\mathbf{u}_n \in U} \left[g_n(\mathbf{x}, \mathbf{u}_n) + V_{n+1}(F_n(\mathbf{x}, \mathbf{u}_n)) \right] \tag{2.16}$$

Equations (2.14) and (2.16) are referred to as the Bellman equations. These equations can be solved recursively by working backwards along the horizon from boundary condition Equation (2.15). At each horizon stage $V_n(\mathbf{x})$ is computed for every state $\mathbf{x}$ using Equation (2.14) starting from boundary condition $V_N(\mathbf{x}) = h(\mathbf{x})$. For computational feasibility, the state space is usually discretized and the state dynamics are projected onto the discretization according to $\mathbf{x}_{n+1} = \text{proj}\,[F_n(\mathbf{x}_n, \mathbf{u}_n)]$.

2.1.3.1 Constraints

State and input constraints are easily handled with dynamic programming. Any $\mathbf{x} \notin \mathbf{X}$ is assigned an arbitrarily large value $V_n(\mathbf{x})$, preventing the state feedback control policy π from ever designing a control input that will lead to any $\mathbf{x} \notin \mathbf{X}$. Control constraints are handled by restricting the optimization Equation (2.14) to only search through feasible controls $\mathbf{u}_n \in \mathbf{U}$, and restricting the control policy Equation (2.16) to choose from feasible $\mathbf{u}_n$.

2.1.3.2 Computational complexity

It is worth mentioning that the computational effort associated with dynamic programming grows considerably with the size of the state space. Quantizing each dimension of the state space $\mathbf{X} \subset \mathbb{R}^{dimX}$ into quant_x levels produces a state space of size $|\mathbf{X}| = \text{quant}_\text{x}{}^{dimX}$. For an N length horizon problem, this amounts to performing $O(N \cdot |\mathbf{X}|) = O(N \cdot \text{quant}_\text{x}{}^{dimX})$ optimization problems of the form Equation (2.14). Although dynamic programming is still much more efficient than exploring every possible state path which amounts to $O(|\mathbf{X}|^N)$ evaluations, performing dynamic programming quickly in even a moderately sized state space presents a considerable challenge. Because of this famous *curse of dimensionality*, dynamic programming is, for the most part, real time prohibitive.

2.1.4 DDP / iLQR

Differential dynamic programming (DDP) [39, 40] and the closely related iterative linear quadratic regulator (iLQR) [41, 42] are dynamic programming methods in which a quadratic approximation to the value function Equation (2.14) is created at each point along the horizon. In creating this quadratic approximation, DDP uses a second order expansion of the system dynamics while iLQR uses a first order expansion. The associated benefit of DDP over iLQR is improved convergence at the

expense of additional computation, however, depending on the application, it can be beneficial to choose faster computation over improved convergence (e.g. in a model predictive control setting in which convergence will never actually happen and computation is a premium). Like the minimum principle, these methods generate a locally optimal control sequence rather than a globally optimal control policy as in dynamic programming. Defining the *state-control value function* $Q_n(\mathbf{x}, \mathbf{u})$ as

$$Q_n(\mathbf{x}, \mathbf{u}) = g_n(\mathbf{x}, \mathbf{u}) + V_{n+1}\big(f_n(\mathbf{x}, \mathbf{u})\big) \tag{2.17}$$

the state value function can be expressed as

$$V_N(\mathbf{x}) = h(\mathbf{x}) \tag{2.18}$$

$$V_n(\mathbf{x}) = Q_n(\mathbf{x}, \mathbf{u}^*) \tag{2.19}$$

where $\mathbf{u}^* = \arg\min_{\mathbf{u}} Q_n(\mathbf{x}, \mathbf{u})$ is the value that minimizes Equation (2.17). Given a nominal trajectory, $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1}$, a local quadratic model of Q_n can be constructed as

$$\begin{aligned} &Q_n(\hat{\mathbf{x}}_n + \delta\mathbf{x}_n, \hat{\mathbf{u}}_n + \delta\mathbf{u}_n) \\ &\qquad \approx Q_n^{(0)} + Q_n^{(\mathbf{x})}\delta\mathbf{x}_n + Q_n^{(u)}\delta\mathbf{u}_n + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_n^\mathsf{T} & \delta\mathbf{u}_n^\mathsf{T}\end{bmatrix}\begin{bmatrix} Q_n^{(xx)} & Q_n^{(xu)} \\ Q_n^{(ux)} & Q_n^{(uu)}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}_n \\ \delta\mathbf{u}_n\end{bmatrix} \end{aligned} \tag{2.20}$$

For given $\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, \delta\mathbf{x}_n$, the value of $\delta\mathbf{u}_n$ which minimizes this local model of Q_n is given by

$$\delta\mathbf{u}_n^* = \arg\min_{\delta\mathbf{u}_n} Q_n = -\left(Q_n^{(uu)}\right)^{-1}\left(Q_n^{(u)} + Q_n^{(ux)}\delta\mathbf{x}_n\right) \tag{2.21}$$

The various partial derivatives $Q_n^{(\cdot)} = \nabla_{(\cdot)} Q(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)$ are determined considering Equation (2.17)

$$
\begin{aligned}
Q_n^{(0)} &= g_n + V_{n+1} & Q_n^{(xx)} &= g_n^{(xx)} + F_n^{(x)\mathsf{T}} V_{n+1}^{(xx)} F_n^{(x)} + V_{n+1}^{(x)} \cdot F_n^{(xx)} \\
Q_n^{(x)} &= g_n^{(x)} + V_{n+1}^{(x)} F_n^{(x)} & Q_n^{(ux)} &= g_n^{(ux)} + F_n^{(u)\mathsf{T}} V_{n+1}^{(xx)} F_n^{(x)} + V_{n+1}^{(x)} \cdot F_n^{(ux)} \\
Q_n^{(u)} &= g_n^{(u)} + V_{n+1}^{(x)} F_n^{(u)} & Q_n^{(uu)} &= g_n^{(uu)} + F_n^{(u)\mathsf{T}} V_{n+1}^{(xx)} F_n^{(u)} + V_{n+1}^{(x)} \cdot F_n^{(uu)}
\end{aligned} \tag{2.22}
$$

where the ij^{th}component of each matrix in the last three equations is defined as

$$
\left(V_{n+1}^{(x)} \cdot F_n^{(xx)}\right)_{ij} = V_{n+1}^{(x)} \cdot \frac{\partial^2 F_n}{\partial x_i \partial x_j} \tag{2.23a}
$$

$$
\left(V_{n+1}^{(x)} \cdot F_n^{(ux)}\right)_{ij} = V_{n+1}^{(x)} \cdot \frac{\partial^2 F_n}{\partial u_i \partial x_j} \tag{2.23b}
$$

$$
\left(V_{n+1}^{(x)} \cdot F_n^{(uu)}\right)_{ij} = V_{n+1}^{(x)} \cdot \frac{\partial^2 F_n}{\partial u_i \partial u_j} \tag{2.23c}
$$

The second order terms of Equation (2.23) are ignored in iLQR, while in DDP they are included. Substituting $\delta \mathbf{u}_n^*$ from Equation (2.21) into the local model Equation (2.20) and simplifying gives a local model for $V_n(\mathbf{x}_n)$ about $\mathbf{x}_n = \hat{\mathbf{x}}_n + \delta \mathbf{x}_n$

$$
\begin{aligned}
V_n(\hat{\mathbf{x}}_n + \delta \mathbf{x}_n) \approx Q_n^{(0)} - \frac{1}{2} Q_n^{(u)\mathsf{T}} (Q_n^{(uu)})^{-1} Q_n^{(u)} + \left[Q_n^{(x)} - Q_n^{(u)} (Q_n^{(uu)})^{-1} Q_n^{(ux)}\right] \delta \mathbf{x}_n \\
+ \frac{1}{2} \delta \mathbf{x}_n^{\mathsf{T}} \left[Q_n^{(xx)} - Q_n^{(xu)} (Q_n^{(uu)})^{-1} Q_n^{(ux)}\right] \delta \mathbf{x}_n
\end{aligned} \tag{2.24}
$$

Equating terms in the Taylor series expansion for $V_n(\mathbf{x}_n)$ gives an update for the partial derivatives of the value function

$$
V_n^{(0)}(\hat{\mathbf{x}}_n) = Q_n^{(0)} - \frac{1}{2} Q_n^{(u)\mathsf{T}} (Q_n^{(uu)})^{-1} Q_n^{(u)} \tag{2.25a}
$$

$$
V_n^{(x)}(\hat{\mathbf{x}}_n) = Q_n^{(x)} - Q_n^{(u)} (Q_n^{(uu)})^{-1} Q_n^{(ux)} \tag{2.25b}
$$

$$
V_n^{(xx)}(\hat{\mathbf{x}}_n) = Q_n^{(xx)} - Q_n^{(xu)} (Q_n^{(uu)})^{-1} Q_n^{(ux)} \tag{2.25c}
$$

A new trajectory $(\mathbf{x}_n, \mathbf{u}_n)_{n=0}^{N-1}$ is simulated using the current measurement of the system state according to

$$\mathbf{x}_0 = \mathbf{x}_0^{\text{meas}} \tag{2.26a}$$

$$\mathbf{u}_n^* = \hat{\mathbf{u}}_n \underbrace{- \left(Q_n^{(uu)}\right)^{-1} Q_n^{(u)} - \left(Q_n^{(uu)}\right)^{-1} Q_n^{(ux)} \left(\mathbf{x}_n - \hat{\mathbf{x}}_n\right)}_{\delta \mathbf{u}_n^*} \tag{2.26b}$$

$$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n^*) \tag{2.26c}$$

Starting from initial condition $V_N(\hat{\mathbf{x}}_N) = h(\hat{\mathbf{x}}_N)$ Equations (2.22) and (2.25) are solved backwards in time from $n = N$ to $n = 0$ constituting the *backwards pass*. Starting from initial condition $\mathbf{x}_0 = \mathbf{x}_0^{\text{meas}}$, a new system trajectory is then simulated according to Equation (2.26) which constitutes the *forward pass*. This simulated trajectory is then used as the new nominal trajectory $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1} := (\mathbf{x}_n, \mathbf{u}_n)_{n=0}^{N-1}$, and the process is restarted.

By creating a local model of the value function through differentials, DDP and iLQR solve two major issues associated with dynamic programming. For one, DDP / iLQR work directly with a continuous state space, so there is no need to artificially discretize the state. Secondly, DDP and iLQR converge much faster than DP as they do not require a visit to each state in the state space during the backward sweep. A stochastic variant of differential dynamic programming suitable for real time computation is proposed in section 5.3.2.

2.1.4.1 Constraints

Choosing an optimal control input which minimizes Equation (2.20) at each stage n in the horizon amounts to a stage-wise quadratic programming problem. Formulating this stage-wise QP problem in the context of DDP remains an active area of research. Box-bounded control input constraints are addressed in [43], general state and control inequality constraints are considered in the recent work of [44]. In this work, state and input constraints are addressed in the stochastic setting in Section 5.3.2.

2.2 Systems with Stochastic Dynamics

This section discusses the basis principles of stochastic systems as relevant to optimal control and reinforcement learning problems. The stochastic systems considered here can be described by the difference equation

$$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, \mathbf{w}_n), \ n = 0, 1, \dots \tag{2.27a}$$

$$\mathbf{x}_0 : \ given \tag{2.27b}$$

$$\mathbf{w}_0 : \ given \tag{2.27c}$$

where $\mathbf{w}_n \in \mathbf{W}$ is a stochastic disturbance input to the system.

2.2.1 Stochastic Optimization

Stochastic optimization refers to a collection of methods for minimizing an objective function when a stochastic effect is present [45]. Consider the objective function $J(\boldsymbol{\theta}, \vec{\mathbf{w}})$ which depends on the decision parameter $\boldsymbol{\theta}$ and the sequence of stochastic disturbances $\vec{\mathbf{w}} = \{\mathbf{w}_0, \mathbf{w}_1, \dots\}$. The parameter $\boldsymbol{\theta}$ is quite general and can represent the terms of a control sequence or the parameters of a parameterized control policy. The goal of stochastic optimization is then to minimize the expected value

$$\min_{\boldsymbol{\theta}} \mathbb{E}[J(\boldsymbol{\theta}, \vec{\mathbf{w}})] = \sum_{\vec{\mathsf{w}}} J(\boldsymbol{\theta}, \vec{\mathsf{w}}) \Pr[\vec{\mathbf{w}} = \vec{\mathsf{w}}] \tag{2.28}$$

2.2.1.1 Sample Average Approximation

Typically, the stochastic optimization Equation (2.28) cannot be solved directly due to the combinatorial difficulty of computing $\mathbb{E}[J(\boldsymbol{\theta}, \vec{\mathbf{w}})]$. An alternative is to first compute the sample average

$$\hat{J}(\boldsymbol{\theta}) = \sum_{k=1}^{K} J(\boldsymbol{\theta}, \vec{\mathbf{w}}^{[k]}) \tag{2.29}$$

and then minimize $\hat{J}(\boldsymbol{\theta})$ through nonlinear programming methods. The approximation $\hat{J}(\boldsymbol{\theta})$ improves as the number of samples K increases in accordance to the Central Limit Theorem which states the difference between the sample average and true average convergence to a zero mean Normal distribution whose variance depends directly on the number of samples K

$$\hat{J}(\boldsymbol{\theta}) - \mathbb{E}[J(\boldsymbol{\theta}, \vec{\mathbf{w}})] \xrightarrow{d} \mathcal{N}\left(0, \frac{\sigma^2}{K}\right)\Bigg(\tag{2.30}$$

where σ^2 is the variance of $J(\boldsymbol{\theta}, \vec{\mathbf{w}})$.

2.2.1.2 Stochastic Approximation

Stochastic approximation is an iterative method which uses noisy measurements to find the root of a function, $H(\boldsymbol{\theta}^*) = \mathbf{0}$, when $H(\boldsymbol{\theta})$ cannot be computed directly[2] but noisy sample observations $\mathbf{y}^{[k]} = H(\boldsymbol{\theta}^{[k]}) + \boldsymbol{\eta}^{[k]}$ are available. It is assumed that $\boldsymbol{\eta}^{[k]}$ is a zero-mean noise process so that $\mathbf{y}^{[k]}$ is an unbiased estimate of $H(\boldsymbol{\theta}^{[k]})$ in the sense that $\mathbb{E}[y^{[k]}] = H(\boldsymbol{\theta}^{[k]})$. The stochastic approximation iteration is

$$\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^{[k]} + \alpha^{[k]}\mathbf{y}^{[k]} \tag{2.31}$$

with the two following conditions imposed on the learning rate $\alpha^{[k]}$

$$\sum_{k=0}^{\infty} \alpha^{[k]} = \infty \qquad \sum_{k=0}^{\infty} \left(\alpha^{[k]}\right)^2 < \infty \tag{2.32}$$

Roughly speaking, the first condition ensures the sequence is non-terminating so that asymptotic convergence properties hold, while the second condition ensures the noise in the samples does not dominate algorithm progress. An intuitive justification of Equation (2.32) in the context of mean estimation is provided in [46]. The aggregate behavior of Equation (2.31) with learning rates Equation (2.32) can be assessed

[2]It may be the case that $H(\boldsymbol{\theta})$ is inaccessible, or it may be too expensive to compute directly.

through the *ODE method* [19,47], which states Equation (2.31) asymptotically tracks the ordinary differential equation

$$\dot{\boldsymbol{\theta}}(t) = H\left(\boldsymbol{\theta}(t)\right) \tag{2.33}$$

2.2.1.3 Gradient Descent Form of Stochastic Approximation

When $H(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}}\mathbb{E}[J(\boldsymbol{\theta}, \vec{\mathbf{w}})]$, stochastic approximation finds a local solution to Equation (2.28) using noisy gradient observations $\mathbf{y}^{[k]} = -\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \vec{\mathbf{w}}^{[k]})$ forming a process known as *stochastic gradient descent* [47]. Convergence of Equation (2.33) can be shown by constructing the Lyapunov function $V(\boldsymbol{\theta}) = \mathbb{E}[J(\boldsymbol{\theta}, \vec{\mathbf{w}})]$ and showing $\frac{dV}{dt} < 0$ through

$$\frac{dV}{dt} = \nabla_{\boldsymbol{\theta}}\mathbb{E}[J(\boldsymbol{\theta}(t), \vec{\mathbf{w}})] \cdot \dot{\boldsymbol{\theta}}(t) = -\left(\nabla_{\boldsymbol{\theta}}\mathbb{E}[J(\boldsymbol{\theta}(t), \vec{\mathbf{w}})]\right)^2 < 0$$

Convergence to $\boldsymbol{\theta}^*$ implies $\mathbf{0} = H(\boldsymbol{\theta}^*) = -\nabla_{\boldsymbol{\theta}}\mathbb{E}[J(\boldsymbol{\theta}^*, \vec{\mathbf{w}})]$, satisfying the necessary conditions for a local minimum assuming $\boldsymbol{\theta}$ is unconstrained (convergence proofs of stochastic gradient descent can be found in [19, 47, 48]). The algorithm proposed in Section 5.3.1 is based on stochastic gradient descent.

2.2.1.4 Fixed Point Form of Stochastic Approximation

A central concept in online learning is the fixed point form of stochastic approximation [47] in which

$$H(\boldsymbol{\theta}) = F(\boldsymbol{\theta}) - \boldsymbol{\theta} \tag{2.34}$$

and F is contractive so that $||F(\boldsymbol{\theta}_a) - F(\boldsymbol{\theta}_b)||_2 \leq \lambda||\boldsymbol{\theta}_a - \boldsymbol{\theta}_b||_2$ for $0 \leq \lambda < 1$. Convergence of Equation (2.33) to equilibrium $\boldsymbol{\theta}^*$ is shown by constructing a Lyapunov function $V(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_2^2$ and employing the ODE method Equation (2.33)

$$\begin{aligned}\frac{dV}{dt} &= (\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) \cdot \dot{\boldsymbol{\theta}}(t) \\ &= (\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) \cdot (F(\boldsymbol{\theta}(t)) - F(\boldsymbol{\theta}^*)) + (\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) \cdot (F(\boldsymbol{\theta}^*) - \boldsymbol{\theta}(t)) \\ &\leq ||\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*||_2 \; ||F(\boldsymbol{\theta}(t)) - F(\boldsymbol{\theta}^*)||_2 - ||\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*||_2^2 \\ &\leq -(1 - \lambda) \; ||\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*||_2^2\end{aligned}$$

Convergence to $\boldsymbol{\theta}^*$ implies $\mathbf{0} = H(\boldsymbol{\theta}^*) = F(\boldsymbol{\theta}^*) - \boldsymbol{\theta}^*$. Convergence can also be proved for general norms $|| \cdot ||_p, \quad p \geq 1$ [47].

2.2.2 Markov Decision Processes

When the disturbance term $\mathbf{w}_n$ in Equation (2.27) obeys the *Markov property*, which roughly states that future behavior of a system is influenced only by the present state, ignoring the sequence of events that lead to the present state, the system dynamics take on a particularly simplified form known as a Markov Decision Process (MDP) in which the state transitions are given in terms of a controlled distribution

$$\mathbf{x}_{n+1} \sim p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) \tag{2.35}$$

When the state space $\mathbf{X}$ is continuous, the distribution is a density function defined by

$$\int\!\!\int_{\mathbf{x}'} p(\mathbf{s}|\mathbf{x}, \mathbf{u}) d\mathbf{s} = \Pr[\mathbf{x}_{n+1} \in \mathbf{x}'|\mathbf{x}_n = \mathbf{x}, \mathbf{u}_n = \mathbf{u}] \tag{2.36}$$

If a discrete space is assumed, the distribution simplifies to a mass function

$$p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) = \Pr[\mathbf{x}_{n+1} = \mathbf{x}'|\mathbf{x}_n = \mathbf{x}, \mathbf{u}_n = \mathbf{u}] \tag{2.37}$$

In this context, the state vector now includes all deterministic and modeled stochastic states. The model $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ can be determined empirically through direct interaction with the environment or through first principles modeling assuming some form of the stochastic effect.

2.3 Stochastic Dynamic Programming

As in the deterministic setting, stochastic dynamic programming (SDP) uses a model of the environment to construct a state-value function, a record of the optimal cost-to-go from each state in the state space. The goal of SDP is to minimize

$$\min_{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}} \mathbb{E}\left[h(\mathbf{x}_N) + \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{u}_n) \;\; \mathbf{x}_0 = \mathbf{x} \right] \tag{2.38}$$

where system dynamics are governed according to the Markov decision process $\mathbf{x}_{n+1} \sim p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$.

2.3.0.1 Finite Horizon SDP

The finite horizon cost of following policy $\mathbf{u}_n = \pi_n(\mathbf{x}_n)$ is given by

$$J^\pi = h(\mathbf{x}_N) + \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \pi_n(\mathbf{x}_n)) \tag{2.39}$$

The expected cost-to-go of following policy $\mathbf{u}_n = \pi_n(\mathbf{x}_n)$ starting from state $\mathbf{x}$ at time n until time N is represented by the *policy value*, $V_n^\pi(\mathbf{x})$

$$V_N^\pi(\mathbf{x}) = \mathbb{E}\left[h(\mathbf{x}_N) \;\; \mathbf{x}_N = \mathbf{x} \right] = h(\mathbf{x}) \tag{2.40}$$

$$\begin{aligned} V_n^\pi(\mathbf{x}) &= \mathbb{E}\left[\sum_{k=n}^{N-1} g_n(\mathbf{x}_k, \pi(\mathbf{x}_k, k)) + h(\mathbf{x}_N) \;\; \mathbf{x}_n = \mathbf{x} \right] \\ &= g(\mathbf{x}, \pi_n(\mathbf{x})) + \mathbb{E}\Big[V_{n+1}^\pi(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x} \Big] \end{aligned} \tag{2.41}$$

The *state value function* $V_n(\mathbf{x})$ results from following the optimal time-varying state-feedback policy $\pi_n^*(\mathbf{x})$, which must satisfy the *Bellman equation*

$$V_n(\mathbf{x}) = \min_{\mathbf{u}\in\mathbf{U}} \Big\{ g_n(\mathbf{x},\mathbf{u}) + \mathbb{E}\Big[V_{n+1}(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x} \Big] \Big\} \Bigg(\tag{2.42}$$

The optimal policy can be inferred directly from the state value function through

$$\pi_n^*(\mathbf{x}) = \arg\min_{\mathbf{u}\in\mathbf{U}} \Big\{ g_n(\mathbf{x},\mathbf{u}) + \mathbb{E}\Big[V_{n+1}(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x} \Big] \Big\} \Bigg(\tag{2.43}$$

These two equations provide a means to recursively compute the state value function exactly by working backward through time starting from N, using a model of the environment $p(\mathbf{x}'|\mathbf{x},\mathbf{u})$

$$V_N(\mathbf{x}) = h(\mathbf{x}) \tag{2.44}$$

$$V_n(\mathbf{x}) = \min_{\mathbf{u}\in\mathbf{U}} \Big\{ g_n(\mathbf{x},\mathbf{u}) + \sum_{\mathbf{x}'\in\mathbf{X}} p(\mathbf{x}'|\mathbf{x},\mathbf{u}) V_{n+1}(\mathbf{x}') \Big\} \Bigg(\tag{2.45}$$

$$\pi_n^*(\mathbf{x}) = \arg\min_{\mathbf{u}\in\mathbf{U}} \Big\{ g_n(\mathbf{x},\mathbf{u}) + \sum_{\mathbf{x}'\in X} p(\mathbf{x}'|\mathbf{x},\mathbf{u}) V_{n+1}(\mathbf{x}') \Big\} \Bigg(\tag{2.46}$$

For this reason, finite horizon dynamic programming in both the deterministic and stochastic case is often referred to as backward dynamic programming.

2.3.0.2 Infinite Horizon SDP

Finite horizon dynamic programming constructs a state value function which explicitly depends on time, even if the instantaneous cost and system dynamics are independent of time. As a result, the optimal state feedback policy, which is inferred directly from the state value function, also depends explicitly on time. The benefit of working in an infinite horizon is that the state value function and therefore the state feedback policy is invariant with time, as long as the instantaneous cost $g(\mathbf{x},\mathbf{u})$

and process dynamics $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ are independent of time [37]. The discounted infinite horizon cost of following policy $\mathbf{u}_n = \pi(\mathbf{x}_n)$ is given by

$$J^{\pi} = \sum_{k=0}^{\infty} \gamma^k g(\mathbf{x}_k, \pi(\mathbf{x}_k)) \tag{2.47}$$

where the discount factor $0 < \gamma < 1$ serves to reduce the impact of costs incurred far into the future on the immediate cost prediction. The expected cost-to-go following policy $\mathbf{u}_n = \pi(\mathbf{x}_n)$ from state $\mathbf{x}$ starting at arbitrary time n is given by the policy value

$$\begin{aligned} V^{\pi}(\mathbf{x}) &= \mathbb{E}\left[\sum_{k=n}^{\infty} \gamma^k g(\mathbf{x}_k, \pi(\mathbf{x}_k)) \;\; \mathbf{x}_n = \mathbf{x}\right] \\ &= g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \, \mathbb{E}\Big[V^{\pi}(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x}\Big] \end{aligned} \tag{2.48}$$

The state value function must satisfy the infinite horizon Bellman equation

$$V(\mathbf{x}) = \min_{\mathbf{u} \in \mathbf{U}} \Big\{ g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}\Big[V(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x}\Big] \Big\} \tag{2.49}$$

and the optimal policy can be inferred directly from the state value function through

$$\pi^*(\mathbf{x}) = \arg\min_{\mathbf{u} \in \mathbf{U}} \Big\{ g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}\Big[V(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x}\Big] \Big\} \tag{2.50}$$

Constructing the state value function is less straight-forward in the infinite horizon case, as working backwards through time is not possible since a terminal time does not exist. Rather, an approximation to state value function, $\hat{V}(\mathbf{x})$, can be solved for iteratively in a process called *value iteration*, treating the resulting Bellman equation

$$\hat{V}(\mathbf{x}) = \min_{\mathbf{u} \in \mathbf{U}} \Big\{ g(\mathbf{x}, \mathbf{u}) + \gamma \sum_{\mathbf{x}' \in \mathbf{X}} p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) \hat{V}(\mathbf{x}') \Big\} \tag{2.51}$$

as a consistency condition. Under mild conditions the operation

$$T[\hat{V}(\mathbf{x})] = \min_{\mathbf{u}\in\mathbf{U}} \left\{ g(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{x}'\in\mathbf{X}} \left(p(\mathbf{x}'|\mathbf{x},\mathbf{u})\hat{V}(\mathbf{x}') \right\} \right(\tag{2.52}$$

is a contraction mapping, so the fixed point iteration

$$\hat{V}^{[k+1]} = T[\hat{V}^{[k]}] \tag{2.53}$$

converges to V as long as each $\mathbf{x} \in \mathbf{X}$ is repeatedly visited. The approximation error after K iterations is bounded by $||\hat{V}^{[k]} - V|| \leq \lambda^K ||\hat{V}^{[0]} - V||$ for the norm $||V|| = \max_{\mathbf{x}} V(\mathbf{x})$. Equations (2.52) and (2.53) together form value iteration. The state value function can also be found in a process known as *policy iteration*, in which the policy value is solved for exactly at each iteration by solving the system of linear equations

$$V^{\pi}(\mathbf{x}) = g(\mathbf{x},\pi(\mathbf{x})) + \gamma \sum_{\mathbf{x}'\in\mathbf{X}} \left(p(\mathbf{x}'|\mathbf{x},\pi(\mathbf{x}))V^{\pi}(\mathbf{x}') \quad \forall \mathbf{x} \in \mathbf{X} \right. \tag{2.54}$$

and making the policy update

$$\pi(\mathbf{x}) \leftarrow \arg\min_{\mathbf{u}\in\mathbf{U}} \left\{ g(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{x}'\in\mathbf{X}} \left(p(\mathbf{x}'|\mathbf{x},\mathbf{u})V^{\pi}(\mathbf{x}') \right\} \right(\quad \forall \mathbf{x} \in \mathbf{X} \tag{2.55}$$

Equations (2.54) and (2.55) form policy iteration. Convergence is guaranteed since $||V^{\pi}||$ must decrease on every iteration [14]. When the control space $\mathbf{U}$ is finite, convergence occurs in a finite number of iterations since there are only finitely many policies in a discrete action and state space. A third process known as *modified policy iteration* combines value iteration with policy iteration. Rather than being solved for exactly, the policy value is updated for several iterations through the update

$$\hat{V}^{\pi}(\mathbf{x}) \leftarrow g(\mathbf{x},\pi(\mathbf{x})) + \gamma \sum_{\mathbf{x}'\in\mathbf{X}} \left(p(\mathbf{x}'|\mathbf{x},\pi(\mathbf{x}))\hat{V}^{\pi}(\mathbf{x}') \quad \forall \mathbf{x} \in \mathbf{X} \right. \tag{2.56}$$

After several iterations of Equation (2.56), the policy is updated according to Equation (2.55). Value iteration is easier to implement than policy iteration, as solving Equation (2.54) exactly is computationally expensive especially when the state space $\mathbf{X}$ is large. However, policy iteration typically converges faster than value iteration since $V^{\pi}(\mathbf{x})$ is exact at each policy update. Modified policy iteration lies somewhere in-between, removing the need to exactly compute the system of Equations (2.54) but providing a better estimate of $V^{\pi}(\mathbf{x})$ at each iteration.

2.4 Reinforcement Learning: Model-Free Value Function Methods

In the infinite horizon setting, dynamic programming provides a framework for iteratively computing the value of a policy which requires a model of the environment, $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$. *Reinforcement learning methods*, unlike dynamic programming, do not require a model of the environment to compute the value of a policy.

2.4.1 Monte Carlo Estimation

Monte Carlo estimation provides a method of policy evaluation based on samples of the discounted infinite horizon cost. Following policy π the cost occurred at each time step is recorded. In the infinite horizon setting the final estimate can be based on the truncated series

$$\tilde{J}^{\pi}(\mathbf{x}_0) = \sum_{n=0}^{N-1} \gamma^n g(\mathbf{x}_n, \pi(\mathbf{x}_n)) \tag{2.57}$$

The full series can be decomposed into the the truncated portion and a bounded term

$$J^{\pi}(\mathbf{x}_0) = \sum_{n=0}^{\infty} \gamma^n g(\mathbf{x}_n, \pi(\mathbf{x}_n)) = \sum_{n=0}^{N-1} \gamma^n g(\mathbf{x}_n, \pi(\mathbf{x}_n)) + \underbrace{\gamma^N \sum_{n=0}^{\infty} \gamma^n g(\mathbf{x}_{n+N}, \pi(\mathbf{x}_{n+N}))}_{\leq \frac{\gamma^N}{1-\gamma} g_{max}}$$

Since all quantities above are positive, the series truncation error is bounded by

$$J^{\pi}(\mathbf{x}_0) - \tilde{J}^{\pi}(\mathbf{x}_0) \ \leq \frac{\gamma^N}{1-\gamma} g_{max}$$

The Monte Carlo update to the state policy value is given by

$$\hat{V}^{\pi}(\mathbf{x}) \leftarrow \hat{V}^{\pi}(\mathbf{x}) + \alpha(\mathbf{x})\Big[\tilde{J}^{\pi}(\mathbf{x}) - \hat{V}^{\pi}(\mathbf{x})\Big] \tag{2.58}$$

The learning rate α can be a function of the number of visits k to state $\mathbf{x}$, $\alpha(\mathbf{x}) = \frac{1}{k(\mathbf{x})}$, so that Equation (2.58) constructs the empirical average or it can be a fixed value $0 < \alpha < 1$ so that Equation (2.58) creates a noisy average with exponentially fading memory.

2.4.2 Temporal-Difference Learning

Like SDP, temporal difference (TD) algorithms provide a means for policy evaluation. Unlike SDP, TD methods do not rely on a model of the environment $p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}))$ to compute the policy value. Assuming a finite state space, the TD policy value update is

$$\hat{V}^{\pi}(\mathbf{x}) \leftarrow \hat{V}^{\pi}(\mathbf{x}) + \alpha(\mathbf{x})\Big[\underbrace{g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \hat{V}^{\pi}(\mathbf{x}') - \hat{V}^{\pi}(\mathbf{x})}_{\text{temporal difference}}\Big] \tag{2.59}$$

Here, $\mathbf{x}$ is the state value at time n and $\mathbf{x}'$ is the observed state value at time $n+1$. The learning rate $\alpha(\mathbf{x})$ is a function of the number of visits k to state $\mathbf{x}$ and satisfies Equation (2.32). The *temporal difference* is the difference between the one-sample estimate of the cost-to-go from state $\mathbf{x}'$ at time $n+1$ and the current policy value estimate of state $\mathbf{x}$ at time n. In view of stochastic approximation of Section 2.2.1.2, this is a stochastic fixed point iteration with $y = g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \hat{V}^{\pi}(\mathbf{x}') - \hat{V}^{\pi}(\mathbf{x})$ providing an unbiased estimate of $g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}[\hat{V}^{\pi}(\mathbf{x}_{n+1})|\mathbf{x}_n = \mathbf{x}] - \hat{V}^{\pi}(\mathbf{x})$. Convergence of the TD update Equation (2.59) implies $\hat{V}^{\pi}(\mathbf{x}) = g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}[\hat{V}^{\pi}(\mathbf{x}_{n+1})|\mathbf{x}_n = \mathbf{x}]$

which satisfies the dynamic programming based policy evaluation Equation (2.48), indicating $\hat{V}^{\pi}(\mathbf{x})$ converges to $V^{\pi}(\mathbf{x})$ for all $\mathbf{x}$ as long as each state is repeatedly visited ($\alpha(\mathbf{x})$ must tend to zero for each $\mathbf{x}$).

2.4.3 Q-learning

If a record of the state-control value function $Q(\mathbf{x}, \mathbf{u})$ was available such that $V(\mathbf{x}) = \min_{\mathbf{u}} Q(\mathbf{x}, \mathbf{u})$, finding the optimal control policy would no longer require a model of the environment as Equation (2.50) reduces to

$$\begin{aligned} \pi^*(\mathbf{x}) &= \arg\min_{\mathbf{u}} \Big\{ g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}\Big[V(\mathbf{x}_{n+1}) \;\; \mathbf{x}_n = \mathbf{x} \Big] \Big\} \\ &= \arg\min_{\mathbf{u}} Q(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{2.60}$$

where the state-control value function satisfies

$$\begin{aligned} Q(\mathbf{x}, \mathbf{u}) &= g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}[V(\mathbf{x}_{n+1}) \; \mathbf{x}_n = \mathbf{x}] \\ &= g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}\Big[\min_{\mathbf{v}} Q(\mathbf{x}_{n+1}, \mathbf{v}) \; \mathbf{x}_n = \mathbf{x} \Big] \end{aligned} \tag{2.61}$$

More generally, the state-control policy function associated with following policy $\pi(\mathbf{x})$, $Q^{\pi}(\mathbf{x}, \mathbf{u})$, satisfies

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}\Big[Q^{\pi}(\mathbf{x}_{n+1}, \pi(\mathbf{x}_{n+1})) \; \mathbf{x}_n = \mathbf{x} \Big] \tag{2.62}$$

as $V^{\pi}(\mathbf{x}) = Q^{\pi}(\mathbf{x}, \pi(\mathbf{x}))$. The breakthrough known as Q-learning constructs such a state-control value function through trial and error interaction with the environment. The Q-learning update

$$\hat{Q}(\mathbf{x}, \mathbf{u}) \leftarrow \hat{Q}(\mathbf{x}, \mathbf{u}) + \alpha(\mathbf{x}, \mathbf{u}) \Big[g(\mathbf{x}, \mathbf{u}) + \gamma \min_{\mathbf{v}} \hat{Q}(\mathbf{x}', \mathbf{v}) - \hat{Q}(\mathbf{x}, \mathbf{u}) \Big] \tag{2.63}$$

is based on stochastic fixed point iteration of Section 2.2.1.2 and can be viewed as a generalization of temporal difference learning. Here $\mathbf{x}$ and $\mathbf{u}$ are the state and control at time n, and $\mathbf{x}'$ is the state observed at time $n+1$. The learning rate $\alpha(\mathbf{x},\mathbf{u})$ is a function of the number of visits k to state-control pair $(\mathbf{x},\mathbf{u})$ and satisfies Equation (2.32). According to stochastic approximation theory, $\hat{Q}(\mathbf{x},\mathbf{u})$ converges to $Q(\mathbf{x},\mathbf{u})$ for all $(\mathbf{x},\mathbf{u})$ provided that all controls continue to be tried from all states, and each state is repeatedly visited ($\alpha(\mathbf{x},\mathbf{u})$ must tend to zero for each $(\mathbf{x},\mathbf{u})$).

2.5 Value Function Approximation

Working with a state space $\mathbf{X}$ that is finite (i.e. discrete and bounded) admits *tabular solutions*, in which the state value function can be described by a simple lookup table representation. Value function approximation (VFA) (also known as approximate dynamic programming, adaptive dynamic programming, and neuro dynamic programming) provides a means to approximately construct the value function when $\mathbf{X}$ becomes large or even infinite (i.e. if $\mathbf{X}$ is continuous), in which case filling out entries of a tabular representation of the value function becomes computationally intractable. Value function approximation is concerned with the weighted least squares problem

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\frac{1}{2}d(\mathbf{x},\boldsymbol{\theta})^2\right] = \min_{\theta} \sum_{\mathbf{x}} \rho(\mathbf{x}) \frac{1}{2}\Big[\underbrace{\tilde{V}^{\pi}(\mathbf{x}) - \hat{V}(\mathbf{x},\boldsymbol{\theta})}_{\triangleq d(\mathbf{x},\boldsymbol{\theta})}\Big]^2 \tag{2.64a}$$

when a state value function is learned and

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\frac{1}{2}d(\mathbf{x},\mathbf{u},\boldsymbol{\theta})^2\right] = \min_{\boldsymbol{\theta}} \sum_{\mathbf{x},\mathbf{u}} \rho(\mathbf{x},\mathbf{u}) \frac{1}{2}\Big[\underbrace{\tilde{Q}^{\pi}(\mathbf{x},\mathbf{u}) - \hat{Q}(\mathbf{x},\mathbf{u},\boldsymbol{\theta})}_{\triangleq d(\mathbf{x},\mathbf{u},\boldsymbol{\theta})}\Big]^2 \tag{2.64b}$$

when a state-control value function is learned. Here $\rho(\cdot)$ is some distribution among the states or state-control pairs, $\tilde{V}^{\pi}(\mathbf{x})$ and $\tilde{Q}^{\pi}(\mathbf{x},\mathbf{u})$ are sample estimates of policy values $V^{\pi}(\mathbf{x})$ and $Q^{\pi}(\mathbf{x},\mathbf{u})$ based on available information (and are not retained in

memory), and $\hat{V}(\mathbf{x}, \boldsymbol{\theta})$, $\hat{Q}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta})$ are parameterized approximations of $\tilde{V}^{\pi}(\mathbf{x})$ and $\tilde{Q}^{\pi}(\mathbf{x}, \mathbf{u})$ (which are retained in memory). These approximations can be constructed as radial basis functions, neural networks, etc. A popular choice is the linear approximation[3] with (possibly nonlinear) basis function $\boldsymbol{\phi}$,

$$\hat{V}(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x})^{\mathsf{T}} \boldsymbol{\theta} \qquad \hat{Q}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x}, \mathbf{u})^{\mathsf{T}} \boldsymbol{\theta} \tag{2.65}$$

There are typically many more states or state-control pairs than elements of $\boldsymbol{\theta}$, so changing one element of $\boldsymbol{\theta}$ changes the estimated value of many states or state-control pairs. The least squares problem Equation (2.64a) can be solved through stochastic gradient descent [20] according to the update

$$\begin{aligned} \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha(\mathbf{x}) \nabla_{\boldsymbol{\theta}} \frac{1}{2} d(\mathbf{x}, \boldsymbol{\theta})^2 \\ &= \boldsymbol{\theta} + \alpha(\mathbf{x}) \left[\tilde{V}^{\pi}(\mathbf{x}) - \hat{V}(\mathbf{x}, \boldsymbol{\theta}) \right] \nabla_{\boldsymbol{\theta}} \hat{V}(\mathbf{x}, \boldsymbol{\theta}) \end{aligned} \tag{2.66}$$

where learning rate $\alpha(\mathbf{x})$ is a function of the number of visits to $\mathbf{x}$ and satisfies Equation (2.32). Similarly, Equation (2.64b) can be solved through the update [20]

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\mathbf{x}, \mathbf{u}) \left[\tilde{Q}^{\pi}(\mathbf{x}, \mathbf{u}) - \hat{Q}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \right] \nabla_{\boldsymbol{\theta}} \hat{Q}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \tag{2.67}$$

The least squares problem can be solved through a variety of other methods including batch least squares such as averaged steepest descent and Gauss-Newton iteration or incremental least squares such as Kalman or Extended Kalman filtering [19]. If a state policy value function is learned, the optimal policies can be formed with a model of the environment through

$$\pi(\mathbf{x}) = \arg\min_{\mathbf{u}} \left\{ g(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E} \left[\hat{V}(\mathbf{x}_{n+1}, \boldsymbol{\theta}) | \mathbf{x}_n = \mathbf{x} \right] \right\} \tag{2.68}$$

[3] It is worth noting that lookup table methods are a special case of linear approximation with as many elements of $\boldsymbol{\theta}$ as states (state VFA) or state-control pairs (state-control VFA), with the basis function serving as an indicator function.

whereas learning a state-control policy value allows construction of the optimal policy in a model-free setting

$$\pi(\mathbf{x}) = \arg\min_{\mathbf{u}} \hat{Q}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}) \tag{2.69}$$

The sample estimates are often formed through dynamic programming or temporal difference updates according to

$$\text{DP estimate of } V^\pi: \quad \tilde{V}^\pi(\mathbf{x}) = g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}\Big[\hat{V}(\mathbf{x}_{n+1}, \boldsymbol{\theta}) \;\; \mathbf{x}_n = \mathbf{x}\Big] \Bigg(= g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \iint_{\mathbf{x}'} p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x})) \hat{V}(\mathbf{x}', \boldsymbol{\theta}) \Bigg) d\mathbf{x}' \tag{2.70a}$$

$$\text{TD estimate of } V^\pi: \quad \tilde{V}^\pi(\mathbf{x}) = g(\mathbf{x}, \pi(\mathbf{x})) + \gamma \hat{V}(\mathbf{x}', \boldsymbol{\theta}) \tag{2.70b}$$

$$\text{TD estimate of } Q^\pi: \quad \tilde{Q}^\pi(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + \gamma \hat{Q}(\mathbf{x}', \pi(\mathbf{x}'), \boldsymbol{\theta}) \tag{2.70c}$$

At each time step $(\mathbf{x}, \mathbf{u})$ is drawn from distribution ρ or generated through direct interaction with the environment, and $\mathbf{x}'$ is drawn from model $p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}))$ or through direct interaction with the environment.

In general, solving the least squares problem does not guarantee the approximation converges to the policy value. For one, there is no guarantee the chosen approximation architecture is capable of accurately representing the policy value. Secondly, the sample estimates given in Equation (2.70) are *biased* estimates of policy values V^π or Q^π since each sample incorporates the policy value approximation $\hat{V}$ or $\hat{Q}$.

3. HYBRID VEHICLE MODEL

3.1 Hybrid Vehicle Background

The requirements of a given vehicle application are specified by speed and propulsion limits, and the vehicle transmission matches these requirements to engine capabilities. Figure 3.1 shows typical engine torque and power curves on the left, and a typical vehicle propulsion requirement curve on the right. The engine is typically sized to deliver a specified minimum torque and power over a range of speeds. In any given application, there is typically some maximum propulsion force required as indicated in the right figure of Fig. 3.1. The corner power location is set by the maximum power available from the engine. The maximum available propulsion force decreases along a curve of constant power past the corner power location. Hydraulic

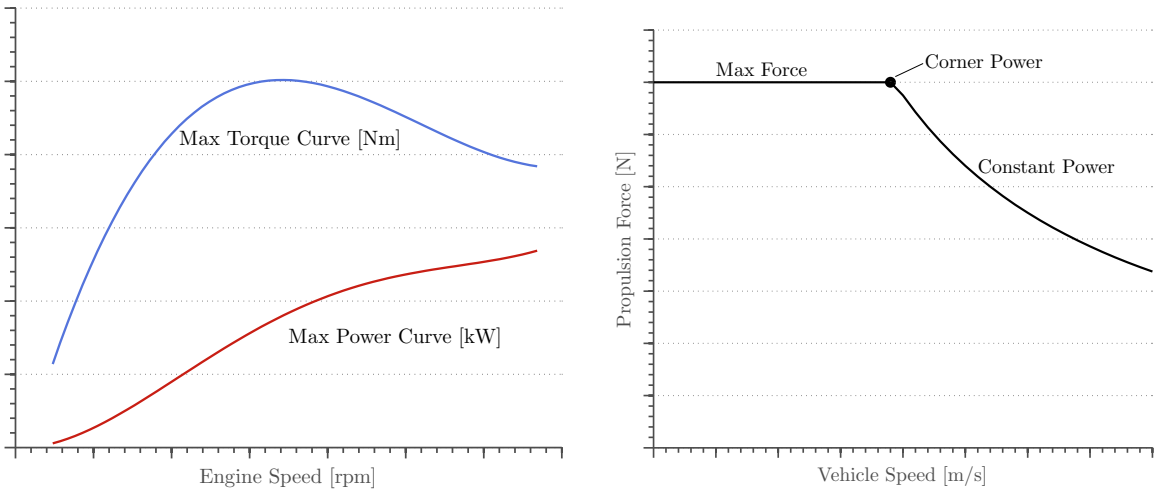

Fig. 3.1. Engine capabilities and vehicle propulsion requirements.

hybrid vehicles (HHV) consist of a primary power path originating from an internal combustion engine and a secondary power path originating from a hydraulic accumulator. The arrangement of the primary and secondary power paths can be divided into

three architectures: parallel in which the secondary power path is in parallel with the primary, series in which the secondary power path is in series with the primary, and series-parallel which combines features of the series and parallel arrangements. One of the defining features of all HHV architectures is regenerative braking, a process by which vehicle kinetic energy is transfered to the hydraulic accumulator to be released during a subsequent propulsion event.

3.1.1 Accumulator Energy Storage

Energy storage in a hydraulic hybrid is accomplished through a hydraulic accumulator, typically of the bladder-type as shown in Fig. 3.2. The top portion contains

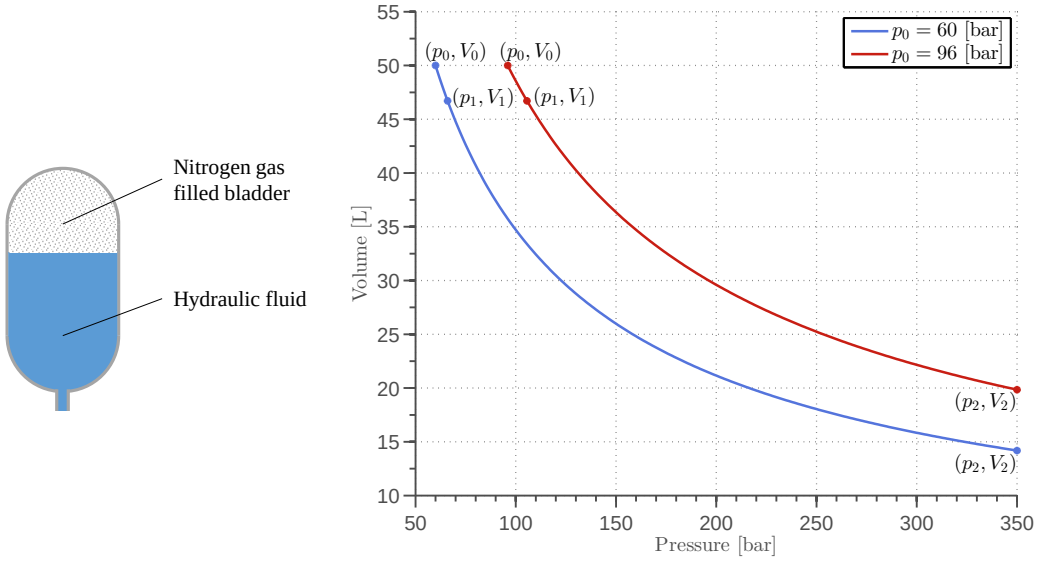

Fig. 3.2. Bladder type hydraulic accumulator. Left: Schematic, Right: $p - V$ curves for two precharge pressures, for a fixed $V_0 = 50 \times 10^{-3}\ m^3$.

a bladder filled with Nitrogen gas. Hydraulic fluid can enter and exit the accumulator through a port on the lower side of the accumulator. The precharge pressure, p_0, is the gas pressure when no hydraulic fluid is present in the accumulator. The minimum operating pressure, p_1, is typically set to 110% of p_0 and, as its name suggests, is the lowest allowable operating pressure ensuring safe accumulator operation. The

maximum allowable operating pressure is shown in the $p-V$ diagram of Fig. 3.2 as p_2. Assuming the Nitrogen mass transfer between the tank and accumulator occurs under isentropic conditions (i.e. without heat transfer, corresponding to mass transfer with a perfectly insulated accumulator), the thermodynamic relationships within the Nitrogen gas bladder are

$$pV = mRT \text{ (Ideal gas law)}, \quad \frac{p}{p_0} = \left(\frac{T}{T_0}\right)^{\gamma/(\gamma-1)}$$

where $\gamma = 1.4$ is the specific heat ratio of Nitrogen. Since the Nitrogen mass is constant throughout accumulator operation, these two equations can be combined to yield the pressure-volume relationship for the Nitrogen gas,

$$pV^{\gamma} = p_0 V_0^{\gamma} = c \tag{3.1}$$

The energy stored in the accumulator between points 1 and 2 on the $p-V$ diagram of Fig. 3.2 is given by,

$$E_{12} = \int_1^2 pdV = -\frac{c^{1/\gamma}}{\gamma}\int_1^2 p^{-1/\gamma}dp = \frac{p_0^{1/\gamma}V_0}{(1-\gamma)}\left(p_2^{(1-1/\gamma)} - p_1^{(1-1/\gamma)}\right)\left(\tag{3.2}$$

where we have used the fact $0 = d\left(pV^{\gamma}\right) = V^{\gamma}dp + \gamma pV^{\gamma-1}dV$ as evident from Equation (3.1). Accumulator energy storage curves are shown in Fig. 3.3. Here, it is assumed that energy storage is in reference to point 1 on the $p-V$ curve (i.e. energy storage is zero at (p_1, V_1)), since this is the lowest pressure allowable during accumulator operation. An interesting observation is the curves for $p_0 = 60$ bar and $p_0 = 96$ bar terminate at nearly the same energy storage level, yet the curve associated with $p_0 = 60$ bar accomplishes a given energy level at a lower associated pressure for a larger range of operation. The accumulator can be designed by considering the energy storage required for a given application. For a given vehicle speed v_{veh}, the kinetic energy $E_K(v_{veh}) = \frac{1}{2}m_{veh}v_{veh}^2$ represents the maximum available energy that can be transfered to the accumulator. Precharge pressure p_0 and accumulator size V_0

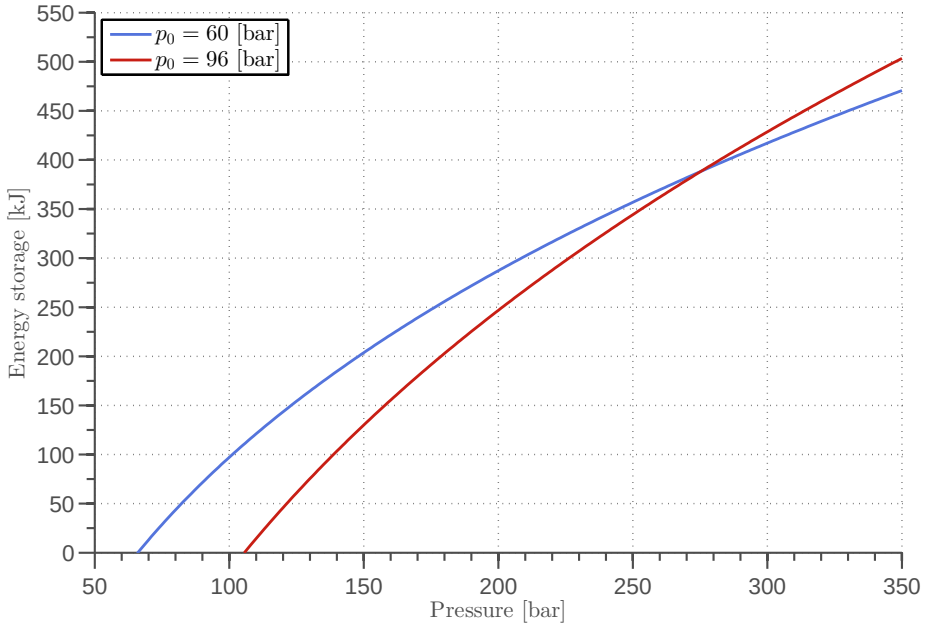

Fig. 3.3. Hydraulic accumulator energy storage curves for $V_0 = 50 \times 10^{-3}\ m^3$, $p_1 = 1.1 \times p_0$.

can therefore be chosen by setting Equation (3.2) equal to the desired value of E_K, using the constraint $p_1 = 1.1 \times p_0$.

3.1.2 Architectures

This subsection reviews basic operation and design considerations for the parallel, series, and series-parallel hydraulic hybrids. A comprehensive comparison between various series-parallel configurations is discussed in [49]. An interesting use of a high-speed flywheel as the secondary energy source in a series-parallel configuration is found in [50]. A novel concept known as the blended hybrid, whereby the hydraulic accumulator is passively disconnected when system differential pressure rises below accumulator pressure, is discussed in [51–53]. The intention of the blended hybrid is to allow the transmission to operate at lower pressures than accumulator pre-charge, lowering losses in the hydraulic circuit. The blended hybrid architecture is adapted to the series and series-parallel configurations in [54].

3.1.2.1 Parallel HHV

The schematic of the parallel HHV is shown in Fig. 3.4. Power from the engine can be supplemented with hydraulic power from a pump/motor unit, which can transfer power to/from the hydraulic accumulator. Essentially, the parallel HHV is a conven-

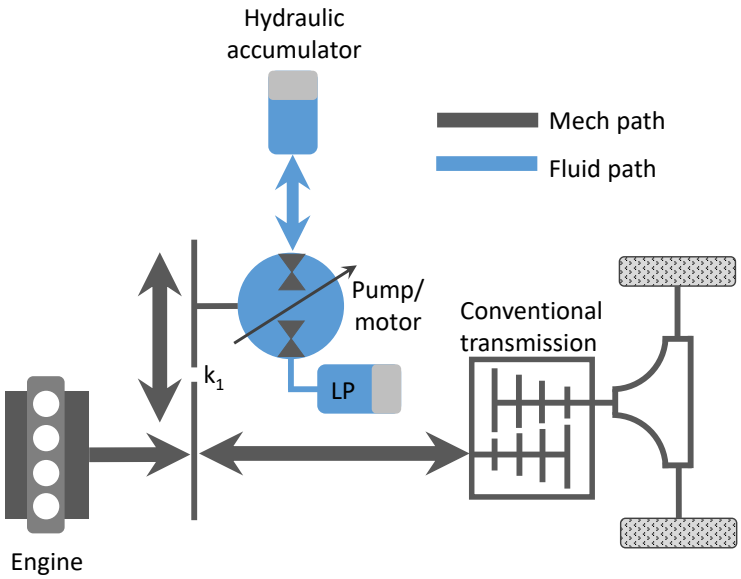

Fig. 3.4. Parallel hybrid hydraulic vehicle.

tional power train augmented with a secondary power path. As such, the engine can be downsized in the sense that maximum power can be achieved by supplementing available engine power with hydraulic power from the pump/motor unit.

3.1.2.2 Series HHV

The schematic of the series HHV with a two-stage output gearbox is shown in Fig. 3.5. Power from the engine is transmitted to a hydraulic pump which converts the mechanical power into pressurized fluid flow. A hydraulic pump/motor unit converts the pressurized fluid flow into mechanical power as the source of vehicle propulsion. During regenerative braking, the pump/motor units operates as a pump charging the hydraulic accumulator by transferring fluid from low pressure to the accumulator. At

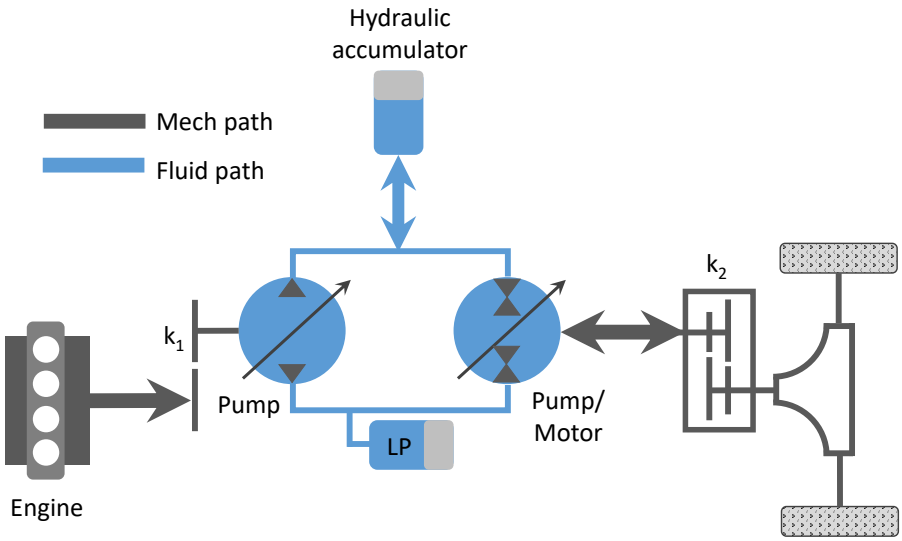

Fig. 3.5. Series hybrid hydraulic vehicle.

low speeds the effective ratio between the motor and wheels, including the drive axle, is $k_2 = k_{2,lo}$, while at higher speeds this ratio changes to $k_2 = k_{2,hi}$. Positive net flow between the pump and motor is transferred into the hydraulic accumulator, while negative net flow indicates fluid is being transferred from the hydraulic accumulator. The pump/motor unit is designed such that, at maximum displacement V_m^{max}, maximum propulsive force can be achieved in low gear at some nominal system differential pressure p°

$$F_p^{max} = \frac{p^\circ V_m^{max}}{2\pi} \frac{k_{2,lo}}{r_{tire}} \tag{3.3}$$

where system differential pressure is the difference between the hydraulic accumulator and low pressure, $p = p_{ha} - p_{lp}$. The pump unit can be designed to deliver required flow rate at high speed through the following flow balance

$$n_{eng}^{max} k_1 V_p^{max} = \frac{v_{veh}^{max}}{r_{tire}} k_{2,hi} V_m^{max} \tag{3.4}$$

Alternatively, the pump unit can be designed such that the engine can be loaded to maximum torque at some nominal system differential pressure p^* through the following torque balance

$$p^* V_p^{max} \frac{k_1}{2\pi} = T_{eng}^{max} \tag{3.5}$$

3.1.2.3 Series-Parallel HHV

The schematic of the series-parallel HHV is shown in Fig. 3.6. A defining feature of

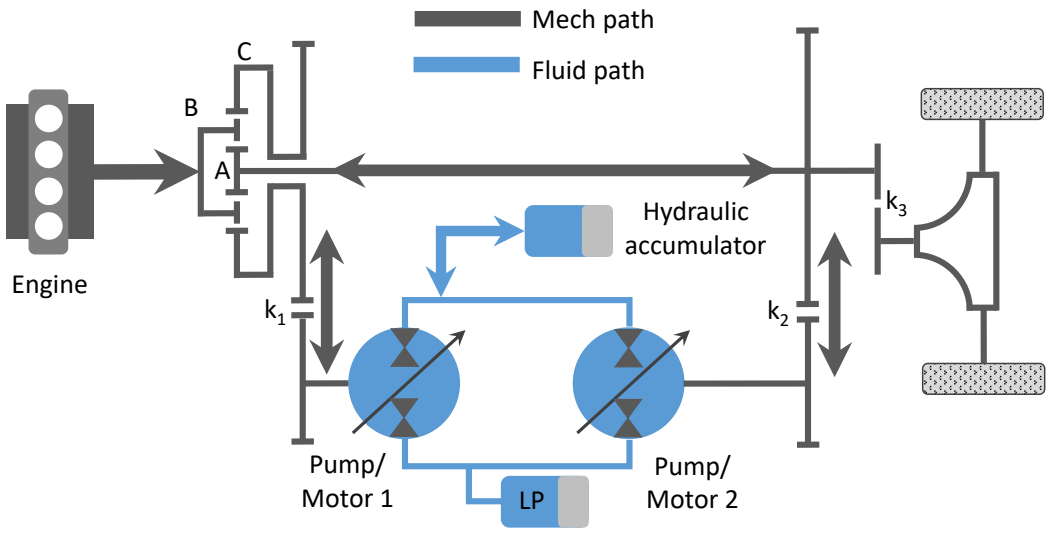

Fig. 3.6. Series-parallel hybrid hydraulic vehicle.

the series-parallel HHV is the planetary gear connected to the engine. The planetary gear allows for power splitting between two separate paths. The engine connects to the planetary gear via carrier gear B, while the hydraulic pump/motor unit 1 connects via ring gear C, and the output shaft and hydraulic pump/motor unit 2 are connected via sun gear A. The planetary gear behavior is defined through the following speed, torque and power relationships between members A, B, C

$$n_A - (1 - k_0)n_B - k_0 n_C = 0 \tag{3.6a}$$

$$T_A = \frac{1}{k_0 - 1} T_B \tag{3.6b}$$

$$T_C = \frac{-k_0}{k_0 - 1} T_B \tag{3.6c}$$

$$\text{mech power path:} \quad P_A = \frac{1}{k_0 - 1} \frac{n_A}{n_B} P_B \tag{3.6d}$$

$$\text{hyd power path:} \quad P_C = \left(1 + \frac{1}{k_0 - 1} \frac{n_A}{n_B}\right) P_B \tag{3.6e}$$

where planetary gear ratio k_0 is determined by the geometry of the planetary gear. The last two equations indicate that the power split between the mechanical path and the hydraulic path is determined by the ratio of vehicle speed to engine speed as indicated by the term $\frac{n_A}{n_B}$. From the last equation in Equation (3.6), the power through the hydraulic path becomes zero when the ratio of sun gear speed to carrier gear speed becomes $\frac{n_A}{n_B} = 1 - k_0$. This condition produces the most efficient point of power transfer within the series-parallel HHV known as the full-mechanical speed point given by

$$v_{mech} = (1 - k_0) n_{eng} \frac{r_{tire}}{k_3} \tag{3.7}$$

Efficiency vs. vehicle speed of the series-parallel HHV compared to the series HHV, assuming a fixed hydraulic path efficiency of 85%, is shown in Fig. 3.7. Efficiency of the series-parallel HHV declines past speed v_{mech}. As such, the planetary gear ratio k_0 and drive gear ratio k_3 can be designed according to Equation (3.7) so v_{mech} occurs at some desired engine speed. Gear ratio k_2 can be designed such that the speed of unit II is limited to some maximum value considering the maximum vehicle speed

$$n_{\mathrm{II}}^{max} = \frac{v_{veh}^{max} k_3 k_2}{r_{tire}} \tag{3.8}$$

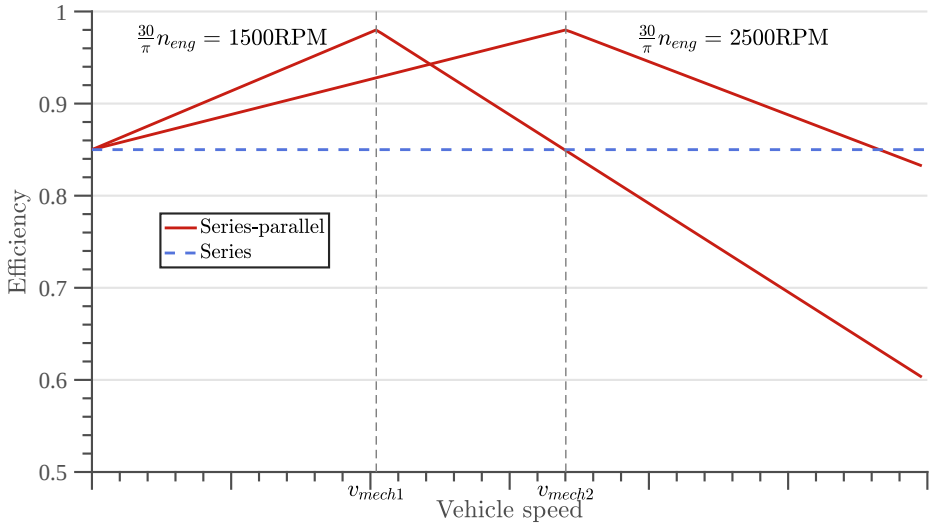

Fig. 3.7. Series-parallel vs. series HHV efficiency.

Gear ratio k_1 can be designed considering maximum engine speed at zero vehicle speed, at which point unit I reaches maximum speed

$$n_{\mathrm{I}}^{max} = \frac{n_A - (1-k_0)n_{eng}^{max}}{k_0} k_1 \Big|_{n_A=0} \tag{3.9}$$

Hydraulic unit II can be designed so that maximum propulsion force is achieved at some nominal system differential pressure, p°

$$F_p^{max} = p^\circ \frac{V_{\mathrm{II}}^{max}}{2\pi} \frac{k_2 k_3}{r_{tire}} \tag{3.10}$$

Finally, hydraulic unit I can be designed so the engine can be loaded to its maximum torque capability at some nominal system differential pressure p^*

$$T_{eng}^{max} = p^* \frac{V_{\mathrm{I}}^{max}}{2\pi} \frac{1-k_0}{k_0} k_1 \tag{3.11}$$

3.2 Series HHV Dynamics

Due to its simple design and superior engine management capabilities, this work focuses on designing an optimal control strategy for the series hybrid shown in Fig. 3.5. The vehicle velocity dynamic is given by

$$\dot{v}_{veh}(t) = \frac{1}{m_{veh}}\Big[\Big(F_p(t) - \tfrac{1}{2}C_d\rho_{air}v_{veh}(t)^2 - m_{veh}g\Big(C_r cos(\phi(t)) + sin(\phi(t))\Big)\Big]\Big(\tag{3.12}$$

where m_{veh} is vehicle mass, ρ_{air} is air density and g is the gravitational constant. The terms C_d and C_r are drag and rolling resistance coefficients associated with the vehicle, where ϕ represents the road grade. The propulsive force F_p is dependent on the differential system pressure[1], p, motor displacement volume V_m, motor torque losses $M_{s,m}$, and is limited by the maximum displacement volume of the motor, V_m^{max}

$$F_p = \left(\frac{V_m}{2\pi}p - M_{s,m}\right)\left(\frac{k_2}{r_{tire}} \tag{3.13}$$

$$\leq \left(\frac{V_m^{max}}{2\pi}p - M_{s,m}\right)\left(\frac{k_2}{r_{tire}} \tag{3.14}$$

$$= F_p^{max}(p) \tag{3.15}$$

The displacement volume of the hydraulic motor, V_m, is determined based on the applied force commanded by the driver, F_p^{cmd}

$$V_m = \frac{2\pi}{p} \quad \frac{F_p^{cmd}r_{tire}}{k_2} + \hat{M}_{s,m}\right)\Bigg(\tag{3.16}$$

The term $\hat{M}_{s,m}$ is a polynomial approximation of the hydraulic motor torque loss term $M_{s,m}$. In general, hydraulic system losses tend to increase as the system differential pressure increases. As such, p must be managed carefully as to satisfy driver propulsion demands ensuring $F_p^{cmd} \leq F_p^{max}(p)$ while simultaneously minimizing the

[1]More generally, for safety, traditional friction brakes can be added so that the propulsion force becomes $F_p = \left(\frac{V_m}{2\pi}p - M_{s,m}\right)\frac{k_2}{r_{tire}} - F_{brake}$. In this work, the friction brake force term F_{brake} is neglected as its role in the drive cycles investigated was negligible.

losses experienced by the hydraulic system. The dynamics of engine speed n_{eng} and intake manifold pressure p_{im} are given by [55, 56]

$$\dot{n}_{eng}(t) = \frac{1}{I_{eng}}\left[T_{cyl}(t) - \frac{k_1}{2\pi}V_p(t)p(t) - k_1 M_{s,p}(t)\right] \tag{3.17}$$

$$\dot{p}_{im}(t) = \frac{RT_{im}}{V_{im}}\left[W_{thr}(t) - \frac{\eta_v V_d}{4\pi RT_{im}}n_{eng}(t)p_{im}(t)\right] \tag{3.18}$$

Here, W_{thr} is throttle mass flow rate, R is the ideal gas constant for air, T_{im} is the intake manifold temperature, η_v is volumetric efficiency of the engine, V_d and V_{im} are the volumes of the engine displacement and intake manifold. The torque produced by the engine cylinders, T_{cyl}, is determined from the engine thermal efficiency, η_t, the lower heating value of the fuel Q_{lhv}, the air-fuel ratio in the cylinders, AFR, and the inducted air mass in the cylinders m_{cyl} [56]

$$m_{cyl} = \frac{\eta_v V_d}{RT_{im}}p_{im} \tag{3.19}$$

$$T_{cyl} = \frac{\eta_t Q_{lhv}}{4\pi AFR}m_{cyl} = \frac{\eta_t \eta_v Q_{lhv} V_d}{4\pi RT_{im}AFR}p_{im} \tag{3.20}$$

The maximum capability of the engine in this work is 125 kW, as the engine speed is limited to 5000 RPM. The maximum torque curve as a function of engine speed and fuel consumption rate, b_f, as a function of engine speed and torque are described by Fig. 3.8 The dynamic of the hydraulic differential system pressure p is

$$\dot{p}(t) = \frac{1}{C_h(p)}\Big[\frac{k_1}{2\pi}V_p(t)n_{eng}(t) - \frac{k_2}{2\pi r_{tire}}V_m(t)v_{veh}(t) - Q_{s,p}(t) - Q_{s,m}(t)\Big] \tag{3.21}$$

where $Q_{s,p}, Q_{s,m}$ are the flow losses of the pump and motor, k_1, k_2 are gear ratios, and V_p is the displacement volume of the hydraulic pump. It is assumed here that low pressure is nearly constant. The capacitance of the hydraulic system [57] is

$$C_h(p) = \frac{V_{ha}p_{ha}^{1/\gamma_{gas}}}{\gamma_{gas}(p+p_{lp})^{1+1/\gamma_{gas}}} + \frac{V_L}{K_L} \tag{3.22}$$

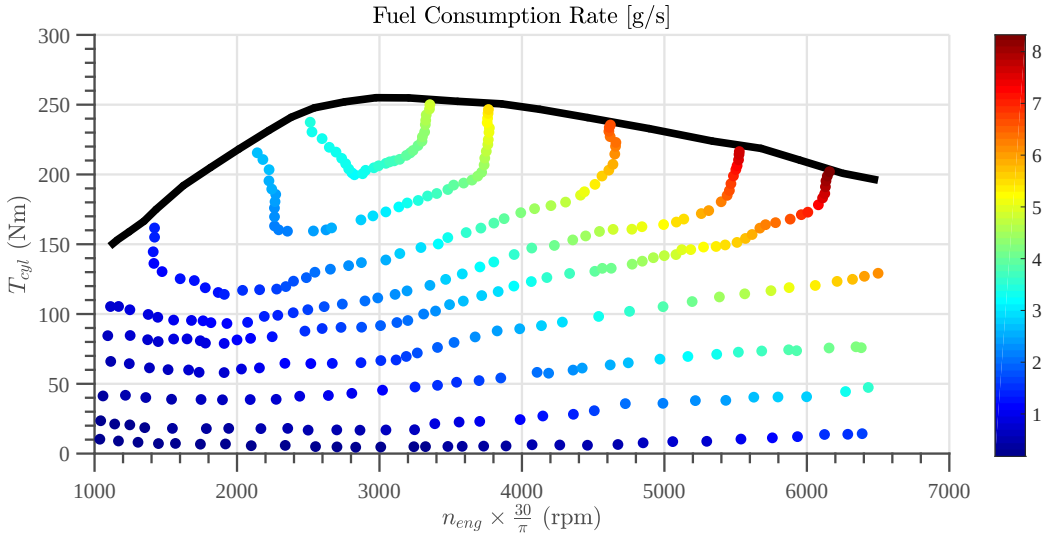

Fig. 3.8. Engine fuel consumption rate, $b_f(n_{eng}, T_{cyl})$, and maximum torque curve, $T_{cyl}^{max}(n_{eng})$.

where V_{ha}, p_{ha} are the pre-charge volume and pressure of the hydraulic accumulator, γ_{gas} is the specific heat ratio of the pressurized gas within the accumulator, p_{lp} is the pressure of the low-pressure system and V_L, K_L are the volume and bulk modulus of the hydraulic lines. Example hydraulic losses and their second order polynomial approximations are shown in Fig. 3.9

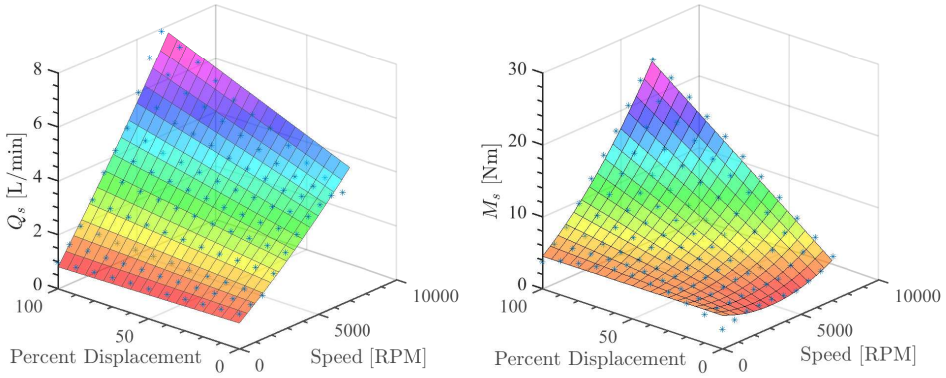

Fig. 3.9. $Q_{s,p}$ and $M_{s,m}$, $p = 250$ bar for 60 cc/rev max displacement volume hydraulic unit. Data points in blue markers, second order polynomial fits $\hat{Q}_{s,p}$ and $\hat{M}_{s,m}$ shown as shaded surface.

4. STATISTICAL MODEL OF DRIVER BEHAVIOR

4.1 Driver Behavior as a Markov Process

Driver behavior is characterized in terms of an acceleration demand, w, which can be inferred from the driver's propulsive force command F_p^{cmd} through[1]

$$w = \frac{1}{m_{veh}}\Big[F_p^{cmd} - \tfrac{1}{2}C_d\rho_{air}v_{veh}^2 - m_{veh}g\left(C_r cos(\phi) + sin(\phi)\right)\Big] \tag{4.1}$$

where ϕ is the road grade, assumed available from measurement or estimation. If the driver acceleration demand w can be forecast along a horizon to some statistical accuracy, then a control strategy which incorporates an underlying statistical model can be designed. It is well known that driver behavior can be modeled effectively as a Markov process [12, 15, 33], a type of stochastic process which adheres to the *Markov property*. The Markov property roughly states that future behavior of the process is influenced only by the present state, unaffected by the sequence of events that lead to the present state. More specifically, the stochastic process $\{w_0, w_1, w_2, \dots\}$ is Markovian if

$$\Pr[w_{n+1} = w^j | \mathcal{F}_n] = \Pr[w_{n+1} = w^j | w_n = w^i] \tag{4.2}$$

where each $w_n \in W$ is a random variable and w^i and w^j are realizations of the random variables w_n and w_{n+1}, respectively. Equation (4.2) states that the probability of the next transition given all prior information up to time n is the same as the probability

[1]It is assumed F_p^{cmd} can be inferred, for example, from driver foot pedal position. During simulation and experiments in this work F_p^{cmd} is the output of a PI feedback process used to track a vehicle speed reference.

of the next transition given the information only of the previous state[2]. If, in addition to satisfying the Markov property, the process is also time invariant then

$$\Pr[w_{n+1} = w^j | w_n = w^i] = \Pr[w_{m+1} = w^j | w_m = w^i] \tag{4.3}$$

for any $n, m \geq 0$. The benefit of working with assumptions Equation (4.2) and Equation (4.3) is that all subsequent computations in the energy management strategies developed in the next chapter are greatly simplified.

In this work the driver acceleration demand w is modeled as a discrete state discrete time Markov process. Each transition is described by the probability distribution matrix (P_{ij}) whose elements are defined as

$$P_{ij} \triangleq \Pr[w_{n+1} = w^j | w_n = w^i] \tag{4.4}$$

The multi-step probability $P_{ij}^{(n)}$ describes the probability of a demand at time n given the value of the demand at time 0

$$P_{ij}^{(n)} \triangleq \Pr[w_n = w^j | w_0 = w^i] \tag{4.5}$$

and, as the notation suggests, is computed by raising matrix (P_{ij}) to the exponent n and selecting the ij^{th} element [58]. The multi-step distribution will be used extensively in the development of a stochastic strategy described in Section 5.3.2.

Driver acceleration demand w is quantized evenly into 19 levels, w^i, $i = 1, 2, \ldots, 19$, between -3 to 3 m/s^2. Any acceleration demand lower than -3 m/s^2 is associated with w^1 while any acceleration demand greater than 3 m/s^2 is associated with w^{19}.

[2]A discrete time deterministic dynamic system described by dynamics $F(x_n, u_n)$ and some initial condition can be viewed as obeying condition Equation (4.2), where $\Pr[x_{n+1} = F(x, u) | x_n = x] = 1$, $\Pr[x_{n+1} \neq F(x, u) | x_n = x] = 0$

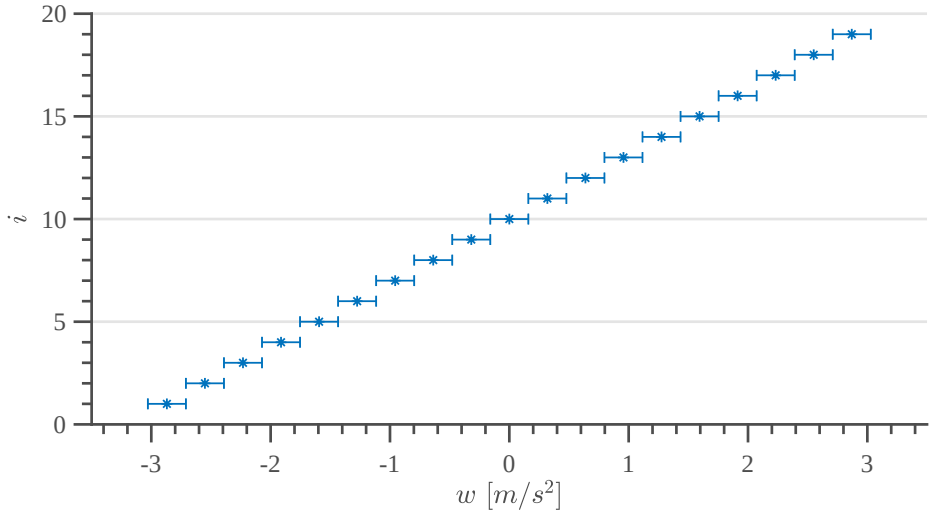

Fig. 4.1. Quantization of driver acceleration demand.

4.2 Learning Driver Behavior

In this work three primary drive cycles are considered, shown in Fig. 4.2. The first drive cycle is the EPA's Urban Dynamometer Driving Schedule (UDDS), a representative urban drive cycle with frequent stops having an average speed of 31.5 km/h and a total run time of approximately 23 minutes. The second drive cycle is the EPA's aggressive urban drive cycle (US06). Having an average speed of 78 km/h with a short runtime of 10 minutes, the US06 cycle was developed by the EPA in response to criticism of the UDDS cycle's inability to represent aggressive, high speed and/or high acceleration driving with rapid speed fluctuations. The third drive cycle, referred to as the GPS cycle, is moderate traffic city driving data from West Lafayette, IN and includes altitude data collected by an on-board GPS device. The GPS cycle has a total runtime of approximately 15 minutes.

A sequence of driver acceleration demands $\{w_0, w_1, w_2, \dots\}$ is created from Equation (4.1) according to $w_n = w(n\Delta t)$, with sampling rate $\Delta t = 1$ second. Estimates of

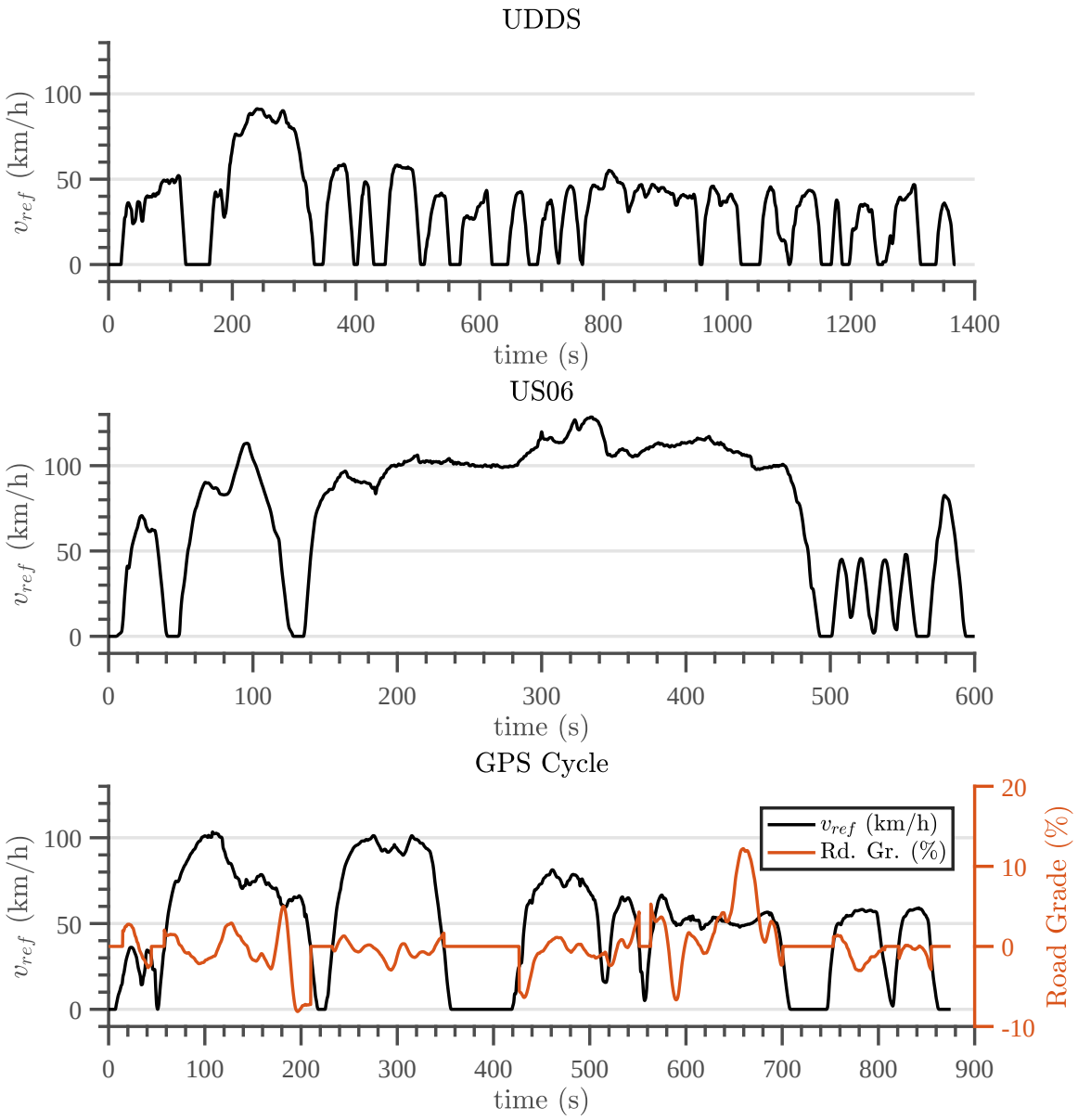

Fig. 4.2. Drive cycles investigated.

the Markov single-step transition probabilities at time step n, $n = 1, 2, 3, \ldots$, denoted $\hat{P}_{ij}^{[n]}$, are determined through a first order filtering process according to [33]

$$\hat{P}_{ij}^{[n]} = \begin{cases} \alpha \mathbb{1}_{ij}^{[n]} + (1-\alpha)\hat{P}_{ij}^{[n-1]} & \text{if } w_n = w^i \\ \hat{P}_{ij}^{[n-1]} & \text{if } w_n \neq w^i \end{cases} \tag{4.6}$$

where $\hat{P}_{ij}^{[0]}$ is an arbitrary initialization and the indicator function $\mathbb{1}_{ij}^{[n]}$ is defined by

$$\mathbb{1}_{ij}^{[n]} = \begin{cases} 1 & \text{if } w_{n+1} = j, w_n = w^i \\ 0 & \text{if } w_{n+1} \neq j, w_n = w^i \end{cases} \tag{4.7}$$

The updates described by Equation (4.6) and Equation (4.7) are performed for all $w^i, w^j \in W$ at each time step n. The parameter $\alpha \in [0, 1]$ is the learning rate that determines the exponential rate at which the dependence on past information is decreased. This estimation process produces an unbiased estimate as now shown. Let $(n_k,\ k = 1, 2, 3, \dots)$ be an indexed sequence of time steps in which the chain is in state $w^i \in W$, and assume that each state w^i is visited an infinite number of times (i.e., $k \to \infty$)

$$\begin{aligned}
\mathbb{E}[\hat{P}_{ij}^{[n_k]}] &= \alpha \mathbb{E}[\mathbb{1}_{ij}^{[n_k]}] + (1-\alpha)\mathbb{E}[\hat{P}_{ij}^{[n_{k-1}]}] \\
&= \alpha \mathbb{E}[\mathbb{1}_{ij}^{[n_k]}] + (1-\alpha)\Big[\alpha \mathbb{E}[\mathbb{1}_{ij}^{[n_{k-1}]}] \\
&\qquad + (1-\alpha)\mathbb{E}[\hat{P}_{ij}^{[n_{k-2}]}]\Big] \\
&= \alpha \mathbb{E}[\mathbb{1}_{ij}] \underbrace{\sum_{m=0}^{k-2} (1-\alpha)^m}_{\to \frac{1}{\alpha}} + \underbrace{(1-\alpha)^{k-1}}_{\to 0} \mathbb{E}[\hat{P}_{ij}^{[n_1]}] \\
&\to \mathbb{E}[\mathbb{1}_{ij}] = \Pr[w_{n+1} = w^j | w_n = w^i] = P_{ij} \text{ as } k \to \infty
\end{aligned}$$

In the third equality above, it is noted that $\mathbb{E}[\mathbb{1}_{ij}^{[n_k]}] = E[\mathbb{1}_{ij}]$ for every n_k since each $\mathbb{1}_{ij}^{[n_k]}$ is a iid copy of the random variable $\mathbb{1}_{ij}$ for each fixed $w^i \in W$. Since $\mathbb{E}[\hat{P}_{ij}] \to P_{ij}$, the estimator is *unbiased*. By a slight abuse of notation, P_{ij} and estimate $\hat{P}_{ij}$ are used interchangeably throughout the remainder of this work.

The transition probability matrix (P_{ij}) is learned according to Equation (4.6) and Equation (4.7) for each drive cycle described in Fig. 4.2. The learning rate is chosen as $\alpha = 0.025$ so that only 20% of the initial estimate $\hat{P}_{ij}^{[0]}$ is retained in memory after 60 transitions from i to j (the influence of $\hat{P}_{ij}^{[0]}$ on $\hat{P}_{ij}^{[n]}$ is $\hat{P}_{ij}^{[0]}(1-\alpha)^n$). The matrices (P_{ij}) shown in Fig. 4.3 are color coded so that dark red indicates a transition

probability that is greater than 0.5, while dark blue indicates a value near 0. All three matrices show a somewhat similar pattern along the diagonal, in that the driver tends to demand an acceleration level at the next time step that is near his or her current demand. However, the degree to which the driver chooses a slightly higher or lower demand at the next time step varies greatly with the drive cycle. In the UDDS cycle the driver has a strong preference to operate along the diagonal, while in the US06 cycle the driver is much more likely to choose an off-diagonal transition. During the GPS cycle, driver behavior appears to be somewhat of a mixture of behavior from UDDS and US06 cycles.

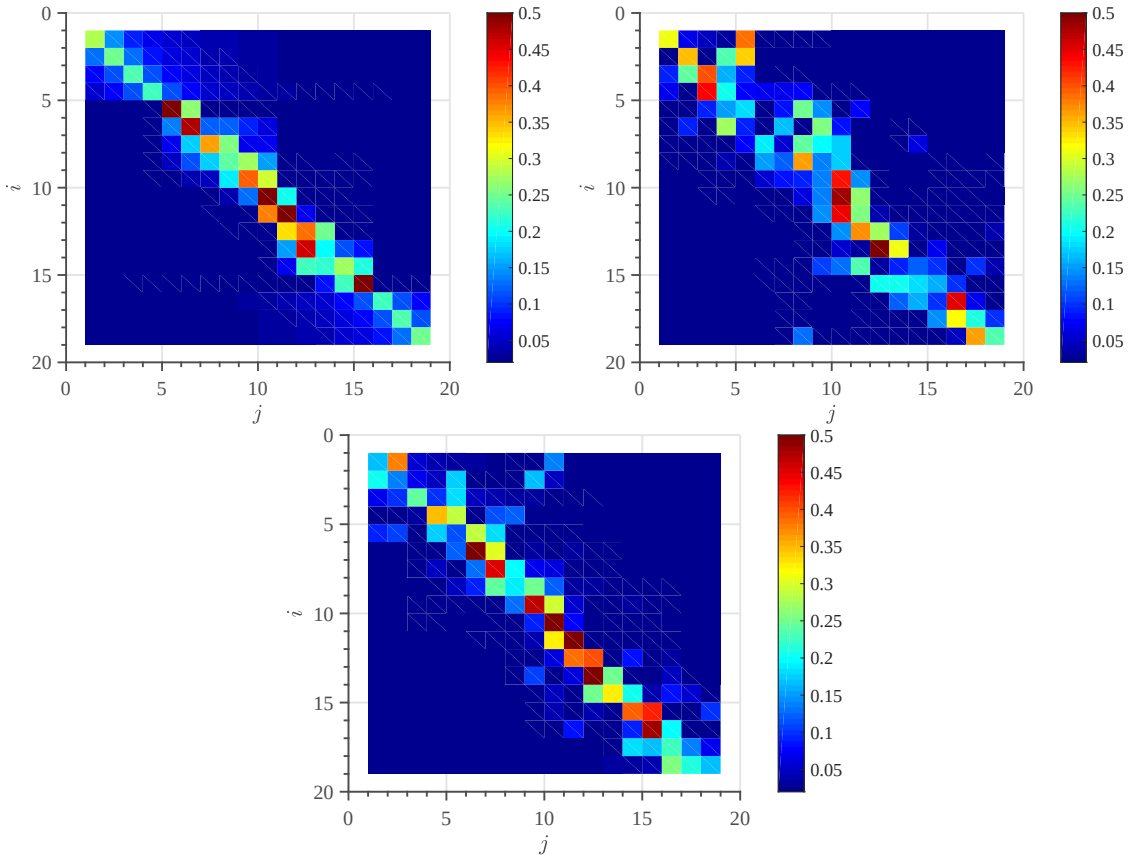

Fig. 4.3. (P_{ij}) for UDDS drive cycle (upper left), US06 drive cycle (upper right), and GPS drive cycle (lower).

The transition probabilities shown in Fig. 4.3 give insight into the singe-step behavior of the driver. However, for the purposes of planning along a horizon it is de-

sirable to understand driver behavior several seconds into the horizon. The multi-step distribution Equation (4.5), $P_{ij}^{(n)} = \Pr[w_n = w^j | w_0 = w^i]$, provides this information. The propagation of the multi-step distribution for each cycle is shown in Fig 4.4. Two initial demands are shown, the left column corresponding to the driver initially demanding a moderately negative acceleration and the right column corresponding to the driver initially demanding a moderately positive acceleration. The effect of small differences in the (P_{ij}) matrices shown in Fig. 4.3 are immediately apparent. For one, the single-step distribution (corresponding to $n = 1$) is very different for each cycle. Secondly, the paths along which the various transition probabilities grow and decay differs from one drive cycle to another.

From the multi-step distribution, the expected value and variance of the driver acceleration demand sequence, for $n = 0, 1, \ldots, N$, can be computed according to

$$\mathbb{E}[w_n | w_0 = w^i] = \sum_{j \in W} \left(P_{ij}^{(n)} w^j \right. \tag{4.8}$$

$$\mathrm{Var}[w_n | w_0 = w^i] = \sum_{j \in W} P_{ij}^{(n)} (w^j)^2 - \quad \sum_{j \in W} \left(P_{ij}^{(n)} w^j \right)^2 \tag{4.9}$$

The expected path of driver acceleration demand given by Equation (4.8) is compared to the sample average for each drive cycle in Fig. 4.5 for three initial demands. Also shown are the standard deviation of driver acceleration demand for each cycle, calculated as $\sigma = \sqrt{\mathrm{Var}[w_n | w_0 = w^i]}$ from Equation (4.9). These quantities provide indication as to the degree to which driver behavior can be anticipated along the horizon, and will be used further in Section 5.3.

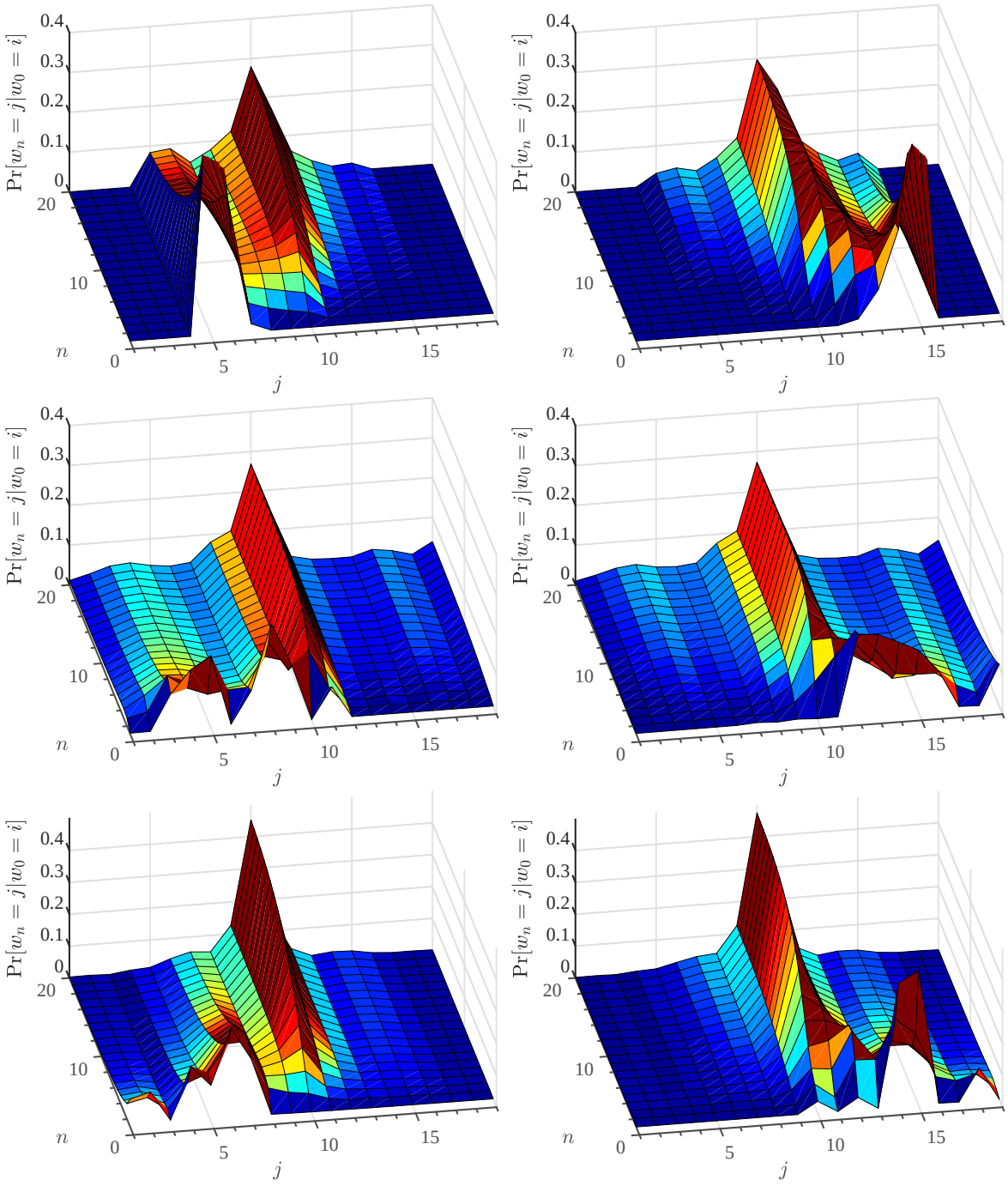

Fig. 4.4. Propagation of $\Pr[w_n = w^j | w_0 = w^i]$ for $i = 5$ (left column) and $i = 15$ (right column). Driver statistics from UDDS cycle (top row), US06 cycle (middle row) and GPS cycle (bottom row).

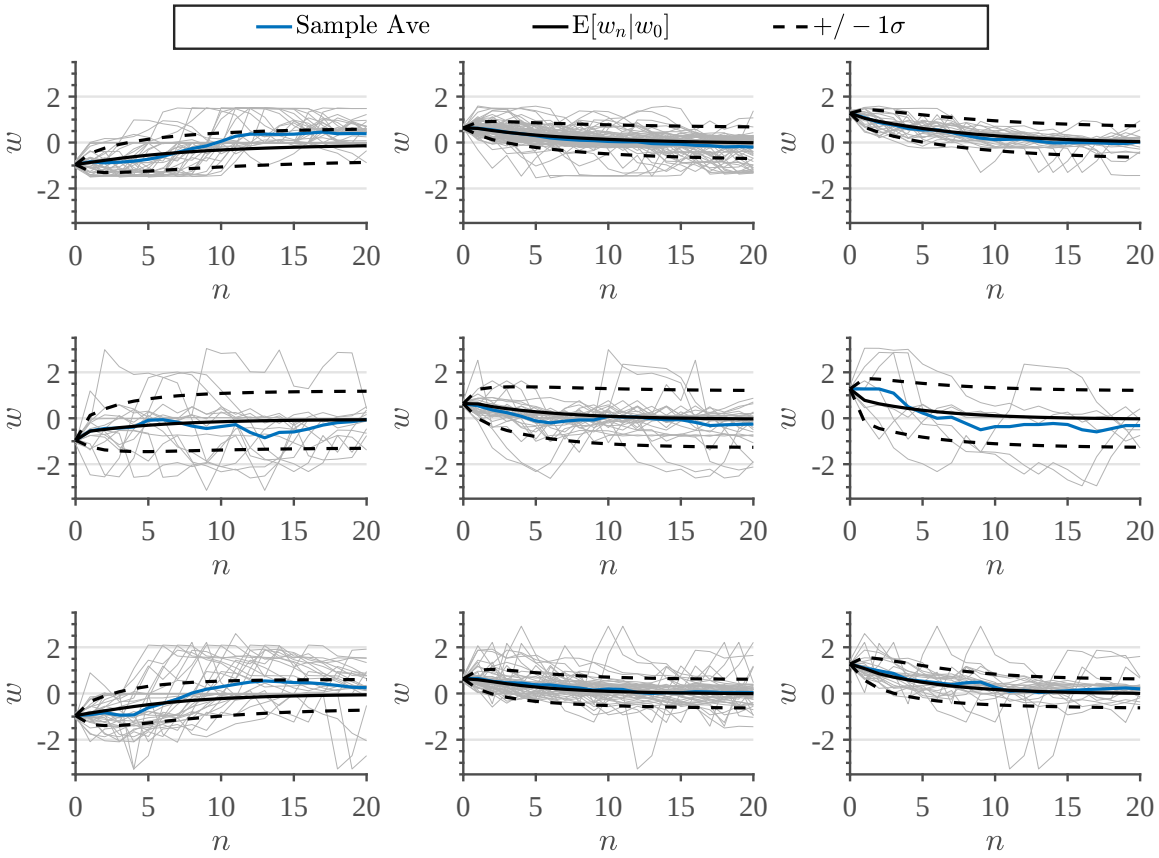

Fig. 4.5. Propagation of $\mathbb{E}[w_n|w_0 = w^i]$. Sample paths shown in light grey. Top row: UDDS cycle, middle row: US06 cycle, bottom row: GPS cycle. Left column: $w_0 = -1\ m/s^2$, middle column: $w_0 = 0.6\ m/s^2$, right column: $w_0 = 1.3\ m/s^2$.

4.3 Long Term Driver Statistics

It was shown in Section 4.2 that the multi-step distribution $P_{ij}^{(n)}$ may be used to generate a reasonable estimation of expected driver behavior along a horizon, given an initial condition corresponding to the driver's immediate demand. The multi-step distribution can also provide valuable information about the driver's longer term statistical behavior. Let the distribution

$$\nu^{ij} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{\infty} \mathbb{1}_{\{w_k = w^j | w_0 = w^i\}} \tag{4.10}$$

denote the long run fraction of time the chain visits state w^j when starting in state w^i. It can be shown, see for instance, [58], that this limit exists for all finite state Markov Chains. Assuming the chain is irreducible[3], then $\nu^{ij} = \nu^j$ for each i so that convergence is independent of the initial state. In this case, ν^j may be interpreted as the fraction of time the driver demands acceleration w^j. Assuming furthermore that the chain is also aperiodic[4], ν^j can be computed directly from the multi-step distribution through

$$\nu^j = \lim_{n\to\infty} P_{ij}^{(n)} \tag{4.11}$$

During numerical experiments it was found that the driver tends to exhibit behavior during low speed driving which differs from behavior during higher speed driving. As a result, two separate models for (P_{ij}) are learned: an aggregate model which is independent of speed and another model specifically for low speed driving below 10 m/s (approximately 23 mph). The distributions of ν^j are shown for each of the three drive cycles in Figs. 4.6 and 4.7. Interestingly, the long term driver behavior distribution shows significant cycle to cycle differences during low speed driving. The aggressive behavior of the driver during the US06 cycle is immediately apparent as more than 54% of low speed driving occurs at high acceleration ($i \geq 16$). In contrast, 43% of low speed driving occurs near coasting ($9 \leq i \leq 11$) during the UDDS cycle.

[3]Roughly speaking, a Markov Chain is said to be *irreducible* if any state of the chain can be reached, eventually, from any initial state. The chain describing driver behavior is clearly irreducible.

[4]Roughly speaking, state i is said to be periodic if i can only be revisited cyclically with period $d > 1, d \in \mathbb{N}$, so that $P_{ii}^{(n)} > 0$ whenever n is a multiple of $d > 1$ and $P_{ii}^{(n)} = 0$ otherwise. Clearly, if a periodic state exists in the chain, convergence of $P^{(n)}$ is not possible since $\lim_{k\to\infty} P^{(kd)} \neq \lim_{k\to\infty} P^{(kd+1)}$. The chain describing driver behavior is not periodic since any state can be revisited immediately at the next timestep, so that each state has period $d = 1$.

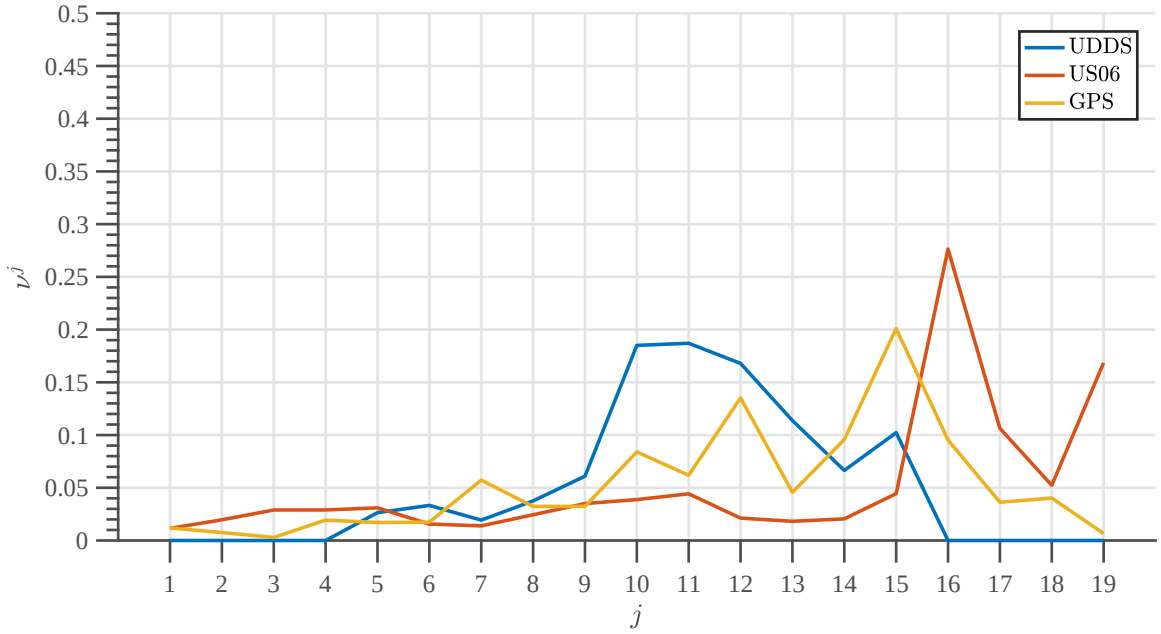

Fig. 4.6. Long term driver behavior ν^i. Statistics at low vehicle speeds $< 10m/s$.

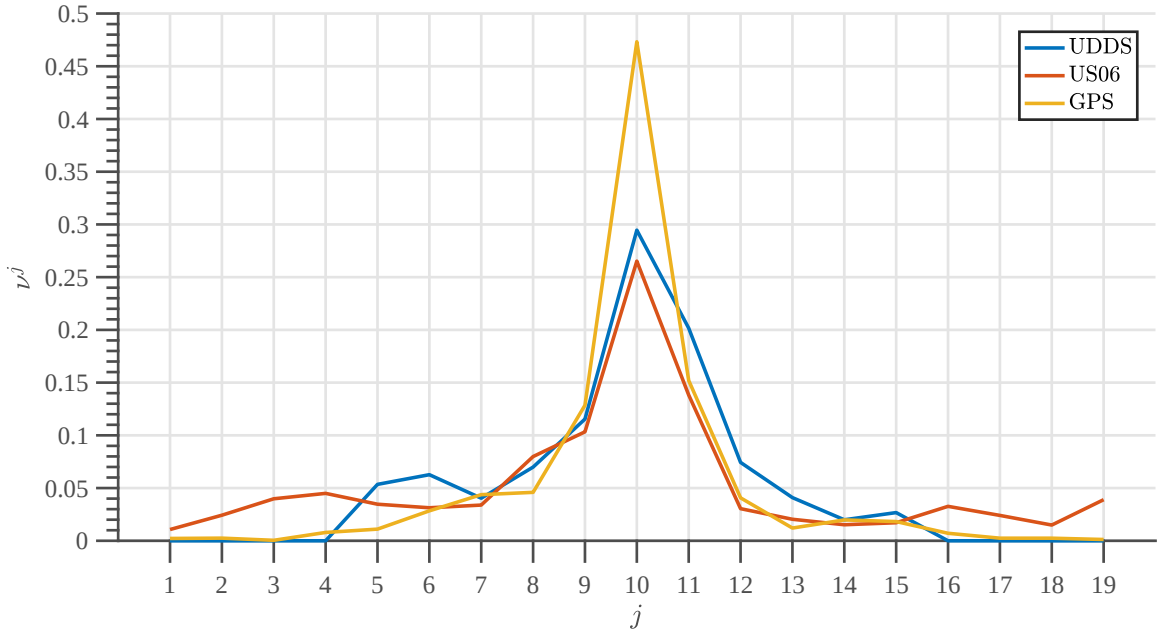

Fig. 4.7. Long term driver behavior ν^i. Aggregate statistics, independent of speed.

5. PREDICTIVE ENERGY MANAGEMENT

Having established models for vehicle and driver dynamics, a model-based predictive energy management strategy can be designed. The goal is to minimize fuel consumption while meeting driver propulsion demands by solving the following finite horizon stochastic optimization problem

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[\sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{u}_n, w_n)\ \mathbf{x}_0, w_0\right] \tag{5.1a}$$

$$\text{subject to}\ \ \mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, w_n) \tag{5.1b}$$

$$\mathbf{x}_n \in \mathbf{X} \tag{5.1c}$$

$$\mathbf{u}_n \in \mathbf{U} \tag{5.1d}$$

Complimentary methods for approximately solving Equation (5.1) are developed. The method developed in Section 5.3.1 performs stochastic optimization based on Monte Carlo sampling, while the methods developed in Sections 5.3.2 and 5.3.3 rely on a dynamic programming approach using the multi step distributions discussed in Section 4.2.

5.1 Embedded System Model

A simplified model of the system dynamics described in Section 3.2 is now developed. This simplified model will serve as the model accessible by various control algorithms developed in subsequent sections. The continuous time embedded system model is defined as

$$\dot{\ell} = v_{veh} \tag{5.2a}$$

$$\dot{v}_{veh} = w \tag{5.2b}$$

$$\dot{n}_{eng} = \frac{1}{I_{eng}} \left[T_{cyl} - \frac{k_1}{2\pi} V_p p - k_1 \hat{M}_{s,p} \right] \tag{5.2c}$$

$$\dot{p} = \frac{1}{C_h(p)} \left[\frac{k_1}{2\pi} V_p n_{eng} - \frac{k_2}{2\pi r_{tire}} V_m v_{veh} - \hat{Q}_{s,p} - \hat{Q}_{s,m} \right] \tag{5.2d}$$

Compared to the model described in Section 3.2, the engine intake manifold dynamics have been neglected and all hydraulic losses are replaced by second order polynomial approximations $\hat{Q}_{s,p}, \hat{Q}_{s,m}, \hat{M}_{s,p}, \hat{M}_{s,m}$. Additionally, the vehicle acceleration dynamic is represented directly by the driver acceleration demand w. The motor displacement volume is once again calculated according to Equation (3.16)

$$V_m = \frac{2\pi}{p} \quad \frac{F_p^{cmd} r_{tire}}{k_2} + \hat{M}_{s,m}$$

where F_p^{cmd} is determined from the driver's acceleration demand w by rearranging Equation (4.1)

$$F_p^{cmd} = m_{veh} w + \tfrac{1}{2} C_d \rho_{air} v_{veh}^2 + m_{veh} g \left[C_r cos(\phi) + sin(\phi) \right] \tag{5.3}$$

The system state and control vectors are defined as

$$\mathbf{x} = \begin{bmatrix} \ell \\ v_{veh} \\ n_{eng} \\ p \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} m_1^{-1} T_{cyl} \\ m_2^{-1} V_p \end{bmatrix} \tag{5.4}$$

respectively. The control inputs are non-dimensionalized versions of cylinder torque and pump displacement volume, with

$$T_{cyl} = m_1 u_1 \tag{5.5a}$$

$$V_p = m_2 u_2 \tag{5.5b}$$

The dynamics of Equation (5.2), represented compactly as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, w, t)$, are numerically integrated using time step Δt by carrying out a Taylor Series Expansion to second order according to

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \dot{\mathbf{x}}(t) + \frac{\Delta t^2}{2} \ddot{\mathbf{x}}(t) + o(\Delta t^2) \tag{5.6}$$

The coefficients $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$ are determined as follows[1] with w and $\mathbf{u}$ assumed as piecewise constant in the interval $[t, t + \Delta t]$

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}, \mathbf{u}, w, t)$$
$$\ddot{\mathbf{x}}(t) = \frac{d\dot{\mathbf{x}}(t)}{dt} = \frac{\partial f}{\partial \mathbf{x}}_{t} f(\mathbf{x}, \mathbf{u}, w, t)$$

The expansion Equation (5.6) is defined in discrete time with timestep Δt as[2]

$$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, w_n) \triangleq \mathbf{x}_n + \Delta t f(t) + \frac{\Delta t^2}{2} \frac{\partial f}{\partial \mathbf{x}}_{t} f(t) \tag{5.7}$$

In this work the embedded system model timestep is chosen as $\Delta t = 1$ second. The horizon length is chosen as $N = 12$ so that the prediction horizon is 12 seconds. It was found through numerical experiments that increasing the horizon beyond 12 timesteps had little to no effect other than increasing computation time.

[1] For simplicity, it is assumed $\frac{\partial f}{\partial t} = 0$

[2] The quantity $f(\mathbf{x}(t), \mathbf{u}(t), w(t), t)$ is represented by shorthand as $f(t)$

5.2 Road Grade Forecasting

Successful predictive energy management is ultimately limited by the ability to forecast the driver's propulsion force command described in Section 5.1,

$$F_p^{cmd} = m_{veh}w + \tfrac{1}{2}C_d\rho_{air}v_{veh}^2 + m_{veh}g\left[C_r cos(\phi) + sin(\phi)\right] \tag{5.3}$$

The largest source of uncertainty is the driver's acceleration demand w, which is modeled as a Markov process and identified in Section 4.2. The vehicle speed v_{veh} can then be anticipated as a result of the forecasted acceleration demand through numerical simulation of the model described by Equation (5.2). What remains to be addressed in Equation (5.3) is the road grade ϕ.

One approach is to model road grade as an independent Markov process as in [34]. The authors of [34] employ stochastic dynamic programming in a finite horizon setting to solve the resulting stochastic optimization problem with reported execution times of 10 to 100 seconds. However, the uncertainty in forecasting F_p^{cmd} along a horizon can be reduced significantly if forecasted road grade incorporated some geometric information as provided by telematics instrumentation, such as a GPS. An assessment on the effect of terrain preview as applied to hybrid electric vehicle control is presented in [59]. Katsargyri [60] uses path forecasting in the form of previewed vehicle speed and road grade in a hybrid electric vehicle. In a similar approach, road grade is previewed along a horizon assuming constant vehicle speed in a conventional vehicle in [30]. Since the state and action spaces are low in [60] and [30], deterministic dynamic programming is used in a finite horizon setting to generate the optimal control trajectory in a model predictive control setup.

The approach taken here incorporates spatially distributed GPS information to develop road grade as a function of vehicle position along the prediction horizon. Unlike previous approaches, future vehicle speed is not assumed known. The segment of road directly ahead of the vehicle is discretized into a grid of n_ℓ equally spaced positions, $r_i, i = 1, 2, \ldots, n_\ell$, so that a sequence of coordinates $(r_i, y_i)_{i=1}^{n_\ell}$ is obtained,

where y is the road altitude. A fit $\hat{y}$ is applied to these coordinates in the form of a multiquadric radial basis function (RBF) with knots $c_i, i = 1, 2, \ldots, n_k$, where $n_k < n_\ell$

$$\hat{y}(\ell) = a_0 + \sum_{i=1}^{n_k} a_i \sqrt{1 + \zeta(\ell - c_i)^2} \tag{5.8}$$

The radial basis function is ideal for this application as its nonlinear basis allows for a high accuracy approximation of road altitude, while the optimal coefficients of its linear weighting structure can be determined efficiently using a least squares projection. The multiquadric form of RBF is specifically chosen as it is differentiable everywhere [61, 62], which will prove valuable when computing road grade. Here, c_i

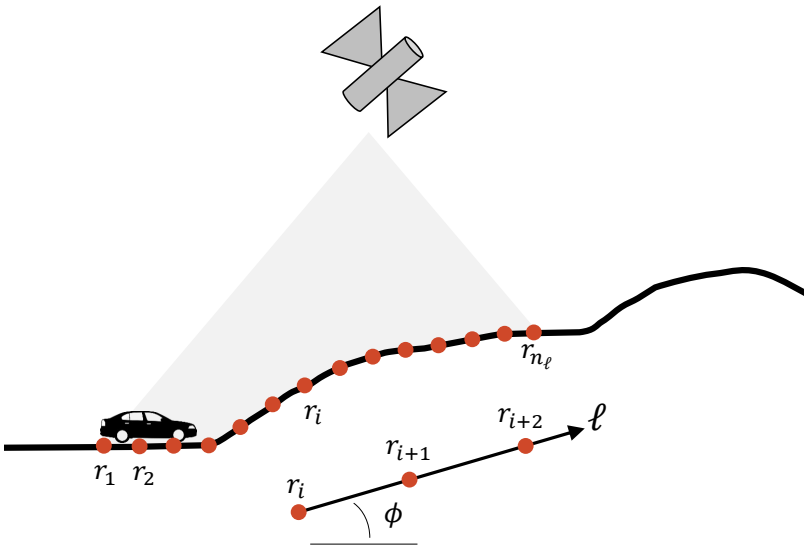

Fig. 5.1. Forecasting road grade along horizon with deterministic, spatially distributed GPS information.

are chosen equally spaced along the grid r_i so that c_1 and c_{n_k} correspond with r_1 and r_{n_ℓ}, respectively, and ζ is a fixed parameter which determines the influence each knot has on the RBF output. The fitting coefficients a_i are calculated in real time using a least squares projection so that the sum of square error $\sum_{i=1}^{n} (y_i - \hat{y}(r_i))^2$ is

minimized. Taking the analytical derivative of $\hat{y}$ from Equation (5.8) with respect to position ℓ gives rate of change in altitude with respect to position

$$\frac{d\hat{y}}{d\ell} = \sum_{i=1}^{n_k} a_i \frac{\zeta(\ell - c_i)}{\sqrt{1 + \zeta(\ell - c_i)^2}} \tag{5.9}$$

from which the road grade model can be computed by taking the inverse sine,

$$\hat{\phi}(\ell) = \sin^{-1}\left(\frac{d\hat{y}}{d\ell}\right) \tag{5.10}$$

Forecasting road grade along the prediction horizon as a function of time is discussed in Sections 5.3.1 and 5.3.2. An example of the road grade estimation applied to real GPS data along a segment of road is shown in Fig. 5.2.

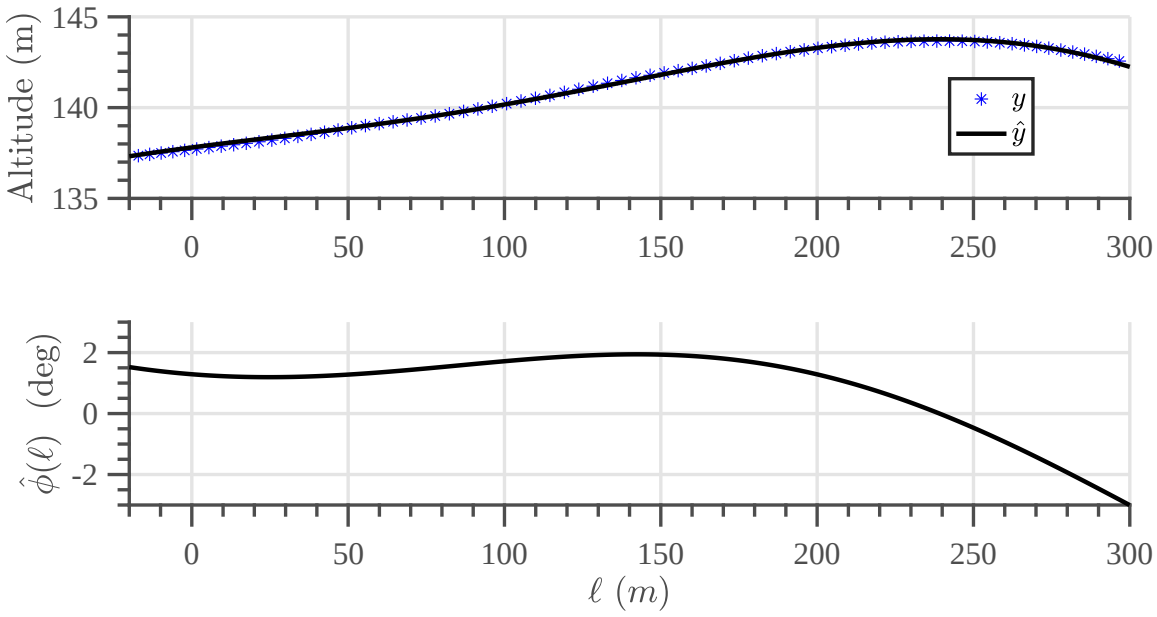

Fig. 5.2. GPS data taken from route in West Lafayette, IN. Positions r_i are set every 20m, with knots c_i placed every 40m, r_1 and c_1 are placed at -20m while r_{n_ℓ} and c_{n_k} are placed at 300m, as referenced to the vehicle's current position. $\zeta = 7.5e-5$.

5.3 Stochastic Control Formulations

The running cost function used in Equation (5.1) is constructed as

$$g_n(\mathbf{x}_n,\mathbf{x}_{n+1},\mathbf{u},w_n) = K_1\left(x_{3,n+1}-x_{3,n}\right)^2 + K_2\,\hat{b}_f(x_3,u_1) + K_3\left(x_4-p^*\right)^2 \times \mathbb{1}_{x_4<p^*} \tag{5.11}$$

Indicator functions are defined as $\mathbb{1}_{a>b}=1$ if $a>b$, $\mathbb{1}_{a>b}=0$ otherwise. The first component of $\tilde{L}$ prevents the engine speed from changing excessively between time steps to prevent undesirable engine operation. The second component is the fuel consumption rate model, $\hat{b}_f$, a polynomial approximation to the actual fuel consumption rate shown in Fig. 3.8. The final term ensures driver demands are satisfied by penalizing system pressures which are lower than a minimum allowable pressure, p^*, which is calculated according to

$$p^* = \max\{p_{req}, p_{set}\} \tag{5.12}$$

The value p_{req} is the pressure required to satisfy driver propulsion force command along the horizon

$$p_{req} = \frac{2\pi}{V_m^{max}} \quad \frac{r_{tire}F_p^{cmd}}{k_2} + \hat{M}_{s,m}\Bigg)\Bigg(\tag{5.13}$$

Equation (5.13) is obtained by rearranging the calculation for motor displacement volume Equation (3.16) and substituting max volume for V_m. Driver propulsion force command F_p^{cmd} is calculated considering the stochastic driver acceleration demand w and resistive forces according to Equation (5.3),

$$F_p^{cmd} = m_{veh}w + \tfrac{1}{2}C_d\rho_{air}v_{veh}^2 + m_{veh}g\left[C_r cos(\phi) + sin(\phi)\right] \tag{5.3}$$

Since F_p^{cmd} is linear in w, it is evident that the statistical model which describes w will directly influence the forecast of driver propulsion force demand and ultimately p_{req} along the horizon.

Satisfying a stochastic driver demand as forecast along a finite horizon can lead to short-sighted planning due to variance in the driver's acceleration demand sequence $\{w_n\}_{n=0}^{N-1}$ and sensitivity of this sequence to the initial demand w_0. By leveraging the long term driver statistics explored in Section 4.3, the value p_{set} in Equation (5.12) provides a pressure target which is independent of initial demand w_0 and does not vary along the horizon thereby allowing for planning beyond the horizon. Recall that ν^j represents the fraction of time the driver demands acceleration w^j, and is calculated from Equation (4.11). The average and standard deviation of non-negative accelerations demands can be determined through

$$w_{ave}^{+} = \frac{\sum_j \nu^j w^j}{\sum_j \nu^j} \ , \quad j \in j^{+} \tag{5.14}$$

$$w_{std}^{+} = \sqrt{\frac{\sum_j \nu^j (w^j)^2}{\sum_j \nu^j} - {w_{ave}^{+}}^2} \ , \quad j \in j^{+} \tag{5.15}$$

where $j^{+} = \{j | w^j \geq 0\}$ is the index set of all non-negative acceleration demands. An acceleration setpoint is now established taking the weighted sum

$$w_{set} = \alpha w_{ave}^{+} + \beta w_{std}^{+} \tag{5.16}$$

In this work, the weights are set as $\alpha = 1$ and $\beta = 1.25$. The value of w_{set} along each drive cycle is shown in Fig. 5.3. The value of w_{set} is observed to jump whenever vehicle speed increases (decreases) above (below) 10 m/s, since two separate Markov chains are retained in memory (one is active at speeds below 10 m/s and a second is active for speeds above 10 m/s) as discussed in Section 4.3.

The intent of this setpoint is to represent a statistically significant driver acceleration demand, so that as a minimum requirement, a differential system pressure should be maintained so that w_{set} can be satisfied instantly, without needing to in-

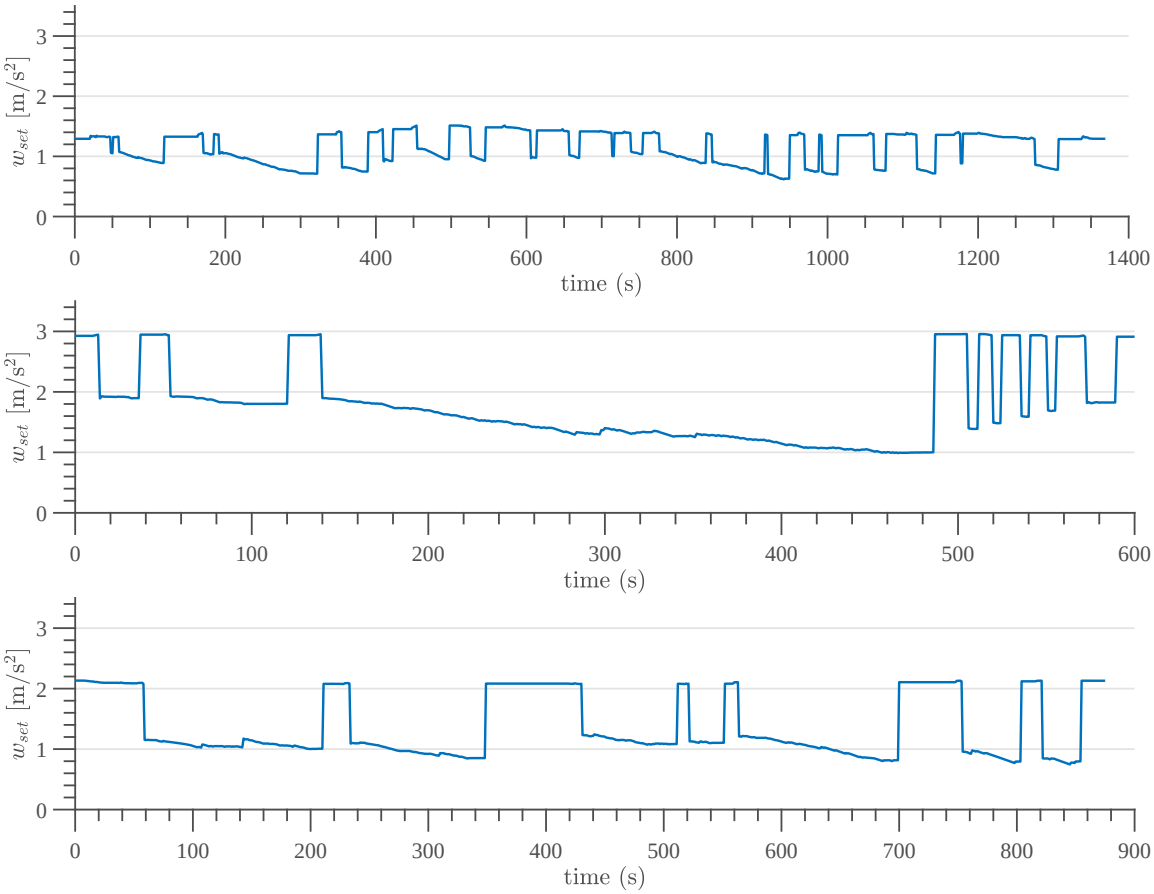

Fig. 5.3. w_{set} for UDDS (top), US06 (middle) and GPS (bottom) cycles.

crease differential system pressure. To this end, the minimum pressure setpoint used in Equation (5.12) is designed as

$$F_p^{set} = m_{veh} w_{set} \tag{5.17a}$$

$$p_{set} = \frac{2\pi}{V_m^{max}} \left(\frac{r_{tire} F_p^{set}}{k_2} + \hat{M}_{s,m} \right) \Bigg(\tag{5.17b}$$

A simple metric for quantifying how well driver demand is met along a drive cycle is discussed in Section 6.3.

5.3.1 Stochastic Gradient Descent with Momentum (SGDM)

This section develops a method to approximately solve Equation (5.1) based on Monte Carlo sampling. The problem is re-formulated as

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[\sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{u}_n, w_n)\ \mathbf{x}_0, w_0\right] \tag{5.18a}$$

$$\text{subject to}\ \ \mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, w_n) \tag{5.18b}$$

State-control constraints are handled with SGDM through penalty functions. The running cost Equation (5.11) is augmented with penalty function $B(\mathbf{x}, \mathbf{u})$

$$g_n(\mathbf{x}_n,\mathbf{x}_{n+1}, \mathbf{u}, w_n) = K_1 \left(x_{3,n+1} - x_{3,n}\right)^2 + K_2\ \hat{b}_f(x_3, u_1) + K_3 \left(x_4 - p^*\right)^2 \times \mathbb{1}_{x_4<p^*} + B(\mathbf{x}, \mathbf{u}) \tag{5.19}$$

where

$$\begin{aligned} B(\mathbf{x}, \mathbf{u}) = {} & b_0 \left(x_3 - x_3^{max}\right)^2 \times \mathbb{1}_{x_3>x_3^{max}} + b_0 \left(x_3 - x_3^{min}\right)^2 \times \mathbb{1}_{x_3<x_3^{min}} \\ & + b_1 \left(\mathbf{u} - \mathbf{u}^{max}\right)^2 \times \mathbb{1}_{\mathbf{u}>\mathbf{u}^{max}} + b_1 \left(\mathbf{u} - \mathbf{u}^{min}\right)^2 \times \mathbb{1}_{\mathbf{u}<\mathbf{u}^{min}} \\ & + b_2 \left(u_1 - T_{cyl}^{max}(x_3)\right)^2 \times \mathbb{1}_{u_1>T_{cyl}^{max}(x_3)} \end{aligned} \tag{5.20}$$

The first component in Equation (5.20) penalizes engine speeds which are outside allowable limits, and likewise, the second component penalizes control inputs which outside physical limits. The final component provides the algorithm with information regarding the maximum torque capabilities of the engine as shown in Fig. 3.8. The intent is to discourage the algorithm from choosing engine torque commands which are beyond the engine's ability, dependent on engine speed.

For convenience we define the horizon cost

$$J(\vec{\mathbf{u}}, \vec{w}) = \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{u}_n, w_n) \tag{5.21}$$

which is a function of the control input sequence $\vec{\mathbf{u}} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}\}$ and the random disturbance input sequence $\vec{w} = \{w_0, w_1, \ldots w_{N-1}\}$. In all that follows, it is assumed that $\mathbf{x}_0$ and w_0 are given so that all expectation computations are conditioned on given values of $\mathbf{x}_0, w_0$. The goal now is to minimize

$$\min_{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}} \mathbb{E}\left[J(\vec{\mathbf{u}}, \vec{w})\right] \tag{5.22}$$

For a given control sequence, the expected value in Equation (5.22) is

$$\mathbb{E}\left[J(\vec{\mathbf{u}}, \vec{w})\right] = \sum_{\vec{\mathbf{w}}} J(\vec{\mathbf{u}}, \vec{\mathbf{w}}) \Pr\left[\vec{w} = \vec{\mathbf{w}}\right] \tag{5.23}$$

Conceptually, if $\nabla_{\vec{\mathbf{u}}} \mathbb{E}[J(\vec{\mathbf{u}}, \vec{w})]$ could be computed directly, a descent with stepsize $\gamma^{[k]}$ of the form

$$\vec{\mathbf{u}}^{[k+1]} = \vec{\mathbf{u}}^{[k]} - \gamma^{[k]} S^{[k]} \nabla_{\vec{\mathbf{u}}} \mathbb{E}[J(\vec{\mathbf{u}}^{[k]}, \vec{w})] \tag{5.24}$$

could be employed, where the order of descent is dependent on the matrix $S^{[k]}$ [63]. Unfortunately, explicitly computing $\nabla_{\vec{\mathbf{u}}} \mathbb{E}[J(\vec{\mathbf{u}}, \vec{w})]$ is generally intractable due to a large number[3] of potential outcomes of the sequence $\vec{w}$, so implementing Equation (5.24) directly is generally not possible.

One approach is to minimizing Equation (5.22) is by approximating Equation (5.23) with the sample average approximation

$$\hat{J}(\vec{\mathbf{u}}) = \frac{1}{K} \sum_{k=1}^{K} J(\vec{\mathbf{u}}, \vec{w}^{[k]}) \tag{5.25}$$

[3]The number of potential outcomes is $|W|^{N-1}$, where $|W|$ is the number of discrete states in the Markov Chain.

where each $J(\vec{\mathbf{u}}, \vec{w}^{[k]})$ is a Monte Carlo sample of the random variable $J(\vec{\mathbf{u}}, \vec{w})$. In general, the approximation $\hat{J}(\vec{\mathbf{u}})$ improves as the number of Monte Carlo samples K increases in accordance with a law of large numbers argument. In [33], Quadratic Programming is employed to minimize Equation (5.25) as applied to the hybrid electric vehicle (HEV) energy management problem. The computational challenge with this approach is K trajectories of any relevant system information must be stored in memory, and the subsequent optimization must be performed considering the entire sample set in the spirit of batch optimization [24, 48]. To reduce the computational burden, [33] removes Monte Carlo samples with comparatively low probability of occurrence from the batch optimization.

The stochastic gradient descent (SGD) update

$$\vec{\mathbf{u}}^{[k+1]} = \vec{\mathbf{u}}^{[k]} - \gamma^{[k]} \nabla_{\mathbf{u}} J(\vec{\mathbf{u}}^{[k]}, \vec{w}^{[k]}) \tag{5.26}$$

is a stochastic form of the idealized descent of Equation (5.24), and is exactly the gradient form of stochastic approximation from Section 2.2.1.2. Stochastic gradient descent finds a locally optimal solution $\vec{\mathbf{u}}^*$ which asymptotically (locally) minimizes the original problem Equation (5.22) [48]. With SGD, only one Monte Carlo sample of the gradient $\nabla_{\mathbf{u}} J(\vec{\mathbf{u}}^{[k]}, \vec{w}^{[k]})$ is required at each iteration offering significantly reduced computational overhead, allowing SGD to process more samples than batch processing in a fixed amount of time. In this way, SGD is competitive with and can even outperform second-order batch optimization methods [64], [65]. The benefit of the sequential optimization approach can understood considering stochastic optimization based on Monte Carlo sampling is as much an estimation problem as it is an optimization problem [66]. The total solution error is a combination of optimization error, which measures an algorithm's ability to determine the optimal solution for the given sampling set, and estimation error, which measures the effect of minimizing an empirical average Equation (5.25) rather than expected cost Equation (5.23). If $\vec{\mathbf{u}}^*$ is

the locally optimal solution determined by a given algorithm, then the total solution error is

$$\underbrace{\hat{J}(\vec{\mathbf{u}}^*) - \min_{\vec{\mathbf{u}}} \mathbb{E}[J(\vec{\mathbf{u}}, \vec{w})]}_{\varepsilon_{tot}} = \underbrace{\hat{J}(\vec{\mathbf{u}}^*) - \min_{\vec{\mathbf{u}}} \hat{J}(\vec{\mathbf{u}})}_{\varepsilon_{opt}} + \underbrace{\min_{\vec{\mathbf{u}}} \hat{J}(\vec{\mathbf{u}}) - \min_{\vec{\mathbf{u}}} \mathbb{E}[J(\vec{\mathbf{u}}, \vec{w})]}_{\varepsilon_{est}}$$

The estimation error generally decreases inversely with K, therefore the total solution error depends on the number of samples that can be processed in the alloted time. The step size sequence $\{\gamma^{[k]}\}_{k\geq 1}$, $\gamma^{[k]} \in \mathbb{R}$ must satisfy the rules given in Section 2.2.1.2 for stochastic approximation:

$$\sum_{k=1}^{\infty} \gamma^{[k]} = \infty, \qquad \sum_{k=1}^{\infty} \left(\gamma^{[k]}\right)^2 < \infty \tag{2.32}$$

The step size schedule chosen here is

$$\gamma^{[k]} = \frac{\gamma_0}{1 + (k-1)\epsilon}, \; k = 1, 2, ... \tag{5.27}$$

where $\epsilon > 0$ is called the decay rate. In this work, we use a slightly modified version of SGD known as stochastic gradient descent with momentum (SGDM) based on Nesterov's Accelerated Gradient (NAG) [67]

$$\vec{\mathbf{v}}^{[k+1]} = \mu \vec{\mathbf{v}}^{[k]} - \gamma^{[k]} \nabla_{\vec{\mathbf{u}}} J(\vec{\mathbf{u}}^{[k]} + \mu \vec{\mathbf{v}}^{[k]}, \vec{w}^{[k]}) \tag{5.28a}$$

$$\vec{\mathbf{u}}^{[k+1]} = \vec{\mathbf{u}}^{[k]} + \vec{\mathbf{v}}^{[k+1]} \tag{5.28b}$$

The quantity $\mathbf{v} \in \mathbb{R}^{dimU}$ is referred to as the velocity term and decays at a rate according to $\mu \in [0, 1)$, known as the momentum parameter. The effect of momentum is to continue pushing the parameter update in directions of previous updates, averaging out oscillations in areas of a rapidly changing gradient. Simultaneously, if several past updates are approximately aligned, the velocity term will act to propel the parameter

update faster than if momentum was absent. The net result is that SGDM tends to move more rapidly towards a local minimum than classical SGD [19, 65, 67, 68]. An attractive feature of NAG is the gradient computation performed in Equation (5.28a) considers a projected estimate of the control sequence, $\vec{\mathbf{u}}^{[k]} + \mu\vec{\mathbf{v}}^{[k]}$, based on the most recent velocity sequence $\vec{\mathbf{v}}^{[k]} = (\mathbf{v}_n^{[k]})_{n=0}^{N-1}$. This projected estimate is in some respect not unlike predictor-corrector methods used to improve stability in numerical solution of ordinary differential equations. The result is improved stability compared to classical momentum, in which the gradient is computed considering only the current value of the control parameter array, particularly when $\mu \approx 1$ [67].

5.3.1.1 Computing the Gradient

This sections proposes a method to iteratively compute the gradient $\nabla_{\vec{\mathbf{u}}} J$ used in the control sequence update Equation (5.28) based on a piecewise linear approximation to the system dynamics along the horizon. The gradient

$$\nabla_{\vec{\mathbf{u}}} J = \begin{bmatrix} \nabla_{\mathbf{u}_0} J & \nabla_{\mathbf{u}_1} J & \dots & \nabla_{\mathbf{u}_{N-1}} J \end{bmatrix} \in \mathbb{R}^{dimU \times N} \tag{5.29}$$

has individual components given by

$$\nabla_{\mathbf{u}_n} J = \sum_{k=0}^{N-1} \left[\frac{\partial g_k}{\partial \mathbf{x}_k} \frac{d\mathbf{x}_k}{d\mathbf{u}_n} + \frac{\partial g_k}{\partial \mathbf{x}_{k+1}} \frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_n} \right]^{\mathsf{T}} + \frac{\partial g_n}{\partial \mathbf{u}_n}^{\mathsf{T}} \tag{5.30}$$

where $\frac{\partial g_k}{\partial \mathbf{x}_k} \in \mathbb{R}^{1 \times dimX}$, $\frac{\partial \mathbf{x}_k}{\partial \mathbf{u}_n} \in \mathbb{R}^{dimX \times dimU}$, $\frac{\partial g_n}{\partial \mathbf{u}_n} \in \mathbb{R}^{1 \times dimU}$. In evaluating Equation (5.30), it will be helpful to define the following matrix

$$\mathcal{C}_n \triangleq \begin{bmatrix} \dfrac{d\mathbf{x}_n}{d\mathbf{u}_0} & \dfrac{d\mathbf{x}_n}{d\mathbf{u}_1} & \dots & \dfrac{d\mathbf{x}_n}{d\mathbf{u}_{N-1}} \end{bmatrix} \in \mathbb{R}^{dimX \times N dimU} \tag{5.31}$$

An efficient recursion for $\mathcal{C}_n$ which can be updated iteratively along the horizon is now developed. Carrying out the first few $\mathcal{C}_n$ gives

$$
\begin{aligned}
\mathcal{C}_1: &\quad \frac{d\mathbf{x}_1}{d\mathbf{u}_0} = \frac{\partial F_0}{\partial \mathbf{u}_0} && \mathbf{0} && \mathbf{0} && \mathbf{0} \quad \cdots \quad \mathbf{0} \\
\mathcal{C}_2: &\quad \frac{d\mathbf{x}_2}{d\mathbf{u}_0} = \frac{\partial F_1}{\partial \mathbf{x}_1}\frac{d\mathbf{x}_1}{d\mathbf{u}_0} && \frac{d\mathbf{x}_2}{d\mathbf{u}_1} = \frac{\partial F_1}{\partial \mathbf{u}_1} && \mathbf{0} && \mathbf{0} \quad \cdots \quad \mathbf{0} \\
\mathcal{C}_3: &\quad \frac{d\mathbf{x}_3}{d\mathbf{u}_0} = \frac{\partial F_2}{\partial \mathbf{x}_2}\frac{d\mathbf{x}_2}{d\mathbf{u}_0} && \frac{d\mathbf{x}_3}{d\mathbf{u}_1} = \frac{\partial F_2}{\partial \mathbf{x}_2}\frac{d\mathbf{x}_2}{d\mathbf{u}_1} && \frac{d\mathbf{x}_3}{d\mathbf{u}_2} = \frac{\partial F_2}{\partial \mathbf{u}_2} && \mathbf{0} \quad \cdots \quad \mathbf{0}
\end{aligned}
$$

By inspection, a recursion for $\mathcal{C}_n$ is given by

$$
\mathcal{C}_{n+1} = \frac{\partial F_n}{\partial \mathbf{x}_n}\mathcal{C}_n + \left[\underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{n \text{ blocks}} \quad \frac{\partial F_n}{\partial \mathbf{u}_n} \quad \underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{N-1-n \text{ blocks}}\right], \quad n = 0, \ldots, N-1 \tag{5.32}
$$

$$
\mathcal{C}_0 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}
$$

$$
\mathbf{0} \in \mathbb{R}^{dimX \times dimU}
$$

In this way, $\mathcal{C}_n$ is updated incrementally at each time step n along the horizon. The individual partial derivatives are calculated considering the system dynamics Equation (5.7)

$$
\frac{\partial F}{\partial \mathbf{x}} = I + h\frac{\partial f}{\partial \mathbf{x}} + \frac{h^2}{2}\left(\frac{\partial f}{\partial \mathbf{x}}\right)^2 \tag{5.33a}
$$

$$
\frac{\partial F}{\partial \mathbf{u}} = I + h\frac{\partial f}{\partial \mathbf{u}} + \frac{h^2}{2}\frac{\partial f}{\partial \mathbf{x}}\frac{\partial f}{\partial \mathbf{u}} \tag{5.33b}
$$

In deriving Equation (5.33) all second order partial derivatives of the form $\frac{\partial^2 f}{\partial \mathbf{x}^2}$ and $\frac{\partial^2 f}{\partial \mathbf{x}\partial \mathbf{u}}$ have been ignored. The gradient Equation (5.29) can now be evaluated with $\mathcal{C}_n$ through

$$
\nabla_{\vec{\mathbf{u}}} J = \text{reshape}\left\{\sum_{n=0}^{N-1}\left[\frac{\partial g_n}{\partial \mathbf{x}_n}\mathcal{C}_n + \frac{\partial g_n}{\partial \mathbf{x}_{n+1}}\mathcal{C}_{n+1}\right]\right\} + \frac{\partial J}{\partial \vec{\mathbf{u}}} \tag{5.34}
$$

where the function *reshape* is used to convert the $1 \times NdimU$ row vector into a $dimU \times N$ matrix and

$$\frac{\partial J}{\partial \vec{\mathbf{u}}} = \left[\frac{\partial g_0}{\partial \mathbf{u}_0}^{\mathsf{T}} \quad \frac{\partial g_1}{\partial \mathbf{u}_1}^{\mathsf{T}} \quad \cdots \quad \frac{\partial g_{N-1}}{\partial \mathbf{u}_{N-1}}^{\mathsf{T}} \right] \in \mathbb{R}^{dimU \times N} \tag{5.35}$$

Finally, $\nabla_{\vec{\mathbf{u}}} J$ is updated iteratively at each time step along the horizon through

$$\nabla_{\vec{\mathbf{u}}} J \leftarrow \nabla_{\vec{\mathbf{u}}} J + \text{reshape} \left\{ \frac{\partial g_n}{\partial \mathbf{x}_n} \mathcal{C}_n + \frac{\partial g_n}{\partial \mathbf{x}_{n+1}} \mathcal{C}_{n+1} \right\} + \frac{\partial g_n}{\partial \mathbf{u}_n}^{\mathsf{T}} \times \mathbb{1}_n \tag{5.36}$$
$$n = 0, \ldots, N-1$$

where $\mathbb{1}_n$ is a N-element row vector such that the k^{th} element is given by

$$\mathbb{1}_n(k) = \begin{cases} 1 & \text{if } k = n+1 \\ 0 & \text{if } k \neq n+1 \end{cases} \tag{5.37}$$

The update Equation (5.36) is initialized with $\nabla_{\vec{\mathbf{u}}} J = \left[\mathbf{0} \quad \cdots \quad \mathbf{0} \right] \in \mathbb{R}^{dimU \times N}$.

5.3.1.2 Monte Carlo Sampling and Variance Reduction

Each Monte Carlo sample $J(\vec{\mathbf{u}}, \vec{w}^{[k]})$ is created by randomly generating the sequence $\{w_n\}_{n=0}^{N-1}$ drawn from the single-step distribution P_{ij} according to

$$w_{n+1} \sim P_{ij} \ , \ \text{where } w^i \triangleq w_n$$

The process of drawing w_{n+1} from P_{ij} is as follows. A sequence of random numbers $\{\omega_0, \omega_1, \ldots, \omega_{N-2}\}$ is generated, where each $\omega_n \in [0, 1]$ is an independent uniform random number. The initial value of w_0 is given and, at each stage $n = 0, \ldots, N-2$, w^i is reset according to $w^i \triangleq w_n$. The value assigned to w_{n+1} is then determined from ω_n according to

$$\begin{aligned} 0 < \omega_n \leq P_{i1} : &\quad w_{n+1} = w^1 \\ P_{i1} < \omega_n \leq P_{i1} + P_{i2} : &\quad w_{n+1} = w^2 \\ P_{i1} + P_{i2} < \omega_n \leq P_{i1} + P_{i2} + P_{i3} : &\quad w_{n+1} = w^3 \\ &\vdots \end{aligned}$$

The general rule for assigning the specific value w^j to w_{n+1} is

$$P_{i1} + \cdots + P_{ij-1} < \omega_n \leq P_{i1} + \cdots + P_{ij} : \quad w_{n+1} = w^j \tag{5.38}$$

The assignment rule Equation (5.38) is performed for $n = 0, \ldots, N-2$. Variance reduction is accomplished with a technique known as PEGASUS [69], in which the Monte Carlo sampling of Equation (5.38) is performed using the same sets of random numbers. A set of K random number sequences is generated before the algorithm is started

$$\begin{aligned} \vec{\omega}^{[1]} &= \{\omega_0, \ldots, \omega_{N-2}\}^{[1]} \\ &\vdots \\ \vec{\omega}^{[k]} &= \{\omega_0, \ldots, \omega_{N-2}\}^{[k]} \end{aligned}$$

At iteration k of SGDM, the k^{th} sequence of random numbers $\vec{\omega}^{[k]}$ is used in the Monte Carlo sampling Equation (5.38). After K iterations, a new point $(\mathbf{x}_0, w_0)$ is measured and brought in as the new initial condition and the process is restarted using the same K sets of random number sequences. The benefit is that for a fixed $(\mathbf{x}_0, w_0)$ initial condition the optimization process reduces to a completely deterministic optimization, resulting in significantly reduced variance in the control sequence between executions of SGDM.

5.3.1.3 Scaling and Final Algorithm

Performance of SGDM is improved significantly by properly scaling the control inputs. The scaling factors m_1, m_2 from Equation (5.5) are determined empirically so that $\nabla_{\vec{\mathbf{u}}} J$ has components of approximately equal magnitude along each dimension, which is a common approach in numerical solution of optimal control problems [25]. The final algorithm is shown in Algorithm 1. Maximum algorithm iterations is set to $K = 200$. For the first 50 iterations the stepsize is held constant at $\gamma = 0.2$, afterwards a decay of $\epsilon = 0.1$ is used. The momentum parameter is set as $\mu = 0.95$. These parameters were finely tuned to deliver optimum performance from SGDM.

Algorithm 1: SGDM

Input: $\mathbf{x}_0, w_0, \vec{\mathbf{u}}, \vec{\mathbf{v}}$

Data: $N, \epsilon, \mu, \gamma_0, K, \{\vec{\omega}^{(1)}, \ldots, \vec{\omega}^{[k]}\}$

for $k = 1 : K$ **do**

Given w_0, generate sample $\{w_1, \ldots, w_{N-1}\}(\vec{\omega}^{[k]})$

$\nabla_{\vec{\mathbf{u}}} J = \mathbf{0} \in \mathbb{R}^{dimU \times N}$

$\mathcal{C}_0 = \mathbf{0} \in \mathbb{R}^{dimX \times N dimU}$

$\vec{\mathbf{u}} := \vec{\mathbf{u}} + \mu \vec{\mathbf{v}}$

for $n = 0 : N - 1$ **do**

$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, w_n)$

Compute $\frac{\partial F_n}{\partial \mathbf{x}_n}, \frac{\partial F_n}{\partial \mathbf{u}_n}, \frac{\partial g_n}{\partial \mathbf{x}_n}, \frac{\partial g_n}{\partial \mathbf{x}_{n+1}}$

$\mathcal{C}_{n+1} = \frac{\partial F_n}{\partial \mathbf{x}_n} \mathcal{C}_n + \left[\underbrace{\mathbf{0} \cdots \mathbf{0}}_{n \text{ blocks}} \; \frac{\partial F_n}{\partial \mathbf{u}_n} \; \underbrace{\mathbf{0} \cdots \mathbf{0}}_{N-1-n \text{ blocks}} \right]$

$\nabla_{\vec{\mathbf{u}}} J \leftarrow \nabla_{\vec{\mathbf{u}}} J + \text{reshape}\left\{ \frac{\partial g_n}{\partial \mathbf{x}_n} \mathcal{C}_n + \frac{\partial g_n}{\partial \mathbf{x}_{n+1}} \mathcal{C}_{n+1} \right\} + \left(\frac{\partial g_n}{\partial \mathbf{u}_n} \right)^{\mathsf{T}} \times \mathbb{1}_n$

end

$\gamma = \dfrac{\gamma_0}{1 + (k-1)\epsilon}$

$\vec{\mathbf{v}} \leftarrow \mu \vec{\mathbf{v}} - \gamma \nabla_{\vec{\mathbf{u}}} J$

$\vec{\mathbf{u}} \leftarrow \vec{\mathbf{u}} + \vec{\mathbf{v}}$

end

Output: $\vec{\mathbf{u}}, \vec{\mathbf{v}}$

5.3.2 Approximate Stochastic Differential Dynamic Programming (ASDDP)

This sections develops *approximate stochastic differential dynamic programming* (ASDDP), a stochastic variant of the classic differential dynamic programming algorithm described in Section 2.1.4, to approximately solve Equation (5.1). The problem is re-formulated as

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[\sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{u}_n, w_n) \;\mathbf{x}_0, w_0\right] \tag{5.39a}$$

$$\text{subject to} \quad \mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, w_n) \tag{5.39b}$$

$$\bar{\mathbf{x}}_{n+1} = \sum_j P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n, w^j) \tag{5.39c}$$

$$D_x \bar{\mathbf{x}}_{n+1} \leq \mathbf{c}_x \tag{5.39d}$$

$$D_u \mathbf{u}_n \leq \mathbf{c}_u \tag{5.39e}$$

Equation (5.39c) is the expected state trajectory along the horizon. Equations (5.39d) and (5.39e) are linear constraints on the expected state and control input trajectories. The state value function is defined as (the derivation can be found in Appendix B)

$$\begin{aligned} V_n(\mathbf{x}_n) &\triangleq \min_{\mathbf{u}_n,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\Big[h(\mathbf{x}_N) + \sum_{k=n}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k, w_k) \;\mathbf{x}_n, w_0 = w^i\Big] \\ &= \min_{\mathbf{u}_n} \mathbb{E}\Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w_n)\big) \;\mathbf{x}_n, w_0 = w^i\Big] \qquad (5.40) \\ &= \min_{\mathbf{u}_n} \sum_j P_{ij}^{(n)} \Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w^j) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w^j)\big)\Big] \qquad (5.41) \end{aligned}$$

With this state value function, the expectation is conditioned on fixed disturbance information available at the start of the horizon, $w_0 = w^i$. As a result, the transition probabilities change along the horizon according to the multi-step transition probability $P_{ij}^{(n)}$. The value function V_n can also be given in terms of the state-control

value function Q_n according to $V_n(\mathbf{x}) = Q_n(\mathbf{x}, \mathbf{u}^*)$ where $\mathbf{u}^* = \arg\min_{\mathbf{u}} Q_n(\mathbf{x}, \mathbf{u})$ and Q_n is defined in a manner consistent with Equation (5.41)

$$\begin{aligned} Q_n(\mathbf{x}_n, \mathbf{u}_n) &= \mathbb{E}\Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w_n)\big) \Big| \mathbf{x}_n, w_0 = w^i \Big] \\ &= \sum_j P_{ij}^{(n)} \Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w^j) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w^j)\big) \Big] \end{aligned} \tag{5.42}$$

Given a nominal trajectory $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1}$ a local model of Q_n to second order is constructed as

$$\begin{aligned} &Q_n(\hat{\mathbf{x}}_n + \delta\mathbf{x}_n, \hat{\mathbf{u}}_n + \delta\mathbf{u}_n) \approx \\ &\qquad Q_n^{(0)} + Q_n^{(x)}\delta\mathbf{x}_n + Q_n^{(u)}\delta\mathbf{u}_n + \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}_n^{\mathsf{T}} & \delta\mathbf{u}_n^{\mathsf{T}}\end{bmatrix} \begin{bmatrix} Q_n^{(xx)} & Q_n^{(xu)} \\ Q_n^{(ux)} & Q_n^{(uu)} \end{bmatrix} \begin{bmatrix}\delta\mathbf{x}_n \\ \delta\mathbf{u}_n\end{bmatrix} \end{aligned} \tag{5.43}$$

Here, $\delta\mathbf{x}_n$ and $\delta\mathbf{u}_n$ are small perturbations in the state and control vectors at time n and $Q_n^{(0)} \triangleq Q_n(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)$. The partial derivatives $Q_n^{(x)}, Q_n^{(u)}, Q_n^{(xx)}, Q_n^{(uu)}, Q_n^{(ux)}$ centered about $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)$ are determined considering Equation (5.42)

$$Q_n^{(a)} = \sum_j P_{ij}^{(n)} \Big[g_n^{(a)}(\hat{q}_n) + V_{n+1}^{(x)}(\mathbf{x}') F_n^{(a)}(\hat{q}_n) \Big] \tag{5.44a}$$

$$Q_n^{(ab)} = \sum_j P_{ij}^{(n)} \Big[g_n^{(ab)}(\hat{q}_n) + F_n^{(a)\mathsf{T}}(\hat{q}_n) V_{n+1}^{(xx)}(\mathbf{x}') F_n^{(b)}(\hat{q}_n) \Big] \tag{5.44b}$$

where $\hat{q}_n \triangleq (\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j)$ and $\mathbf{x}' \triangleq F_n(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j)$. To reduce computational burden, the second order derivatives $F_n^{(xx)}, F_n^{(ux)}, F_n^{(uu)}$ have been neglected in the last equation of (5.44). For given $\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, \delta\mathbf{x}_n$, the unconstrained value of $\delta\mathbf{u}_n$ which minimizes the local model Equation (5.43) is

$$\delta\mathbf{u}_n^* = \arg\min_{\delta\mathbf{u}_n} Q_n = -\left(Q_n^{(uu)}\right)^{-1}\left(Q_n^{(u)} + Q_n^{(ux)}\delta\mathbf{x}_n\right) \tag{5.45}$$

Substituting $\delta \mathbf{u}_n^*$ into the local model Equation (5.43) and simplifying gives a local second order model for $V_n(\mathbf{x})$ about the nominal trajectory $(\hat{\mathbf{x}}_n)_{n=0}^{N-1}$ for arbitrary $\mathbf{x}$ where $\delta \mathbf{x}_n = \mathbf{x} - \hat{\mathbf{x}}_n$

$$\begin{aligned} V_n(\mathbf{x}) \approx Q_n^{(0)} - \frac{1}{2} Q_n^{(u)\mathsf{T}} (Q_n^{(uu)})^{-1} Q_n^{(u)} + \left[Q_n^{(x)} - Q_n^{(u)} (Q_n^{(uu)})^{-1} Q_n^{(ux)} \right] (\mathbf{x} - \hat{\mathbf{x}}_n) \\ + \frac{1}{2} (\mathbf{x}_n - \hat{\mathbf{x}}_n)^{\mathsf{T}} \left[Q_n^{(xx)} - Q_n^{(xu)} (Q_n^{(uu)})^{-1} Q_n^{(ux)} \right] (\mathbf{x} - \hat{\mathbf{x}}_n) \end{aligned} \tag{5.46}$$

For fixed $\hat{\mathbf{x}}_n$, the partial derivatives of Equation (5.46) are evaluated at arbitrary $\mathbf{x}$ according to

$$V_N^{(x)}(\mathbf{x}) = h^{(x)}(\mathbf{x}) \tag{5.47a}$$

$$V_N^{(xx)}(\mathbf{x}) = h^{(xx)}(\mathbf{x}) \tag{5.47b}$$

$$V_n^{(x)}(\mathbf{x}) = [Q_n^{(x)} - Q_n^{(u)} (Q_n^{(uu)})^{-1} Q_n^{(ux)}] + [Q_n^{(xx)} - Q_n^{(xu)} (Q_n^{(uu)})^{-1} Q_n^{(ux)}](\mathbf{x} - \hat{\mathbf{x}}_n) \tag{5.47c}$$

$$V_n^{(xx)}(\mathbf{x}) = Q_n^{(xx)} - Q_n^{(xu)} (Q_n^{(uu)})^{-1} Q_n^{(ux)} \tag{5.47d}$$

Starting from initial condition $V_N(\hat{\mathbf{x}}_N) = h(\hat{\mathbf{x}}_N)$, Equation (5.44) and Equation (5.47) are evaluated backwards in time along the horizon about the nominal trajectory $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1}$ which constitutes the *backward pass*.

The next step is to update the nominal trajectory $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N-1}$ by simulating the system forward in time along the horizon, which constitutes the *forward pass*. Unlike the classic deterministic case of DDP, the forward pass is uncertain in the stochastic setting as state trajectory $(\mathbf{x}_n)_{n=0}^{N-1}$ depends on the realization of the stochastic disturbance trajectory $(w_n)_{n=0}^{N-1}$. The expected nominal state trajectory is generated for a given control sequence considering disturbance information available at the beginning of the horizon according to

$$\begin{aligned}\bar{\mathbf{x}}_{n+1} &= \mathbb{E}[F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n, w_n)|\bar{\mathbf{x}}_n, w_0 = w^i] \\ &= \sum_j \Bigg(P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n, w^j) \end{aligned} \tag{5.48}$$

Starting from initial condition $\mathbf{x}_0 = \mathbf{x}_0^{\text{meas}}$, a new system trajectory is simulated forward in time along the horizon $n = 0, \ldots, N-1$ according to Equation (5.49) which represents the *forward pass*

$$\bar{\mathbf{x}}_0 = \mathbf{x}_0^{\text{meas}} \ , \ w_0 = w_0^{\text{meas}} \tag{5.49a}$$

$$\mathbf{u}_n^* = \hat{\mathbf{u}}_n \underbrace{- \left(Q_n^{(uu)}\right)^{-1} \left[Q_n^{(u)} - Q_n^{(ux)} \left(\bar{\mathbf{x}}_n - \hat{\mathbf{x}}_n\right)\right]}_{\delta \mathbf{u}_n^*} \Bigg(\tag{5.49b}$$

$$\bar{\mathbf{x}}_{n+1} = \sum_j \Bigg(P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n^*, w^j) \Bigg(\Bigg(\tag{5.49c}$$

The new nominal trajectory is updated according to $\{\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n\}_{n=0}^{N-1} := \{\bar{\mathbf{x}}_n, \mathbf{u}_n^*\}_{n=0}^{N-1}$ and the process is restarted.

5.3.2.1 State - Control Constraints

Minimizing the local model of Q_n given by Equation (5.43) is an unconstrained quadratic optimization problem, whose solution is given by Equation (5.45). However, with some modification the problem of minimizing Equation (5.43) subject to state and control input constraints in a stochastic environment can be addressed. A first order expansion about $(\bar{\mathbf{x}}_n, \hat{\mathbf{u}}_n)$ is taken to produce an approximation to the system dynamics that is linear in the control input

$$\begin{aligned}\bar{\mathbf{x}}_{n+1} &= \sum_j P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n, w^j) \\ &\approx \sum_j P_{ij}^{(n)} \left[F_n(\bar{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j) + F_n^{(u)}(\bar{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j)\delta\mathbf{u}_n\right]\end{aligned} \tag{5.50}$$

The state and control vectors are constrained according to

$$D_x \bar{\mathbf{x}}_{n+1} \leq \mathbf{c}_x \tag{5.51a}$$

$$D_u \left[\hat{\mathbf{u}}_n + \delta\mathbf{u}_n\right] \leq \mathbf{c}_u \tag{5.51b}$$

Combining these equations leads to the following constrained quadratic programming problem, which is solved with an active set strategy [70]

$$\min_{\delta\mathbf{u}_n} \quad \frac{1}{2}\delta\mathbf{u}_n^\mathsf{T} Q_n^{(uu)} \delta\mathbf{u}_n + \left(Q_n^{(u)} + \delta\mathbf{x}_n^\mathsf{T} Q_n^{(xu)}\right)\delta\mathbf{u}_n \tag{5.52a}$$

$$\text{subject to} \quad D\delta\mathbf{u}_n \leq \mathbf{c} \tag{5.52b}$$

$$D = \begin{bmatrix} D_x \sum_j P_{ij}^{(n)} F_n^{(u)}(\bar{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j) \\ D_u \end{bmatrix} \tag{5.52c}$$

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_x - \sum_j P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j) \\ \mathbf{c}_u - D_u \hat{\mathbf{u}}_n \end{bmatrix} \tag{5.52d}$$

Solving the quadratic programming problem described by Equation (5.52) constrains the expected state trajectory along the horizon considering control input constraints.

5.3.2.2 Modification for Global Convergence

A standard modification is made to ensure the Hessian matrix $Q_n^{(uu)}$ is positive definite at all stages along the horizon. In this way, convergence occurs even far from

the solution when $Q_n^{(uu)}$ may not be positive definite. A simple method is used based on Hessian modification in standard Newton iteration [70, 71],

$$Q_n^{(uu)} := Q_n^{(uu)} + \tau I \tag{5.53a}$$

where

$$\tau = \begin{cases} \delta - \lambda_{min}\left(Q_n^{(uu)}\right), & \delta > \lambda_{min}\left(Q_n^{(uu)}\right) \\ 0, & \delta \leq \lambda_{min}\left(Q_n^{(uu)}\right) \end{cases} \tag{5.53b}$$

The modification performed by Equation (5.53) ensures the smallest eigenvalue of $Q_n^{(uu)}$ is no less than $\delta > 0$, which in this work is set to $\delta = 0.003$. It is worthing noting that the same control input scalings m_1 and m_2 used in Section 5.3.1 are used for the ASDDP algorithm. The benefit of using input scalings here is that the eigenvalues of $Q_n^{(uu)}$ have approximately the same magnitude. The ASDDP algorithm is summarized in Algorithm 2.

5.3.2.3 Remarks on Computational Complexity of ASDDP

In retrospect the value function shown in (5.41) is similar to a stochastic variant of DDP presented in [39] in which V_n is explicitly dependent on the stochastic state. However, here Equation (5.41) is not explicitly dependent on the stochastic state due to the fact that ASDDP incorporates the multi-step Markov transition probability $P_{ij}^{(n)}$. As such, (5.41) must only be evaluated for every $w^j \in W$, not for every $(w^i, w^j) \in W \times W$. This significantly reduces the computational complexity of the *backward pass* from $O(|W|^2)$ to $O(|W|)$ making ASDDP more suitable for real time implementation.

Algorithm 2: ASDDP

Input: $\mathbf{x}_0, w_0, (\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N}$

$\hat{\mathbf{x}}_0 := \mathbf{x}_0, w^i := w_0$

——*Backward Pass*——

$\{Q_n^{(x)}, Q_n^{(u)}, Q_n^{(xx)}, Q_n^{(uu)}, Q_n^{(ux)}\} = 0$

for $n = N-1 : 0$ **do**

  foreach $w^j \in W$ **do**

    $\mathbf{x}_{n+1} = F_n(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, w^j)$

    if *n=N-1* **then**

      $V_{n+1}^{(x)} = h^{(x)}(\mathbf{x}_{n+1}), \quad V_{n+1}^{(xx)} = h^{(xx)}(\mathbf{x}_{n+1})$

    end

    else

      $V_{n+1}^{(x)} = A + B\big[\mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1}\big], \;\; V_{n+1}^{(xx)} = B$

    end

    $Q_n^{(x)} = Q_n^{(x)} + P_{ij}^{(n)}\big[g_n^{(x)} + V_{n+1}^{(x)} F_n^{(x)}\big]$

    $Q_n^{(u)} = Q_n^{(u)} + P_{ij}^{(n)}\big[g_n^{(u)} + V_{n+1}^{(x)} F_n^{(u)}\big]$

    $Q_n^{(xx)} = Q_n^{(xx)} + P_{ij}^{(n)}\big[g_n^{(xx)} + F_n^{(x)T} V_{n+1}^{(xx)} F_n^{(x)}\big]$

    $Q_n^{(uu)} = Q_n^{(uu)} + P_{ij}^{(n)}\big[g_n^{(uu)} + F_n^{(u)T} V_{n+1}^{(xx)} F_n^{(u)}\big]$

    $Q_n^{(ux)} = Q_n^{(ux)} + P_{ij}^{(n)}\big[g_n^{(ux)} + F_n^{(u)T} V_{n+1}^{(xx)} F_n^{(x)}\big]$

    $Q_n^{(xu)} = Q_n^{(ux)T}$

  end

  Modify $Q_n^{(uu)}$ according to Equation (5.53)

  $A = Q_n^{(x)} - Q_n^{(u)}\big[Q_n^{(uu)}\big]^{-1} Q_n^{(ux)}$, $B = Q_n^{(xx)} - Q_n^{(xu)}\big[Q_n^{(uu)}\big]^{-1} Q_n^{(ux)}$

end

——*Forward Pass*——

$\bar{\mathbf{x}}_0 := \mathbf{x}_0$

for $n = 0 : N-1$ **do**

  $\delta\mathbf{x}_n := \bar{\mathbf{x}}_n - \hat{\mathbf{x}}_n$

  Solve QP subproblem Equation (5.52) for $\delta\mathbf{u}_n$

  $\mathbf{u}_n^* := \hat{\mathbf{u}}_n + \delta\mathbf{u}_n$

  $\bar{\mathbf{x}}_{n+1} = \sum_j \big(P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n^*, w^j)$

end

Output: $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N} := (\bar{\mathbf{x}}_n, \mathbf{u}_n^*)_{n=0}^{N}$

5.3.3 Average Path Differential Dynamic Programming (APDDP)

We now develop *average path differential dynamic programming* (APDDP) to approximately solve Equation (5.1). The problem is re-formulated as

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \sum_{n=0}^{N-1} g_n(\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{u}_n, \bar{w}_n)\ \mathbf{x}_0, w_0 \tag{5.54a}$$

$$\text{subject to}\quad \bar{w}_n = \sum_j P_{ij}^{(n)} w^j \tag{5.54b}$$

$$\mathbf{x}_{n+1} = F_n(\mathbf{x}_n, \mathbf{u}_n, \bar{w}_n) \tag{5.54c}$$

$$\bar{\mathbf{x}}_{n+1} = \sum_j P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n, w^j) \tag{5.54d}$$

$$D_x \bar{\mathbf{x}}_{n+1} \leq \mathbf{c}_x \tag{5.54e}$$

$$D_u \mathbf{u}_n \leq \mathbf{c}_u \tag{5.54f}$$

Average path differential dynamic programming is identical to the ASDDP method described in Section 5.3.2 except the state-control value function is constructed as

$$Q_n(\mathbf{x}_n, \mathbf{u}_n) = g_n(\mathbf{x}_n, \mathbf{u}_n, \bar{w}_n) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, \bar{w}_n)\big) \tag{5.55}$$

where the average disturbance path is defined as

$$\bar{w}_n = \sum_j P_{ij}^{(n)} w^j \tag{5.56}$$

Compared to ASDDP, the primary benefit with APDDP is a significant reduction in computational burden since the summations $\sum_j P_{ij}^{(n)}$ associated with stochastic computations are nearly eliminated during the backward pass. Through numerical experimentation it was found that APDDP had trouble meeting driver demand when using the same calibrations from ASDDP (i.e. K_3 from Equation (5.11) and α, β from Equation (5.16)). This is likely due to the fact that whereas ASDDP is evaluating all possible values of the disturbance $w_n = w^j, j \in W$ during creation of the state-control

value function Equation (5.42), APDDP only evaluates the average value $\bar{w}_n$ during creation of the state-control value function Equation (5.55). As a result, APDDP will ignore the impact of disturbance values which deviate from the averaged disturbance value along the horizon. To remedy this, gains K_3, α, and β were increased until APDDP was able to satisfy driver demands. Meeting driver demand is discussed further in a quantitative manner in Section 6.3. The APDDP algorithm is summarized in Algorithm 3.

5.3.4 Block Diagram of Stochastic Control Algorithms

The implementation of SGDM, ASDDP, and APDDP is shown in Fig. 5.4. Each

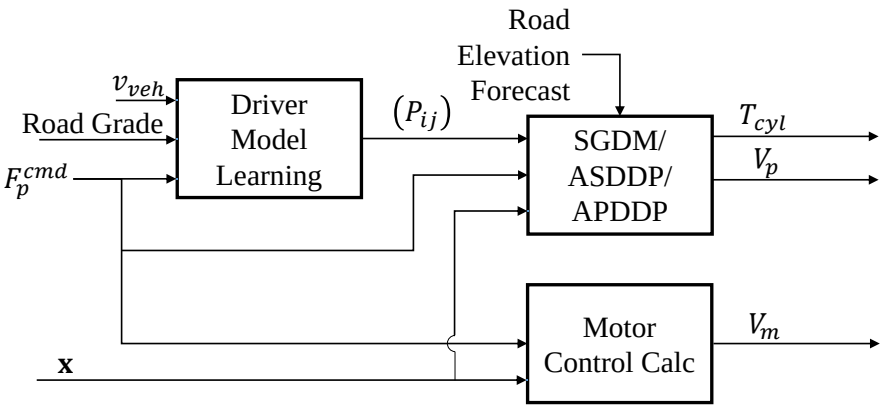

Fig. 5.4. Stochastic algorithm block diagram.

of these algorithms relies on the learned statistical model of driver behavior (P_{ij}) to form decisions along the horizon $n = 0, 1, \ldots, N-1$. The sequence $(\mathbf{x}_n^*, \mathbf{u}_n^*)_{n=0}^{N-1}$ is recomputed every T_s seconds. The motor displacement volume, V_m, is updated according to Equation (3.16). Using the scaling factors of Equation (5.5), the inputs T_{cyl} and V_p are formed using the first element from the control sequence

$$\begin{bmatrix} T_{cyl} \\ V_p \end{bmatrix} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \mathbf{u}_0^* \tag{5.57}$$

The driver model learning process is described by Equations (4.6) and (4.1), motor displacement volume calculation is given by Equation (3.16).

Algorithm 3: APDDP

Input: $\mathbf{x}_0, w_0, (\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N}$

$\hat{\mathbf{x}}_0 := \mathbf{x}_0, w^i := w_0$

——*Backward Pass*——

$\{Q_n^{(x)}, Q_n^{(u)}, Q_n^{(xx)}, Q_n^{(uu)}, Q_n^{(ux)}\} = 0$

for $n = N-1 : 0$ **do**

- $\bar{w}_n = \sum_j \left(P_{ij}^{(n)} w^j \right.$
- $\mathbf{x}_{n+1} = F_n(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n, \bar{w}_n)$
- **if** *n=N-1* **then**
  - $V_{n+1}^{(x)} = h^{(x)}(\mathbf{x}_{n+1}), \quad V_{n+1}^{(xx)} = h^{(xx)}(\mathbf{x}_{n+1})$
- **end**
- **else**
  - $V_{n+1}^{(x)} = A + B\big[\mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1}\big], \;\; V_{n+1}^{(xx)} = B$
- **end**
- $Q_n^{(x)} = g_n^{(x)} + V_{n+1}^{(x)} F_n^{(x)}$
- $Q_n^{(u)} = g_n^{(u)} + V_{n+1}^{(x)} F_n^{(u)}$
- $Q_n^{(xx)} = g_n^{(xx)} + F_n^{(x)T} V_{n+1}^{(xx)} F_n^{(x)}$
- $Q_n^{(uu)} = g_n^{(uu)} + F_n^{(u)T} V_{n+1}^{(xx)} F_n^{(u)}$
- $Q_n^{(ux)} = g_n^{(ux)} + F_n^{(u)T} V_{n+1}^{(xx)} F_n^{(x)}$
- $Q_n^{(xu)} = Q_n^{(ux)T}$
- Modify $Q_n^{(uu)}$ according to Equation (5.53)
- $A = Q_n^{(x)} - Q_n^{(u)}\big[Q_n^{(uu)}\big]^{-1} Q_n^{(ux)}$, $B = Q_n^{(xx)} - Q_n^{(xu)}\big[Q_n^{(uu)}\big]^{-1} Q_n^{(ux)}$

end

——*Forward Pass*——

for $n = 0 : N-1$ **do**

- $\delta \mathbf{x}_n := \mathbf{x}_n - \hat{\mathbf{x}}_n$
- Solve QP subproblem Equation (5.52) for $\delta \mathbf{u}_n$
- $\mathbf{u}_n^* := \hat{\mathbf{u}}_n + \delta \mathbf{u}_n$
- $\bar{\mathbf{x}}_{n+1} = \sum_j \left(P_{ij}^{(n)} F_n(\bar{\mathbf{x}}_n, \mathbf{u}_n^*, w^j) \right.$

end

Output: $(\hat{\mathbf{x}}_n, \hat{\mathbf{u}}_n)_{n=0}^{N} := (\mathbf{x}_n, \mathbf{u}_n^*)_{n=0}^{N}$

5.4 Benchmark Strategies

Two benchmark strategies are provided as a means to evaluate SGDM, ASDDP, and APDDP. First, a baseline strategy based on instantaneous optimization is representative of that which can be achieved without consideration of upcoming driver demands or road elevation. Second, a theoretical best strategy is created to demonstrate the best which can be achieved when all cycle information available is provided to the decision making process. Like SGDM, ASDDP, and APDDP, the baseline strategy is implementable as a real time control algorithm, whereas the theoretical best strategy is not.

5.4.1 Baseline: Instantaneous Optimization

A baseline strategy based on instantaneous optimization (InstOpt) is created, similar to that developed in [1]. The control inputs are generated to minimize the instantaneous fuel consumption rate considering current operating conditions and neglecting the effect of future driver demands and road elevation. The strategy is described in Fig. 5.5. Pump displacement volume is controlled according to a proportional-integral (PI) controller processes to maintain some minimum pressure in the accumulator denoted as p_{ref}. This minimum pressure reference is held fixed at some nominal value and gradually raised if the driver propulsion force demand is not satisfied. The engine is managed to deliver the minimum speed that can satisfy the power demanded by the pump. If the accumulator pressure falls to some level ϵ below p_{ref}, engine speed may be commanded to increase according to a limited PI controller process. A minimum engine speed is set so that the pump can always provide enough flow to satisfy the motor flow demand, unless pump displacement volume is zero in which case this flow-based engine speed command is zero. The motor displacement is controlled according to Equation (3.16). Parameters of the baseline strategy were iteratively calibrated so the strategy performed well on all three drive cycles, with emphasis placed on performance under the UDDS drive cycle. Once established, these

parameters were unchanged from one cycle to the next. The reference pressure was set to 150 bar, with precharge pressure set to 135 bar (90% of the reference pressure). Justification for the 150 bar reference pressure is established with Fig. 6.1 in Section 6.

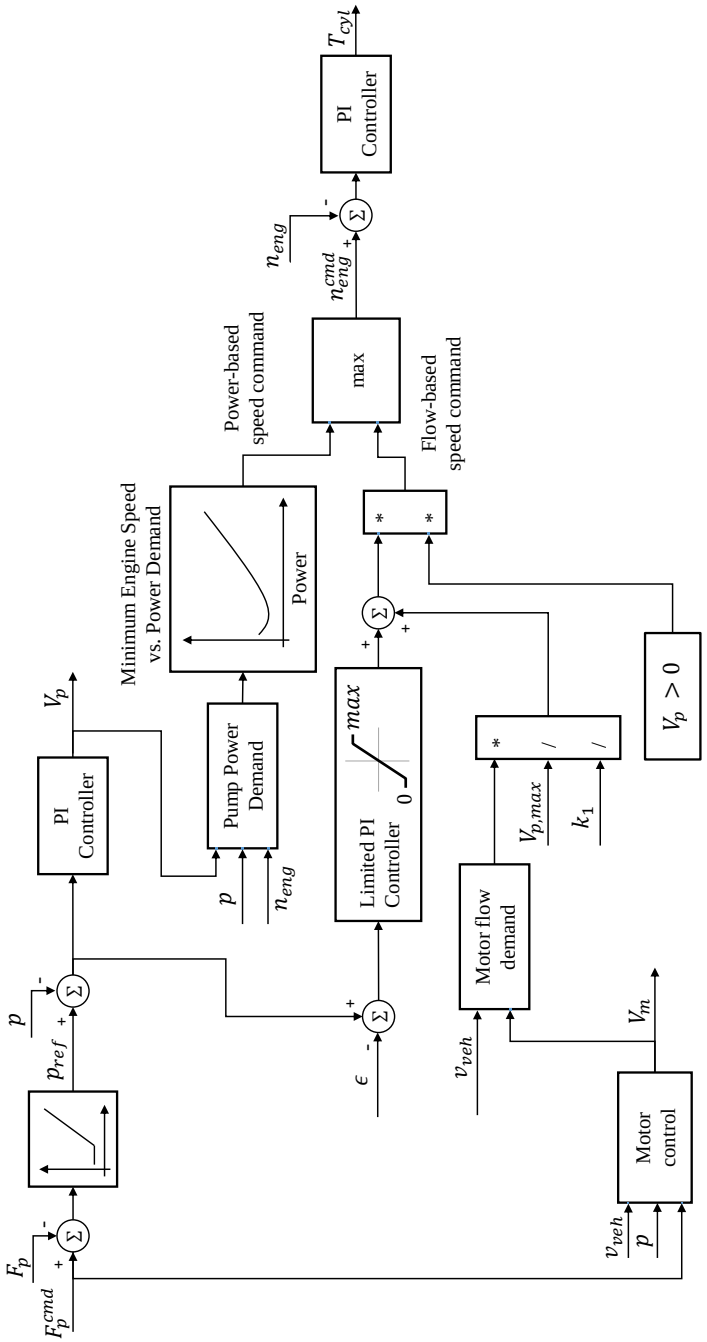

Fig. 5.5. Instantaneous optimization strategy (InstOpt).

5.4.2 Theoretical Best: Deterministic Differential Dynamic Programming with Driver Forecast

The classic (deterministic) differential dynamic programming algorithm discussed in Section 2.1.4 is used to generate a theoretically best controller to serve as a basis for comparison. The implementation of *DDP with driver forecast* (DDP for short) is shown in Fig. 5.6. Unlike the stochastic algorithms discussed in Section 5.3, DDP

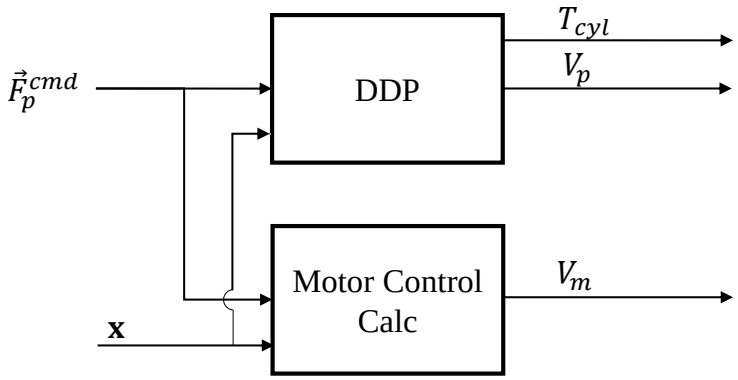

Fig. 5.6. Theoretial best strategy: DDP with driver forecast.

has full access to the propulsion force command sequence along the horizon, $\vec{F}_p^{cmd}$. Consequently, the DDP algorithm is not actually implementable in practice. The values for T_{cyl} and V_p are generated every T_s seconds according to Equation (5.57). The value for V_m is updated every 0.01 seconds according to Equation (3.16).

6. SIMULATION

Simulation is performed in Matlab Simulink for the series-hybrid configuration shown in Fig. 3.5. A mid-size sport utility vehicle is simulated with parameters shown in Table 6.1. The system is designed so that maximum propulsion force, F_p^{max}, can be achieved when differential system pressure is 290 bar when the vehicle is in low gear. The distribution of driver propulsion force command for each of the cycles

Table 6.1. Series-Hybrid SUV Parameters.

Description	**Symbol**	**Value**	**Units**
Vehicle mass	m_{veh}	2091	kg
Max eng. power	P_{eng}^{max}	125	kW
Max propulsion force	F_p^{max}	6500	N
Max vehicle speed	v_{veh}^{max}	125	km/h
Dynamic tire radius	r_{tire}	0.35	-
Aero drag coefficient	C_d	1.62	-
Rolling resistance coefficient	C_r	0.010	-
Engine inertia	I_{eng}	0.5	kg·m^2
Gear ratio 1	k_1	1	-
Gear ratio 2: lo, hi	$k_{2,lo}, k_{2,hi}$	10, 6.67	-
Gear ratio 2 lo/hi thresh	$v_{veh,hi}$	20	m/s
Displacement vol. of hyd. pump	V_p^{max}	63	cc/rev
Displacement vol. of hyd. motor	V_m^{max}	50	cc/rev
Hyd. accumulator precharge vol.	V_{ha}	50	L
Hyd. accumulator precharge press.	p_{ha}	70	bar
Max differential system press.	p_{max}	350	bar
Low-pressure accum press.	p_{lp}	10	bar

investigated is shown in Fig. 6.1. This distribution indicates the fraction of time the driver spends commanding various levels of propulsion force. For example, in the UDDS cycle the driver commands a propulsion force between -500 and 500N for approximately 55% of the cycle. At the far extreme a propulsion force between 5500

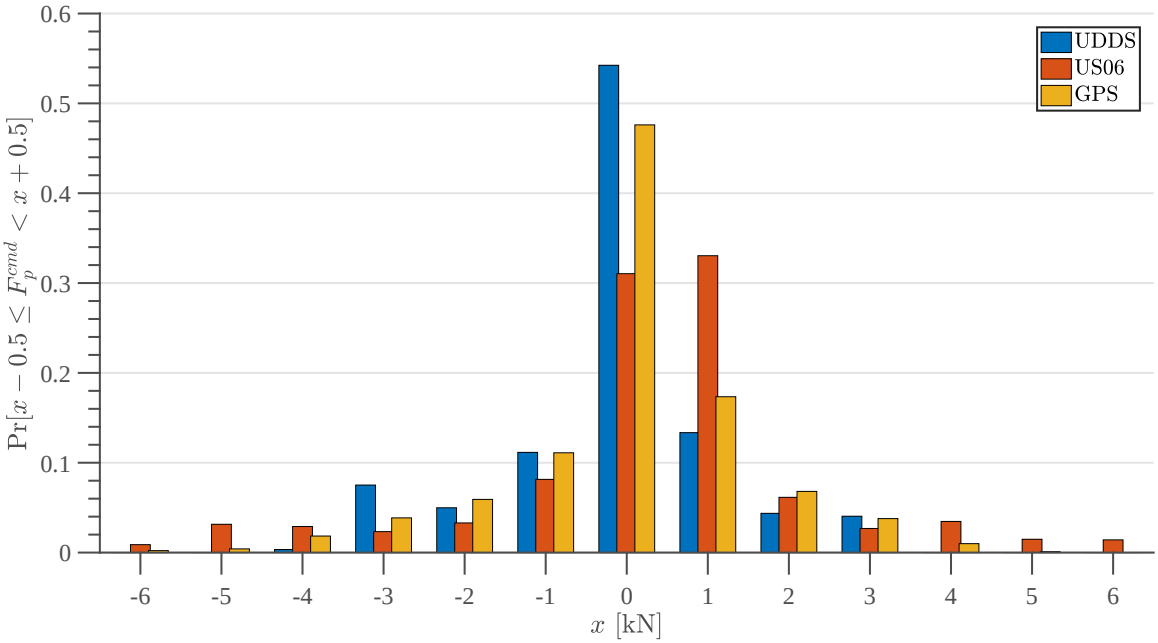

Fig. 6.1. Driver propulsion force command distribution for each drive cycle.

and 6500N is requested during the US06 cycle for approximately 1.4% of the cycle (8.4 seconds). Recall that the reference differential system pressure for the baseline strategy InstOpt is $p_{ref} = 150$ bar, so that a 3500N propulsion force can be generated in low gear at the reference pressure. Referring to Fig. 6.1, a propulsion force of 3500N covers the majority of driving demands for the cycles investigated. When a propulsion force greater than 3500N is commanded, the baseline strategy will need to increase the differential system pressure as described in Fig. 5.5.

6.1 Simulation Setup

The simulation configuration is shown in Fig. 6.2. The vehicle dynamics block contains the engine, vehicle and hydraulics dynamics described in Section 3.2. The algorithm block contains the embedded system model described in Section 5.1 and one of the algorithms described in Chapter 5 (either SGDM, ASDDP, APDDP, DDP, or InstOpt). The road elevation forecast block described in Section 5.2 provides elevation information along the horizon. The SGDM, ASDDP, APDDP and DDP

algorithms generate control inputs T_{cyl} and V_p every $T_s = 0.1$ seconds and input V_m every 0.01 seconds. InstOpt generates all three contorl inputs every 0.01 seconds.

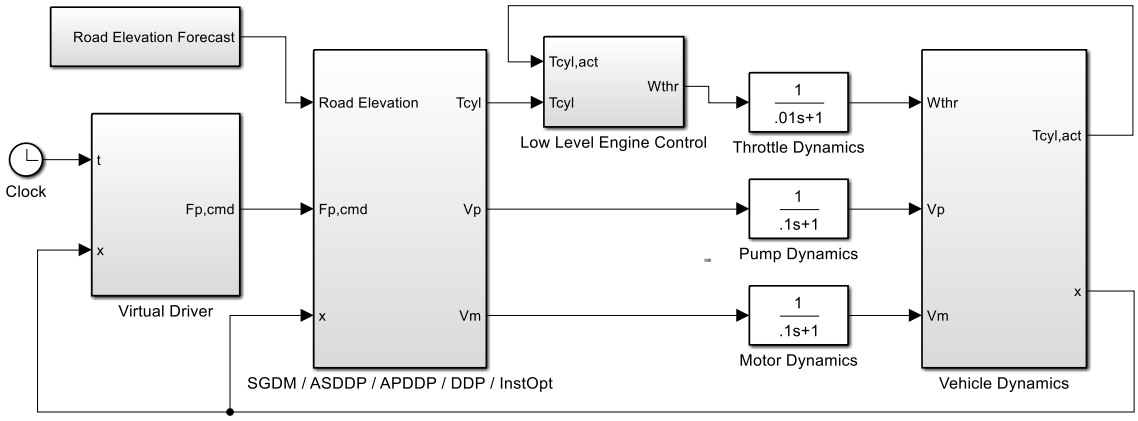

Fig. 6.2. Stochastic algorithm block diagram.

A virtual driver is created which generates propulsion force command F_p^{cmd} along the three drive cycles of Fig. 4.2. The virtual driver is a PI controller which tracks the drive cycle's reference vehicle speed v_{veh}^{ref} according to

$$F_p^{cmd}(t) = k_p \left(v_{veh}^{ref}(t) - v_{veh}(t) \right) + \int_0^t k_i \left(v_{veh}^{ref}(\tau) - v_{veh}(\tau) \right) d\tau \tag{6.1}$$

The gains k_p and k_i were tuned so that even a small speed tracking error $v_{veh}^{ref} - v_{veh}$ results in a large propulsion force command. To ensure excellent speed tracking for all three cycles the penalty K_3 from cost rate function Equation (5.11) is made large so that the tracking of F_p^{cmd} is also excellent, as will be shown.

In the low level engine control block the cylinder torque control input, T_{cyl}, is converted into an engine throttle mass flow command, W_{thr}, though a simple PI controller

$$W_{thr}(t) = k_p \left(T_{cyl}(t) - T_{cyl,act}(t) \right) + \int_0^t k_i \left(T_{cyl}(\tau) - T_{cyl,act}(\tau) \right) d\tau \tag{6.2}$$

6.2 Cycle Analysis

In this section some results of the SGDM, ASDDP, DDP and InstOpt algorithms are compared qualitatively. Reference speed tracking and state / control trajectories are examined.

6.2.1 UDDS Cycle

A segment of the UDDS drive cycle is shown in Fig. 6.3. This segment corresponds

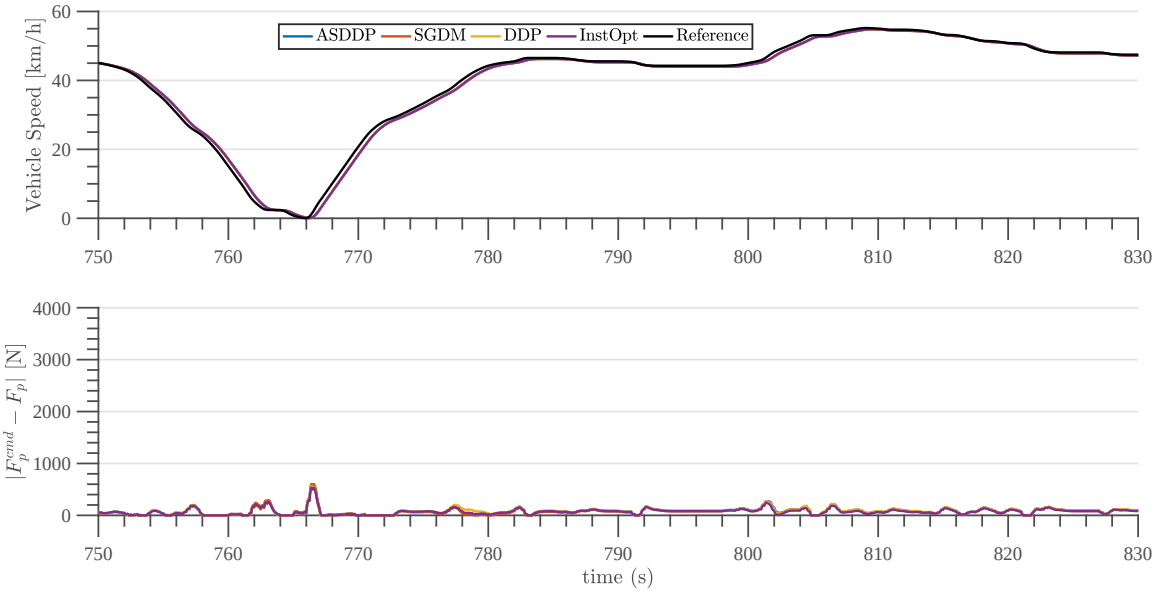

Fig. 6.3. Segment of UDDS Cycle.

to the driver just finishing a sequence of stop and go driving and beginning a phase of cruising at moderate speed. The speed and propulsion force tracking are excellent under all four algorithms.

State and control input trajectories are shown in Fig. 6.4. The stochastic strategies (SGDM and ASDDP) keep differential system pressure higher during the stop and go driving segment when acceleration demands become large, then lower differential system pressure once the cruising segment begins. The DDP with driver forecast strategy (DDP), which can foresee upcoming acceleration demands, only raises sys-

tem pressure briefly to meet the strong acceleration demand near time $t = 765$ s. The baseline strategy based on instantaneous optimization (InstOpt) raises engine speed and differential system pressure in a pattern which is somewhat similar to SGDM and ASDDP. However, it can be seen that the stochastic strategies have an advantage in that differential system pressure is allowed to drop down as low as 100 bar during the cruising phase where higher pressures are not required (thereby resulting in higher hydraulic displacement volumes and overall improved efficiency). Comparing the two stochastic strategies, ASDDP tends to adjust T_{cyl} and V_p more rapidly than SGDM, perhaps indicating that ASDDP converges more quickly than SGDM.

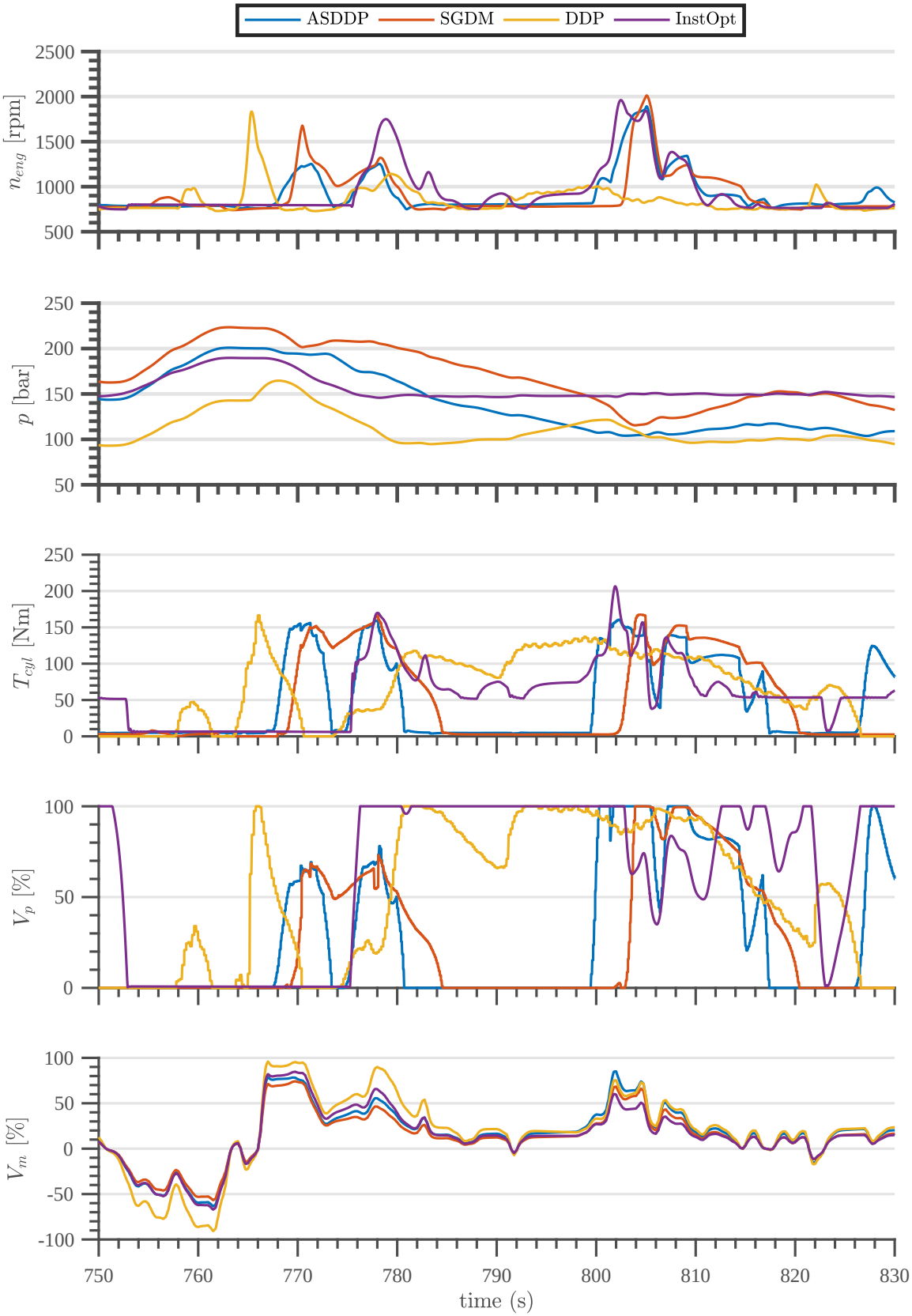

Fig. 6.4. State and control trajectories over segment of UDDS Cycle.

6.2.2 US06 Cycle

A segment of the aggressive US06 drive cycle is shown in Fig. 6.5. This segment corresponds to aggressive accelerations near the start of the cycle.

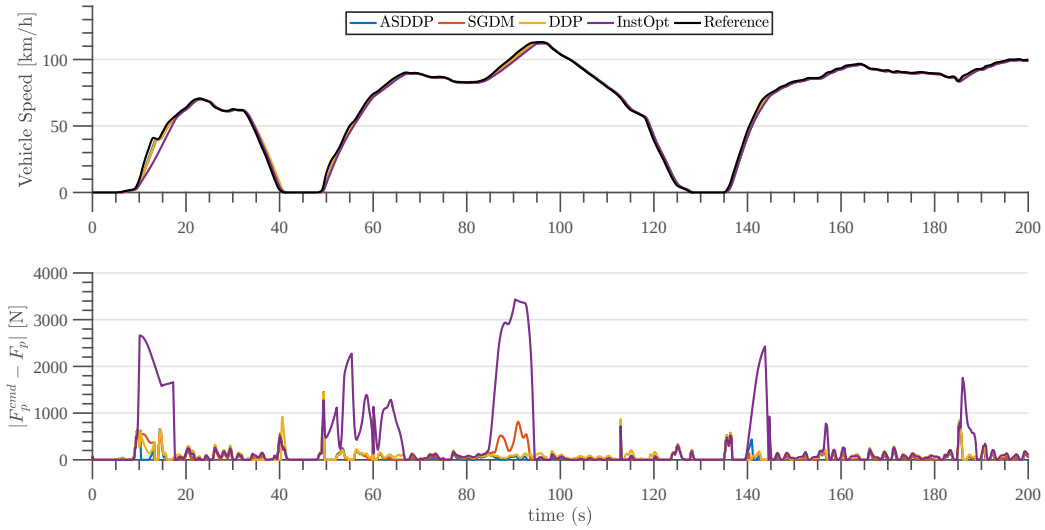

Fig. 6.5. Segment of US06 Cycle.

The speed tracking performance of each algorithm is very good, with the exception of InstOpt. Large differences between the commanded and actual propulsion force are seen under InstOpt, indicating difficulty meeting the driver demand. The situation becomes more apparent when the trajectories of engine speed and differential system pressure are examined, shown in Fig. 6.6. It is interesting to note that SGDM, ASDDP, and DDP increase the differential system pressure just before the start of the aggressive acceleration event near time $t = 10$ seconds. In this way, SGDM, ASDDP and DDP are well positioned to accommodate the driver's aggressive acceleration demand. The InstOpt strategy, which is provided no information regarding upcoming behavior, maintains differential system pressure at the minimum 150 bar until just before $t = 10$ seconds. Near $t = 10$ seconds, InstOpt rapidly increases T_{cyl} and V_p in an attempt to meet the driver demand.

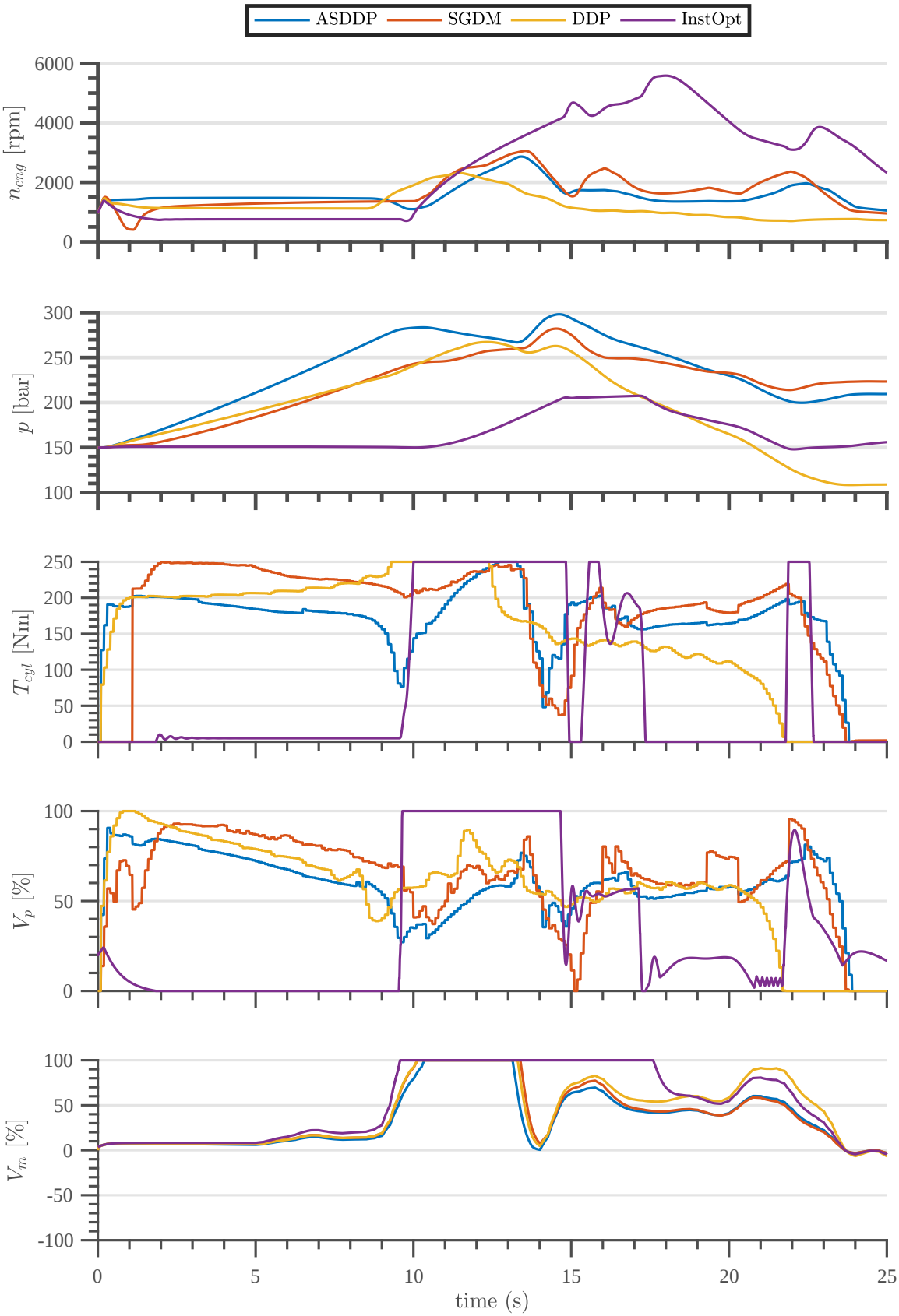

Fig. 6.6. State and control trajectories over segment of US06 Cycle.

6.2.3 GPS Cycle

A segment of the GPS drive cycle is shown in Fig. 6.7. This segment corresponds

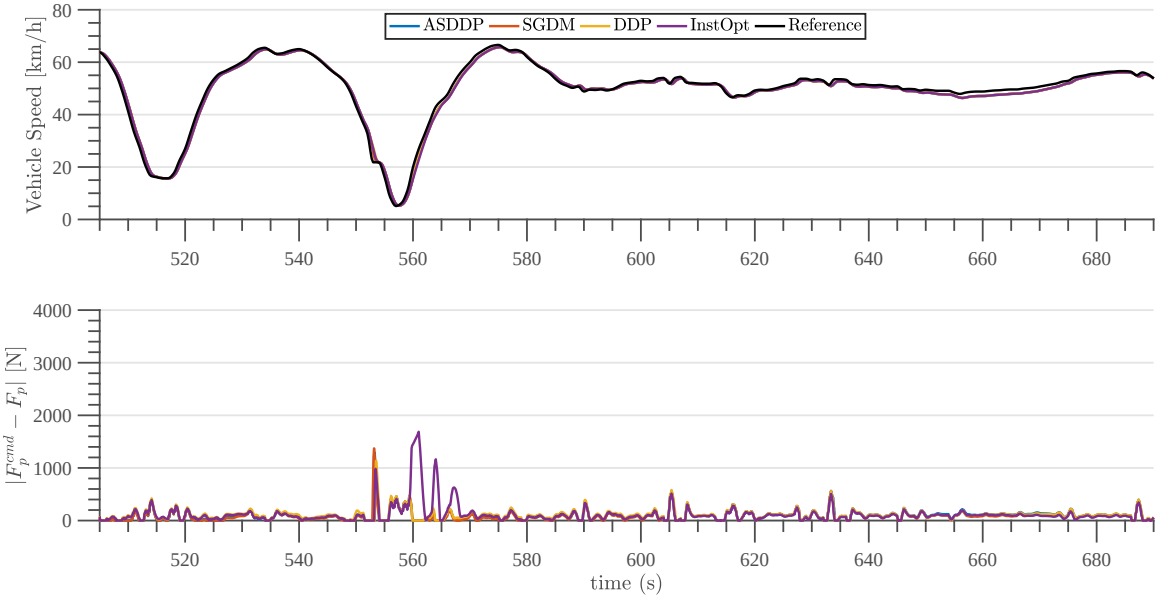

Fig. 6.7. Segment of GPS Cycle.

to the driver just finishing a sequence of stop and go driving and beginning a phase of cruising at moderate speed. Trajectories of engine speed and differential system pressure are shown in Fig. 6.8. SGDM and ASDDP tend to keep differential system pressure higher during stop and go driving, then lowering differential system pressure during the cruising phase. Interestingly, ASDDP generates engine speed and differential system pressure trajectories which nearly match DDP during the cruising phase.

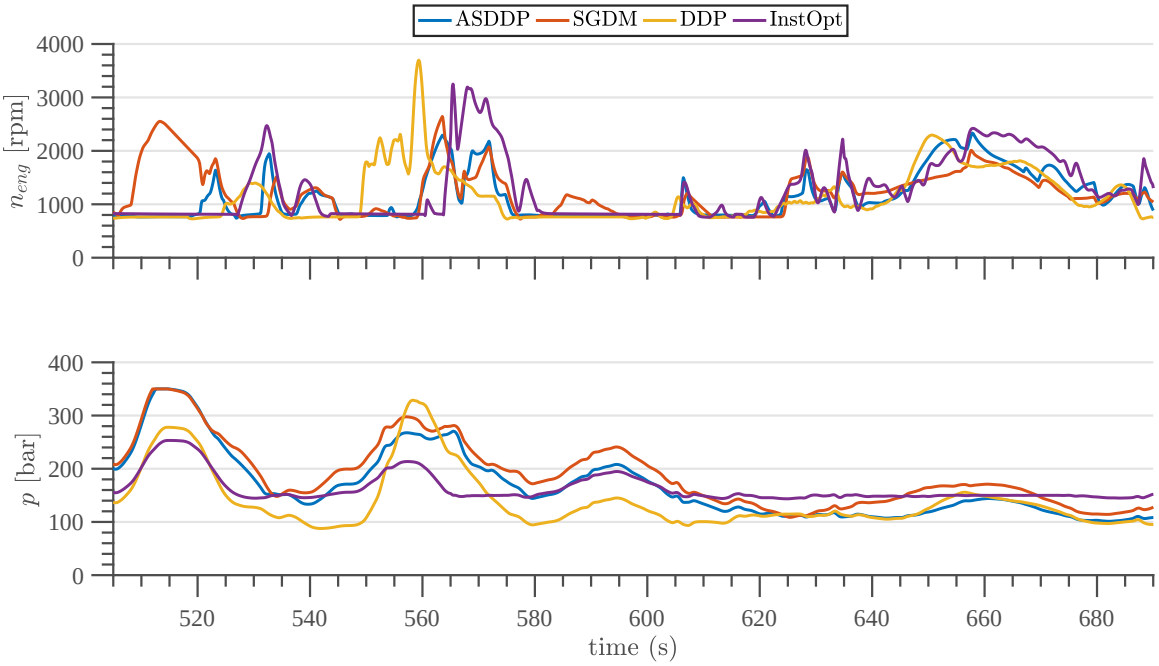

Fig. 6.8. Engine speed and differential system pressure over segment of GPS Cycle.

6.3 Performance Metrics

To evaluate the performance of each controller quantitatively two metrics are defined. The first metric is simply the fuel consumed along the entire cycle

$$\text{Fuel Consumption} = \int_0^T b_f(n_{eng}(t), T_{cyl}(t))\ dt \tag{6.3}$$

where b_f is the fuel consumption rate of the engine described in Fig. 3.8. The second metric indicates how well the driver demand is met along the cycle through a modified speed tracking integral

$$\text{Tracking Metric} = \frac{1}{\text{cycle dist [km]}} \int_0^T v_{veh}^{ref}(t) - v_{veh}(t)\ \times \mathbb{1}_{V_m(t)=V_m^{max}}\ dt \tag{6.4}$$

where the indicator function is given by

$$\mathbb{1}_{V_m(t)=V_m^{max}} = \begin{cases} 1 & \text{if } V_m(t) = V_m^{max} \\ 0 & \text{otherwise} \end{cases}$$

Recall in Section 3.2 it was shown that the propulsion force is limited by the differential system pressure. The tracking metric ultimately measures how well a particular controller can anticipate and/or react to the propulsion force commanded by the driver by properly managing the differential system pressure along the drive cycle. The units of the tracking metric are meters per kilometer, measuring the average distance in meters the vehicle has regressed from the reference cycle per kilometer as a result of insufficient differential system pressure. The inclusion of the indicator function in the tracking metric definition reduces sensitivity to the virtual driver controller gains described in Equation (6.1). A lower tracking metric score indicates better performance. A score of 0 - 2 m/km indicates that driver demand is (nearly) perfectly met along the entire drive cycle. A score much greater than 4 m/km (a score of 4 m/km is equivalent to one car length per kilometer) may indicate noticeable discrepancies between commanded and produced propulsion force.

6.3.1 Learning Progression

This section investigates how well SGDM, ASDDP, and APDDP progressively optimize fuel usage and drivability as each cycle is repeated. Each row of driver model (P_{ij}) is initialized to a Gaussian-like distribution, centered around w^i. On each subsequent run (P_{ij}) is adapted to the driver behavior as described in Section 4.2. At the end of each run the elements of (P_{ij}) are stored in memory and then used as the initial conditions for the following run.

Learning progression under the UDDS cycle is shown in Fig. 6.9. The results from the DDP and InstOpt benchmark strategies are also plotted, but since these strategies do not adapt to driver behavior their performance metrics are constant across the cycle

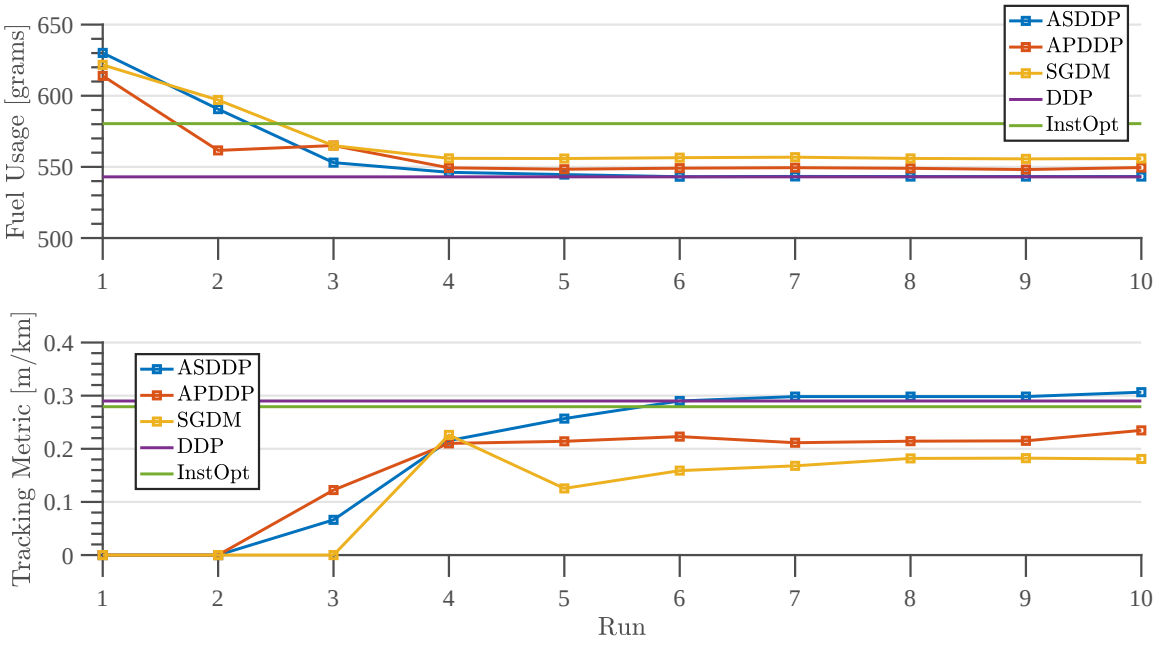

Fig. 6.9. UDDS cycle metrics.

runs. As (P_{ij}) is adapted to the UDDS drive cycle, fuel usage improves quickly while the tracking metric is increased only slightly (note the scale of the tracking metric). Interestingly, convergence for both algorithms has nearly been achieved by the end of the fourth run. Learning progression under the GPS and US06 cycles are shown in Figs. 6.10 and 6.11. As with the UDDS cycle, convergence has nearly occurred after the second or third run.

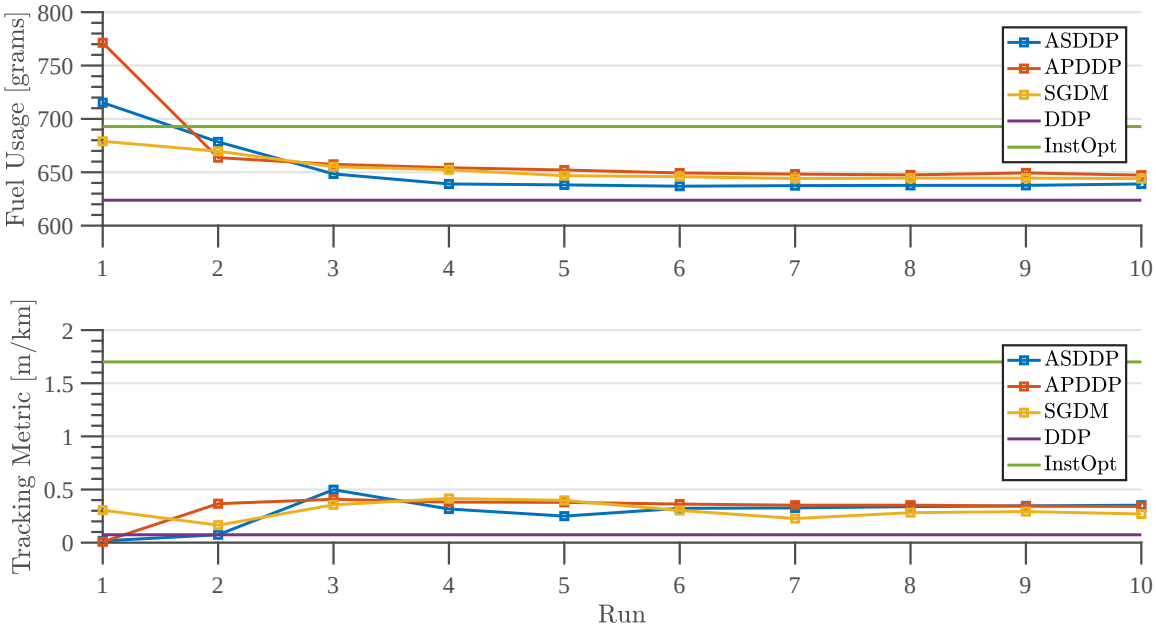

Fig. 6.10. GPS cycle metrics.

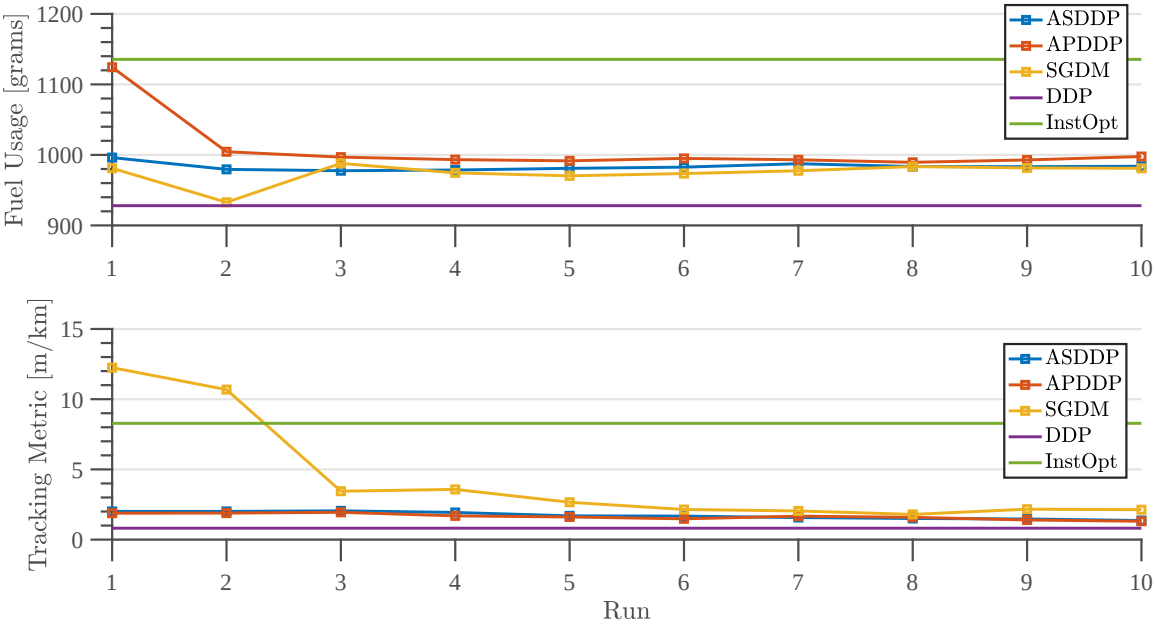

Fig. 6.11. US06 cycle metrics.

The final fuel usage and tracking metric results after 10 repeated runs of each cycle are tabulated in Tables 6.2 and 6.3.

Table 6.2. Fuel usage results, percent relative to DDP.

Cycle / Alg	UDDS	US06	GPS Cycle	GPS Cycle (without rd. gd. forecast)
DDP	100.0%	100.0%	100.0%	-
SGDM	102.4%	106.3%	102.2%	103.4%
ASDDP	100.0%	105.8%	102.3%	104.8%
APDDP	101.2%	107.3%	103.8%	104.2%
InstOpt	106.9%	122.4%	111.1%	-

Table 6.3. Tracking metric results [m/km].

Cycle / Alg	UDDS	US06	GPS Cycle
DDP	0.29	0.81	0.07
SGDM	0.18	2.02	0.29
ASDDP	0.31	1.36	0.36
APDDP	0.23	1.31	0.34
InstOpt	0.28	8.28	1.70

6.3.2 Cross Training

To better understand the benefit of learning cycle-specific driver behavior, a cross training simulation is performed where each cycle is repeatedly run as in the previous section, but the statistical driver model (P_{ij}) is initialized on statistics obtained from other cycles. The same metrics from the previous section are examined. In order to simplify the presentation, only the results from the ASDDP and APDDP algorithms are shown. The results from DDP and InstOpt are also included as reference points. The progression of the fuel usage and tracking metrics and shown over six runs. On run zero the driver behavior learning mechanism is frozen so that the effect of running any given cycle on statistics learned from repeatedly running another cycle is determined. After run zero is complete the driver behavior learning mechanism is allowed to run as normal.

The cross trained simulation results for the UDDS cycle are shown in Fig. 6.12. The blue curves show ASDDP results obtained by initializing (P_{ij}) with driver statistics obtained from the GPS and US06 cycles. Likewise, the red curves show APDDP

results obtained in a similar manner. Interestingly, when (P_{ij}) is initialized with US06 statistics (dashed curves) the InstOpt outperforms the ASDDP strategy in terms of fuel usage until during the second run of the UDDS cycle (22-45 minutes) in which driver learning is active. Similarly, InstOpt outperforms APDDP fuel usage until during the third run of UDDS (45-67 minutes) in which driver learning is active. This result highlights the importance of adapting to relevant statistics if a stochastic strategy is to be employed.

The cross trained simulation results for the US06 cycle are shown in Fig. 6.13. Fuel usage results remain relatively constant across the six runs. However, the tracking metric improves significantly after the first run of the US06 cycle in which driver learning is active (10 minutes). Cross trained results from the GPS cycle are shown in Fig. 6.14. Regardless of (P_{ij}) initialization ASDDP and APDDP outperform InstOpt during the first run in which driver learning is active.

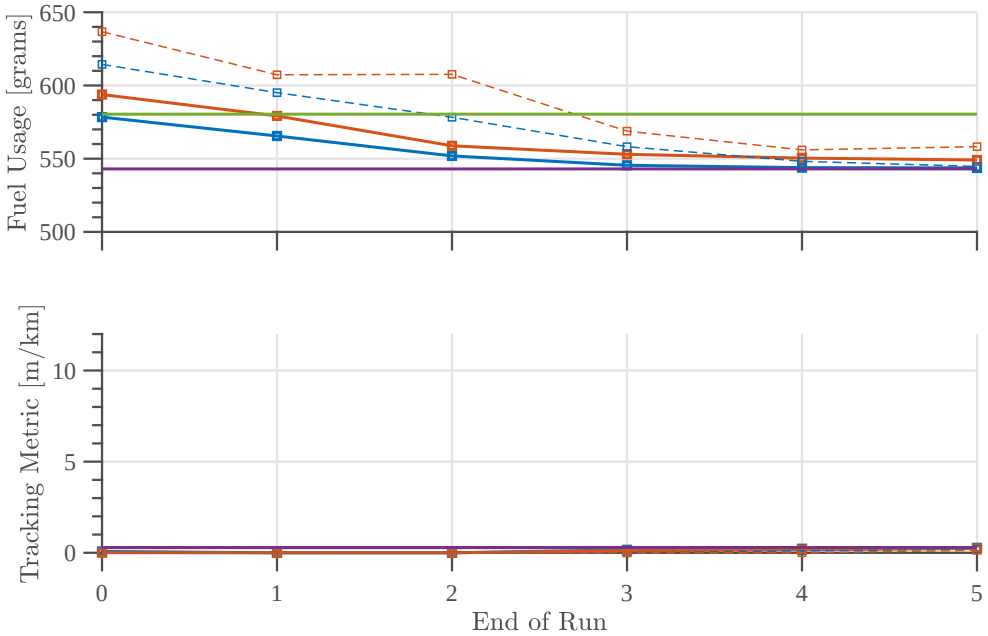

Fig. 6.12. UDDS cycle cross training metrics. Blue: ASDDP using stats from GPS (solid), US06 (dashed). Red: APDDP using stats from GPS (solid), US06 (dashed). Purple: DDP and Green: InstOpt.

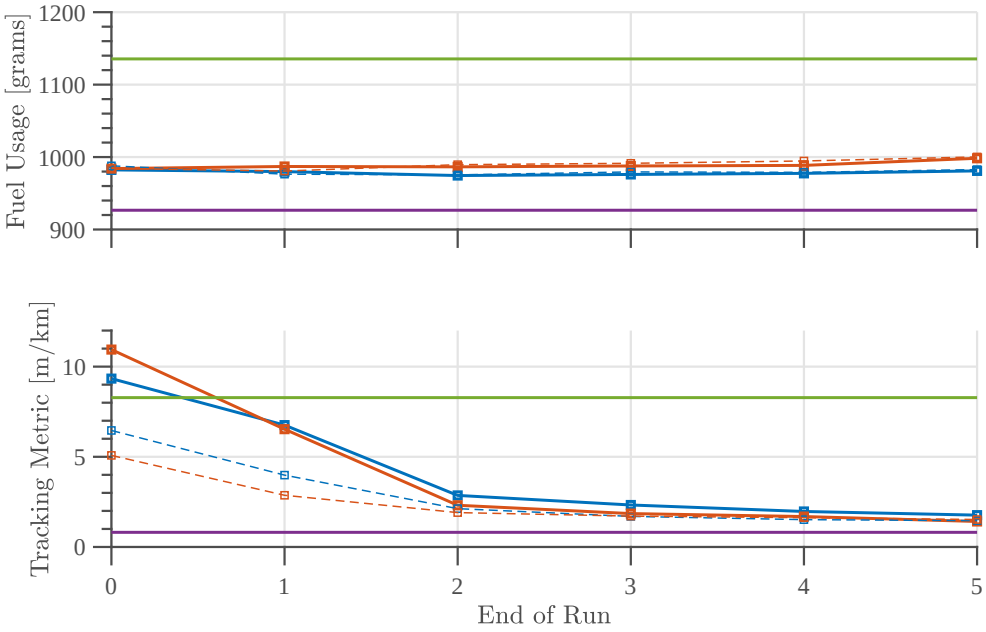

Fig. 6.13. US06 cycle cross training metrics. Blue: ASDDP using stats from UDDS (solid), GPS (dashed). Red: APDDP using stats from UDDS (solid), GPS (dashed). Purple: DDP and Green: InstOpt.

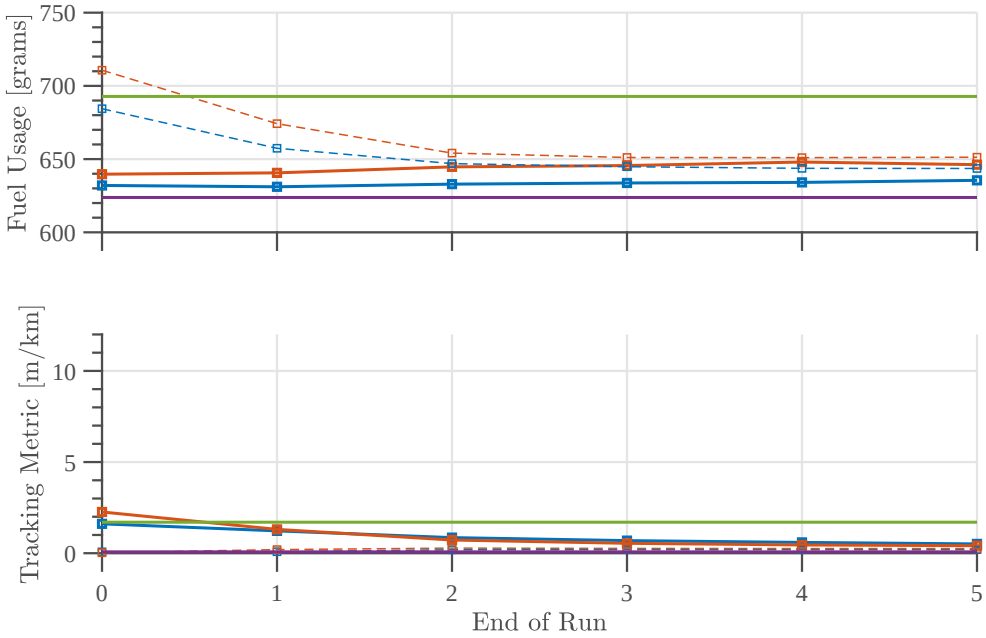

Fig. 6.14. GPS cycle cross training metrics. Blue: ASDDP using stats from UDDS (solid), US06 (dashed). Red: APDDP using stats from UDDS (solid), US06 (dashed). Purple: DDP and Green: InstOpt.

6.4 Computation Times

The average computation times of the three stochastic algorithms are shown in Table 6.4. The values indicate how much faster than real time each algorithm executes. These values were obtained by running each algorithm in the full simulation setup shown in Fig. 6.2 and comparing the simulation run time to elapsed wall-clock time. The simulations were carried out on a laptop equipped with a 2.6 GHz i7 processor. ASDDP runs nearly twice as fast as SGDM, and the APDDP runs nealy five times faster than ASDDP. The massive increase in speed associated with APDDP over the other algorithms can be attributed to the fact that APDDP is not considering the true stochasticity of the problem, resulting in a significantly reduced computational burden.

Table 6.4. Computation times.

Algorithm	**Average sim:real time**
SGDM	3.4:1
ASDDP	7:1
APDDP	34:1

7. EXPERIMENT

An experimental setup is used to demonstrate the real time potential of the ASDDP algorithm on a processor with limited computational resources. A secondary objective is to demonstrate a model predictive control approach can successfully control a series hydraulic hybrid using a simplified control-oriented model of the real physics.

7.1 Experimental Hardware

The series hybrid test rig at the Maha Fluid Power Research Center is shown in Fig. 7.1. An electric motor, referred to as the engine simulator, is directly connected to a hydraulic pump, unit 1. The engine simulator is a 126 kW Schenck three phase induction motor, capable of providing a 300 Nm torque at 4000 RPM. Hydraulic unit 1 is a Sauer S90 42 cc/rev variable displacement swash plate type pump. An electric motor/generator, referred to as the load simulator, is used to simulate vehicle inertia and road load. The load simulator is a 186 kW Reliance motor, capable of producing a 500 Nm torque at 3600 rpm. A second hydraulic pump/motor is connected directly to the load simulator, referred to as unit 2. Hydraulic unit 2 is a Sauer S90 75 cc/rev variable displacement swash plate type pump. The engine and load simulators are coupled to ABB manufactured ACS800 variable frequency drives. These drives control the output frequency which facilitates a control over the speed and torque of the two simulators. The ABB drives have transient and steady state speed control accuracy better than 0.1 %. A hydraulic power supply pressurizes a low pressure line to replace leakage losses, and an accumulator is connected to the high pressure line for energy recovery. Data acquisition and control was conducted using the cRIO 9074 controller, a product by National Instruments. The cRIO 9704 has a single core 400 MHz processor and 128 MB of RAM.

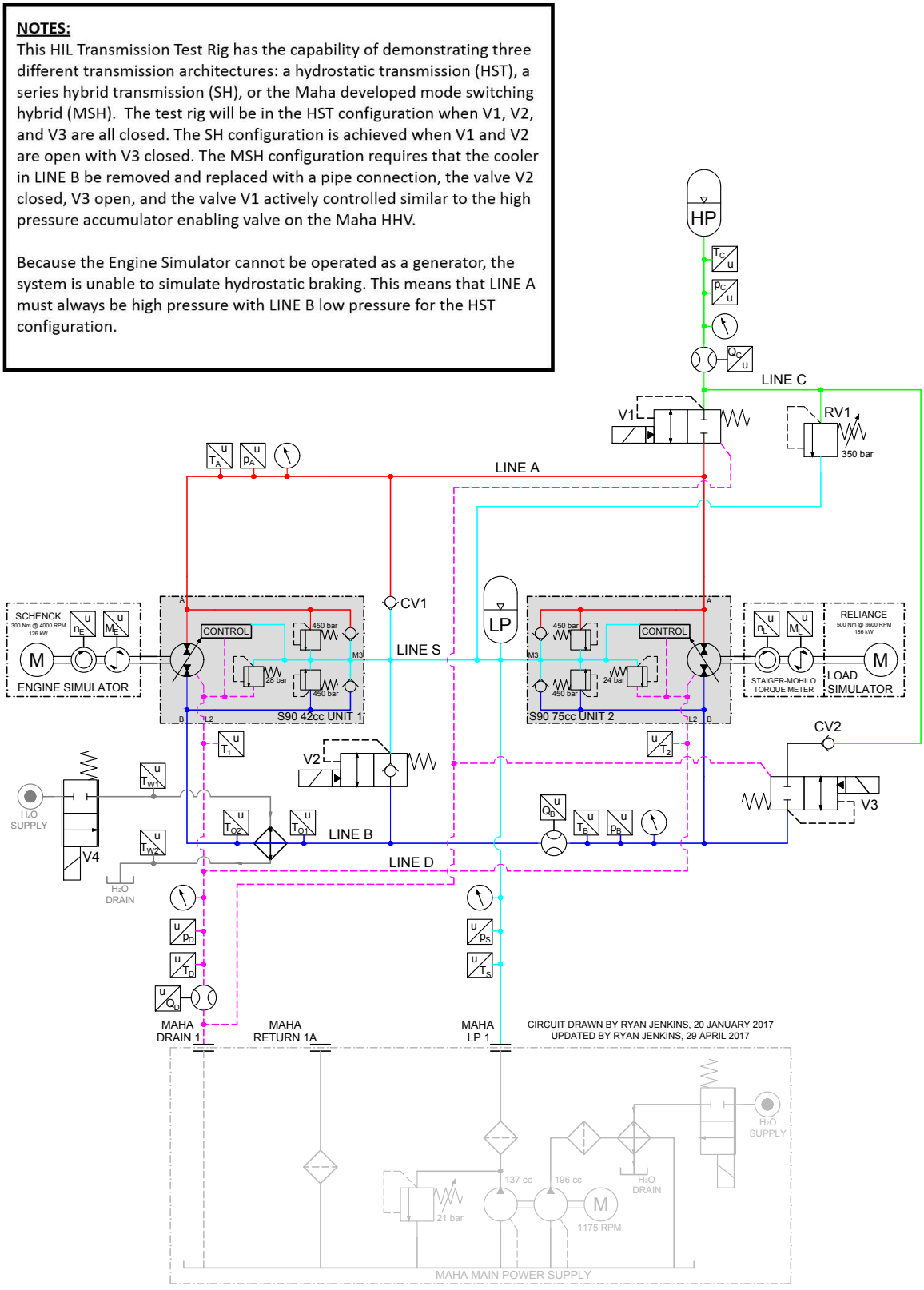

Fig. 7.1. Series hybrid test rig setup at the Maha Fluid Power Research Lab.

7.2 Experiment Setup

The experiment was carried out on the test rig shown in Fig. 7.1. The load simulator was setup to simulate a lightweight passenger vehicle with parameters listed in Table 7.1. The engine simulator is provided a reference speed command generated

Table 7.1. Series-Hybrid Experiment Parameters.

Description	**Symbol**	**Value**	**Units**
Vehicle mass	m_{veh}	1520	kg
Max propulsion force	F_p^{max}	4000	N
Max vehicle speed	v_{veh}^{max}	60	km/h
Engine simulator inertia	I_{eng}	0.38	kg-m^2
Load simulator inertia	I_{load}	0.50	kg-m^2
Virtual axle ratio	k_{axle}	4:1	-
Dynamic tire radius	r_{tire}	0.31	-
Aero drag coefficient	C_d	1.62	-
Rolling resistance coefficient	C_r	0.010	-
Displacement vol. of hyd. pump	V_p^{max}	42	cc/rev
Displacement vol. of hyd. motor	V_m^{max}	75	cc/rev
HP accumulator precharge vol.	V_{ha}	20	L
HP accumulator precharge press.	p_{ha}	80	bar
LP accumulator precharge vol.	V_{la}	20	L
LP accumulator precharge press.	p_{la}	12	bar
Max hi pressure	$p_{A,max}$	240	bar
Low-pressure reservoir press.	p_{lp}	25	bar

by the ASDDP algorithm in the following manner. As described in Section 5.3.2, an optimal state-control sequence $(\mathbf{x}_n^*, \mathbf{u}_n^*)_{n=0}^{N-1}$ is generated every $T_s = 0.5$ seconds. The value $\mathbf{x}_0^*$ is simply the measured state feedback information. Value $\mathbf{x}_1^*$ is the predicted optimal value of the state at the next horizon time step, where the horizon time is $\Delta t = 1$ second according to Equation (5.7). The reference engine speed provided to the engine simulator can be computed as the following linearly interpolated value[1]

$$n_{eng}^{cmd} = n_{eng,0}^* + \left(n_{eng,1}^* - n_{eng,0}^*\right)\left(\frac{T_s}{\Delta t}\right) \tag{7.1}$$

[1]At the time of experimentation n_{eng}^{cmd} was implemented with a discrete time first order low pass filter which emulates Equation (7.1)

The pump displacement command V_p is generated using Equation (5.57) and the motor displacement command V_m is generated every 0.01 seconds using Equation (3.16).

7.3 Data-Simulation Comparison

A simulation is constructed to emulate the test rig setup. The purpose of this simulation is to validate the modeling equations shown in Chapter 3 and the simulation approach taken in Chapter 6. The simulation setup is shown in Fig. 7.2. The

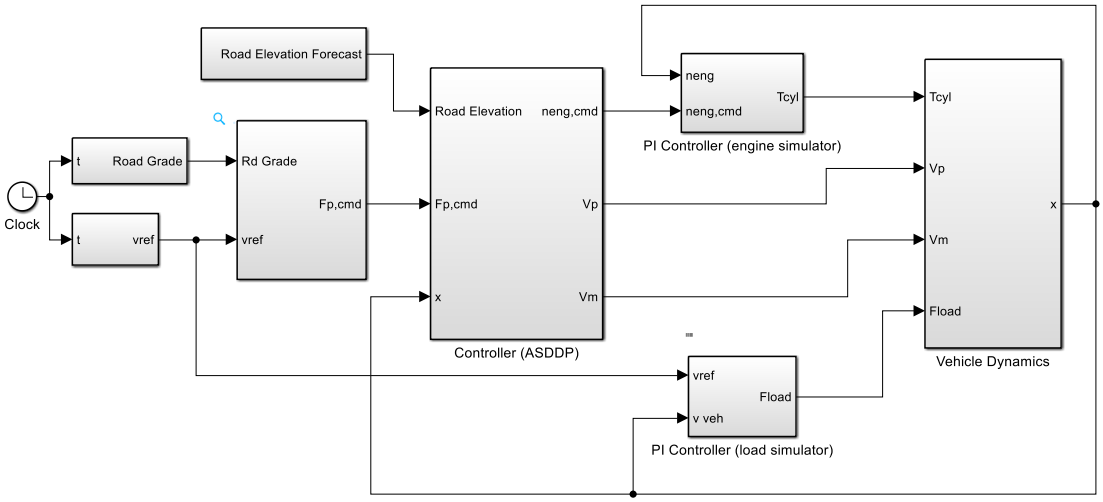

Fig. 7.2. Block diagram of experimental setup.

propulsion force command, F_p^{cmd}, is generated completely open loop according to

$$F_p^{cmd} = m_{veh} a_{veh}^{ref} + \tfrac{1}{2} C_d \rho_{air} (v_{veh}^{ref})^2 + m_{veh} g \left[C_r cos(\phi) + sin(\phi) \right]$$

The term a_{veh}^{ref} is a numerical derivative of the vehicle reference speed. The engine, vehicle, and hydraulic dynamics are the same as given in Section 3.2. The only exceptions are the resistive forces in Equation (3.12) are replaced with F_{load} created by the load simulator block, and T_{cyl} from Equation (3.20) is replaced with the value created by the engine simulator block. The gains of the PI controllers used for the

engine and load simulators were tuned to match the performance characteristics of the real electric units.

The first four minutes of the GPS cycle are carried out in the experiment. A plot of vehicle speed is shown in Fig. 7.3. The vehicle speed profile matches very

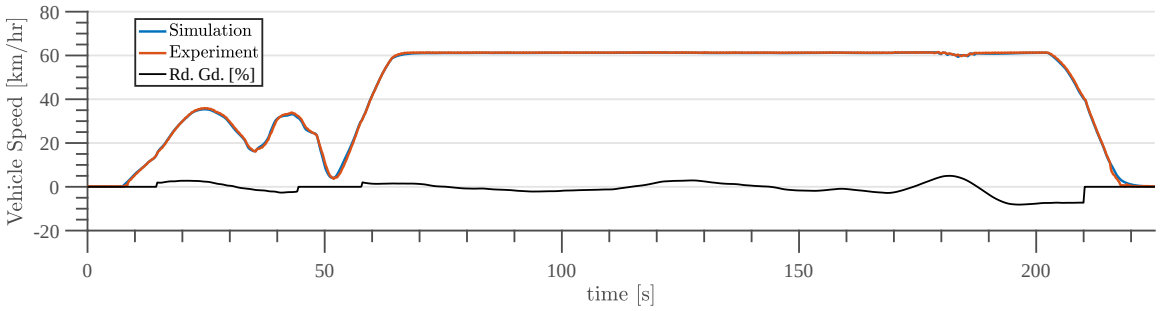

Fig. 7.3. Segment of GPS cycle.

well between the experiment and simulation. Engine speed and pressure of the high pressure accumulator are shown in Fig. 7.4.

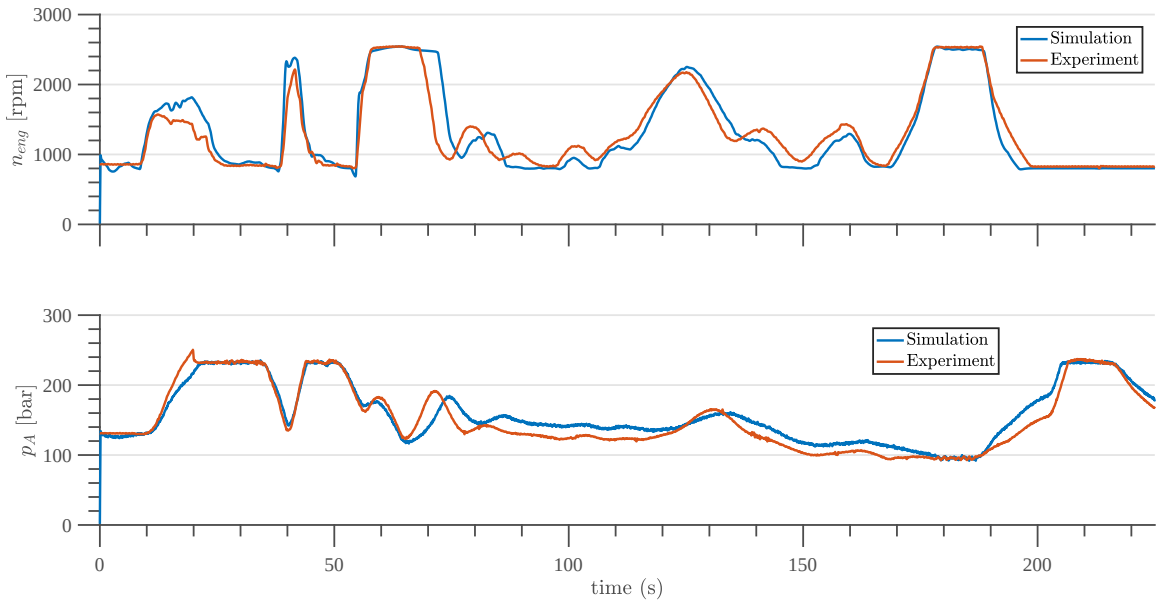

Fig. 7.4. Engine speed and high pressure trajectories over segment of GPS cycle.

Agreement between the simulation and experimental data is again very good, will some slight deviations seen during periods of vehicle acceleration. The control inputs

are shown in Fig. 7.5. Overall, agreement between simulation and experimental data is very good.

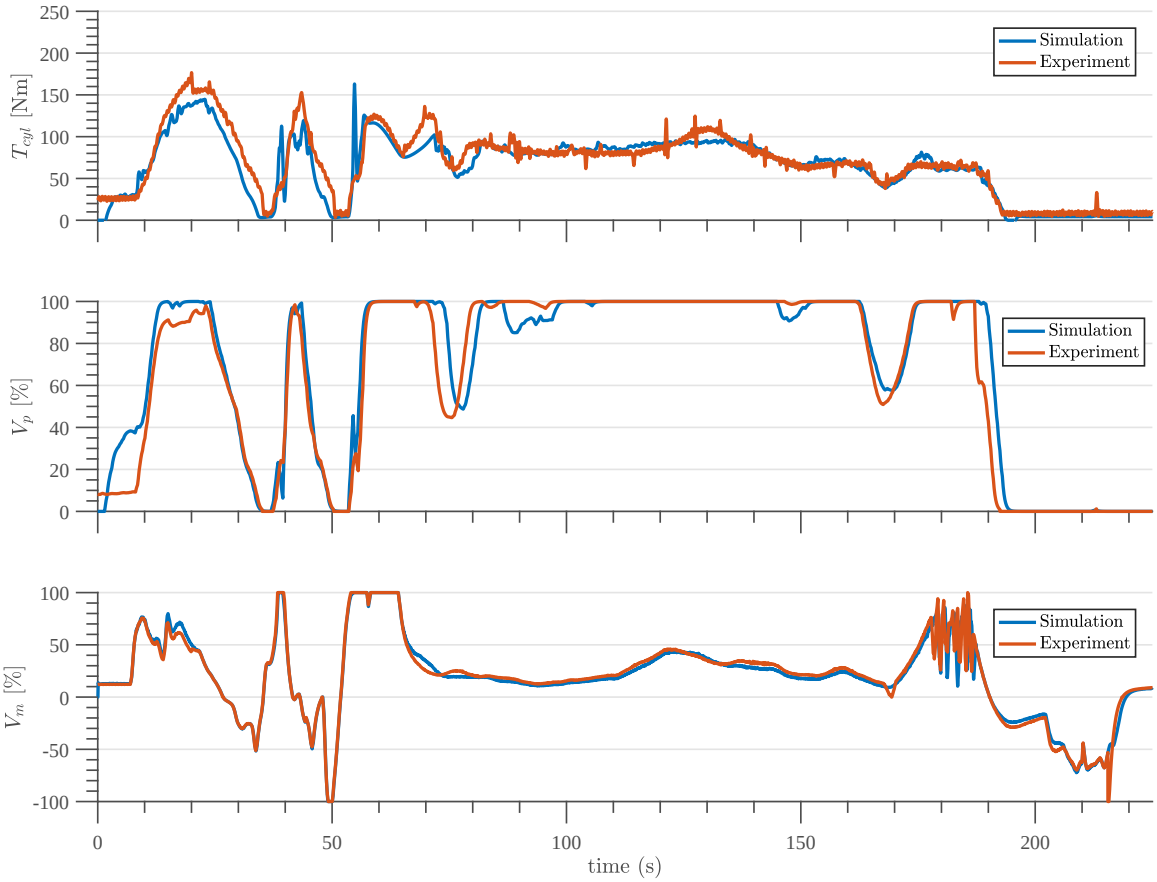

Fig. 7.5. Control input trajectories over segment of GPS cycle.

Near time 180 seconds a high frequency oscillation is observed in the volumetric displacement of hydraulic unit 2. It is worthwhile to note this effect is captured nearly perfectly in simulation. For safety reasons, a small amount of logic was built into the controller which reduces the displacement volume of unit 2 if the high pressure accumulator drops below p_{set} (described by Equation (5.17b)). As shown in Fig. 7.6 the high pressure accumulator drops below p_{set} near time 180 seconds, explaining the rapid adjustments in unit 2 displacement volume. To investigate this further, the gain K_1 from Equation (5.11), which penalizes changes in engine speed between each horizon timestep, is reduced from a value of 0.1 to 0.01 in simulation. The comparison

between the nominal simulation (with $K_1 = 0.1$) and the modified simulation (with $K_1 = 0.01$) is shown in Fig. 7.6. Remarkably, the rapid oscillation is eliminated in

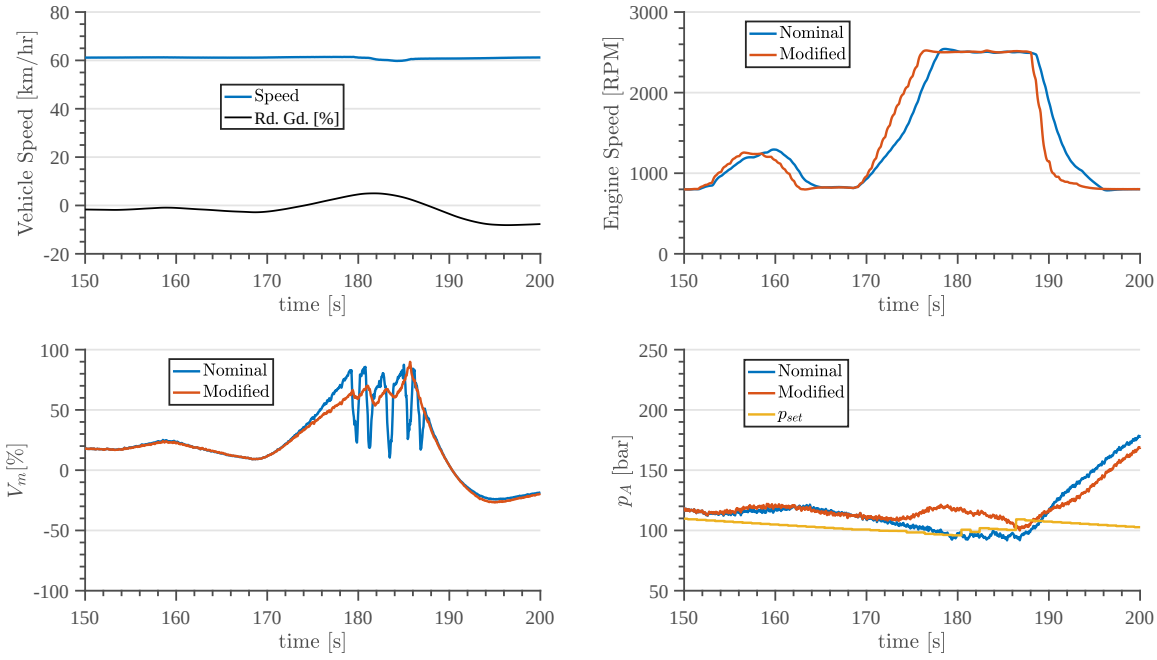

Fig. 7.6. Simulation comparison with $K_1 = 0.1$ (nominal simulation) and $K_1 = 0.01$ (modified simulation).

the modified simulation. This can be explained considering the differences in engine speed observed in Fig. 7.6. In both simulations, ASDDP anticipates the need for a higher engine speed near time 170 seconds in response to the upcoming increase in road grade. The modified simulation is allowed to increase engine speed at a slightly faster rate, and is therefore able to maintain a pressure in the high pressure accumulator which is above the p_{set} limit. This phenomenon gives some credence to the predictive abilities of the ASDDP algorithm.

8. CONCLUSIONS AND FUTURE DIRECTIONS

Real time optimal control (aka model predictive control aka receding horizon control) is a powerful framework for hybrid vehicle energy management. It allows us to derive controllers which consider upcoming conditions and past statistics. By incorporating an adaptive element the controller can be continuously adjusted to maximize performance for the specific operating environment.

In this work a Markov chain model of driver behavior was employed. It was shown that the transition probabilities can be adapted in minutes to the drive cycle, even when initialized on values obtained from a cycle with completely incorrect characteristics. The multi-step transition probabilities were shown to be an effective tool for anticipating driver behavior along a prediction horizon. Adapting the Markov chain model in real time seems to be critical when employing a stochastic strategy. As seen in Section 6.3.2, a poorly tuned statistical model can lead to performance which is worse than a strategy incorporating no statistical information at all. Three computational methods for real time energy management in a HHV when driver behavior and vehicle route are not known in advance were presented. When the Markov chain model is correctly adapted to the drive cycle, these methods produce fuel consumption results which are reasonably close to a theoretically best controller which has full access to driver behavior. Furthermore, each method significantly outperforms a baseline controller which is not provided any statistical driver behavior information. Road elevation forecasting provides some further gains in fuel reduction, even on a moderately level terrain found in Lafayette, IN.

Of the three computational methods developed in 5.3, the ASDDP algorithm seems to provide the most benefit in terms of execution time and fuel consumption results. Experimental results indicate ASDDP has real time run potential on a resource limited processor. When executed on a 400 MHz processor with 128 MB of

RAM, the ASDDP algorithm successfully controlled a series hybrid test rig. During the experiment, the controller update timestep was set at $T_s = 0.5$ seconds, which is not unreasonable for high level control of a powertrain.

8.1 Future Directions

8.1.1 Adjusting P_{ij} to Driving Indicators

In this work the Markov chain transition probabilities, P_{ij}, are adapted in real time. However, these values are not altered in response to various indicators such as traffic signals, upcoming traffic congestion, entering / exit a high speed segment of road, etc. For example, if a red light is being approached the likelihood of a deceleration command in the very near future becomes quite high, regardless of past behavior. Adjusting matrix (P_{ij}) in response to these indicators could provide substantial prediction benefit. On-board telematics could provide a means to inform the algorithm of upcoming indicators.

8.1.2 MPDDP

Average path differential dynamic programming (APDDP) developed in Section 5.3.3 was competitive with ASDDP in terms of fuel consumption but executed in a fraction of the time. The improved speed of APDDP can be attributed to the fact that each timestep along the horizon APDDP considers only a single disturbance transition, whereas ASDDP considers $|W|$ transitions. A hybrid algorithm could foresee-ably consider several likely transitions plus several transitions at outer variances of the disturbance path (as seen for example in Fig. 4.5) for a total of $1 < y < |W|$ transition evaluations. Such a strategy (possibly *multi-path differential dynamic programming*?) could potentially offer nearly 100% of the performance benefits of ASDDP at a considerably reduced computational cost. A mechanism for selecting which transitions to consider at each horizon timestep would be required.

8.1.3 Multi-Stage Markov Chain Modeling

More can be done in the way of Markov chain modeling. The Markov chain used in this work was a single-stage model of the form

$$P_{ij} \triangleq \Pr[w_{n+1} = w^j | w_n = w^i]$$

In words, the probability of the next transition is based only on the present disturbance value. A more sophisticated model could use information about past disturbances to make better predictions about the next transition, such as

$$P_{(i_1,i_2)j} \triangleq \Pr[w_{n+1} = w^j | w_n = w^{i_1}, w_{n-1} = w^{i_2}]$$

The hope is that by including more information to the prediction, the prediction becomes more accurate. The downside is that learning time may increase which could offset prediction benefits (recall the single stage model shown above can be effectively learned in roughly 20-30 minutes). Additionally, incorporating such a multi-stage model may add computational complexity to the algorithm which needs to be considered.

REFERENCES

REFERENCES

[1] R. Kumar, "A power management strategy for hybrid output coupled power-split transmission to minimize fuel consumption," Ph.D. dissertation, Purdue University, West Lafayette, 2010.

[2] W. J. Midgley and D. Cebon, "Comparison of regenerative braking technologies for heavy goods vehicles in urban environments," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 226, no. 7, pp. 957–970, 2012.

[3] N. Jalil, N. A. Kheir, and M. Salman, "A rule-based energy management strategy for a series hybrid vehicle," in *American Control Conference, 1997. Proceedings of the 1997*, vol. 1. IEEE, 1997, pp. 689–693.

[4] L. Serrao, S. Onori, and G. Rizzoni, "A comparative analysis of energy management strategies for hybrid electric vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 133, 2011.

[5] G. Paganelli, M. Tateno, A. Brahma, G. Rizzoni, and Y. Guezennec, "Control development for a hybrid-electric sport-utility vehicle: strategy, implementation and field test results," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 6. IEEE, 2001, pp. 5064–5069.

[6] R. Kumar and M. Ivantysynova, "Towards an optimal energy management strategy for hybrid hydraulic powertrains based on output-coupled power split principle," in *Proceedings of the 5th FPNI PhD Symposium*, 2008, pp. 41–51.

[7] K. Williams and M. Ivantysynova, "Towards an optimal energy management strategy for hybrid hydraulic powertrains based on dual stage power split principle," in *Proceedings of the 5th FPNI PhD Symposium*, 2008, pp. 27–40.

[8] G. Paganelli, S. Delprat, T.-M. Guerra, J. Rimaux, and J.-J. Santin, "Equivalent consumption minimization strategy for parallel hybrid powertrains," in *Vehicular Technology Conference, 2002. VTC Spring 2002. IEEE 55th*, vol. 4. IEEE, 2002, pp. 2076–2081.

[9] 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, no. 4-5, pp. 509–524, 2005.

[10] L. Serrao, S. Onori, and G. Rizzoni, "Ecms as a realization of pontryagin's minimum principle for hev control," in *American Control Conference, 2009. ACC'09.* IEEE, 2009, pp. 3964–3969.

[11] N. Kim, S. Cha, and H. Peng, "Optimal control of hybrid electric vehicles based on pontryagin's minimum principle," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1279–1287, 2011.

[12] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Computation*, vol. 11, pp. 229–242, 1999.

[13] D. P. Filev and I. Kolmanovsky, "Markov chain modeling approaches for on board applications," in *American Control Conference*, Baltimore, Maryland, 2010, pp. 4139–4145.

[14] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: Wiley, 2005.

[15] C.-C. Lin, H. Peng, and J. Grizzle, "A stochastic control strategy for hybrid electric vehicles," in *American Control Conference*, Boston, Massachusetts, 2004, pp. 4710–4715.

[16] ——, "Optimization of powertrain operating policy for feasibility assessment and calibration: Stochastic dynamic programming approach," in *American Control Conference*, Anchorage, AK, 2002, pp. 1425–1430.

[17] D. F. Opila, X. Wang, R. McGee, and J. Grizzle, "Real-time implementation and hardware testing of a hybrid vehicle energy management controller based on stochastic dynamic programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 2, p. 021002, 2013.

[18] R. Kumar and M. Ivantysynova, "Investigation of various power management strategies for a class of hydraulic hybrid powertrains: Theory and experiments," in *Proc. of the 6th FPNI PhD Symposium*, 2010, pp. 87–99.

[19] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific, 1996.

[20] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[21] W. Powell, *Approximate Dynamic Programming*. Hoboken, New Jersey: Wiley, 2011.

[22] R. Johri, "Neuro-dynamic programming and reinforcement learning for optimal energy management of a series hydraulic hybrid vehicle considering engine transient emissions," Ph.D. dissertation, University of Michigan, Ann Arbor, 2011.

[23] Z. D. Asher, D. A. Baker, and T. H. Bradley, "Prediction error applied to hybrid electric vehicle optimal fuel economy," *IEEE Transactions on Control Systems Technology*, 2017.

[24] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2015.

[25] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. New York, NY, USA: Cambridge University Press, 2009.

[26] H. Borhan, A. Vahidi, A. Phillips, M. Kuang, and I. Kolmanovsky, "Predictive energy management of a power-split hybrid electric vehicle," in *2009 American Control Conference*, St. Louis, MO, 2009.

[27] T. Deppen, A. G. Alleyne, K. A. Stelson, and J. J. Meyer, "Optimal energy use in a light weight hydraulic hybrid passenger vehicle," *Journal of Dynamic Systems, Measurement, and Control*, vol. 134, 2012.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[29] G.-E. Katsargyri, I. V. Kolmanovsky, J. Michelini, M. L. Kuang, A. M. Phillips, M. Rinehart, and M. A. Dahleh, "Optimally controlling hybrid electric vehicles using path forecasting," in *American Control Conference, 2009. ACC'09.* IEEE, 2009, pp. 4613–4617.

[30] E. Hellstrom, "Look-ahead control of heavy vehicles," Ph.D. dissertation, Linkoping University Institute of Technology, Linkoping, 2010.

[31] M. Cannon, P. Couchman, and B. Kouvaritakis, *MPC for Stochastic Systems*, R. Findeisen, F. Allgower, and L. Biegler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[32] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[33] S. DiCairano, D. Bernardini, A. Bemporad, and I. Kolmanovsky, "Stochastic mpc with learning for driver-predictive vehicle control and its application to hev energy management," *IEEE Trans. on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, 2014.

[34] X. Zeng and J. Wang, "A parallel hybrid electric vehicle energy management strategy using stochastic model predictive control with road grade preview," *IEEE Transactions on Control System Technology*, vol. 23, no. 6, pp. 2416–2423, 2015.

[35] C. Sun, X. Hu, S. J. Moura, and F. Sun, "Velocity predictors for predictive energy management in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1197–1204, 2015.

[36] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical model predictive control framework for autonomous ground vehicles," in *American Control Conference, 2008.* IEEE, 2008, pp. 3719–3724.

[37] D. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 1*, 3rd ed. Belmont, Massachusetts: Athena Scientific, 2005.

[38] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 2012.

[39] D. Jacobson and D. Mayne, *Differential Dynamic Programming*. New York, NY: Elsevier, 1970.

[40] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in neural information processing systems*, 2008, pp. 1465–1472.

[41] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[42] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4906–4913.

[43] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1168–1175.

[44] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 695–702.

[45] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[46] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.

[47] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Cambridge: Cambridge University Press New Delhi, 2008.

[48] L. Bottou, "Online algorithms and stochastic approximations," *Online Learning and Neural Networks*, 1998.

[49] B. Carl, M. Ivantysynova, and K. Williams, "Comparison of operational characteristics in power split continuously variable transmissions," Sae technical paper, Tech. Rep., 2006.

[50] R. Kumar, M. Ivantysynova, and K. Williams, "Study of energetic characteristics in power split drives for on highway trucks and wheel loaders," SAE Technical Paper, Tech. Rep., 2007.

[51] M. Sprengel and M. Ivantysynova, "Investigation and energetic analysis of a novel hydraulic hybrid architecture for on-road vehicles," in *13th Scandinavian International Conference on Fluid Power; June 3-5; 2013; Linköping; Sweden*, no. 092. Linköping University Electronic Press, 2013, pp. 87–98.

[52] T. Bleazard, H. Haria, M. Sprengel, and M. Ivantysynova, "Optimal control and performance based design of the blended hydraulic hybrid," in *ASME/BATH 2015 Symposium on Fluid Power and Motion Control*. American Society of Mechanical Engineers, 2015.

[53] M. Sprengel, T. Bleazard, H. Haria, and M. Ivantysynova, "Implementation of a novel hydraulic hybrid powertrain in a sports utility vehicle," *IFAC-PapersOnLine*, vol. 48, no. 15, pp. 187–194, 2015.

[54] M. Sprengel and M. Ivantysynova, "Recent developments in a novel blended hydraulic hybrid transmission," SAE Technical Paper, Tech. Rep., 2014.

[55] A. Gibson and I. Kolmanovsky, "Modeling and analysis of engine torque modulation for shift quality improvement," in *2006 SAE World Congress*, Detroit, Michigan, 2006.

[56] J. Heywood, *Internal Combustion Engine Fundamentals*. NY, New York: McGraw-Hill, 1988.

[57] R. Rahmfeld, "Development and control of energy saving energy saving hydraulic servo drives for mobile systems," Ph.D. dissertation, Tech. Univ. of Hamburg-Harbur, Hamburg, 2002.

[58] G. F. Lawler, *Introduction to stochastic processes*. CRC Press, 2006.

[59] C. Zhang, A. Vahidi, P. Pisu, X. Li, and K. Tennant, "Role of terrain preview in energy management of hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1139–1147, March 2010.

[60] G.-E. Katsargyri, "Optimally controlling hybrid electric vehicles using path forecasting," Master's thesis, Massachusetts Institute of Technology, Cambridge, 2008.

[61] M. J. L. Orr. (1996) Introduction to radial basis function networks. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.215.2807

[62] B. Fornberg and N. Flyer, "The gibbs phenomenon for radial basis functions," in *Gibbs Phenomenon in Various Representations and Applications, Sampling Publishing*, 2006.

[63] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. New York, NY: Springer, 2008.

[64] L. Bottou, "Stochastic learning," *Advanced Lectures on Machine Learning*, pp. 146–168, 2004.

[65] R. Battiti, "First- and second-order methods for learning: Between steepest descent and newtons method," *Neural Computation*, pp. 144–166, 1992.

[66] L. Bottou and Y. LeCun, "Large scale online learning," in *NIPS*, S. Thrun, L. K. Saul, and B. Schlkopf, Eds. MIT Press, 2003, pp. 217–224.

[67] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. of the 30th International Conference on Machine Learning*, Atlanta, Georgia, 2013.

[68] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 12, pp. 145–151, 1999.

[69] A. Y. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 406–415.

[70] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York, NY: Springer, 2006.

[71] L.-Z. Liao and C. A. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

APPENDIX

A. DRIVER BEHAVIOR STATISTICS

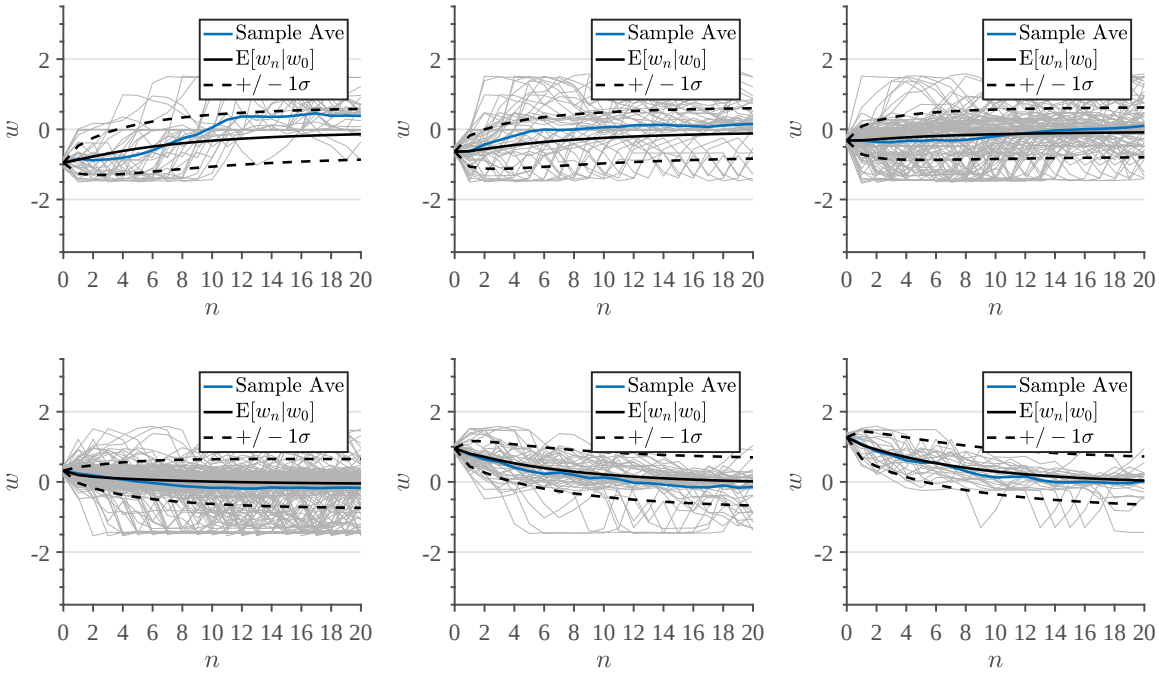

Fig. A.1. Propagation of $\mathbb{E}[w_n|w_0 = w^i]$. Sample paths shown in light grey. UDDS cycle.

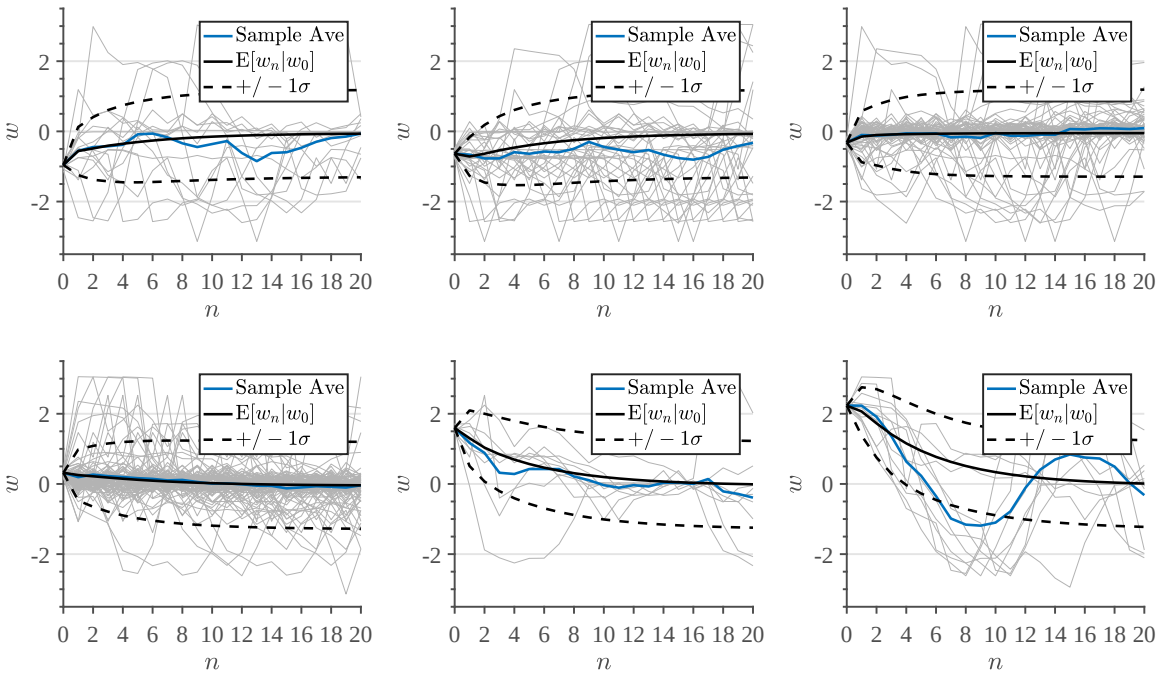

Fig. A.2. Propagation of $\mathbb{E}[w_n|w_0 = w^i]$. Sample paths shown in light grey. US06 cycle.

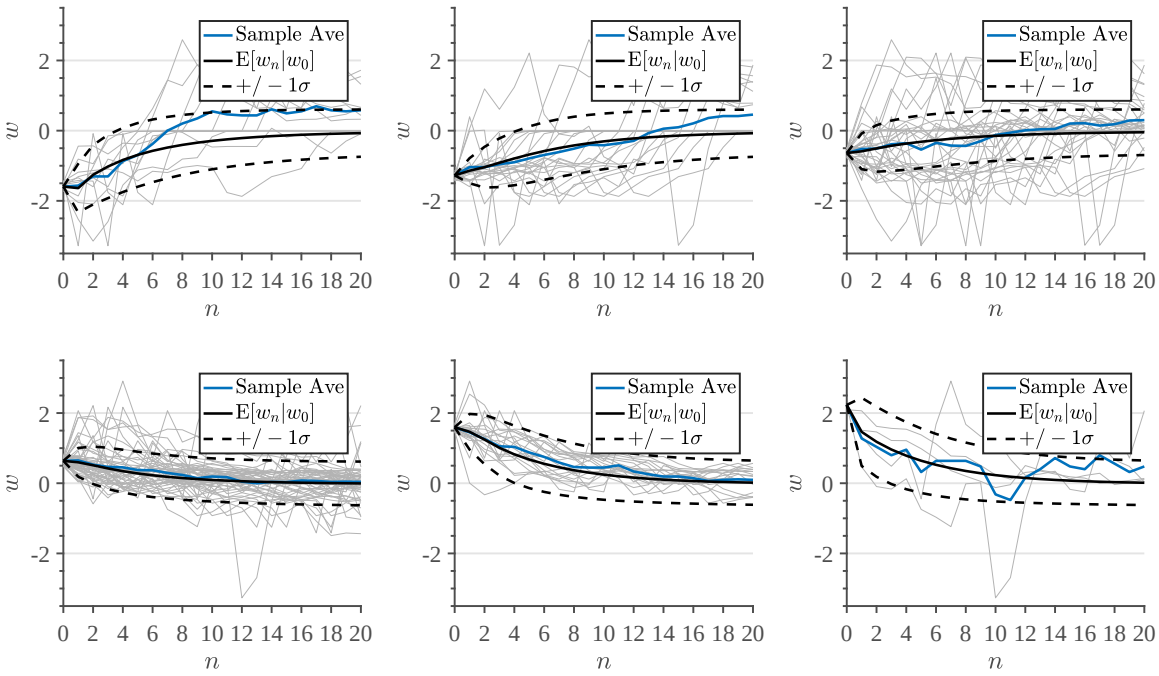

Fig. A.3. Propagation of $\mathbb{E}[w_n|w_0 = w^i]$. Sample paths shown in light grey. GPS cycle.

B. VALUE FUNCTION DERIVATION FOR ASDDP

Define

$$\bar{\mathbf{X}}_n(\mathbf{x}, \mathbf{u}) = \{\mathbf{x}^j | \mathbf{x}^j = F_n(\mathbf{x}, \mathbf{u}, w^j),\ w^j \in W\} \subset \mathbf{X}$$

as the set of all states reachable from $\mathbf{x}$ under control input $\mathbf{u}$ at time n. The finite horizon value function is given by [1]

$$\begin{aligned}
V_n(\mathbf{x}_n) &= \min_{\mathbf{u}_n,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\Big[h(\mathbf{x}_N) + \sum_{k=n}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k, w_k)\ \mathbf{x}_n, w_0 = w^i\Big] \\
&= \min_{\mathbf{u}_n,\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n) + h(\mathbf{x}_N) + \sum_{k=n+1}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k, w_k)\ \mathbf{x}_n, w_0 = w^i\right] \\
&= \min_{\mathbf{u}_n} \Bigg\{ \mathbb{E}\Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n)\ \mathbf{x}_n, w_0 = w^i\Big] + \\
&\qquad \min_{\mathbf{u}_{n+1},\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[h(\mathbf{x}_N) + \sum_{k=n+1}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k, w_k)\ \mathbf{x}_n, w_0 = w^i\right] \Bigg\} \\
&= \min_{\mathbf{u}_n} \Bigg\{ \mathbb{E}\Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n)\ \mathbf{x}_n, w_0 = w^i\Big] + \\
&\qquad \sum_{\mathbf{x}^j \in \bar{\mathbf{X}}_n(\mathbf{x}_n, \mathbf{u}_n)} \Pr\Big[\mathbf{x}_{n+1} = \mathbf{x}^j\ \mathbf{x}_n, \mathbf{u}_n, w_0 = w^i\Big] \times \\
&\qquad \underbrace{\min_{\mathbf{u}_{n+1},\ldots,\mathbf{u}_{N-1}} \mathbb{E}\left[h(\mathbf{x}_N) + \sum_{k=n+1}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k, w_k)\ \mathbf{x}_{n+1}, w_0 = w^i\right]}_{V_{n+1}(\mathbf{x}_{n+1})} \Bigg\} \\
&= \min_{\mathbf{u}_n} \sum_j P_{ij}^{(n)} \Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w^j) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w^j)\big)\Big]
\end{aligned} \tag{B.1}$$

[1] Conditional expectation $\mathbb{E}[X] = \sum_y \Pr[Y = y]\mathbb{E}[X|Y = y]$ is used in the second to last equality

with boundary condition $V_N(\mathbf{x}) = h(\mathbf{x})$. The last equality used the following

$$\begin{aligned}\Pr\big[\mathbf{x}_{n+1} = \mathbf{x}^j \;\; \mathbf{x}_n, \mathbf{u}_n, w_0 = w^i\big] &= \Pr\big[w_n = w^j \;\; w_0 = w^i\big] \\ &= P_{ij}^{(n)}\end{aligned}$$

where $\mathbf{x}^j \triangleq F_n(\mathbf{x}_n, \mathbf{u}_n, w^j) \in \bar{\mathbf{X}}_n(\mathbf{x}_n, \mathbf{u}_n)$. Equation (5.41) is equivalent to

$$V_n(\mathbf{x}_n) = \min_{\mathbf{u}_n} \mathbb{E}\Big[g_n(\mathbf{x}_n, \mathbf{u}_n, w_n) + V_{n+1}\big(F_n(\mathbf{x}_n, \mathbf{u}_n, w_n)\big) \;\; \mathbf{x}_n, w_0 = w^i\Big]$$

VITA

VITA

Kyle Williams received the degree Bachelor of Science in Mechanical Engineering from Purdue University in May 2005. He was a summer intern at Caterpillar, Inc. during the summers of 2004 and 2005. He received his Master's Degree in Mechanical Engineering from Purdue in August 2007. During the following years he worked for Parker Hannifin and Caterpillar, Inc. He is currently a control engineer with Caterpillar's Large Power Systems Division. He received his PhD from Purdue in May 2018. His interests are predictive control for stochastic systems, with an emphasis on energy and vehicle systems.

PUBLICATIONS

K. Williams and M. Ivantysynova. "Approximate Stochastic Differential Dynamic Programming for Hybrid Vehicle Energy Management", *IEEE Transactions on Control Systems Technology* (submitted)

K. Williams, R. Kumar and M. Ivantysynova. "Robust control for a dual stage power split transmission with energy recovery," in *Proceedings of the 6th International Fluid Power Conference*, Dresden, Germany, Vol. 1, pp.127-144, April 2008

K. Williams and M. Ivantysynova "Towards an optimal energy management strategy for hybrid hydraulic powertrains based on dual stage power split principle," in *Proceedings of the 5th FPNI PhD Symposium*, Krakow, Poland, pp 27 - 40, July 2008

R. Kumar, K. Williams and M. Ivantysynova. "Study of energetic characteristics in power split drives for on-highway trucks and wheel loaders," in *Proceedings of the SAE International Commercial Vehicle Engineering Congress*, Chicago, Illinois, 2007

K. Williams, "Energy recovery for hydraulic hybrid power split drives," Master's thesis, Purdue University, 2007

B. Carl, M. Ivantysynova and K. Williams, "Comparison of Operational Characteristics in Power Split Continuously Variable Transmissions", SAE Technical Paper 2006-01-3468, 2006