REAL-TIME, SELF-LEARNING IDENTIFICATION AND STOCHASTIC OPTIMAL CONTROL OF ADVANCED POWERTRAIN SYSTEMS

by

Andreas Malikopoulos

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2008

Doctoral Committee:

Professor Dionissios N. Assanis, Co-Chair
Professor Panos Y. Papalambros, Co-Chair
Professor James S. Freudenberg
Professor A. Galip Ulsoy
Assistant Professor Domitilla Del Vecchio

To my wife, Voula, and
to my daughter, Georgina

ACKNOWLEDGMENTS

When I came to Ann Arbor, I could never imagine that during the years to come I would have the most wonderful and productive time in my life. I had the chance to have come across a lot of wise and wonderful people. Their guidance and support have been invaluable.

I was extremely fortunate to have two remarkably supportive advisors, Professor Dennis Assanis and Professor Panos Papalambros. They provided friendly warmth, challenges, continuing support throughout my studies while affording me a tremendous amount of freedom and responsibility.

Professor Assanis is the reason that I decided to attend the University of Michigan for graduate school. I had come across his research activities at the Automotive Research Center (ARC) while still undecided about the graduate school I should attend. It took me just seconds to realize that this was the research environment I was looking for. When I met him and expressed my strong interest to join his group, I got impressed by his depth of thought, his openness, and his critical eye. Professor Assanis offered an extraordinary depth of knowledge in advanced powertrain systems, and he was a noble teacher and mentor. I would like to sincerely thank him for giving me the opportunity to join his research group at the University of Michigan.

During the first year of my studies, I had the privilege to meet Professor Papalambros while taking the Design Optimization class. I believe that this meeting had a significant impact in shaping my research interests and professional goals. Professor Papalambros provided continuing advice and developed my sense of the academic community in engineering. He provided challenges and he offered a blinding depth of

knowledge in design optimization. He promoted a respect for rigorous work, and constantly pushed the boundaries of his expertise while maintaining extraordinary standards of quality. I would like to sincerely thank him for his invaluable guidance and support throughout my studies.

In addition to my advisors, many people contributed meaningfully by providing feedback and perspective that helped define my direction. I would like to thank the committee members of my dissertation, Professor James Freudenberg, Professor Galip Ulsoy, and Professor Domitilla Del Vecchio for their valuable feedback and comments on my dissertation and publications. During my interaction with Professor Freudenberg, while I was taking his class in linear feedback control systems, he contributed directly to my understanding in control theory and the related applications in advanced powertrain systems.

I also owe special thanks to Professor Demosthenis Teneketzis for broadening and deepening my understanding in stochastic control and centralized stochastic systems. He provided feedback and excellent references; his classes in stochastic process and stochastic control along with our numerous interactions were instrumental in enhancing my knowledge in this area.

I would also like to express my appreciation to a plethora of other individuals, who spent a considerably amount of time in meeting with me and providing assistance. Dr. Michael Kokkolaras was always willing to provide and share his knowledge in optimization while working together in various projects. His feedback in various aspects of my research activities was always helpful and valuable. Professor Rudy Schmerl provided a tremendous amount of assistance in enhancing my technical communication skills. I enjoyed all our meetings and discussions. Professor Zoran Filipi was instrumental in developing my understanding of modeling and simulation of advanced powertrain systems. Dr. George Lavoie was always willing to share his knowledge and expertise while provided helpful feedback on my work. Dr. Aristotelis Babajimopoulos provided

unlimited assistance in any computer related issue. Finally, I would like to thank all my lab-mates and staff, past and present, in Autolab and Optimal Design Laboratory.

This research was supported by the Automotive Research Center (ARC), a U.S. Army Center of Excellence in Modeling and Simulation of Ground Vehicles at the University of Michigan. This support is gratefully acknowledged.

I owe my gratitude to my father, Alexandros, and to my mother, Ioanna, who have been always supportive in any aspect of life. The knowledge that my success will bring to them happiness and pride was an extra motivator for me.

This dissertation is dedicated to my wife, Voula, and to my daughter, Georgina. I would not be able to complete my PhD without the boundless love, support, and patience of my wonderful loving wife. I will always be indebted to her for prioritizing my work and success against her interests and professional goals. Georgina's smile was a tremendous source of energy for me since she was born. Looking at this little angle always rejuvenated me, and immersed me in new energy and tranquility.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

The following nomenclature is used consistently in the dissertation.

k	Discrete time steps (decision epochs)
s_k	System state at time k
y_k	System output at time k
w_k	Disturbance at time k
v_k	Measurement error at time k
a_k	Control action at time k
$\mathcal{S}$	State space
$\mathcal{A}$	Control space
$\mathcal{W}$	Disturbance space
$A(\cdot)$	Feasible action set
f	Function in the state equation
h	Function in the observation equation
π	Control policy
μ	Control functions
J^{π}	Cost corresponding to a control policy π
z^k	System observation at time k
$P^{\pi}(\cdot\|\cdot)$	Conditional distribution of states for a given control policy π
$R(\cdot\|\cdot)$	State transition cost
$\mathbf{P}(\cdot,\cdot)$	Transition probability matrix
$\mathbf{R}(\cdot,\cdot)$	Transition cost matrix

M^{θ}	Family of models parameterized by the parameter θ
$\hat{\theta}_k$	Estimation of the parameter at time k
θ°	True parameter
Θ	Parameter set
T_1	First entrance time of the state
T_n	nth entrance time of state
μ_i	Mean recurrence time of state i
$\tilde{\mathcal{S}}$	Predictive Optimal Decision-making (POD) domain
$\tilde{\mathcal{S}}_i$	Predictive Representation Node (PRN) for each state i
$\bar{R}_i(\cdot \mid \cdot)$	Predictive Representation Node (PRN) value for each state i
R^i_{PRN}	Predictive Representation Node (PRN) expected evaluation function for each state i
$\mu_{\tilde{\mathcal{S}}_i}$	Mean recurrence time of each Predictive Representation Node (PRN)
$\boldsymbol{\rho}$	Vector of the stationary distribution of the chain
I_C	Indicator function of a given set C
V	Number of visits of the chain
$\bar{V}$	Mean number of visits of the chain
$\bar{\pi}$	Lookahead control policy by the Predictive Optimal Stochastic Control Algorithm (POSCA)
J	Accumulated cost incurred by dynamic programming
$\tilde{J}$	Accumulated cost incurred by the Predictive Optimal Stochastic Control Algorithm (POSCA)
$\bar{J}$	Lookahead cost incurred by the Predictive Optimal Stochastic Control Algorithm (POSCA)
Γ	Strategic form game
r	Player in a strategic form game
R^r	Payoff function for each player r in a strategic form game

A^r	Set of feasible strategies for each player r
a^r	Strategy for each player r in a strategic form game
R_r'	Mapping of the payoff functions in decentralized learning

ABSTRACT

REAL-TIME, SELF-LEARLING IDENTIFICATION AND STOCHASTIC OPTIMAL CONTROL OF ADVANCED POWERTRAIN SYSTEMS

by

Andreas Malikopoulos

Co-Chairs: Dionissios N. Assanis and Panos Y. Papalambros

Increasing demand for improving fuel economy and reducing emissions without sacrificing performance has stimulated significant research on and investment in advanced internal combustion engine technologies. These technologies have introduced a number of controllable variables that have enhanced our ability to optimize engine operation. Current engine calibration methods for deriving the optimal values of the controllable variables generate a static tabular relationship between the variables and steady-state operating points or specific driving conditions (e.g., vehicle speed profiles for highway and city driving). These methods, however, seldom guarantee optimal engine operation for common driving habits (e.g., stop-and-go driving, rapid acceleration, or rapid braking). Each individual driving style is different and rarely meets those driving conditions of testing for which the engine has been calibrated to operate optimally.

This dissertation develops the theory and algorithms that succeed in making the engine of a vehicle an autonomous intelligent system capable of learning the optimal values of the controllable variables in real time while the driver drives the vehicle. The engine is treated as a controlled stochastic system, and engine calibration is formulated as

a sequential decision-making problem under uncertainty that addresses the system identification and stochastic control problem simultaneously.

Specifically, the theory for building models suited for sequential decision-making under uncertainty is reviewed. These models formalize the framework in which an intelligent or rational system can select control actions so that a long-term reward is maximized. The theory is extended to portray a real-time computational learning model with which the state estimation and system identification problem can be solved. A lookahead control algorithm is implemented that provides the decision-making mechanism suitable for real-time implementation. The algorithm solves the stochastic control problem by utilizing accumulated data acquired over the learning process of the computational model. The increase of the problem's dimensionality, when more than one controllable variable is considered, is addressed by a decentralized learning control scheme. This scheme draws from multi-agent learning research in a range of areas, including reinforcement learning and game theory, to coordinate optimal behavior among the controllable variables.

Various case studies, including cart-pole balancing, vehicle cruise-control, and gasoline and diesel engine calibration, were conducted. In the engine calibration problem, the engine was shown to progressively perceive the driver's driving style and eventually learn its optimal calibration for this driving style.

The theory and algorithms developed in this dissertation may reduce considerably the existing discrepancy between the gas mileage estimate displayed on the vehicle's window sticker and the actual one. This would allow every driver to realize optimal fuel economy and pollutant emissions as fully as possible with respect to his/her driving habits.

CHAPTER 1

INTRODUCTION

Increasing demand for improving fuel economy and reducing emissions without sacrificing performance has induced significant research on and investment in advanced internal combustion engine technologies. These technologies, such as fuel injection systems, variable geometry turbocharging, variable valve actuation, and exhaust gas recirculation, have introduced a number of new engine variables that can be controlled to optimize engine operation. In particular, the determination of the optimal values of these variables, referred to as engine calibration, have been shown to be critical for achieving high engine performance and fuel economy while meeting emission standards. Consequently, engine calibration is defined as a procedure that optimizes one or more engine performance criteria, e.g., fuel economy, emissions, or engine performance with respect to the engine controllable variables. This dissertation develops the theory and algorithms that succeed in making the engine of a vehicle an autonomous intelligent system capable of learning the optimal values of the controllable variables in real time while the driver drives the vehicle.

1.1 Motivation

Current calibration methods generate a static tabular relationship between the optimal values of the controllable variables and steady-state operating points or specific driving conditions (e.g., vehicle speed profiles for highway and city driving). This

relationship is incorporated into the electronic control unit (ECU) of the engine to control engine operation. While the engine is running, values in the tabular relationships are interpolated to provide the values of the controllable variables for each engine operating point. These calibration methods, however, seldom guarantee optimal engine operation for common driving habits (e.g., stop and go driving, rapid acceleration, or rapid braking). Each individual driving style is different and rarely meets those driving conditions of testing for which the engine has been calibrated to operate optimally. Consumers often complain that their new cars simply cannot achieve the gas mileage estimate displayed on the window sticker or featured in advertisements.

1.2 Advancements in Engine Electronic Control Units

Advanced internal combustion engine technologies have led to increased opportunities for use of Electronic Control Units (ECUs). Current ECUs perform a variety of control tasks using engine calibration static maps that provide the values of several controllable variables. These values are then used as references by actuators to maintain optimal engine operation. In traditional ECU development processes, engine calibration maps are generated experimentally by extensive steady-state engine operation and step function changes of engine speed and load [1-4]. This is usually accompanied by simple transient operation limited by dynamometer capabilities and simulation technologies [5]. However, steady-state and simple transient engine operation is only partially indicative of actual engine operation in a vehicle. Increased sophistication of ECUs coupled with engine technologies can lead to significant calibration improvements.

Advances in computing technology have enabled simulation-based methods such as Hardware in the Loop (HiL) and Software in the Loop (SiL) test systems [6, 7]. HiL systems have been widely utilized as powerful methods for implementing engine calibration maps. These systems involve a real-time simulation engine model and a

vehicle system connected to the ECU hardware. HiL systems allow the ECU development through simulation of powertrain components and vehicle system. SiL systems are more recent approaches, in which the engine model is integrated with the ECU software, and run on a computer [8]. SiL allows selective tests of single calibration tasks and separate modules of the ECU early in the development process. An essential requirement of HiL and SiL systems is the availability of an engine model capable of generating physical and consistent outputs of a combustion engine based on actual inputs.

1.3 State-of-the-art Engine Calibration Methods

HiL and SiL systems aim to provide automated software tools for generating and validating calibration maps during the ECU development process. Various methods for deriving these maps at steady-state and limited transient engine operation have been extensively reported in the literature [9-12]. These efforts have been valuable in understanding steady-state operation, and optimizing fuel economy and emissions in the last years [13]. However, continuously optimal engine operation has not yet been possible. State-of-the-art engine calibration methods rely on static correlations for steady-state operating points accompanied by transient vehicle testing. The calibration process, its duration, and its cost grow exponentially with the number of controllable variables, and optimal calibration for the entire feasible engine operating domain cannot be guaranteed. Even for engines with simple technologies, achievement of optimal calibration may become impractical [10]. In addition, current calibration methods cannot guarantee optimal engine operation in transient cases encountered in driving styles of different drivers [14]. Transient operation constitutes the largest segment of engine operation over a driving cycle compared to the steady-state one [15, 16]. Fuel consumption and emissions during transient operation are extremely complicated [16], vary significantly with each particular driving cycle [17, 18], and are highly dependent

upon the calibration [18, 19]. Engine operating points, during the transient period before their steady-state value is reached, are associated with different Brake-Specific Fuel Consumption (BSFC) values, depending on the directions, as shown in Figure 1.1, from which they have been arrived, illustrated qualitatively in Figure 1.2. Pollutant emissions such as NO_x, and particulate matters, demonstrate the same qualitative behavior, as shown by Hagena *et al.* [13].

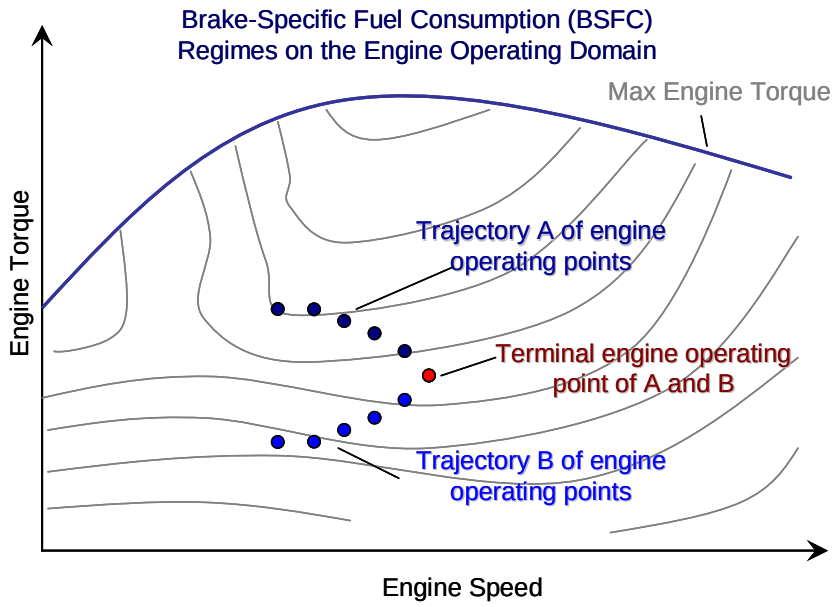

Figure 1.1 − Two trajectories A, and B, of engine operating points ending at the same operating point

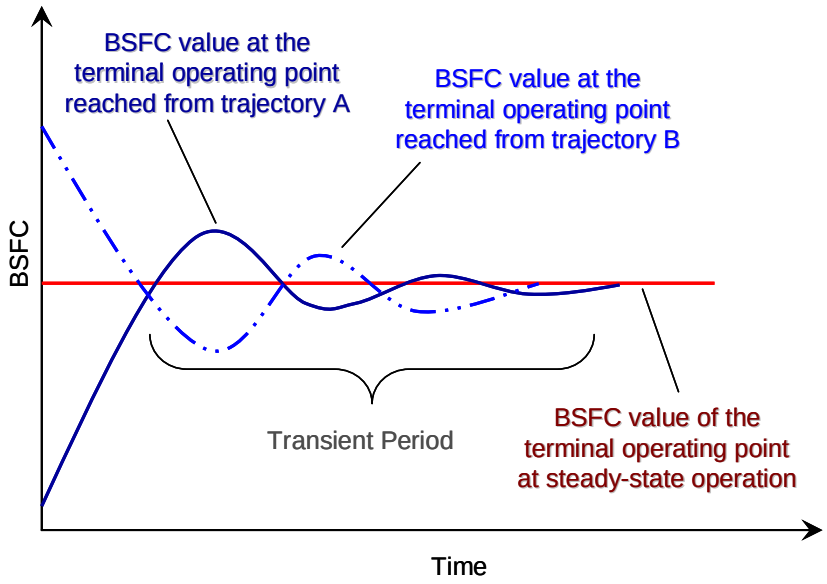

Figure 1.2 − BSFC value of the terminal engine operating point as reached from trajectories A, and B

1.3.1 Design of Experiments

Exhaustive investigation of the controllable variables with respect to all potential engine operating points requires a huge amount of testing. With the increasing number of these variables in current engine technologies, the required effort for testing grows exponentially. Design of Experiments (DoE) [20-23] is typically used to reduce the scope of the experiments required to derive the optimal engine calibration correlation under steady-state operating conditions. The main objective of this method is to expedite dynamometer tests significantly using a smaller subset of tests. This subset is utilized either in implementing engine calibration experimentally or in developing mathematical models for evaluating engine output. Using these models, optimization methods can determine the engine calibration static correlations between steady-state operating points and the controllable engine variables [24].

DoE has been widely used as the baseline calibration method for the last several years. Major applications include catalyst system optimization [25], optimization of variable valvetrains for performance and emissions [26-28], and implementation of

dynamic model-based engine calibrations. DoE employs statistics to generate a matrix of test points to explore the behavior or a physical system. The method relies on statistical data to determine a set of models that describe the system responses when some variables vary. In applying DoE to engine calibration, the objective is to define the coefficients of polynomial equations that can represent engine output over the range of the various controllable variables.

1.3.2 Dynamic Model-Based Calibration Systems

Dynamic model-based calibration methods utilize high-fidelity dynamic engine models. The data required to develop these models are obtained by operating the engine through a set of transient dynamometer tests while the engine calibration is perturbed in real time by a reconfigurable rapid prototyping control system. The predictive engine model produced in this fashion utilizes a combination of equation-based and neural network methods. DoE-experimental calibration is well suited only for steady-state engine operation over some driving cycle. In contrast, dynamic modeling produces a transient or dynamic engine model capable of predicting engine operating cycle. The steady-state optimal engine calibration can be produced from the transient engine model as a sub-set of the transient engine operation. Guerrier *et al.* [9] employed DoE and advanced statistical modeling to develop empirical models to enhance the powertrain control module calibration tables. Stuhler *et al.* [2] implemented a standardized and automated calibration environment, supporting the complexity of gasoline direct injection engines, for an efficient calibration process using an online DoE to decrease the calibration cost. Rask *et al.* [10] developed a dynamic-based calibration method to rapidly generate optimized maps for a V6 engine equipped with two-step variable valve actuation and intake cam phasing. Engine models employed in dynamic model-based calibration methods can predict engine output over transient operation within the data

utilized to calibrate the models. However, not all the correlations of optimal values of the controllable engine variables associated with the transient operating points can be quantified explicitly; to pre-specify the entire transient engine operation is impractical, and thus, engine calibration correlations cannot be optimized for these cases *a priori*.

1.3.3 Calibration using Artificial Neural Networks

Various approaches have been proposed for using artificial neural networks (ANNs) to promote modeling and calibration of engines [11, 29-32]. Neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. ANNs are application-specific and exhibit unpredictable behavior when previously unfamiliar data are presented to them. These difficulties increase if a nonlinear dynamic presentation of a system is to be realized, because of the increasing number of possibilities related to the dynamics and the interactions between the input signals. ANNs are suited for formulating objective functions, evaluating the specified engine performance indices with respect to the controllable engine variables and, thus, deriving the engine calibration correlations. They are computationally efficient for optimization requiring hundreds of function evaluations. However, optimal engine calibration for the entire engine operating domain is seldom guaranteed even for steady-state operating points. Moreover, the correlations between optimal values of the controllable engine variables and the transient operating points, overall, cannot be quantified explicitly, prohibiting *a priori* optimal engine calibration.

1.3.4 Simulation-Based Calibration Systems

Research efforts in addressing transient operation have focused on simulation-based methods to derive calibration maps for transients of particular driving cycles. Burk

et al. [33] presented the necessary procedures required to utilize co-simulation techniques with regard to predicting engine drive cycle performance for a typical vehicle. Jacquelin *et al*. [34] utilized analytical tools to run the FTP-75 driving cycle through pre-computed engine performance maps, depending on engine speed, load, intake and exhaust cam centerline positions. Atkinson *et al.* [14] implemented a dynamic system to provide optimal calibration for transient engine operation of particular driving cycles. These methods utilize engine models sufficiently accurate to portray fuel economy and feed-gas emissions during transient engine operation. However, identifying all possible transients, and thus deriving optimal values of the controllable variables through calibration maps for those cases *a priori*, is infeasible.

1.4 Research Objective

This dissertation reports research work towards implementing the theory and the algorithmic implementation that allow the engine of a vehicle to become an autonomous intelligent system. The engine should be able to realize in real time its optimal calibration with respect to both steady-state and transient operating points designated by the driver's driving style.

The problem of deriving the optimal values of the controllable variables for engine operating point transitions compromises of two major sub-problems. The first concerns exploitation of the information acquired from the engine operation to identify its behavior, that is, how an engine representation can be built by observing engine operating point transitions. In control theory, this is addressed as a state estimation and system identification problem. The second concerns assessing the engine output with respect to alternative values of the controllable variables (control policies), and selecting those that optimize specified engine performance indices. This forms a stochastic control problem.

In this context, the engine is treated as a controlled stochastic system and engine calibration is formulated as a sequential decision-making problem under uncertainty.

The research objective is to make the engine capable of learning the optimal values of the engine variables in real time while the driver drives the vehicle. The engine progressively perceives the driver's driving style and eventually learns the optimal calibration for this driving style. The longer the engine runs during a particular driving style, the better the engine's specified performance criteria will be. The engine's ability to learn its optimum operation is not limited, however, to a particular driving style. The engine can learn to operate optimally for different drivers, although the drivers should indicate their identities before starting the vehicle. The engine can then adjust its calibration to be optimal for a particular driver based on what it has learned in the past regarding his/her driving style.

The ultimate goal of the research work reported here is to fully exploit the engine's given technology in terms of the maximum specified performance criteria, e.g., engine power, fuel economy, and pollutant emissions that can be achieved. It aims to provide an answer to the following question: "For an engine with a given technology, what are the maximum performance criteria that a driver can get with respect to his/her driving habits?"

1.5 Outline of Dissertation

The dissertation is organized as follows. Chapter 2 presents the theory for building computational models suitable for real-time sequential decision-making under uncertainty. These models are essential traits of any intelligent or rational system that selects control actions after every perception, so that a long-term reward (cost) is maximized (minimized). Research efforts in implementing these models are reviewed in that chapter. Chapter 3 portrays a real-time computational learning model suitable to

solve the state estimation and system identification sub-problem. Chapter 4 introduces the algorithmic structure of the decision-making mechanism suitable for real-time implementation. The algorithm solves the stochastic control sub-problem by utilizing accumulated data acquired over the learning process of the computational model. The increase of the problem's dimensionality, when more than one controllable variable is considered, is addressed by a decentralized learning control scheme presented in Chapter 5. This scheme draws from multi-agent learning research in a range of areas, including reinforcement learning, and game theory, to coordinate optimal behavior among the various controllable variables. The engine is considered as a cooperative multi-agent system, in which the subsystems, i.e., controllable variables, are treated as autonomous rational agents who strive interactively and jointly to optimize engine performance indices. In Chapter 6, the research contributions are summarized and future work is proposed. Relevant references are included at the end of each chapter.

1.6 References

[1] Roepke, K. and Fischer, M., "Efficient Layout and Calibration of Variable Valve Trains," SAE Transactions-Journal of Engines, v. 110, 2001, SAE 2001-01-0668.

[2] Stuhler, H., Kruse, T., Stuber, A., Gschweitl, K., Piock, W., Pfluegl, H., and Lick, P., "Automated Model-Based GDI Engine Calibration Adaptive Online DoE Approach," SAE 2002 World Congress, Detroit, Michigan, March 3-7, 2002, SAE 2002-01-0708.

[3] Brooks, T., Lumsden, G., and H.Blaxill, "Improving Base Engine Calibrations for Diesel Vehicles Through the Use of DoE and Optimization Techniques," Powertrain and Fluid Systems Conference and Exhibition, San Antonio, Texas, USA, September 24-27, 2005, SAE 2005-01-3833.

[4] Knafl, A., Hagena, J. R., Filipi, Z. S., and Assanis, D. N., "Dual-Use Engine Calibration: Leveraging Modern Technologies to Improve Performance-Emissions Tradeoff," SAE Word Congress, Detroit, Michigan, April 11-14, 2005, SAE 2005-01-1549.

[5] Steiber, J., Trader, A., Reese, R., Bedard, M., Musial, M., and Treichel, B., "Development of an Engine Test Cell for Rapid Evaluation of Advanced Powertrain Technologies Using Model-Controlled Dynamometers," SAE Transactions-Journal of Passenger Cars- Electronics & Electrical Systems, v. 115, 2006, SAE 2006-01-1409.

[6] Schuette, H. and Ploeger, M., "Hardware-in-the-Loop Testing of Engine Control Units - A Technical Survey," SAE 2007 World Congress and Exhibition, Detroit, Michigan, April 16-19, 2007, SAE 2007-01-0500.

[7] Philipp, O., Buhl, M., Diehl, S., Huber, M., Roehlich, S., and Thalhauser, J., "Engine ECU Function Development Using Software-in-the-Loop Methodology," SAE 2005 World Congress and Exhibition, Detroit, Michigan, 2005, SAE 2005-01-0049.

[8] Caraceni, A., Cristofaro, F. D., Ferrana, F., and Scala, S., "Benefits of Using a Real-Time Engine Model During Engine ECU Development," SAE 2003 World Congress and Exhibition, March 3-6, 2003, SAE 2003-01-1049.

[9] Guerrier, M. and Cawsey, P., "The Development of Model-Based Methodologies for Gasoline IC Engine Calibration," SAE Transactions-Journal of Engines, v. 113, 2004, SAE 2004-01-1466.

[10] Rask, E. and Sellnau, M., "Simulation-Based Engine Calibration: Tools, Techniques, and Applications," SAE Transactions-Journal of Engines, v. 113, 2004, SAE 2004-01-1264.

[11] Wu, H., "Decentralized Iterative Learning Control for a Class of Large Scale Interconnected Dynamical Systems," Journal of Mathematical Analysis and Applications, vol. 327, pp. 233-45, 2007.

[12] Jankovic, M. and Magner, S., "Fuel Economy Optimization in Automotive Engines," Proceedings of the 2006 American Control Conference, Minneapolis, MN, USA, 2006.

[13] Hagena, J. R., Filipi, Z. S., and Assanis, D. N., "Transient Diesel Emissions: Analysis of Engine Operation During a Tip-In," SAE 2006 World Congress, Detroit, Michigan, April 3-6, 2006, SAE 2006-01-1151.

[14] Atkinson, C. and Mott, G., "Dynamic Model-Based Calibration Optimization: An Introduction and Application to Diesel Engines," SAE World Congress, Detroit, Michigan, April 11-14, 2005, SAE 2005-01-0026.

[15] Rakopoulos, C. D., Giakoumis, E. G., Hountalas, D. T., and Rakopoulos, D. C., "The Effect of Various Dynamic, Thermodynamic and Design Parameters on the Performance of a Turbocharged Diesel Engine Operating under Transient Load Conditions," SAE 2004 World Congress and Exhibition, Detroit, Michigan, April 8-11, 2004, SAE 2004-01-0926.

[16] Wijetunge, R. S., Brace, C. J., Hawley, J. G., Vaughan, N. D., Horroocks, R. W., and Bird, G. L., "Dynamic Behavior of a High-Speed, Direct-Injection Diesel Engine," SAE Transactions-Journal of Engines, v. 108, 1999, SAE 1999-01-0829.

[17] Clark, N. N., Gautam, M., Rapp, B. L., Lyons, D. W., Graboski, M. S., McCormick, R. L., Alleman, T. L., and National, P. N., "Diesel and CNG Transit Bus Emissions Characterization by Two Chassis Dynamometer Laboratories: Results and Issues," SAE Transactions-Journal of Fuels and Lubricants, v. 108, 1999, SAE 1999-01-1469.

[18] Samulski, M. J. and Jackson, C. C., "Effects of Steady-State and Transient Operation on Exhaust Emissions from Nonroad and Highway Diesel Engines," SAE Transactions-Journal of Engines, v. 107, 1998, SAE 982044.

[19] Green, R. M., "Measuring the Cylinder-to-Cylinder EGR Distribution in the Intake of a Diesel Engine During Transient Operation," SAE Transactions-Journal of Engines, v. 109, 2000, SAE 2000-01-2866.

[20] Clarke, G. M. and Kempson, R. E., Introduction to the Design and Analysis of Experiments, Hodder Arnold, November 1996.

[21] Hicks, C. R. and Turner, K. V., Fundamental Concepts in the Design of Experiments, 5th edition, Oxford University Press, USA, March 1999.

[22] Diamond, W. J., Practical Experiment Designs : for Engineers and Scientists, 3rd edition, John Wiley & Sons, February 2001.

[23] Montgomery, D. C., Design and Analysis of Experiments, 4th edition, John Wiley and Sons, 1997.

[24] Papalambros, P. Y. and Wilde, D. J., Principles of Optimal Design: Modeling and Computation, 2nd edition, Cambridge University Press, July 2000.

[25] Edwards, S. P., Grove, D. M., and Wynn, H. P., Statistics for Engine Optimization, 1st edition, John Wiley & Sons Canada, December 2000.

[26] Amann, M., Buckingham, J., and Kono, N., "Evaluation of HCCI Engine Potentials in Comparison to Advanced Gasoline and Diesel Engines," Powertrain and Fluid Systems Conference and Exhibition, Toronto, Ontario, Canada, September 16-19, 2006, SAE 2006-01-3249.

[27] Ghauri, A., Richardson, S. H., and Nightingale, C. J. E., "Variation of Both Symmetric and Asymmetric Valve Events on a 4-Valve SI Engine and the Effects on Emissions and Fuel Economy," SAE 2000 World Congress, Detroit, Michigan, March 6-9, 2000, SAE 2000-01-1222.

[28] Regner, G., Teng, H., Wieren, P. V., Park, J. I., Park, S. Y., and Yeom, D. J., "Performance Analysis and Valve Event Optimization for SI Engines Using Fractal Combustion Model," Powertrain and Fluid Systems Conference and Exhibition, Toronto, Ontario, Canada, September 16-19, 2006, SAE 2006-01-3238.

[29] Ayeb, M., Theuerkauf, H. J., and Winsel, T., "SI Engine Emissions Model Based on Dynamic Neural Networks and D- Optimality," SAE World Congress, Detroit, Michigan, April 11-14, 2005, SAE 2005-01-0019.

[30] Brahma, I., He, Y., and Rutland, C. J., "Improvement of Neural Network Accuracy for Engine Simulations," Powertrain and Fluid Systems Conference and Exhibition, Pittsburgh, Pennsylvania, September 27-30, 2003, SAE 2003-01-3227.

[31] Lowe, D. and Zapart, K., "Validation of Neural Networks in Automotive Engine Calibration," Proceedings of the Fifth International Conference on Artificial Neural Networks, pp. 221-6, Cambridge, UK, 1997.

[32] Meyer, S. and Greff, A., "New Calibration Methods and Control Systems with Artificial Neural Networks," SAE 2002 World Congress, Detroit, Michigan, March 4-7, 2002, SAE 2002-01-1147.

[33] Burk, R., Jacquelin, F., and Wakeman, R., "A Contribution to Predictive Engine Calibration Based on Vehicle Drive Cycle Performance," SAE 2003 World Congress, Detroit, Michigan, USA, March 3-6, 2003, SAE 2003-01-0225.

[34] Jacquelin, F., Burk, R., and Wakeman, R. J., "Cam Phaser Actuation Rate Performance Impact on Fuel Consumption and NOx Emissions Over the FTP-75 Drive Cycle," SAE 2003 World Congress, Detroit, Michigan, USA, 2003, SAE 2003-01-0023.

CHAPTER 2

SYSTEM IDENTIFICATION AND STOCHASTIC CONTROL

This chapter presents the underlying theory for building computational models suitable for real-time sequential decision-making under uncertainty. The engine is treated as a controlled stochastic system and engine calibration is formulated as a sequential decision-making problem under uncertainty. The goal is to make the engine an autonomous intelligent system that can select the values of the controllable variables in real time, for each engine operating point transition, which optimize specified engine performance criteria. In essence, we seek an optimal calibration that can be achieved for steady-state and transient engine operating points resulting from the driver's driving style.

2.1 Modeling Engine Operation as a Stochastic Process

Engines are streamlined syntheses of complex physical processes determining a convoluted dynamic system. They are operated with reference to engine operating points and the values of various engine controllable variables. At each operating point, these values highly influence engine performance criteria, e.g., fuel economy, emissions, or acceleration. This influence becomes more important at engine operating point transitions designated partly by the driver's driving style and partly by the engine's controllable variables. Consequently, the engine is a system whose behavior is not completely

foreseeable, and its future evolution (operating point transitions) depends on the driver's driving style.

We seek a method to derive the optimal values of the controllable variables for each engine operating point transition that optimize engine performance criteria. These values are selected at points of time referred to as decision epochs (or stages), when the time domain can be either discrete or continuous. In our case, discrete time is employed because of the discreteness in the values of the controllable variables (control actions). The engine output is sampled at the decision epochs.

The engine performance criteria are treated as controlled random functions, the engine is treated as a controlled stochastic system, and engine operation is modeled as a stochastic process. The problem of engine calibration is thus reformulated as a sequential decision-making problem under uncertainty. The goal is to select the values of the controllable variables for each engine operating point in real time that optimize the random functions representing the engine performance criteria.

2.2 Sequential Decision-Making Problems Under Uncertainty

Sequential decision models [1, 2] are mathematical abstractions of situations in which decisions must be made in several decision epochs while incurring a certain cost (or reward) at each epoch. Each decision may influence the circumstances under which future decisions will be made, and thus, the decision maker must balance his/her desire to minimize (maximize) the cost (reward) of the present decision against his/her desire to avoid future situations where high cost is inevitable.

An example of a decision-making process involves portfolio management. An investor must balance his/her desire to achieve immediate return against a desire to avoid investments in areas where low long-term yield is probable. The current decision will depend also on his/her assessment of the areas with high long-term profit. This

assessment is a problem of state estimation, and the decision to be made by the decision maker is formulated as a stochastic control problem, which requires a solution of the estimation problem first. The prediction of areas with high profit can be expressed as the conditional expectation $E\{r_{k+1} \mid r_k\}$ of the next period's value given the present period's value. Consequently, the state estimation problem requires the knowledge of the probability distribution of the value of the areas to be invested. The estimation of the relevant probability distribution is the problem of identification (or system identification). In most situations of decision-making in a stochastic environment, the problems of identification, estimation, and control are all tied together. The deterministic and stochastic system models aim to provide the mathematical framework for analysis of the sequential decision-making problem in deterministic and stochastic environments, respectively.

2.2.1 Deterministic System Model

A broadly applicable model of discrete-time deterministic optimal control of a dynamic system over a finite number of stages $M \in \mathbb{N}$ (a finite horizon) has two principal features: (a) an underlying discrete-time dynamic system, and (b) a cost function that is additive over time. The dynamic system expresses the evolution of the system's "state," under the influence of decisions made at discrete instances of time. The deterministic dynamic system [3] is described by the general system equation

$$\mathbf{s}_{k+1} = \mathbf{f}_k(\mathbf{s}_k, \boldsymbol{\alpha}_k),\ k = 0,1,...,M\text{-}1, \tag{2.1}$$

where $\mathbf{s}_k \in \mathbb{R}^n$ is the column vector of system's states, which belong to some state space $\mathcal{S}$, and $\boldsymbol{\alpha}_k \in \mathbb{R}^m$ is the input at time k; $\boldsymbol{\alpha}_k$ represents the vector of control actions chosen by the decision maker from some feasible action set $A(s_k)$, which is a subset of some control space $\mathcal{A}$. The system output is represented by the equation

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{s}_k, \boldsymbol{\alpha}_k),\ k = 0,1,...,M\text{-}1, \tag{2.2}$$

where $\mathbf{y}_k \in \mathbb{R}^p$.

Without loss of generality, the following discussion will refer to the one-dimensional deterministic system model, namely,

$$s_{k+1} = f_k(s_k, \alpha_k),\ k = 0,1,...,M\text{-}1, \tag{2.3}$$

$$y_k = h_k(s_k, \alpha_k). \tag{2.4}$$

An important property of the system described by Eq. (2.3) is that the current state s_k and the sequence of control actions $a_k, a_{k+1}, ..., a_{k+m}$ determine the state s_{k+m+1} independently of the past values of state and control actions $s_{k-1}, s_{k-2}, ..., a_{k-1}, a_{k-2}, ...$, that is, there is a function $f_{k+m+1,k}$ such that

$$s_{k+m+1} = f_{k+m+1,k}(s_k, a_k, ..., a_{k+m}). \tag{2.5}$$

The cost incurred at the *k*th decision epoch is given by a function $c(s_k, a_k)$. We seek a finite sequence of functions $\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$, defined as a control policy, which minimizes the total cost over *M* decision epochs. The functions μ_k specify the control $a_k = \mu(s_k)$ that will be chosen when at *k*th decision epoch the state is s_k. Consequently, the total cost corresponding to a policy $\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$, and initial state s_0 is given by

$$J^{\pi}(s_0) = \sum_{k=0}^{M-1} c(s_k, \mu(s_k)), \tag{2.6}$$

where the states $s_1, s_2, ..., s_{M-1}$ are generated from s_0 and π via the systems equation

$$s_{k+1} = f_k(s_k, \mu(s_k)), \quad k = 0, 1, ..., M-1. \tag{2.7}$$

At each initial state s_0 and π, there is a corresponding sequence of control actions $a_0, a_1, ..., a_{M-1}$, where $a_k = \mu(s_k)$ and s_k is generated by Eq. (2.7). An alternative formulation of the problem includes the selection of a sequence of control actions, rather than a policy π, minimizing the following total cost

$$J(s_0) = \sum_{k=0}^{M-1} c(s_k, a_k). \tag{2.8}$$

The deterministic optimal control problem is representative of a plethora of sequential decision-making problems of practical interest, and it constitutes the basis of the stochastic system model.

2.2.2 Stochastic System Model

The stochastic system model [3, 4] establishes the mathematical framework for the representation of dynamic systems such as engines that evolve stochastically over time. The discrete-time stochastic optimal control problem is obtained from the deterministic problem when the system includes a stochastic disturbance or noise at time k, w_k, in its portrayal. Consequently, Eq. (2.3) is replaced by the equation

$$s_{k+1} = f_k(s_k, \alpha_k, w_k), \quad k = 0, 1, ..., M-1. \tag{2.9}$$

The sequence $\{w_k, k \geq 0\}$ is a stochastic process with a given probability law; that is, the joint probability distribution of the random variables $w_0, w_1, ..., w_k$ is known for

each k. When the state is not directly observed, it is necessary to augment the state Eq. (2.9) with the equation

$$y_k = h_k(s_k, v_k),\ k = 0,1,...,M\text{-}1, \tag{2.10}$$

where y_k is the observation or system's output and v_k is the measurement error or noise. The sequence $\{v_k, k \geq 0\}$ is a stochastic process with known probability distribution. If in Eq. (2.10) $h_k(s_k, v_k) = s_k$, then the system is completely observed, namely, $y_k = s_k$, whereas if $y_k \neq s_k$ the system is partially observed.

The system's state s_k depends upon the input sequence $a_0, a_1, ..., a_{M-1}$ as well as the random variables $w_0, w_1, ..., w_k$, Eq. (2.9). Consequently, s_k is a random variable; the system output $y_k = h_k(s_k, v_k)$ is a function of the random variables $s_0, s_1, ..., v_0, v_1, ...$, and thus, is also a random variable. Similarly, the sequence of control actions $a_k = \mu(s_k)$, $\{a_k, k \geq 0\}$, constitutes a stochastic process.

Definition 2.1 [4]. The random variables $s_0, w_0, w_1, ..., v_0, v_1, ...$, are addressed as basic random variables, since the sequences $\{s_k, k \geq 0\}$ and $\{a_k, k \geq 0\}$ are constructed from them.

As shown in the previous section, the deterministic system model, Eq. (2.3), imposes the property that the state s_{k+1} at time k+1 is completely determined once s_k and a_k are known. It is desirable for the stochastic system model, Eq. (2.9), to retain an analogous property by imposing a condition directly on the basic random variables. That is, whether the conditional probability distribution of s_{k+1} given s_k and a_k is independent of previous values of states and control actions. Suppose the control policy $\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$ is employed. The corresponding stochastic processes $\{s_k^\pi, k \geq 0\}$, $\{y_k^\pi, k \geq 0\}$, and $\{a_k^\pi, k \geq 0\}$, are defined by

$$s_{k+1}^\pi = f_k(s_k^\pi, a_k^\pi, w_k),\ s_0^\pi = s_0, \tag{2.11}$$

$$y_k^\pi = h_k(s_k^\pi, v_k).$$

Suppose further that the values realized by the random variables s_k and a_k are known. These values are insufficient to determine the value of s_{k+1} since w_k is not known. The value of s_{k+1} is statistically determined by the conditional distribution of s_{k+1} given s_k and a_k, namely

$$P^\pi_{s_{k+1}|s_k,a_k}(\cdot \mid s_k, a_k). \tag{2.12}$$

For any subset $\mathcal{S}_{k+1}$, that is, the state space at time *k*+1, and from Eq. (2.11), we have

$$P^\pi_{s_{k+1}|s_k,a_k}(\mathcal{S}_{k+1} \mid s_k, a_k) = P^\pi_{w_k|s_k,a_k}(\mathcal{W}_k \mid s_k, a_k), \tag{2.13}$$

where $\mathcal{W}_k := \{w \mid f_k(s_k, a_k, w) \in \mathcal{S}_k\}$.

The interpretation of Eq. (2.13) is that the conditional probability of reaching the state space $\mathcal{S}_{k+1}$ at time *k*+1 given s_k and a_k is equal to the probability of being at the disturbance space $\mathcal{W}_k$ at time *k*. Suppose that the previous values of the random variables s_m and a_m, $m \le k-1$ are known. The conditional distribution of s_{k+1} given these values will be

$$P^\pi_{s_{k+1}|s_k,a_k}(\mathcal{S}_{k+1} \mid s_k, ..., s_0, a_k, ..., a_0) = P^\pi_{w_k|s_k,a_k}(\mathcal{W}_k \mid s_k, ..., s_0, a_k, ..., a_0). \tag{2.14}$$

The conditional probability distribution of s_{k+1} given s_k and a_k can be independent of the previous values of states and control actions, if it is guaranteed that for every control policy π, w_k is independent of the random variables s_m and a_m, $m \le k-1$.

Kumar and Varaiya [4] proved that this property is imposed under the following assumption.

Assumption 2.1. The basic random variables $s_0, w_0, w_1, ..., v_0, v_1, ...,$ are all independent.

Assumption 2.1 imposes a condition directly to the basic random variables which eventually yields that the state s_{k+1} depends only on s_k and a_k. Moreover, the conditional probability distributions do not depend on the control policy π, and thus, the suuperscript π can be dropped

$$P^{\pi}_{s_{k+1}|s_k,a_k}(s_{k+1} \mid s_k, ..., s_0, a_k, ..., a_0) = P^{\pi}_{w_k|s_k,a_k}(s_{k+1} \mid s_k, a_k), \tag{2.15}$$

or $P_{s_{k+1}|s_k,a_k}(s_{k+1} \mid s_k, ..., s_0, a_k, ..., a_0) = P_{s_{k+1}|s_k,a_k}(s_{k+1} \mid s_k, a_k).$

A stochastic process $\{s_k, k \geq 0\}$ satisfying the condition of Eq. (2.15) is called a *Markov Process* and the condition is addressed as a *Markov property*.

Definition 2.2 [5]. A Markov process is a random process $\{s_k, k \geq 0\}$ with the property that given the values of the process from time zero up through the current time, the conditional probability of the value of the process at any future time depends only on its value at the current time. That is, the future and past are conditionally independent given the present.

Definition 2.3 [6]. When the state of a Markov process is discrete, then the process is called a Markov chain.

A large class of sequential decision-making problems under uncertainty can be modeled as a Markov Decision Process (MDP). MDP [7] provides the mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of the decision maker. Decisions are made at points of time referred to as decision epochs, and the time domain can be either discrete or continuous.

2.3 Markov Decision Process

The Markov decision process model consists of five elements: (a) decision epochs; (b) states; (c) actions; (d) the transition probability matrix; and (e) the transition cost (or reward) matrix. In this framework, the decision maker is faced with the problem of influencing system behavior as it evolves over time, by making decisions (choosing actions). The objective of the decision maker is to select the course of action (control policy) which causes the system to perform optimally with respect to some predetermined optimization criterion. Decisions must anticipate costs (or rewards) associated with future system states-actions.

At each decision epoch *k*, the system occupies a state $s_k = i$ from the finite set of all possible system states $\mathcal{S}$

$$\mathcal{S} = \{1, 2, ..., N\},\ N \in \mathbb{N}. \tag{2.16}$$

In this state $s_k \in \mathcal{S}$, the decision maker has available a set of allowable actions, $\alpha_k \in A(s_k),\ A(s_k) \subseteq \mathcal{A}$, where $\mathcal{A}$ is the finite action space

$$\mathcal{A} = \bigcup_{s_k \in \mathcal{S}} A(s_k). \tag{2.17}$$

The decision-making process occurs at each of a sequence of decision epochs $k = 0, 1, 2, ..., M$, $M \in \mathbb{N}$. At each epoch, the decision maker observes a system's state $s_k = i, i \in \mathcal{S}$, and executes an action $\alpha_k \in A(s_k)$, from the feasible set of actions $A(s_k) \subseteq \mathcal{A}$ at this state. At the next epoch, the system transits to the state $s_{k+1} = j, j \in \mathcal{S}$ imposed by the conditional probabilities $p(s_{k+1} = j \mid s_k = i, \alpha_k)$, designated by the transition probability matrix $\mathbf{P}(\cdot,\cdot)$. The conditional probabilities of $\mathbf{P}(\cdot,\cdot)$, $p : \mathcal{S} \times \mathcal{A} \to [0,1]$, satisfy the constraint

$$\sum_{j=1}^{N} p(s_{k+1} = j \mid s_k = i, \alpha_k) = 1, \tag{2.18}$$

where N is the cardinality of $\mathcal{S}$, $N = |\mathcal{S}|$.

Following this state transition, the decision maker receives a cost associated with the action α_k, $R(s_{k+1} = j \mid s_k = i, \alpha_k), R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as imposed by the transition cost matrix $\mathbf{R}(\cdot,\cdot)$.

2.3.1 The Cost of a Markov Control Policy

The solution to an MDP can be expressed as an admissible control policy so that a given performance criterion is optimized over all admissible policies Π. An admissible policy consists of a sequence of functions

$$\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}, \tag{2.19}$$

where μ_k maps states s_k into actions $\alpha_k = \mu_k(s_k)$ and is such that $\mu_k(s_k) \in A(s_k), \forall s_k \in \mathcal{S}$.

A Markov policy π determines the probability distribution of state process $\{s_k, k \geq 0\}$ and the control process $\{a_k, k \geq 0\}$. Different policies will lead to different probability distributions. In optimal control problems, the objective is to derive the optimal control policy that minimize (maximize) the accumulated cost (reward) incurred at each state transition per decision epoch. If a policy π is fixed, the cost incurred by π when the process starts from an initial state s_0 and up to the time horizon M is

$$J^{\pi}(s_0) = \sum_{k=0}^{M-1} R_k(s_{k+1} = j \mid s_k = i, a_k), \tag{2.20}$$

$\forall i, j \in \mathcal{S}, \forall a_k \in A(s_k).$

The accumulated cost $J_\pi(s_0)$ is a random variable since s_k and a_k are random variables. Hence the expected accumulated cost of a Markov policy is given by

$$J^\pi(s_0) = \underset{\substack{s_k \in S \\ a_k \in A(s_k)}}{E} \{\sum_{k=0}^{M-1} R_k(s_{k+1} = j \mid s_k = i, a_k)\} = \underset{\substack{s_k \in S \\ a_k \in A(s_k)}}{E} \{\sum_{k=0}^{M-1} R_k(s_{k+1} = j \mid s_k = i, \mu_k(s_k))\}. \quad (2.21)$$

The expectation is with respect to the probability distribution of $\{s_k, k \geq 0\}$ and $\{a_k, k \geq 0\}$ determined by the Markov policy π. Eq. (2.21) can readily be evaluated in terms of the transition probability matrix as follows:

$$J^\pi(s_0) = \sum_{k=0}^{M-1} \sum_{j=1}^{N} P_k(s_{k+1} = j \mid s_k = i, a_k) \cdot R_k(s_{k+1} = j \mid s_k = i, a_k). \quad (2.22)$$

Consequently, the control policy that minimizes Eq. (2.22) is defined as the optimal Markov policy π^*. Dynamic programming (DP) has been widely employed as the principal method for computing global optimal policies in sequential decision-making problems under uncertainty. Algorithms, such as value iteration, policy iteration, and linear programming, have been extensively utilized in solving deterministic and stochastic optimal control problems, Markov and semi-Markov decision problems, min-max control problems, and sequential games. While the nature of these problems may vary significantly, their underlying structures are very similar.

2.4 Dynamic Programming Algorithm

The Dynamic Programming (DP) [8] algorithm rests on the *principle of optimality*. The principle of optimality states the following fact [1]. Let $\pi^* = \{\mu_0^*, \mu_1^*, ..., \mu_{M-1}^*\}$ be an optimal policy for a finite decision-making problem, and

assume that, when using π^*, a given state $s_k = i$ occurs at time k with positive probability. Consider the sub-problem whereby the process occupies the state $s_k = i$ at time k and wishes to minimize the accumulated cost from time k to time M, namely,

$$\underset{\substack{s_k \in S \\ a_k \in A(s_k)}}{E} \{R_M(s_M) + \sum_{k=k}^{M-1} R_k(s_{k+1} \mid s_k, a_k)\}, \tag{2.23}$$

where $R_M(s_M) = R_M(s_{M+1} \mid s_M, a_M)$.

Then the truncated policy $\{\mu_k^*, \mu_{k+1}^*, ..., \mu_{M-1}^*\}$ is optimal for the sub-problem.

The principle of optimality essentially suggests that an optimal policy can be constructed in piecemeal fashion, first constructing an optimal policy for the "tail sub-problem" involving the last decision epoch, then extending the optimal policy to the "tail sub-problem" involving the last two decision epochs, and continuing in this manner until an optimal policy for the entire problem is derived. The DP algorithm is founded on the principle of backward induction. It proceeds sequentially, by solving all the tail sub-problems of a given number of decision epochs, utilizing the solution of the tail sub-problems of shorter number of decision epochs. The DP algorithm is stated as follows. For every initial state s_0, the optimal cost $J^{\pi^*}(s_0)$, given by Eq. (2.22), of the sequential decision-making problem under uncertainty is equal to $J_0(s_0)$, given by the last step of the following algorithm, which proceeds backward in time from the decision epoch M-1 to decision epoch 0:

$$J_M(s_M) = R_M(s_M) = R_M(s_{M+1} \mid s_M, a_M), \tag{2.24}$$

$$J_k(s_k) = \min_{a_k \in A(s_k)} \underset{s_k \in S}{E} \{R_M(s_M) + J_{k+1}(f_k(s_k, a_k, w_k))\}, k = 0, 1, ..., M-1.$$

2.4.1 Optimal Control Policy

The optimal policy $\pi^* = \{\mu_0^*, \mu_1^*, ..., \mu_M^*\}$ for the finite horizon problem can be derived by

$$\pi^* = \arg\min_{\pi \in \Pi} J_0(s_0). \tag{2.25}$$

The finite-horizon model is appropriate when the decision-maker's "lifetime" is known, namely, the terminal epoch of the decision-making sequence. For problems with a very large number of decision epochs, however, the infinite-horizon model is more appropriate. In this context, the overall expected undiscounted cost is:

$$J_0(s_0) = \lim_{M \to \infty} \min_{\pi \in \Pi} J_0(s_0). \tag{2.26}$$

This relation is valuable computationally and analytically, and it holds under certain conditions [3].

2.5 Engine Identification and Stochastic Control: Problem Definition

The problem of deriving the optimal values of the controllable variables for engine operating transitions involves two major sub-problems. The first is exploitation of the information acquired from the engine output to identify its behavior. That is, how an engine representation can be built by observing engine operating point transitions is the state estimation and system identification problem. The second is assessment of the engine output with respect to alternative values of the controllable variables (control policies), and selecting those that optimize specified engine performance criteria, e.g., fuel economy, emissions, engine power, etc. The latter forms a stochastic control problem. Although computational considerations in many instances lead us to treat these three aspects separately, in our approach they are considered simultaneously in solving the engine calibration problem in real time, while the engine is running the vehicle.

The stochastic system model, Eq. (2.9), can provide a systematic treatment of the general engine identification and stochastic control problem. The model adapted to the engine, as illustrated in Figure 2.1, is repeated here for easy reference

$$s_{k+1} = f_k(s_k, a_k, w_k),\ k = 0,1,...,M-1, \tag{2.27}$$

$$y_k = h_k(s_k, v_k),$$

where s_k represents the engine operating point, which belongs to the finite engine operating space $\mathcal{S}$, and a_k represent the values of the controllable variables at time *k*. These values belong to some feasible action set $A(s_k)$, which is a subset of the control space $\mathcal{A}$. The sequence $\{w_k, k \geq 0\}$ is an unknown disturbance representing the driver while commanding the engine through the gas pedal position. This sequence is treated as a stochastic process with an unknown probability distribution; y_k is the observation or engine's output, and v_k is the measurement sensor error or noise. The sequence $\{v_k, k \geq 0\}$ is a stochastic process with unknown probability distribution.

The initial engine operating point s_0 along with the sequences of gas pedal position $\{w_k, k \geq 0\}$, and the sensor error $\{v_k, k \geq 0\}$ are assumed to be independent, which is a reasonable assumption in reality. In this context, the engine is treated as a stochastic system and engine operation can be considered as a stochastic process $\{s_k, k \geq 0\}$ which satisfies the condition of Eq. (2.15). Consequently, engine operation can be modeled as a Markov decision process with the cost function at each state (engine operating point) transition to be represented by the engine output (e.g., fuel economy, emissions, and engine power). The problem of engine calibration is thus formulated as a sequential decision-making problem under uncertainty.

At each decision epoch *k*, the engine operates at a given state s_k designated partly by the driver w_{k-1}, and partly by the controllable variable a_{k-1} at time *k*-1. On that basis

the self-learning controller selects a value a_k. One decision epoch later, the engine transits to a new state s_{k+1} and the controller observes the engine output $y_{k+1} = h_{k+1}(s_{k+1}, v_{k+1}) = R_k(s_{k+1} \mid s_k, a_k)$ associated with this state transition, as illustrated in Figure 2.2 .

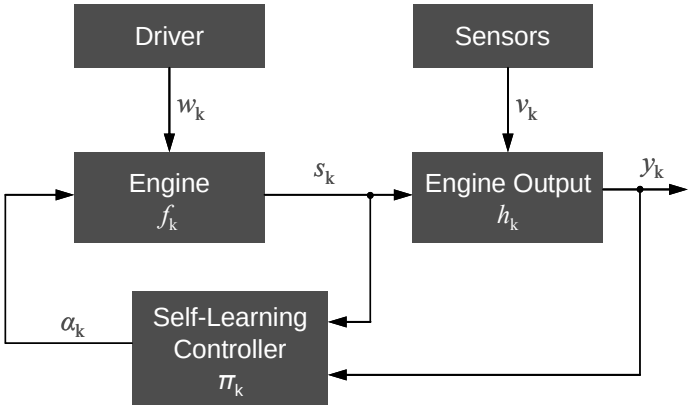

Figure 2.1 − The stochastic system model adapted for the engine calibration problem.

The controller (decision maker) is faced with the problem of influencing engine operation as it evolves over time by selecting values of the controllable variables. The goal of the controller is to select the optimal control policy (optimum values of the controllable variables) for the sequences of engine operating point transitions, corresponding to the driver's driving style, that cause the engine to perform optimally with respect to some predetermined performance criterion (cost function).

A key aspect of this process is that decisions are not viewed in isolation since the controller simultaneously solves the state estimation and system identification sub-problem by using the conditional probabilities of the sequence $\{s_k, k \geq 0\}$ given the sequence $\{a_k, k \geq 0\}$. Consequently, in the stochastic control problem the self-learning controller can select those values that balance the desire to minimize the cost function of the next engine operating transition against the desire to avoid future operating point

transitions where high cost is inevitable. This approach aims to provide engine calibration that can capture transient engine designated by the driver's driving style.

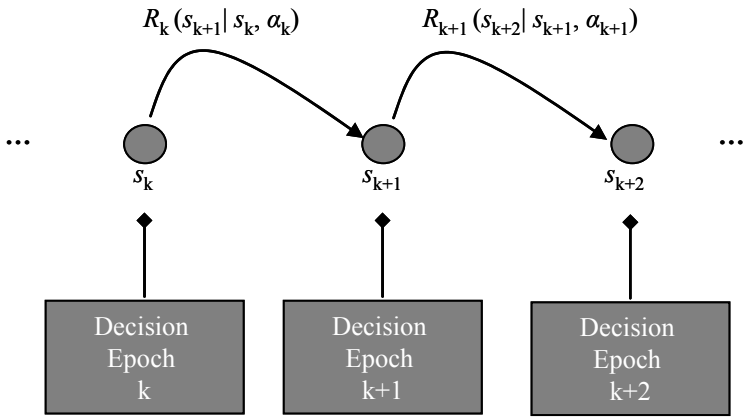

Figure 2.2 − Sequential decision-making problem.

2.6 References

[1] Bertsekas, D. P., Dynamic Programming and Optimal Control (Volumes 1 and 2), Athena Scientific, September 2001.

[2] Bertsekas, D. P. and Shreve, S. E., Stochastic Optimal Control: The Discrete-Time Case, 1st edition, Athena Scientific, February 2007.

[3] Bertsekas, D. P. and Tsitsiklis, J. N., Neuro-Dynamic Programming (Optimization and Neural Computation Series), 1st edition, Athena Scientific, May 1996.

[4] Kumar, P. R. and Varaiya, P., Stochastic Systems, Prentice Hall, June 1986.

[5] Kemeny, J. G. and Snell, J. L., Finite Markov Chains, 1st edition, Springer, December 5, 1983.

[6] Krishnan, V., Probability and Random Processes, 1st edition, Wiley-Interscience, July 11, 2006.

[7] Puterman, M. L., Markov Decision Processes: Discrete Stochastic Dynamic Programming, 2nd Rev. edition, Wiley-Interscience, 2005.

[8] Bellman, R., Dynamic Programming. Princeton, NJ, Princeton University Press, 1957.

CHAPTER 3

REAL-TIME, SELF-LEARNING SYSTEM IDENTIFICATION

Modeling dynamic systems incurring stochastic disturbances for deriving a control policy is a ubiquitous task in engineering. However, in some instances obtaining a model of a system may be impractical or impossible. Alternative approaches employ a simulation-based stochastic framework, in which the system interacts with its environment in real time and obtains information that can be processed to produce an optimal control policy. In this context, the problem of developing a policy for controlling the system's behavior is formulated as a sequential decision-making problem under uncertainty. This problem involves two major sub-problems: (a) the state estimation and system identification problem, and (b) the stochastic control problem. The first is exploitation of the information acquired from the system output to identify its behavior, that is, how a state representation can be built by observing the system's state transitions. The second is assessment of the system output with respect to alternative control policies, and selecting those that optimize specified performance criteria.

This chapter reports research on implementing a computational model suitable to solve the state estimation and system identification sub-problem. The evolution of the system is modeled as a Markov Decision Process (MDP). A state-space representation is constructed through a learning mechanism which can then be used in solving the stochastic control problem. The model allows decision making based on gradually

enhanced knowledge of system response as it transitions from one state to another, in conjunction with actions taken at each state.

3.1 Reinforcement Learning Algorithms

Deriving a control policy for dynamic systems is an off-line process in which various methods from control theory are utilized iteratively. These methods aim to determine the policy that satisfies the system's physical constraints while optimizing specific performance criteria. A challenging task in this process is to derive a mathematical model of the system's dynamics that can adequately predict the response of the physical system to all anticipated inputs. Exact modeling of complex engineering systems, however, may be infeasible or expensive. Alternative methods have been developed enabling the real-time implementation of control policies for systems when an accurate model is not available. In this framework, the system interacts with its environment, and obtains information enabling it to improve its future performance by means of costs (or rewards) associated with control actions taken. This interaction portrays the learning process conveyed by the progressive enhancement of the system's "knowledge" regarding the course of action (control policy) that maximizes the accumulated rewards with respect to the system's operating point (state) transitions. The environment is assumed to be non-deterministic; namely, taking the same action in the same state on two different decision time steps (decision epochs or stages), the system may transit to a different state and incur a dissimilar cost in the subsequent step. Consequently, the problem of developing a policy for controlling the system's behavior is formulated as a sequential decision-making problem under uncertainty.

Dynamic programming (DP) has been widely employed as the principal method for analysis of sequential decision-making problems [1]. Algorithms, such as value iteration and policy iteration, have been extensively utilized in solving deterministic and

stochastic optimal control problems, Markov and semi-Markov decision problems, minimax control problems, and sequential games. However, the computational complexity of these algorithms in some occasions may be prohibitive and can grow intractably with the size of the problem and its related data, referred to as the DP "curse of dimensionality" [2]. In addition, DP algorithms require the realization of the conditional probabilities of state transitions and the associated costs, implying *a priori* knowledge of the system dynamics. However, even if the transition probabilities are known, the problem of analytic computation might be too hard, and one might seek an approximation method that exploits the possibilities of simulation.

Simulation-based methods for solving sequential decision-making problems under uncertainty have been primarily developed in the field of Reinforcement Learning (RL) [2-4]. RL has aimed to provide algorithms, founded on DP, for learning control policies when analytical methods cannot be used effectively, or the system's state transition probabilities are not known [5]. A major influence on research leading to current RL algorithms has been Samuel's method [6, 7], used to modify a heuristic evaluation function for deriving optimal board positions in the game of checkers. In this algorithm, Samuel represented the evaluation function as a weighted sum of numerical features and adjusted the weights based on an error derived from comparing evaluations of current and predicted board positions. This approach was refined and extended by Sutton [8, 9] to introduce a class of incremental learning algorithms, Temporal Difference (TD). TD algorithms are specialized for deriving optimal policies for incompletely known systems, using past experience to predict their future behavior. Watkins [10] extended Sutton's TD algorithms and developed an algorithm for systems to learn how to act optimally in controlled Markov domains by explicitly utilizing the theory of DP. A strong condition implicit in the convergence of Q-learning to an optimal control policy is that the sequence of decision epochs that forms the basis of learning must include an infinite number of decision epochs for each initial state and action. Q-learning is considered the most

popular and efficient model-free learning algorithm in deriving optimal control policies in Markov domains [11]. Schwartz [12] explored the potential of adapting Q-learning to an average-cost framework with his R-learning algorithm; Bertsekas and Tsitsiklis [3] presented a similar to Q-learning average-cost algorithm. Mahadevan [13] surveyed reinforcement-learning average-cost algorithms and showed that these algorithms do not always produce bias-optimal control policies.

The aforementioned algorithms consist of evaluation functions attempting to successively approximate the Bellman equation. These evaluation functions assign to each state the total cost expected to accumulate over time starting from a given state when a policy π is employed. Although many of these algorithms are eventually guaranteed to find suboptimal policies in sequential decision-making problems under uncertainty, their use of the accumulated data acquired over the learning process is inefficient, and they require a significant amount of experience to achieve acceptable performance [11]. This requirement arises due to the formation of these algorithms in deriving optimal policies without learning the system models *en route*; that is they do not solve the state estimation and system identification problem simultaneously.

Algorithms for computing optimal policies by learning the models are especially important in applications in which real-world experience is considered expensive. Sutton's Dyna architecture [14, 15] exploits strategies which simultaneously utilize experience in building the model and adjust the derived policy. Prioritized sweeping [11] and Queue-Dyna [16] are similar methods concentrating on the interesting subspaces of the state-action space. Barto *et al.* [4] developed another method, called Real-Time Dynamic Programming (RTDP), referring to the cases in which concurrently executing DP and control processes influence one another. RTDP focuses the computational effort on the state-subspace that the system is most likely to occupy. However, these methods are specific to problems in which the system needs to achieve particular goal states and the initial cost of every goal state is zero.

3.2 Identification and Stochastic Adaptive Control

Adaptive control algorithms provide a systematic treatment in deriving optimal control policies in stochastic systems where exact modeling is not available *a priori*. In this context, the evolution of the system is modeled as a countable state controlled Markov chain whose transition probability is specified up to an unknown parameter taking values in a compact metric space.

In general, the analysis of optimal control in dynamic systems starts with a given state space model

$$s_{k+1} = f_k(s_k, \alpha_k, w_k), \tag{3.1}$$

$$y_k = h_k(s_k, v_k). \tag{3.2}$$

Stating that the model is given implies that the functions f_k, h_k, and the probability distribution of the basic random variables $s_0, w_0, w_1, ..., v_0, v_1, ...,$ are all known. Moreover, the conditional distributions of (s_{k+1}, y_k) given (s_k, a_k) are also known. This is the off-line information in contrast to the on-line one which at time k consists of the observations $z^k = (y^k, a^{k-1})$. Consequently, the off-line information specifies a unique system. However, the off-line information is often insufficient to characterize a model completely; that is, the system model is not known initially. In this context, we are faced with the problem of how to make a rational choice of the control values $a_0, a_1, ...,$ when the model is unspecified. This problem can be reformulated as a problem with partial observation. Nevertheless, the resulting computational burden increases making this formulation impractical except in the case when the unspecified system model is known to belong to a finite set.

Suppose the system model is unknown. As we make more observations z^k, i.e., as k increases, we ought to be able to characterize better the system model. The abstract

framework that describes and analyzes this learning process is quite simple. A particular model is specified by a triple [17]

$$M := (f, h, \mathbb{P}), \tag{3.3}$$

where f is the function in the state equation, h is the function in the observation equation, and $\mathbb{P}$ is the probability distribution of the basic random variables. It is assumed that the off-line information is such as to guarantee that the true system model belongs to the family of models $M^\theta := (f^\theta, h^\theta, \mathbb{P}^\theta)$ parameterized by the finite-dimensional vector θ which is known to belong to the set Θ. Consequently, it is known *a priori* that the true system model corresponds to some true parameter $\theta^\circ \in \Theta$, which is unidentified initially. At time *k*, an estimation $\hat{\theta}_k$ of the true parameter is made based on the on-line observation z^k. If the estimation or identification procedure is sufficient then $\hat{\theta}_k$ should approach θ° as *k* increases.

The initial uncertainty about the system is reflected in the parameterization, i.e., the function $\theta \to (f^\theta, h^\theta, \mathbb{P}^\theta)$, and the size of the parameter set Θ. This size may be large or small, and the parameterization may be more or less complex, i.e., linear vs. nonlinear, one-to-one vs. many-to-one. In practice, however, the set of models $\{M^\theta\}$ can only approximately represent the true system. Better approximations will lead to more complex parameterizations and a larger model set Θ making the identification problem more complicated. Consequently, the choice of parameterization must keep a balance between the demand for accuracy and the need to limit the computational burden.

The stochastic adaptive control problem has been extensively reported in the literature. Mandl [18] considered an adaptive control scheme providing a minimum contrast estimate of the unknown model of the systems at each decision epoch (stage), and then applying the optimal feedback control corresponding to this estimate. If the system satisfies a certain "identifiability condition", the sequence of parameter estimates

converges almost surely to the true parameter. Borkar and Variaya [19] removed this identifiability condition and showed that when Θ is finite, the maximum likelihood estimate $\hat{\theta}_k$ converges almost surely to a random variable. Borkar and Variaya [20], and Kumar [21] examined the performance of the adaptive control scheme of Mandl without the "identifiability condition," but under varying degrees of generality of the state, control, and model spaces with the attention restricted to the maximum likelihood estimate. Doshi and Shreve [22] proved that if the set of allowed control laws is generalized to include the set of randomized controls, then the cost of using this scheme will almost surely equal to the optimal cost achievable if the true parameter were known. Kumar and Becker [23] implemented a novel approach to the adaptive control problem when a set of possible models is given including a new criterion for selecting a parameter estimate. This criterion is obtained by a deliberate biasing of the maximum likelihood criterion in favor of parameters with lower optimal costs. These results were extended by assuming that a finite set of possible models is not available [24]. Sato, Abe, and Takeda [25-27] proposed a learning controller for Markovian decision problems with unknown probabilities. The controller was designed to be asymptotically optimal considering a conflict between estimation and control for determination of a control policy. Kumar [28], and Varaiya [29] have provided comprehensive surveys of the aforementioned research efforts.

Certainty Equivalence Control (CEC) [17, 28] is a common approach in addressing stochastic adaptive control problems. The unknown system parameter is estimated at each decision epoch while assuming that the decision maker selects a control action as if the estimated parameter is the true one. The major drawback of this approach is that the decision maker may get locked in a false parameter when there is a conflict between learning and control. Forcing controls, different actions from those imposed by the certainty equivalence control, at some random decision epochs are often utilized to address this issue. The certainty equivalence control employing a forcing strategy is

optimal in stochastic adaptive optimization problems with the average-cost-per-unit-time criterion.

Various stochastic adaptive control schemes have considered the classical example of the multi-armed bandit problem. Lai and Robbins [30] developed a solution methodology for bandits with independent identically distributed arms by introducing the average-cost-per-unit-time criterion. Ananthanam, Varaiya, and Walrand [31], and Agrawal, Hedge, and Teneketzis [32] generalized this result by developing various extensions of the Lai-Robbins formulation in the multi-armed bandit problem. Agrawal, Teneketzis, and Ananthanam [33] developed a "translation scheme" which along with the construction of an "extended probability space" solved the controlled Markov chain problem by converting it to a form similar to that for the controlled Markov independent sequence problem [34]. These results were utilized by Graves and Lai [35] to develop adaptive control rules considering compact parameter set and general state-space while assuming finite set of admissible policies. In these adaptive control schemes, the best possible performance depends on the on-line forcing strategy. Agrawal and Teneketzis [36] studied the rate of forcing to asses the performance of a certainty equivalence control with forcing for the multi-armed bandit problem and the adaptive control of Markov chains.

Although the aforementioned research work has successfully led to asymptotically optimal adaptive control schemes, their underlying framework imposes limitations in implementing these schemes on the engine calibration problem defined as a sequential decision-making problem under uncertainty in Chapter 2. In particular, in the engine calibration problem, the engine model is assumed to be completely unknown, and thus, parameterization cannot be developed. Moreover, the requirement of real-time derivation of the values of the engine controllable variables over an unknown horizon imposes an additional computational burden in implementing such control schemes.

In this chapter, a computational model suited for real-time sequential decision-making under uncertainty modeled as controlled Markov chain is proposed. The model consists of a new state-space representation that addresses the state estimation and system identification sub-problem for the entire system (engine) operating domain. Furthermore, it utilizes an evaluation function suitable for lookahead control algorithms, and thus, for real-time implementation.

3.3 Finite State Controlled Markov Chains

The evolution of the system (engine) is modeled as a Markov Decision Process with a finite state space (engine operating domain) $\mathcal{S}$ and action space (values of the engine controllable variables) $\mathcal{A}$. So, a discrete-time Markov process $\{s_k, k \geq 0\}$ is considered that takes values in some countable set $\mathcal{S}$, that is, the state belongs to a finite state space $\mathcal{S} = \{1, 2, ..., N\}, N \in \mathbb{N}$. The control action a_k takes values in a pre-specified space $\mathcal{A}$. Consequently, the process is a controlled Markov chain. The transition probabilities are specified by the $N \times N$ matrix valued function on $\mathcal{A}$,

$$a \to \mathbf{P}(a) := \{\mathbf{P}_{ij}(a), \forall i, j \in \mathcal{S}\}, \forall a \in \mathcal{A} \tag{3.4}$$

with the interpretation

$$\text{Prob}\{s_{k+1} = j \mid s_k = i, s_{k-1}, ..., s_0, a_k, ..., a_0\} = \mathbb{P}_{ij}(a_k) = p_{ij},\ k \geq 0, \forall i, j \in \mathcal{S}. \tag{3.5}$$

Definition 3. 1 [37]. The chain $\{s_k, k \geq 0\}$ is called homogeneous if

$$\mathbb{P}_{ij}(s_{k+1} = j \mid s_k = i) = \mathbb{P}_{ij}(s_1 = j \mid s_0 = i),\ \forall k \geq 0, \forall i, j \in \mathcal{S}. \tag{3.6}$$

Theorem 3. 1 [38]. The transition probability matrix $\mathbf{P}(a)$ is a stochastic matrix, that is

(a) $\mathbf{P}(a)$ has non-negative entries, or $p_{ij} \geq 0, \forall i, j \in \mathcal{S}$,

(b) the sum of its rows is equal to one, or $\sum_{j \in \mathcal{S}} p_{ij} = 1, \forall i \in \mathcal{S}$.

Proof. The proof is provided by Grimmett and Stirzaker [38].

□

Definition 3. 2 [39]. The *n*-step transition probability matrix $\mathbb{P}_{ij}^{(m,m+n)}(a_k)$ is the matrix of *n*-step transition probabilities $\mathbb{P}_{ij}^{(m,m+n)}(a_k) = \mathbb{P}_{ij}^{(m,m+n)}(s_{m+n} = j \mid s_m = i)$.

The *n*-step transition probability matrix satisfies the Chapman-Kolmogorov equation,

$$p_{ij}^{(m,m+n)} = \sum_{l \in \mathcal{S}} p_{il}^{(m)} \cdot p_{lj}^{(n)}, \tag{3.7}$$

or $\mathbf{P}_a^{(m,m+n)}(a_k) = \mathbf{P}_a^n$.

3.3.1 Classification of States in a Markov Chain

The evolution of a Markov chain can be seen as the motion of a notional particle which jumps between the states of the state space $i \in \mathcal{S}$ at each decision epoch. The classification of the states in a Markov chain aims to provide insight towards modeling appropriately the evolution of a controlled dynamic system.

Definition 3. 3 [38]. A Markov state $i \in \mathcal{S}$ is called recurrent (or persistent), if

$$\mathbb{P}_{ij}(s_k = i \text{ for some } k \geq 0 \mid s_0 = i) = 1, \tag{3.8}$$

that is, the probability of eventual return to state i, having started from i, is one.

The first time the chain $\{s_k, k \geq 0\}$ visits a state $i \in \mathcal{S}$ is given by

$$T_1(i) := \min\{k \geq 1 : s_k = i\}. \tag{3.9}$$

$T_1(i)$ is called the *first entrance time* or *first passage time* of state i. It may happen that $s_k \neq i$ for any $k \geq 1$. In this case, $T_1(i) = \min \varnothing$, which is taken to be ∞. Consequently, if the chain never visits state i for any time $k \geq 1$, $T_1(i) = \infty$. Given that the chain starts in state i, the conditional probability that the chain returns to state i in finite time is

$$f_{ii} := \mathbb{P}(T_1(i) < \infty \mid s_0 = i). \tag{3.10}$$

Consequently, for a recurrent state i $f_{ii} = 1$. Furthermore, if the expected time for the chain to return to a recurrent state i is finite, the state is said to be positive recurrent; otherwise, the state is said to be null recurrent. The *n*th entrance time of state i is given by

$$T_n(i) := \min\{k \geq T_{n-1}(i) : s_k = i\}. \tag{3.11}$$

Definition 3. 4 [38]. The mean recurrence time μ_i of a state i is defined as

$$\mu_i := E\{T_1(i) \mid s_0 = i\}. \tag{3.12}$$

Definition 3. 5 [38]. The period $d(i)$ of a state i is defined by

$$d(i) := gcd\{n : T_n(i) > 0\}, \tag{3.13}$$

that is, the greatest common divisor of the decision epochs at which return is possible. The state i is periodic if $d(i) > 1$ and aperiodic if $d(i) = 1$.

Definition 3. 6 [38]. A Markov state is called ergodic, if it is positive recurrent, and aperiodic.

Definition 3. 7 [38]. If the chain started from state i and visits state j, that is $\mathbb{P}_{ij}^{(n)}(s_n = j \mid s_0 = i) > 0$ for some $n > 0$, it is said that i communicates with j, and it is denoted $i \to j$. It is said that i and j intercommunicate if $i \to j$ and $j \to i$, in which case it is denoted $i \leftrightarrow j$.

Definition 3. 8 [39]. A Markov chain is called *irreducible* if all states intercommunicate in a finite number of decision epochs, that is, $\mathbb{P}_{ij}^{(n)}(s_n = j \mid s_0 = i) > 0, \ \forall i, j \in \mathcal{S}$.

3.4 The Predictive Optimal Decision-Making Computational Model

The Predictive Optimal Decision-making (POD) [40] learning model implemented in this dissertation consists of a new state-space system representation. The state-space representation accumulates gradually enhanced knowledge of the system's transition from each state to another in conjunction with actions taken for each state. This knowledge is expressed in terms of transition probabilities and an expected evaluation function associated with each Markov state. The major differences between the proposed computational model and the existing RL methods are: (a) the model solves the state estimation and system identification sub-problem for the entire system's operating domain by learning the transition probability and cost matrices, and (b) the model utilizes an evaluation function suitable for lookahead control algorithms, and thus, for real-time implementation. While the model's knowledge regarding the transition probabilities is advanced, a real-time lookahead control algorithm, which is developed in Chapter 4, can realize the control actions in the stochastic control sub-problem. This approach is especially appealing to learning engineering systems in which the initial state is not fixed

[41, 42], and recursive updates of the evaluation functions to approximate the Bellman equation would demand a huge number of iterations to achieve the desired system performance.

The model considers controlled systems that their evolution is modeled as a Markov chain, with the following assumptions.

Assumption 3. 1. The Markov chain is homogeneous.

Assumption 3. 2. The Markov chain is ergodic, that is, the states are positive recurrent and aperiodic.

Assumption 3. 3. The Markov chain is irreducible. Consequently, each state i of the Markov chain intercommunicates with each other $i \leftrightarrow j, \forall i, j \in \mathcal{S}$, that is, each system's state can be reached with a positive probability from any other state in finite decision epochs.

3.4.1 Construction of the POD State Space Representation

The new state-space representation defines the POD domain $\tilde{\mathcal{S}}$, which is implemented by a mapping $\mathcal{H}$ from the Cartesian product of the finite state space and action space of the Markov chain $\{s_k, k \geq 0\}$

$$\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \tilde{\mathcal{S}}, \tag{3.14}$$

where $\mathcal{S} = \{1, 2, ..., N\}$, $N \in \mathbb{N}$ denotes the Markov state space, and $\mathcal{A} = \bigcup_{s_k \in \mathcal{S}} A(s_k), \forall s_k = i \in \mathcal{S}$ stands for the finite action space. Each state of the POD domain represents a Markov state transition from $s_k = i \in \mathcal{S}$ to $s_{k+1} = j \in \mathcal{S}$ for all $k \geq 0$, as illustrated in Figure 3.1, that is

$$\tilde{\mathcal{S}} := \left\{ \tilde{s}_{k+1}^{ij} \mid \tilde{s}_{k+1}^{ij} \equiv s_k = i \underset{\mu(s_k) \in A(s_k)}{\to} s_{k+1} = j, \sum_{j=1}^{N} p(s_{k+1} = j \mid s_k = i, \alpha_k) = 1, N = |\mathcal{S}| \right\}, \tag{3.15}$$

$\forall i, j \in \mathcal{S},\ \forall \mu(s_k) \in A(s_k).$

Definition 3. 9. The mapping H generates an indexed family of subsets, $\tilde{\mathcal{S}}_i$, for each Markov state $s_k = i \in \mathcal{S}$, defined as Predictive Representation Nodes (PRNs). Each PRN is constituted by the set of POD states $\tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}_i$ representing the state transitions from the state $s_k = i \in \mathcal{S}$ to all other Markov states

$$\tilde{\mathcal{S}}_i = \left\{ \tilde{s}_{k+1}^{ij} \mid s_k = i \underset{\mu(s_k) \in A(s_k)}{\rightarrow} s_{k+1} = j, \forall j \in \mathcal{S} \right\}. \tag{3.16}$$

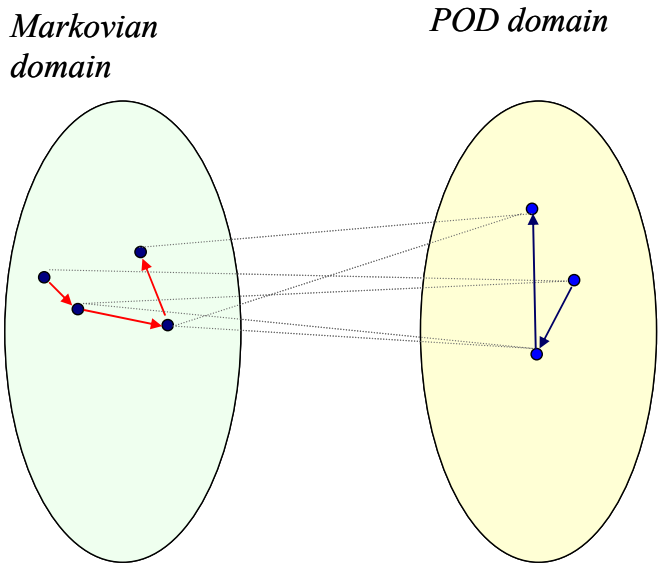

Figure 3.1 − Construction of the POD domain.

PRNs partition the POD domain insofar as the POD underlying structure captures the state transitions in the Markov domain as depicted in Figure 3.2, namely

$$\tilde{\mathcal{S}} = \bigcup_{\tilde{s}_k^{ij} \in \tilde{\mathcal{S}}_i} \tilde{\mathcal{S}}_i, \text{ with} \tag{3.17}$$

$\bigcap_{\tilde{s}_k^{ij} \in \tilde{\mathcal{S}}_i} \tilde{\mathcal{S}}_i = \varnothing.$

PRNs, constituting the fundamental aspect of the POD state representation, provide an assessment of the Markov state transitions along with the actions executed at each state. This assessment aims to establish a necessary embedded property of the new state representation so as to consider the potential transitions that can occur in subsequent decision epochs. The assessment is expressed by means of the PRN value, $\overline{R}_i(\tilde{s}_{k+1}^{ij} \mid \mu(s_i))$, which accounts for the minimum expected cost that can be achieved by transitions occurring inside a PRN.

Definition 3. 10. The PRN value $\overline{R}_i(\tilde{s}_{k+1}^{ij} \mid \mu(s_i))$ is defined as

$$\overline{R}_i(\tilde{s}_{k+1}^{ij} \mid \mu(s_k = i)) := \min_{\mu(s_k) \in \mathcal{A}} \sum_{j=1}^{N} p(s_{k+1} = j \mid s_k = i, \mu(s_k)) \cdot R(s_{k+1} = j \mid s_k = i, \mu(s_k)), \quad (3.18)$$

$\forall \tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}, \forall i, j \in \mathcal{S}, \forall \mu(s_k) \in A(s_k), \text{and } N = \mid \mathcal{S} \mid .$

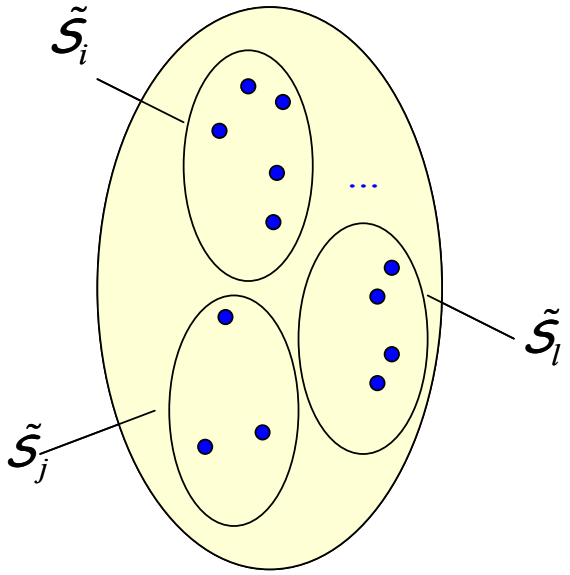

Figure 3.2 − Partition of POD through the PRNs.

The PRN value is exploited by POD state representation as an evaluation metric to estimate the subsequent Markov state transitions. The estimation property is founded on the assessment of POD states by means of an expected evaluation function, $R^{i}_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k))$, defined as

$$R^{i}_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) = \{ p(s_{k+1} = j \mid s_k = i, \mu(s_k)) \cdot R(s_{k+1} = j \mid s_k = i, \mu(s_k)) + \\ + \overline{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1}))\}, \tag{3.19}$$

$$\forall \tilde{s}^{jm}_{k+2} \in \tilde{\mathcal{S}}, \forall i, j, m \in \mathcal{S}, \ \forall \mu(s_k), \in A(s_k), \forall \mu(s_{k+1}) \in A(s_{k+1}).$$

Consequently, employing the POD evaluation function through Eq. (3.19), each POD state, $\tilde{s}^{ij}_{k+1} \in \tilde{\mathcal{S}}_i$, is comprised of an overall cost corresponding to: (a) the expected cost of transiting from state $s_k = i$ to $s_{k+1} = j$ (implying also the transition from the PRN $\tilde{\mathcal{S}}_i$ to $\tilde{\mathcal{S}}_j$); and (b) the minimum expected cost when transiting from $s_{k+1} = j$ to any other Markov state at *k*+2 (transition occurring into $\tilde{\mathcal{S}}_j$).

3.4.2 Self-Learning System Identification

While the system interacts with its environment, the POD model learns the system dynamics in terms of the Markov state transitions. The POD state representation attempts to provide a process in realizing the sequences of state transitions that occurred in the Markov domain, as infused in PRNs. The different sequences of the Markov state transitions are captured by the POD states and evaluated through the expected evaluation functions given in Eq. (3.19). Consequently, the lowest value of the expected evaluation function at each POD state essentially estimates the subsequent Markov state transitions with respect to the actions taken. As the process is stochastic, however, the real-time learning method still has to build a decision-making mechanism of how to select actions, namely, how to solve the stochastic control problem. This problem is addressed in Chapter 4.

The learning performance is closely related to the exploration-exploitation strategy of the action space. More precisely, the decision maker has to exploit what is already known regarding the correlation involving the admissible state-action pairs that minimize the costs, and also to explore those actions that have not yet been tried for these pairs to assess whether these actions may result in lower costs. A balance between an exhaustive exploration of the environment and the exploitation of the learned policy is fundamental to reach nearly optimal solutions in a few decision epochs and, thus, to enhance the learning performance. This exploration-exploitation dilemma has been extensively reported in the literature. Iwata *et al.* [43] proposed a model-based learning method extending Q-learning and introducing two separated functions based on statistics and on information by applying exploration and exploitation strategies. Ishii *et al.* [44] developed a model-based reinforcement learning method utilizing a balance parameter, controlled through variation of action rewards and perception of environmental change. Chan-Geon *et al.* [45] proposed an exploration-exploitation policy in Q-learning consisting of an auxiliary Markov process and the original Markov process. Miyazaki *et al.* [46] developed a unified learning system realizing the tradeoff between exploration and exploitation. Hernandez-Aguirre *et al.* [47] analyzed the problem of exploration-exploitation in the context of the approximately correct framework and studied whether it is possible to set bounds on the complexity of the exploration needed to achieve a fixed approximation error over the action value function with a given probability.

An exhaustive exploration of the environment is necessary to evade premature convergence on a sub-optimal solution even if this may result in both sacrificing the system's performance in the short run and increasing the learning time. In our case it is assumed that, for any state $s_k = i \in \mathcal{S}$, all actions of the feasible action set $\mu(s_k = i) \in A(s_k = i)$ are selected by the decision maker at least once. At the early decision epochs and until full exploration of the action set $A(s_k = i), \forall i \in \mathcal{S}$ occurs, the

mapping from the states to probabilities of selecting the actions is constant; namely, the actions for each state are selected randomly with the same probability

$$p(\mu(i)\,|\,i) = \frac{1}{|A(i)|}, \forall \mu(i) \in A(i), \forall i \in \mathcal{S}. \tag{3.20}$$

When the exploration phase is complete, a lookahead control algorithm can be utilized to build up the decision-making mechanism.

3.4.3 Stationary Distributions and the Limit Theorem

The behavior of a Markov chain after a long time k has elapsed is described by the stationary distributions and the limit theorem. The sequence $\{s_k, k \geq 0\}$ does not converge to some particular state $i \in \mathcal{S}$ since it enjoys the inherent random fluctuation which is specified by the transition probability matrix. Subject to certain conditions, the distribution of $\{s_k, k \geq 0\}$ settles down.

Definition 3. 11 [38]. The vector $\boldsymbol{\rho}$ is called a stationary distribution of the chain if $\boldsymbol{\rho}$ has entries $(\rho_i, i \in \mathcal{S})$ such that:

(a) $\rho_i \geq 0$ for all i, and $\sum_{i \in \mathcal{S}} \rho_i = 1$,

(b) $\boldsymbol{\rho} = \boldsymbol{\rho} \cdot \mathbf{P}$, that is $\rho_i = \sum_{j \in \mathcal{S}} \rho_j \cdot \mathbb{P}_{ji}$, where $\mathbb{P}_{ji}$ is the transition probability $\mathbb{P}_{ji}(s_{k+1} = i \,|\, s_k = j)$, for all i.

If the transition probability matrix of a Markov chain $\mathbf{P}_{ij}$ is raised to a higher power, the resulting matrix is also a transition probability matrix. If the matrix is kept raising to higher powers, then the elements in any given column start converging to the same number. This property can be illustrated further in the following simple example. Let us consider a Markov chain with two states $\mathcal{S} = \{1,2\}$ and a transition probability matrix

$$\mathbf{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}.$$

This matrix represents the one-step transition probabilities of the states. Consequently, if the chain is at state 1 there is a probability of 0.7 that it will remain there and of 0.3 that it will transit to state 2. Similarly, if the chain is at state 2, there is a probability of 0.4 that it will transit to state 1 and of 0.6 that it will remain at state 2. If this matrix is raised to the second order, the resulting matrix yields the two-step transition probabilities

$$\mathbf{P}^2 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} . \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{bmatrix}.$$

The elements of the two-step transition probability matrix essentially return the conditional probability that the chain will transit to a particular state within two decision epochs. Consequently, the value $\mathbb{P}_{12}^2(s_{k+1} = 2 \mid s_k = 1) = 0.39$ in the above matrix is the conditional probability that the chain will go from state 1 to state 2 in two decision epochs. If the one-step transition probability matrix is raised to the 8th power, it is noticed that the elements in any given column start converging to 0.57 and 0.43, respectively, namely,

$$\mathbf{P}^8 = \begin{bmatrix} 0.5715 & 0.4285 \\ 0.5714 & 0.4286 \end{bmatrix}.$$

These numbers constitute the stationary distribution of the chain, vector $\boldsymbol{\rho}$, that is,

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \begin{bmatrix} 0.57 \\ 0.43 \end{bmatrix}.$$

The limit theorem states that if a chain is irreducible with positive recurrent states, the following limit exists

$$\rho_j = \lim_{n\to\infty} \mathbb{P}_{ij}^n(s_{k+1} = j \mid s_k = i) = \mathbb{P}(s_n = j). \tag{3.21}$$

Theorem 3. 2 ("Limit Theorem") [38]. An irreducible Markov chain has a stationary distribution $\boldsymbol{\rho}$ if and only if all the states are positive recurrent. Furthermore, $\boldsymbol{\rho}$ is the unique stationary distribution and is given by $\rho_i = \mu_i^{-1}$ for each $i \in \mathcal{S}$, where μ_i is the mean recurrence time of i.

Proof. The proof is provided by Grimmett and Stirzaker [38].

□

Stationary distributions have the following property

$$\boldsymbol{\rho} = \boldsymbol{\rho} \cdot \mathbf{P}^n, \forall n \geq 0 \tag{3.22}$$

3.4.4 Convergence of POD Model

As the system interacts with its environment, the POD state representation realizes the sequences of state transitions that occurred in the Markov domain, as infused in PRNs. In this section, it is shown that this realization determines the stationary distribution of the Markov chain.

Definition 3. 12. Given a set $C \subset \mathbb{R}$ and a variable x, the indicator function, denoted by $I_C(x)$, is defined by

$$I_C(x) := \begin{cases} 1, x \in C \\ 0, x \notin C \end{cases} \tag{3.23}$$

Lemma 3. 1. Each PRN is irreducible, that is $\tilde{\mathcal{S}}_i \leftrightarrow \tilde{\mathcal{S}}_j, \forall i, j \in \mathcal{S}$.

Proof. At the decision epoch k, the state transition from i to j corresponds to the $\tilde{s}_k^{ij}$ inside the PRN $\tilde{\mathcal{S}}_i$. The next state transition will occur from the state j to any other Markov state. Consequently, by Definition 3.9, the next state transition will occur in $\tilde{\mathcal{S}}_j$. By Assumption 3.3, all states intercommunicate with each other, that is, $i \leftrightarrow j, \forall i, j \in \mathcal{S}$. So PRNs intercommunicate and thus they are irreducible. The lemma is proved.

□

The number of visits of the chain to the state $j \in \mathcal{S}$ between two successive visits to state $i \in \mathcal{S}$ at the decision epoch $k = M$, that is, the number of visits of the POD state $\tilde{s}_M^{ij} \in \tilde{\mathcal{S}}$, is given by

$$V(\tilde{s}_M^{ij}) := \sum_{k=1}^{M} I_{\{s_k = j\} \cap \{T_1(i) \geq k\}}(s_k) \tag{3.24}$$

where $T_1(i)$ is the time of the first return to state $i \in \mathcal{S}$.

Definition 3. 13. The mean number of visits of the chain to the state $j \in \mathcal{S}$ between two successive visits to state $i \in \mathcal{S}$ is

$$\bar{V}(\tilde{s}_M^{ij}) := E\{V(\tilde{s}_M^{ij}) \mid s_k = i\}, \tag{3.25}$$

or $\bar{V}(\tilde{s}_M^{ij}) := \sum_{k=1}^{M} \mathbb{P}(s_k = j, T_1(i) \geq k \mid s_0 = i).$

Definition 3. 14. The mean recurrence time time $\mu_{\tilde{\mathcal{S}}_i}$ that the chain spends at the PRN $\tilde{\mathcal{S}}_i$ is

$$\mu_{\tilde{\mathcal{S}}_i} := \sum_{j \in \mathcal{S}} \bar{V}(\tilde{s}_M^{ij}) = \sum_{j \in \mathcal{S}} \sum_{k=1}^{M} \mathbb{P}(s_k = j, T_1(i) \geq k \mid s_0 = i). \tag{3.26}$$

Lemma 3. 2. The mean recurrence time of each PRN $\tilde{\mathcal{S}}_i, \mu_{\tilde{\mathcal{S}}_i}$, is equal to the mean recurrence time of state $i \in \mathcal{S}$, μ_i.

Proof. It was shown (Lemma 3.1) that each time the Markov chain transits from one state $i \in \mathcal{S}$ to a state $j \in \mathcal{S}$ there is a corresponding transition from the PRN $\tilde{\mathcal{S}}_i$ to $\tilde{\mathcal{S}}_j$. Consequently, the number of visits of the chain to the state $i \in \mathcal{S}$ is equal to the number of visits to the PRN $\tilde{\mathcal{S}}_i$. Taken the expectation of this number yields the mean recurrence time, by Definition 3. 13. The lemma is proved.

□

Proposition 3. 1. If A, B, and C are some events and

$$\mathbb{P}(A \mid B \cap C) = \mathbb{P}(A \mid B), \tag{3.27}$$

then

$$\mathbb{P}(A \cap C \mid B) = \mathbb{P}(A \mid B) \cdot \mathbb{P}(C \mid B) \tag{3.28}$$

Proof.

$$\mathbb{P}(A \cap C \mid B) = \frac{\mathbb{P}(A \cap B \cap C)}{\mathbb{P}(B)} \tag{3.29}$$

using the identity $\mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(A \cap B)$, Eq. (3.29) yields

$$\frac{\mathbb{P}(A \mid C \cap B) \cdot \mathbb{P}(C \cap B)}{\mathbb{P}(B)} = \frac{\mathbb{P}(A \mid B) \cdot \mathbb{P}(C \cap B)}{\mathbb{P}(B)} \text{ by using Eq. (3.27)}$$

$$= \frac{\mathbb{P}(A \mid B) \cdot \mathbb{P}(C \cap B)}{\mathbb{P}(B)} = \frac{\mathbb{P}(A \mid B) \cdot \mathbb{P}(C \mid B) \cdot \mathbb{P}(B)}{\mathbb{P}(B)}$$

$$= \mathbb{P}(A \mid B) \cdot \mathbb{P}(C \mid B).$$

□

It remains to present the main result of the POD computational model, namely, that the realization of the sequences of state transitions that occurred in the Markov domain as infused by the PRNs determines the stationary distribution of the Markov chain.

Theorem 3. 3. The POD state representation generates the stationary distribution $\boldsymbol{\rho}$ of the Markov chain. Moreover, the stationary probability is given by the mean recurrence time of each PRN $\tilde{\mathcal{S}}_i, \rho_i = \mu_{\tilde{S}_i}^{-1}$.

Proof. Since the chain is ergodic with irreducible states, it is guaranteed that the chain has a unique stationary distribution, and for each state $i \in \mathcal{S}$ the stationary probability is equal to $\rho_i = \mu_i^{-1}$.

$\rho_i \cdot \mu_i =$

$= \rho_i \cdot \mu_{\tilde{S}_i}$ by Lemma 3. 2

$$= \sum_{j \in S} \sum_{k=1}^{M} \mathbb{P}(s_k = j, T_1(i) \geq k \mid s_0 = i) \cdot \mathbb{P}(s_0 = i) \tag{3.30}$$

$$= \sum_{j \in S} \sum_{k=1}^{M} \mathbb{P}(s_k = j, T_1(i) \geq k, s_0 = i) \tag{3.31}$$

by using the identity $\mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(A \cap B)$.

For $k = 1$, Eq. (3.31) yields

$$\sum_{j \in S} \mathbb{P}(s_k = j, T_1(i) \geq 1, s_0 = i) = 1. \tag{3.32}$$

For $k \geq 2$, Eq. (3.30) yields

$$\sum_{j \in S} \sum_{k=1}^{M} \mathbb{P}(s_k = j, T_1(i) \geq k \mid s_0 = i) \cdot \mathbb{P}(s_0 = i)$$

$$= \sum_{j\in \mathrm{S}} \sum_{k=1}^{M} \mathbb{P}(s_k = j, s_m \neq i \text{ for } 1 \le m \le k-1, s_0 = i) \tag{3.33}$$

$$= \sum_{j\in \mathrm{S}} \sum_{k=1}^{M} \mathbb{P}(s_k = j \mid s_0 = i) \cdot \mathbb{P}(s_m \neq i \text{ for } 1 \le m \le k-1 \mid s_0 = i) \cdot \mathbb{P}(s_0 = i)$$

$$= \sum_{j\in \mathrm{S}} \sum_{k=1}^{M} \mathbb{P}(s_k = j \mid s_0 = i) \cdot \mathbb{P}(s_m \neq i \text{ for } 1 \le m \le k-1, s_0 = i)$$

$$= \sum_{k=1}^{M} \left(\sum_{j\in \mathrm{S}} \mathbb{P}(s_k = j \mid s_0 = i) \right) \cdot \mathbb{P}(s_m \neq i \text{ for } 1 \le m \le k-1, s_0 = i)$$

$$= \sum_{k=1}^{M} \mathbb{P}(s_m \neq i \text{ for } 1 \le m \le k-1, s_0 = i) \tag{3.34}$$

by using the identity $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$, Eq. (3.34) becomes

$$\sum_{k=1}^{M} \mathbb{P}(s_0 = i) + \mathbb{P}(s_m \neq i \text{ for } 1 \le m \le k-1) - \mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-1)$$

Since the Markov chain is homogeneous (Assumption 3. 1)

$$= \sum_{k=1}^{M} \{\mathbb{P}(s_0 = i) + \mathbb{P}(s_0 \neq i) + \mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-3) -$$
$$\mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-1)\}$$

$$= \sum_{k=1}^{M} \{\mathbb{P}(s_0 = i) + \mathbb{P}(s_0 \neq i)\} + \lim_{k\to\infty} \left(\mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-3)\right) -$$
$$- \lim_{k\to\infty} \left(\mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-1)\right), \tag{3.35}$$

since the Markov states are irreducible (Assumption 3. 3)

$\lim_{k\to\infty} \left(\mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-3)\right) = 0$, and

$\lim_{k\to\infty} \left(\mathbb{P}(s_m \neq i \text{ for } 0 \le m \le k-1)\right) = 0$.

Eq. (3.35) becomes

$$= \sum_{k=1}^{M} \{\mathbb{P}(s_0 = i) + \mathbb{P}(s_0 \neq i)\} = \sum_{k=1}^{M} \{1\} = 1.$$

We have shown that

$$\rho_i \cdot \mu_i = \rho_i \cdot \mu_{\tilde{S}_i} = 1.$$

Consequently, the stationary distribution is given by the mean recurrence time of each PRN $\tilde{\mathcal{S}}_i, \mu_{\tilde{\mathcal{S}}_i}$

$$\rho_i = \frac{1}{\mu_{\tilde{S}_i}}.$$

□

3.5 Concluding Remarks

In this chapter, a computational model suited for real-time sequential decision-making under uncertainty was implemented. The evolution of the system was modeled as a Markov chain. A state-space representation was constructed through a learning mechanism and used in solving the state estimation and system identification problem. The model accumulates gradually enhanced knowledge of system response as it transitions from one state to another, in conjunction with actions taken at each state. As the system interacts with its environment, the state representation of the model realizes the sequences of state transitions that occurred in the Markov domain, as infused in the Predictive Representation Nodes (PRNs). It was shown that this realization determines the stationary distribution of the Markov chain (Theorem 3. 3). Utilizing this model, a lookahead control algorithm can be employed simultaneously to address the stochastic control problem in real time. This problem is addressed in Chapter 4.

3.6 References

[1] Bertsekas, D. P. and Shreve, S. E., Stochastic Optimal Control: The Discrete-Time Case, 1st edition, Athena Scientific, February 2007.

[2] Gosavi, A., "Reinforcement Learning for Long-Run Average Cost," European Journal of Operational Research, vol. 155, pp. 654-74, 2004.

[3] Bertsekas, D. P. and Tsitsiklis, J. N., Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3), 1st edition, Athena Scientific, May 1996.

[4] Sutton, R. S. and Barto, A. G., Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning), The MIT Press, March 1998.

[5] Borkar, V. S., "A Learning Algorithm for Discrete-Time Stochastic Control," Probability in the Engineering and Information Sciences, vol. 14, pp. 243-258, 2000.

[6] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, vol. 3, pp. 210-229, 1959.

[7] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers. II -Recent progress," IBM Journal of Research and Development, vol. 11, pp. 601-617, 1967.

[8] Sutton, R. S., Temporal Credit Assignment in Reinforcement Learning, PhD Thesis, University of Massachusetts, Amherst, MA, 1984.

[9] Sutton, R. S., "Learning to Predict by the Methods of Temporal Difference," Machine Learning, vol. 3, pp. 9-44, 1988.

[10] Watkins, C. J., Learning from Delayed Rewards, PhD Thesis, Kings College, Cambridge, England, May 1989.

[11] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement Learning: a Survey," Journal of Artificial Intelligence Research, vol. 4, 1996.

[12] Schwartz, A., "A Reinforcement Learning Method for Maximizing Undiscounted Rewards," Proceedings of the Tenth International Conference on Machine Learning, pp. 298-305, Amherst, Massachusetts, 1993.

[13] Mahadevan, S., "Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results," Machine Learning, vol. 22, pp. 159-195, 1996.

[14] Sutton, R. S., "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," Proceedings of the Seventh International Conference on Machine Learning, pp. 216-224, Austin, TX, USA, 1990.

[15] Sutton, R. S., "Planning by Incremental Dynamic Programming," Proceedings of the Eighth International Workshop on Machine Learning (ML91), pp. 353-357, Evanston, IL, USA, 1991.

[16] Peng, J. and Williams, R. J., "Efficient Learning and Planning Within the Dyna Framework," Proceedings of the IEEE International Conference on Neural Networks (Cat. No.93CH3274-8), pp. 168-74, San Francisco, CA, USA, 1993.

[17] Kumar, P. R. and Varaiya, P., Stochastic Systems, Prentice Hall, June 1986.

[18] Mandl, P., "Estimation and Control in Markov Chains," Advances in Applied Probability, vol. 6, pp. 40-60, 1974.

[19] Borkar, V. and Varaiya, P., "Adaptive control of Markov chains. I. Finite parameter set," IEEE Transactions on Automatic Control, vol. AC-24, pp. 953-7, 1979.

[20] Borkar, V. and Varaiya, P., "Identification and adaptive control of Markov chains," SIAM Journal on Control and Optimization, vol. 20, pp. 470-89, 1982.

[21] Kumar, P. R., "Adaptive Control With a Compact Parameter Set," SIAM Journal on Control and Optimization, vol. 20, pp. 9-13, 1982.

[22] Doshi, B. and Shreve, S. E., "Strong Consistency of a Modified Maximum Likelihood Estimator for Controlled Markov Chains," Journal of Applied Probability, vol. 17, pp. 726-34, 1980.

[23] Kumar, P. R. and Becker, A., "A new Family of Optimal Adaptive Controllers for Markov Chains," IEEE Transactions on Automatic Control, vol. AC-27, pp. 137-46, 1982.

[24] Kumar, P. R. and Lin, W., "Optimal Adaptive Controllers for Unknown Markov Chains," IEEE Transactions on Automatic Control, vol. AC-27, pp. 765-74, 1982.

[25] Sato, M., Abe, K., and Takeda, H., "Learning Control of Finite Markov Chains with Unknown Transition Probabilities," IEEE Transactions on Automatic Control, vol. AC-27, pp. 502-5, 1982.

[26] Sato, M., Abe, K., and Takeda, H., "An Asymptotically Optimal Learning Controller for Finite Markov Chains with Unknown Transition Probabilities," IEEE Transactions on Automatic Control, vol. AC-30, pp. 1147-9, 1985.

[27] Sato, M., Abe, K., and Takeda, H., "Learning Control of Finite Markov Chains with an Explicit Trade-off Between Estimation and Control," IEEE Transactions on Systems, Man and Cybernetics, vol. 18, pp. 677-84, 1988.

[28] Kumar, P. R., "A Survey of Some Results in Stochastic Adaptive Control," SIAM Journal on Control and Optimization, vol. 23, pp. 329-80, 1985.

[29] Varaiya, P., "Adaptive Control of Markov Chains: A Survey," Proceedings of the IFAC Symposium, pp. 89-93, New Delhi, India, 1982.

[30] Lai, T. L. and Robbins, H., "Asymptotically Efficient Adaptive Allocation Rules," Advances Appl. Math., vol. 6, pp. 4-22, 1985.

[31] Anantharam, V., Varaiya, P., and Walrand, J., "Asymptotically Efficient Allocation Rules for the Multiarmed Bandit Problem with Multiple Plays. I. IID Rewards," IEEE Transactions on Automatic Control, vol. AC-32, pp. 968-76, 1987.

[32] Agrawal, R., Hedge, M. V., and Teneketzis, D., "Asymptotically Efficient Adaptive Allocation Rules for the Multiarmed Bandit Problem With Switching Cost," IEEE Transactions on Automatic Control, vol. 33, pp. 899-906, 1988.

[33] Agrawal, R., Teneketzis, D., and Anantharam, V., "Asymptotically Efficient Adaptive Allocation Schemes for Controlled Markov Chains: Finite Parameter Space," IEEE Transactions on Automatic Control, vol. 34, pp. 1249-59, 1989.

[34] Agrawal, R., Teneketzis, D., and Anantharam, V., "Asymptotically Efficient Adaptive Allocation Schemes for Controlled I.I.D. Processes: Finite Parameter Space," IEEE Transactions on Automatic Control, vol. 34, pp. 258-267, 1989.

[35] Graves, T. L. and Tze Leung, L., "Asymptotically Efficient Adaptive Choice of Control Laws in Controlled Markov Chains," SIAM Journal on Control and Optimization, vol. 35, pp. 715-43, 1997.

[36] Agrawal, R. and Teneketzis, D., "Certainty Equivalence Control with Forcing: Revisited," Proceedings of, pp. 2107, Tampa, FL, USA, 1989.

[37] Gubner, J. A., Probability and Random Processes for Electrical and Computer Engineers, 1st edition, Cambridge University Press, June 5, 2006.

[38] Grimmett, G. R. and Stirzaker, D. R., Probability and Random Processes, 3rd edition, Oxford University Press, July 16, 2001.

[39] Krishnan, V., Probability and Random Processes, 1st edition, Wiley-Interscience, July 11, 2006.

[40] Malikopoulos, A. A., Papalambros, P. Y., and Assanis, D. N., "A State-Space Representation Model and Learning Algorithm for Real-Time Decision-Making Under Uncertainty," Proceedings of the 2007 ASME International Mechanical Engineering Congress and Exposition, Seattle, Washington, November 11-15, 2007.

[41] Malikopoulos, A. A., Papalambros, P. Y., and Assanis, D. N., "A Learning Algorithm for Optimal Internal Combustion Engine Calibration in Real Time," Proceedings of the ASME 2007 International Design Engineering Technical Conferences Computers and Information in Engineering Conference, Las Vegas, Nevada, September 4-7, 2007.

[42] Malikopoulos, A. A., Assanis, D. N., and Papalambros, P. Y., "Real-Time, Self-Learning Optimization of Diesel Engine Calibration," Proceedings of the 2007 Fall Technical Conference of the ASME Internal Combustion Engine Division, Charleston, South Carolina, October 14-17, 2007.

[43] Iwata, K., Ito, N., Yamauchi, K., and Ishii, N., "Combining Exploitation-Based and Exploration-Based Approach in Reinforcement Learning," Proceedings of the Intelligent Data Engineering and Automated - IDEAL 2000, pp. 326-31, Hong Kong, China, 2000.

[44] Ishii, S., Yoshida, W., and Yoshimoto, J., "Control of Exploitation-Exploration Meta-Parameter in Reinforcement Learning," Journal of Neural Networks, vol. 15, pp. 665-87, 2002.

[45] Chan-Geon, P. and Sung-Bong, Y., "Implementation of the Agent Using Universal On-Line Q-learning by Balancing Exploration and Exploitation in Reinforcement Learning," Journal of KISS: Software and Applications, vol. 30, pp. 672-80, 2003.

[46] Miyazaki, K. and Yamamura, M., "Marco Polo: a Reinforcement Learning System Considering Tradeoff Exploitation and Exploration under Markovian Environments," Journal of Japanese Society for Artificial Intelligence, vol. 12, pp. 78-89, 1997.

[47] Hernandez-Aguirre, A., Buckles, B. P., and Martinez-Alcantara, A., "The Probably Approximately Correct (PAC) Population Size of a Genetic Algorithm," 12th IEEE Internationals Conference on Tools with Artificial Intelligence, pp. 199-202, 2000.

CHAPTER 4

REAL-TIME STOCHASTIC CONTROL

This chapter presents the algorithmic implementation that provides the decision-making mechanism suitable for real-time implementation. The algorithm solves the stochastic control sub-problem by utilizing accumulated data acquired over the learning process of the POD model developed in Chapter 3. A lookahead control algorithm is proposed that assigns at each state the control actions that minimize the transition cost of the next two decision epochs. The principle of the algorithm is founded on the theory of stochastic control problems known as games against nature. The efficiency of the POD model and the lookahead algorithm is demonstrated on four applications: (a) the single cart-pole balancing problem; (b) a vehicle cruise-control problem; (c) a gasoline engine that learns the optimal spark advance over aggressive acceleration profiles; and (d) a diesel engine that learns the optimal injection timing in a segment of a driving cycle.

4.1 The Predictive Optimal Stochastic Control Algorithm

The POD state representation attempts to provide an efficient process in realizing the state transitions that occurred in the Markov domain. The different sequences of the state transitions are captured by the POD states and evaluated through the expected evaluation functions. Consequently, the lowest value of the expected evaluation function at each PRN essentially estimates the Markov state transitions that will occur. As the process is stochastic, however, it is still necessary for the decision maker to build a

decision-making mechanism for making decisions (selecting control actions). The Predictive Optimal Stochastic Control Algorithm (POSCA), proposed in this dissertation, aims to provide this mechanism.

The principle of POSCA is founded on the theory of stochastic control problems with unknown disturbance distribution, also known as games against nature. The decision-making mechanism is modeled as a stochastic game between the decision maker (controller) and an "opponent" (environment). The solution of this game is derived utilizing the mini-max theorem. Each POD state $\tilde{s}_{k+1}^{ij} \in \tilde{\mathcal{S}}_i$ corresponds to a completed game that started at the Markov state $s_k = i \in \mathcal{S}$ and ended up at $s_{k+1} = j \in \mathcal{S}$. At state $s_k = i$, the decision maker has a set of strategies (control actions) $\mu(s_k) \in A(s_k)$ available to play. Similarly, the environment's set of strategies are the Markov states $\mathcal{S} = \{1, 2, ..., N\}$, $N \in \mathbb{N}$. During the learning process of the POD model, this game has been played insofar as the decision maker forms a belief about the environment's behavior by fully exploring all available strategies, $\mu(s_k) \in A(s_k)$. This property arises when the state representation converges to the stationary distribution of the Markov chain. Consequently, at state $s_k = i \in \mathcal{S}$, the decision maker can select those control actions by means of the PRN expected evaluation functions, $R_{PRN}^{i}(\tilde{s}_{k+1}^{ij}, \mu(s_k))$. However, to handle the uncertainty of this prediction, the decision maker seeks a policy $\pi^* \in \Pi$, which guarantees the best performance in the worst possible situation, namely,

$$\pi^*(s_k) = \underset{\bar{\mu}_k(s_k) \in A(s_k)}{\arg\min} \left\{ \max_{s_{k+1} \in \mathcal{S}} \left[R_{PRN}^{i}(\tilde{s}_{k+1}^{ij}, \mu(s_k)) \right] \right\}. \tag{4.1}$$

This approach is especially appealing for real-time implementation when the time between decision epochs is small. In this situation, the controller needs to select control actions quickly and there is not enough time to search for an optimal policy for a relatively distant future.

The implementation of the POD model and POSCA is illustrated in Figure 4.1. At each decision epoch k, the POD model observes the system's state $s_k = i \in \mathcal{S}$ and control action $a_k \in A(s_k)$ selected from the feasible action set $A(s_k)$, which is a subset of some control space $\mathcal{A}$. At the next decision epoch, the system transits to another state $s_{k+1} = j \in \mathcal{S}$ imposed by the transition probability $\mathbb{P}_{ij}(\cdot)$, and receives a numerical cost $R_{ij}(\cdot)$. The POD state representation realizes the sequences of state transitions $\mathbb{P}_{ij}(\cdot)$ that occurred in the Markov domain and the associated costs $R_{ij}(\cdot)$. When the POD model converges to the stationary distribution, POSCA is employed to derive the control policy π^* by means of Eq. (4.1).

4.1.1 Performance Bound of POSCA

This section evaluates the performance bound of POSCA in terms of the accumulated cost over the decision epochs. The following Lemma aims to provide a useful step toward presenting the main result (Theorem 4. 1).

Lemma 4. 1 [1]. Let $f : \mathcal{S} \to [-\infty, \infty]$ and $g : \mathcal{S} \times \mathcal{A} \to [-\infty, \infty]$ be two functions such that

$$\min_{a \in \mathcal{A}} \big(g(i,a) \big) > -\infty, \forall i \in \mathcal{S}. \tag{4.2}$$

Then we have

$$\min_{\mu(i) \in \mathcal{A}} \max_{i \in \mathcal{S}} [f(i) + g(i, \mu(i))] = \max_{i \in \mathcal{S}} [f(i) + \min_{a \in \mathcal{A}} g(i,a)],$$

where $\mu : \mathcal{S} \to \mathcal{A}$, such that $a = \mu(i)$, and $\mathcal{S}, \mathcal{A}$ are some sets.

Proof. The proof is provided by Bertsekas [1].

□

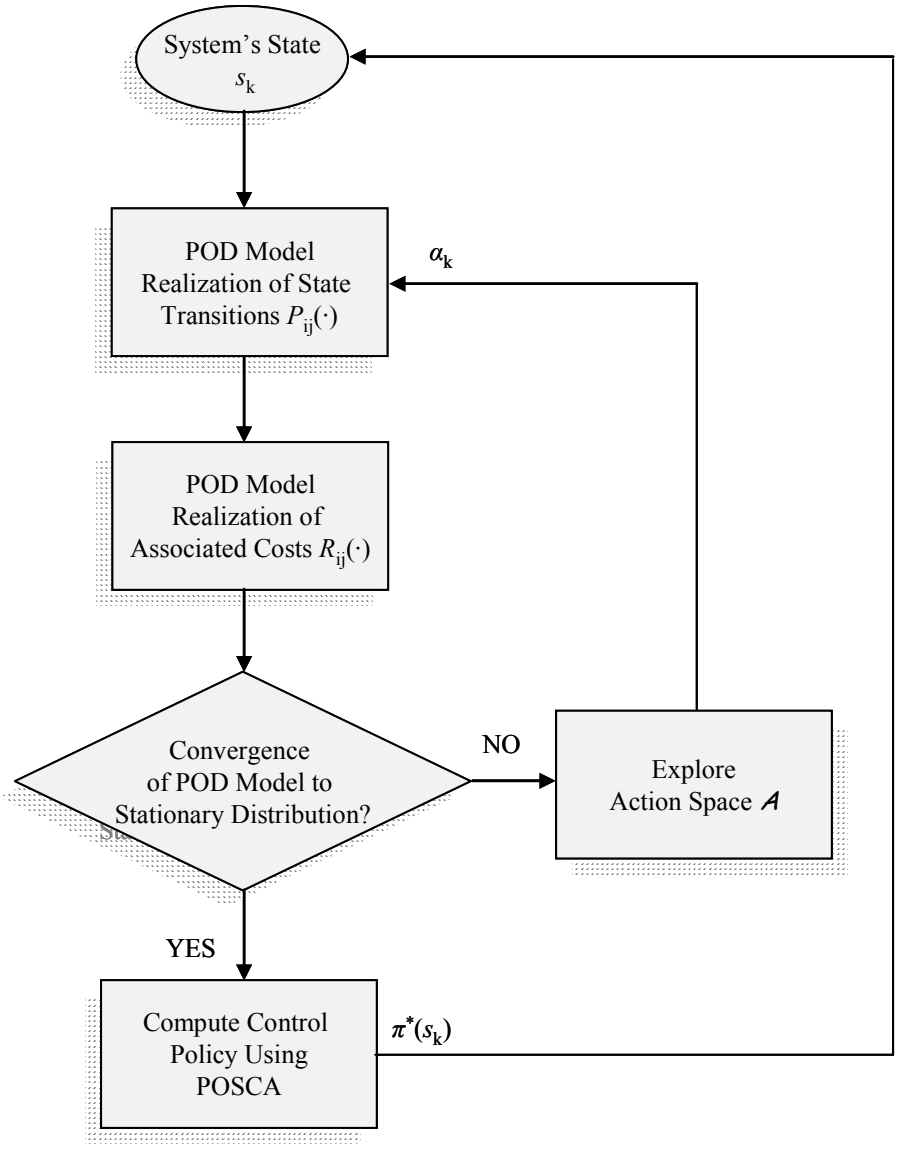

Figure 4.1 − Implementation of POD model and POSCA.

Assumption 4. 1. The accumulated cost incurred at each decision epoch k is bounded, that is, $\exists\ \delta > 0$ such that $J_k(s_k) < \delta$.

Theorem 4. 1. The accumulated cost $\tilde{J}_k(s_k)$ incurred by the two step lookahead policy $\bar{\pi} = \{\bar{\mu}_0, \bar{\mu}_1, ..., \bar{\mu}_{M-1}\}$ of POSCA

$$\bar{\pi}(s_k) = \underset{\bar{\mu}_k(s_k)\in A(s_k)}{\arg\min} \max_{s_{k+1}\in\mathcal{S}} \left[R(s_{k+1} \mid s_k, \alpha_k) + \min_{a_{k+1}\in A(s_{k+1})} E_{s_{k+2}\in\mathcal{S}} \left\{ R(s_{k+2} \mid s_{k+1}, \alpha_{k+1}) \right\} \right], \tag{4.3}$$

is bounded by the accumulated cost $J_k(s_k)$ incurred by the minimax control policy $\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$ of Dynamic Programming (DP), namely,

$$\pi = \underset{\bar{\mu}_k(s_k) \in A(s_k)}{\arg\min} \max_{s_{k+1} \in \mathcal{S}} \left[R(s_{k+1} \mid s_k, \alpha_k) + J_{k+1}(s_{k+1}) \right]. \tag{4.4}$$

with probability 1.

Proof. Suppose that the chain starts at a state $s_0 = i, i \in \mathcal{S}$ at time $k = 0$ and ends up at $k = M$. We consider the problem of finding a policy $\pi = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$ with $\mu_k(s_k) \in \mathcal{A}$ for all $s_k \in \mathcal{S}$ and k that minimizes the cost function

$$J^{\pi}(s_k) = \max_{s_{k+1} \in \mathcal{S}} \left[R_M(s_M \mid s_{M-1}, a_{M-1}) + \sum_{k=0}^{M-2} R_k(s_{k+1} \mid s_k, a_k) \right]. \tag{4.5}$$

The DP algorithm for this problem takes the following form starting from the tail sup-problem

$$J_M(s_M) = \min_{\mu_M(s_M) \in A(s_M)} \max_{s_{M+1} \in \mathcal{S}} \left[R_M(s_{M+1} \mid s_M, a_M) \right] = R_M(s_M), \text{ and} \tag{4.6}$$

$$J_k(s_k) = \min_{\mu_k(s_k) \in A(s_k)} \max_{s_{k+1} \in \mathcal{S}} \left[R(s_{k+1} \mid s_k, \alpha_k) + J_{k+1}(s_{k+1}) \right], \tag{4.7}$$

where $R_M(s_M)$ is the cost of the terminal decision epoch.

Following the steps of the DP algorithm proposed by Bertsekas [1], the optimal accumulated cost $J^{\pi^*}(s_0)$ starting from the last decision epoch and moving backwards is

$$J^{\pi^*}(s_0) = \min_{\mu_0(s_0) \in A(s_0)} ... \min_{\mu_{M-1}(s_{M-1}) \in A(s_{M-1})} \max_{s_0 \in \mathcal{S}} ... \max_{s_M \in \mathcal{S}} \left[R_M(s_M \mid s_{M-1}, a_{M-1}) + \sum_{k=0}^{M-2} R_k(s_{k+1} \mid s_k, a_k) \right]. \tag{4.8}$$

By applying Lemma 4. 1, we can interchange the min over μ_{M-1} and the max over $s_0,...,s_{M-2}$. The required assumption of Lemma 4. 1 (Eq. (4.2)) is implied by Assumption 4. 1. Eq. (4.8) yields

$$J^{\pi^*}(s_0) = \min_{\mu_0(s_0)\in A(s_0)} ... \min_{\mu_{M-2}(s_{M-2})\in A(s_{M-2})}$$
$$\left[\max_{s_0\in\mathcal{S}} ... \max_{s_{M-2}\in\mathcal{S}}\left[\sum_{k=0}^{M-3} R_k(s_{k+1}\mid s_k,a_k) + \max_{s_{M-1}\in\mathcal{S}}\left[R_{M-1}(s_{M-1}\mid s_{M-2},a_{M-2}) + J_M(s_M)\right]\right]\right] \tag{4.9}$$
$$= \min_{\mu_0(s_0)\in A(s_0)} ... \min_{\mu_{M-2}(s_{M-2})\in A(s_{M-2})}\left[\max_{s_0\in\mathcal{S}} ... \max_{s_{M-2}\in\mathcal{S}}\left[\sum_{k=0}^{M-3} R_k(s_{k+1}\mid s_k,a_k) + J_{M-1}(s_{M-1})\right]\right].$$

By continuing backwards in similar way we obtain

$$J^{\pi^*}(s_0) = J_0(s_0). \tag{4.10}$$

Consequently, an optimal policy for the minimax problem can be constructed by minimizing the RHS of Eq. (4.5).

The cost incurred by policy $\bar{\pi} = \{\bar{\mu}_0, \bar{\mu}_1, ..., \bar{\mu}_{M-1}\}$ of POSCA at each decision epoch k is

$$\bar{J}_M(s_M) = \tag{4.11}$$

$$= \min_{\bar{\mu}_M(s_M)\in A(s_M)} \max_{s_{M+1}\in\mathcal{S}}\left[R_M(s_{M+1}\mid s_M,\alpha_M) + \min_{a_{M+1}\in A(s_{M+1})} E_{s_{M+2}\in\mathcal{S}}\left\{R_{M+1}(s_{M+2}\mid s_{M+1},\alpha_{M+1})\right\}\right]$$
$$= R_M(s_{M+1}\mid s_M,a_M) = R_M(s_M), \tag{4.12}$$

since the terminal decision epoch is at $k = M$, and thus, $R_{M+1}(s_{M+2}\mid s_{M+1},\alpha_{M+1}) = 0$.

$$\bar{J}_{M-1}(s_{M-1}) =$$

$$= \min_{\bar{\mu}_{M-1}(s_{M-1}) \in A(s_{M-1})} \max_{s_M \in \mathcal{S}} \left[R_{M-1}(s_M \mid s_{M-1}, \alpha_{M-1}) + \min_{a_M \in A(s_M)} \underset{s_{M+1} \in \mathcal{S}}{E} \left\{ R_M(s_{M+1} \mid s_M, \alpha_M) \right\} \right], \quad (4.13)$$

$$\bar{J}_{M-2}(s_{M-2}) =$$

$$= \min_{\bar{\mu}_{M-2}(s_{M-2}) \in A(s_{M-2})}$$

$$\max_{s_{M-1} \in \mathcal{S}} \left[R_{M-2}(s_{M-1} \mid s_{M-2}, \alpha_{M-2}) + \min_{a_{M-1} \in A(s_{M-1})} \underset{s_M \in \mathcal{S}}{E} \left\{ R_{M-1}(s_M \mid s_{M-1}, \alpha_{M-1}) \right\} \right], \quad (4.14)$$

…

$$\bar{J}_0(s_0) = \min_{\bar{\mu}_0(s_0) \in A(s_0)} \max_{s_1 \in \mathcal{S}} \left[R_0(s_1 \mid s_0, \alpha_0) + \min_{a_1 \in A(s_1)} \underset{s_2 \in \mathcal{S}}{E} \left\{ R_1(s_2 \mid s_1, \alpha_1) \right\} \right]. \quad (4.15)$$

Performing the same task as we did with DP algorithm by starting from the last epoch of the decision-making process and moving backwards, the accumulated cost incurred by POSCA $\tilde{J}_k(s_k)$ is

$$\tilde{J}_M(s_M) =$$

$$= \min_{\bar{\mu}_M(s_M) \in A(s_M)} \max_{s_{M+1} \in \mathcal{S}} \left[R_M(s_{M+1} \mid s_M, \alpha_M) + \min_{a_{M+1} \in A(s_{M+1})} \underset{s_{M+2} \in \mathcal{S}}{E} \left\{ R_{M+1}(s_{M+2} \mid s_{M+1}, \alpha_{M+1}) \right\} \right]$$
$$= R_M(s_{M+1} \mid s_M, a_M) = R_M(s_M) = J_M(s_M), \quad (4.16)$$

$$\tilde{J}_{M-1}(s_{M-1}) =$$

$$= \min_{\bar{\mu}_{M-1}(s_{M-1}) \in A(s_{M-1})}$$

$$\max_{s_M \in \mathcal{S}} \left[R_{M-1}(s_M \mid s_{M-1}, \alpha_{M-1}) + \min_{a_M \in A(s_M)} \underset{s_{M+1} \in \mathcal{S}}{E} \left\{ R_M(s_{M+1} \mid s_M, \alpha_M) \right\} \right] + \tilde{J}_M(s_M) \quad (4.17)$$

$$= \min_{\bar{\mu}_{M-1}(s_{M-1}) \in A(s_{M-1})} \max_{s_M \in \mathcal{S}} \left[R_{M-1}(s_M \mid s_{M-1}, \alpha_{M-1}) \right] + \tilde{J}_M(s_M), \quad (4.18)$$

since $R_M(s_{M+1} \mid s_M, \alpha_M) = 0$.

$$\tilde{J}_{M-1}(s_{M-1}) = \min_{\bar{\mu}_M(s_M)\in A(s_M)} \max_{s_{M+1}\in\mathcal{S}} \left[R(s_{M+1} \mid s_M, \alpha_M) + \tilde{J}_M(s_M) \right] = J_{M-1}(s_{M-1}), \tag{4.19}$$

since $\tilde{J}_M(s_M)$ is a constant quantity.

$$\begin{aligned} &\tilde{J}_{M-2}(s_{M-2}) = \\ &= \min_{\bar{\mu}_{M-2}(s_{M-2})\in A(s_{M-2})} \\ &\max_{s_{M-1}\in\mathcal{S}} \left[R_{M-2}(s_{M-1} \mid s_{M-2}, \alpha_{M-2}) + \min_{a_{M-1}\in A(s_{M-1})} E_{s_M\in\mathcal{S}} \left\{ R_{M-1}(s_M \mid s_{M-1}, \alpha_{M-1}) \right\} \right] + \\ &+ \tilde{J}_{M-1}(s_{M-1}). \end{aligned} \tag{4.20}$$

However,

$$\begin{aligned} &\min_{\bar{\mu}_{M-2}(s_{M-2})\in A(s_{M-2})} \\ &\max_{s_{M-1}\in\mathcal{S}} \left[R_{M-2}(s_{M-1} \mid s_{M-2}, \alpha_{M-2}) + \min_{a_{M-1}\in A(s_{M-1})} E_{s_M\in\mathcal{S}} \left\{ R_M(s_M \mid s_{M-1}, \alpha_{M-1}) \right\} \right] \le \\ &\min_{\mu_{M-2}(s_{M-2})\in A(s_{M-2})} \max_{s_{M-1}\in\mathcal{S}} \left[R_{M-1}(s_{M-1} \mid s_{M-2}, a_{M-2}) \right], \end{aligned} \tag{4.21}$$

since the LHS of the inequality will return a cost $R_{M-2}(\cdot \mid \cdot, \cdot)$ which is not only maximum over when the chain transits from s_{M-2} to s_{M-1} but also minimum when the chain transits from s_{M-1} to s_M. So, the LHS can be at most equal to the cost which is maximum over the transition from s_{M-2} to s_{M-1}. Consequently, comparing the accumulated cost of POSCA in Eq. (4.20) with the one resulted from the DP at the same decision epoch, namely,

$$J_{M-2}(s_{M-2}) = \min_{\mu_{M-2}(s_{M-2})\in A(s_{M-2})} \max_{s_{M-1}\in\mathcal{S}} \left[R_{M-1}(s_{M-1} \mid s_{M-2}, a_{M-2}) + J_{M-1}(s_{M-1}) \right], \tag{4.22}$$

we conclude that

$$\tilde{J}_{M-2}(s_{M-2}) \le J_{M-2}(s_{M-2}) \tag{4.23}$$

By continuing backward with similar arguments we have

$$\tilde{J}_0(s_0) \le J_0(s_0) = J^{\pi^*}(s_0). \tag{4.24}$$

Consequently, the accumulated cost resulting from the control policy $\bar{\pi} = \{\bar{\mu}_0, \bar{\mu}_1, ..., \bar{\mu}_{M-1}\}$ of POSCA is bounded by the accumulated cost of the optimal minimax control policy of DP with probability 1.

□

4.2 Application: Single Cart-Pole Balancing Problem

The overall performance of the POD model and POSCA is evaluated on the basis of its application to the inverted pendulum balancing problem. The inverted pendulum involves a pendulum hinged to the top of a wheeled cart as illustrated in Figure 4.2. The objective of POD is to balance the pendulum having no prior knowledge about the system dynamics, utilizing only real-time measurements.

Realizing the balance control policy of a single inverted pendulum without *a priori* knowledge of the system's model has been extensively reported in the literature for the evaluation of learning algorithms. Anderson [2] implemented a neural network reinforcement-learning method to generate successful action sequences. Two neural networks having a similar structure were employed to learn two functions: (a) an action function mapping the current state into control actions, and (b) an evaluation action mapping the current state into an evaluation of that state. These two networks were trained utilizing reinforcement learning by evaluating the performance of the network and compared to real-time measurements. Williams *et al.* [3] proposed a learning architecture for training a neural network controller to provide the appropriate control force to balance

the inverted pendulum. One network for the identification of the plant dynamics and one for the controller were employed. Zhidong *et al.* [4] implemented a "neural-fuzzy BOXES" control system by neural networks and utilized reinforcement learning for the training. Jeen-Shing *et al.* [5] proposed a defuzzification method incorporating a genetic algorithm to learn the defuzzification factors. Mustapha *et al.* [6] developed an actor-critic reinforcement learning algorithm represented by two adaptive neural-fuzzy systems. Si *et al.* [7] proposed a generic on-line learning control system similar to Anderson's utilizing neural networks and evaluated it through its application to both a single and double cart-pole balancing problem. The system utilizes two neural networks, and employs the action- dependent heuristic dynamic programming to adapt the weights of the networks.

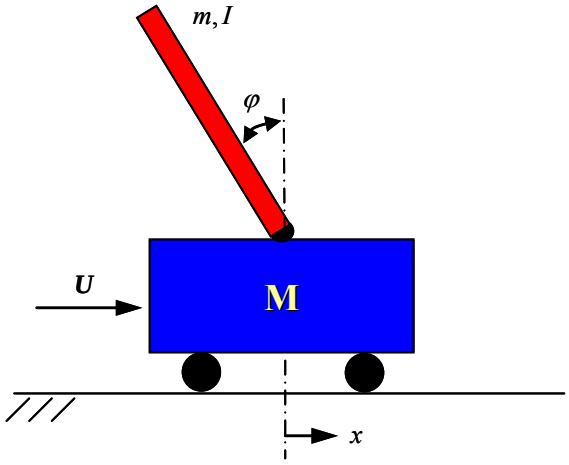

Figure 4.2 − The inverted pendulum.

In the implementation of the POD on the single inverted pendulum presented here, two major variations are considered: (a) a single look-up table-based representation is employed for the controller to develop the mapping from the system's Markov states to optimal actions, and (b) two of the system's state variables are selected to represent the Markov state. The latter introduces uncertainty and thus a conditional probability

distribution associating the state transitions with respect to the actions taken. Consequently, the POD model is evaluated in deriving the optimal policy (balance control policy) in a sequential decision making problem under uncertainty.

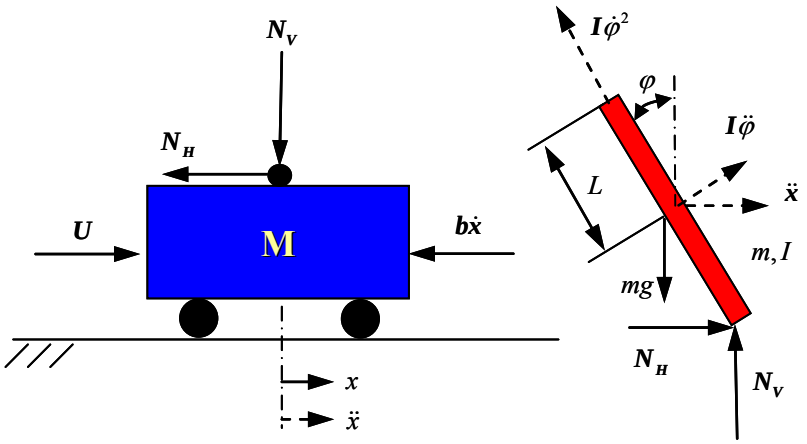

Figure 4.3 − Free body diagram of the system.

The governing equations, derived from the free body diagram of the system, shown in Figure 4.3, are:

$$(M+m)\ddot{x}+b\dot{x}+mL\ddot{\varphi}\cos\varphi-mL\dot{\varphi}^2\sin\varphi=U, \tag{4.25}$$

$$mL\ddot{x}\cos\varphi+(I+mL^2)\ddot{\varphi}+mgL\sin\varphi=0,$$

where $M=0.5$ kg, $m=0.2$ kg, $b=0.1\ \dfrac{\text{N sec}}{\text{m}}$,

$I=0.006$ kg m^2, $g=9.81\ \dfrac{\text{m}}{\text{sec}^2}$, and $L=0.3$ m.

The goal of the learning controller is to realize in real time the force, U, of a fixed magnitude to be applied either to the right or the left direction so that the pendulum stands balanced when released from any angle, φ, between 3° and -3°. The system is simulated by numerically solving the nonlinear differential equations (4.25) employing

the explicit Runge-Kutta method with a time step of τ =0.02 sec. The simulation is conducted by observing the system's states and executing actions (control force U) with a sample rate T =0.02 sec (50 Hz). This sample rate defines a sequence of decision-making epochs, $k = 0,1,2,...,M,\ M \in \mathcal{N}$.

The system is fully specified by four state variables: (a) the position of the cart on the track, $x(t)$; (b) the cart velocity, $\dot{x}(t)$; (c) the pendulum's angle with respect to the vertical position, $\varphi(t)$; and (d) the angular velocity, $\dot{\varphi}(t)$. However, to incorporate uncertainty, the Markov states are selected to be only the pair of the pendulum's angle and angular velocity, namely, the finite state space $\mathcal{S}$ is defined as

$$\mathcal{S} = \{i \mid i = (\varphi, \dot{\varphi})\}. \tag{4.26}$$

Consequently, at state $s_k = i \in \mathcal{S}$ and executing a control force value, U_k , the system will end up at state $s_{k+1} = j \in \mathcal{S}$ with a conditional probability $p(s_{k+1} = j \mid s_k = i, U_k)$. The control force, U_k , selects values from the finite set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k = i) = [-3N, 3N], \forall k \geq 0, \forall i \in \mathcal{S}, \tag{4.27}$$

where $i = 1,2,...,N,\ N = |\mathcal{S}|$.

Each state of the POD state space $\tilde{\mathcal{S}}$ represents a Markov state transition from $s_k = i \in \mathcal{S}$ to $s_{k+1} = j \in \mathcal{S}$ for all $k \geq 0$, that is

$$\tilde{\mathcal{S}} := \left\{ \tilde{s}_{k+1}^{ij} \mid \tilde{s}_{k+1}^{ij} \equiv s_k = i \underset{\mu(s_k) \in A(s_k)}{\rightarrow} s_{k+1} = j,\ \sum_{j=1}^{N} p(s_{k+1} = j \mid s_k = i, U_k) = 1, N = |\mathcal{S}| \right\}. \tag{4.28}$$

The decision-making process occurs at each of a sequence of epochs $k = 0,1,2,...,M,\ M \in \mathbb{N}$. At each decision epoch k , the learning controller observes the

system's state $s_k = i \in \mathcal{S}$, and executes a control force value $U_k \in A(s_k)$. At the next decision epoch, the system transits to another state $s_{k+1} = j \in \mathcal{S}$ imposed by the conditional probability $p(s_{k+1} = j \mid s_k = i, U_k)$, and receives a numerical cost $R(s_{k+1} = j \mid s_k = i, U_k)$ (the pendulum's angle φ). The control policy $\bar{\pi} = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$, where the functions μ_k specify the control $U_k = \mu(s_k)$, is derived by means of the following equation

$$\bar{\pi}(s_k) = \underset{\bar{\mu}_k(s_k) \in A(s_k)}{\arg\min} \left\{ \max_{s_{k+1} \in \mathcal{S}} \left[R^i_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) \right] \right\}, \tag{4.29}$$

where

$$R^i_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) = \{ p(s_{k+1} = j \mid s_k = i, \mu(s_k)) \cdot R(s_{k+1} = j \mid s_k = i, \mu(s_k)) + \\ + \bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1})) \}, \tag{4.30}$$

where

$$\bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1})) := \\ := \min_{\mu(s_{k+1}) \in \mathcal{A}} \sum_{j=1}^{N} p(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})) \cdot R(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})), \tag{4.31}$$

$\forall \tilde{s}^{jm}_{k+2} \in \tilde{\mathcal{S}}, \forall i, j \in \mathcal{S}, \forall \mu(s_k) \in A(s_k)$, and $N = |\mathcal{S}|$.

The inverted pendulum is simulated repeatedly for different initial angles, φ, between 3° and -3° utilizing the POD learning method. The simulation lasts for 50 sec and each complete simulation defines one iteration. If at any instant during the simulation, the pendulum's angle, φ, becomes greater than 3° or less than -3°, this constitutes a failure, denoted by stating that there was one iteration associated with a failure. If, however, no failure occurs during the simulation, this is denoted by stating that there was one iteration associated with no failure.

4.2.1 Simulation Results

After completing the learning process, the controller employing the POD learning method realizes the balance control policy of the pendulum, as illustrated in Figure 4.4. In some instances, however, the system's response demonstrates some overshoots or delays during the transient period, shown in Figure 4.5. This can be handled by a denser parameterization of the state-space or adding a penalty in long transient responses. The efficiency of the POD learning method in deriving the optimal balance control policy that stabilizes the system is illustrated in Figure 4.6. It is noted that after POD realizes the optimal policy in 749 failures and, afterwards, as the number of iterations continues, no further failures occur.

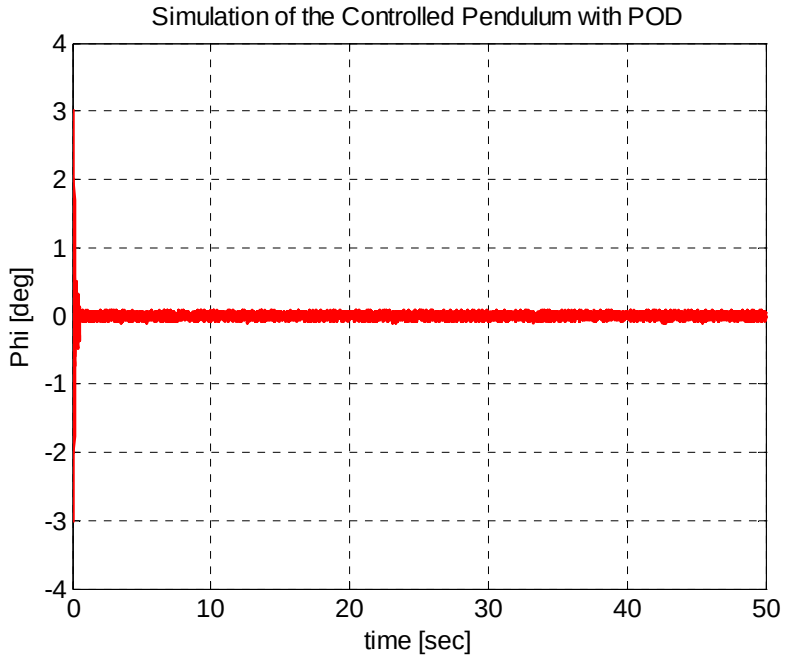

Figure 4.4 − Simulation of the system after learning the balance control policy with POD for different initial conditions.

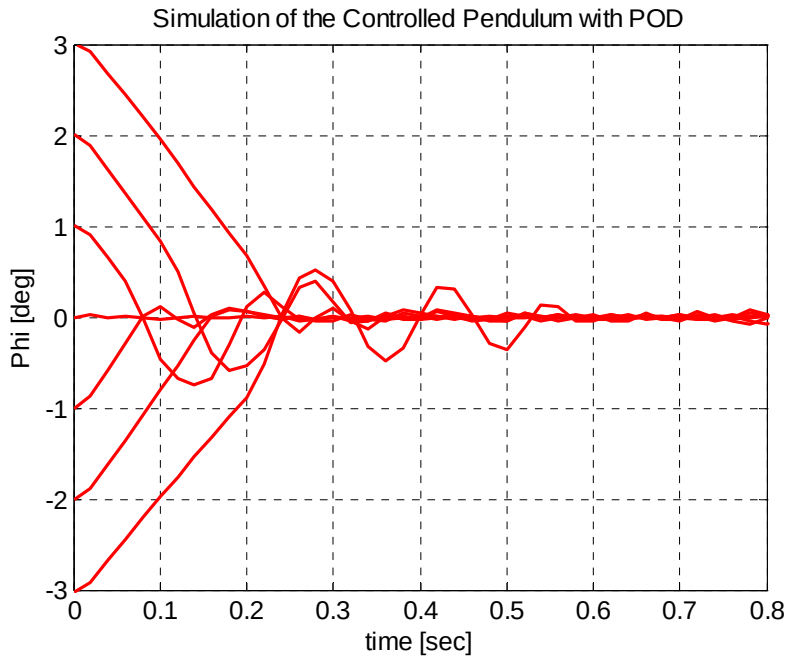

Figure 4.5 − Simulation of the system after learning the balance control policy with POD for different initial conditions (zoom in).

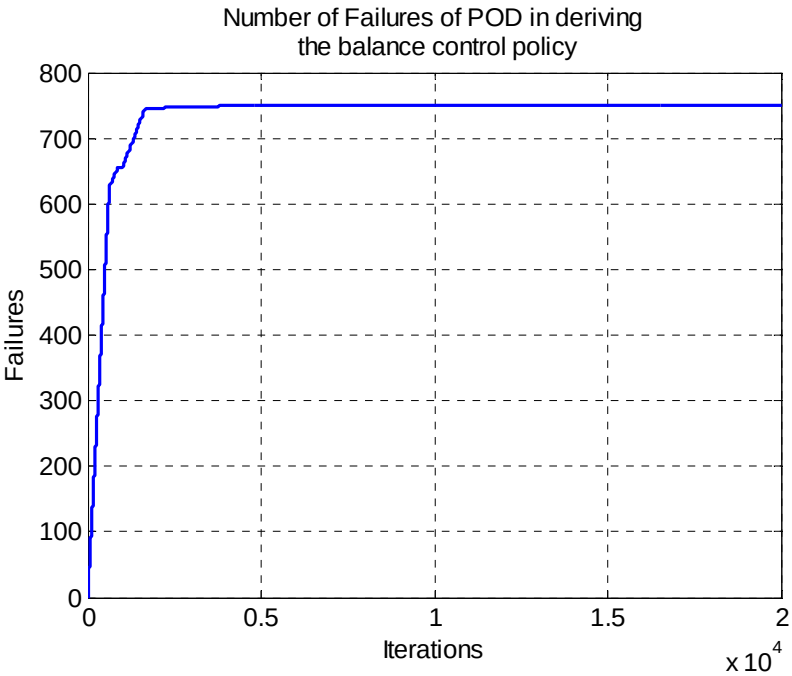

Figure 4.6 − Number of failures until POD derives the balance control policy.

4.3 Application: Autonomous Vehicle Cruise Control

In this section, the overall performance of the POD model and POSCA is demonstrated on a vehicle cruise-control problem. Cruise control automatically regulates the vehicle's longitudinal velocity by suitably adjusting the gas pedal position. A vehicle cruise-control system is activated by the driver who desires to maintain a constant speed in long highway driving. The driver activates the cruise controller while driving at a particular speed, which is then recorded as the desired or set-point speed to be maintained by the controller. The main goal in designing a cruise control algorithm is to maintain vehicle speed smoothly but accurately, even under large variation of plant parameters (e.g., the vehicle's varying mass in terms of the number of passengers) and road grade. In the case of passenger cars, however, vehicle mass may change noticeably but is within a small range. Therefore, powertrain behavior might not vary significantly.

The objective of the POD learning cruise controller is to realize in real time the control policy (gas pedal position) that maintains the vehicle speed as set by the driver under a great range of different road grades. Implementing learning vehicle cruise controllers has been addressed previously, employing learning and active control approaches. Zhang *et al.* [8] implemented learning control based on pattern recognition to regulate in real time the parameters of a PID cruise controller. Shahdi *et al.* [9] proposed an active learning method to extract the driver's behavior and to derive control rules for a cruise control system. However, no attempt has been reported in implementing a learning automotive vehicle cruise controller utilizing the principle of reinforcement learning, i.e., enabling the controller to improve its performance over time by learning from its own failures through a reinforcement signal from the external environment, and thus attempting to improve future performance.

The software package enDYNA by TESIS [10], suitable for real-time simulation of internal combustion engines, is used to evaluate the performance of the POD learning

cruise controller. The software simulates the longitudinal vehicle dynamics with a highly variable drivetrain including the modules of starter, brake, clutch, converter, and transmission. In the driving mode the engine is operated by means of the usual vehicle control elements just as a driver would do. In addition, a mechanical parking lock and the uphill grade can be set. The driver model is designed to operate the vehicle at given speed profiles (driving cycles). It actuates the starter, accelerator, clutch and brake pedals according to the profile specification, and also shifts gears. In this example, an existing vehicle model is selected representing a midsize passenger car carrying a 1.9-L turbocharged diesel engine.

When activated, the learning cruise controller bypasses the driver model and takes over the vehicle's cruising. The Markov states are defined to be the pair of the transmission gear and the difference between the desired and actual vehicle speed, ΔV, namely,

$$\mathcal{S} = \{i \mid i = (gear, \Delta V)\}. \tag{4.32}$$

The actions, a, correspond to the gas pedal position and can take values from the feasible set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k = i) = [0, 0.7], \tag{4.33}$$

where $i = 1, 2, ..., N,\ N = |\mathcal{S}|$.

To incorporate uncertainty the vehicle is simulated in a great range of different road grades from 0° to 10°. Each state of the POD state space represents a Markov state transition from $s_k = i \in \mathcal{S}$ to $s_{k+1} = j \in \mathcal{S}$ for all $k \geq 0$, that is

$$\tilde{\mathcal{S}} := \left\{ \tilde{s}_{k+1}^{ij} \mid \tilde{s}_{k+1}^{ij} \equiv s_k = i \underset{\mu(s_k) \in A(s_k)}{\rightarrow} s_{k+1} = j,\ \sum_{j=1}^{N} p(s_{k+1} = j \mid s_k = i, a_k) = 1, N = |\mathcal{S}| \right\}. \tag{4.34}$$

The decision-making process occurs at each of a sequence of epochs $k = 0,1,2,...,M$, $M \in \mathbb{N}$. At each decision epoch k, the cruise controller observes the system's state $s_k = i \in \mathcal{S}$, and selects a pedal position $a \in A(s_k)$. At the next decision epoch, the system transits to another state $s_{k+1} = j \in \mathcal{S}$ imposed by the conditional probability $p(s_{k+1} = j \mid s_k = i, a_k)$, and receives a numerical cost $R(s_{k+1} = j \mid s_k = i, a_k)$ (difference between the desired and actual vehicle speed). The control policy $\bar{\pi} = \{\mu_0, \mu_1, ..., \mu_{M-1}\}$, where the functions μ_k specify the pedal position $a_k = \mu(s_k)$, is derived by means of the following equation

$$\bar{\pi}(s_k) = \arg\min_{\bar{\mu}_k(s_k) \in A(s_k)} \left\{ \max_{s_{k+1} \in \mathcal{S}} \left[R^i_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) \right] \right\}, \tag{4.35}$$

where

$$R^i_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) = \{ p(s_{k+1} = j \mid s_k = i, \mu(s_k)) \cdot R(s_{k+1} = j \mid s_k = i, \mu(s_k)) + \\ + \bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1})) \}, \tag{4.36}$$

where

$$\bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1})) := \\ := \min_{\mu(s_{k+1}) \in \mathcal{A}} \sum_{j=1}^{N} p(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})) \cdot R(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})), \tag{4.37}$$

$\forall \tilde{s}^{jm}_{k+2} \in \tilde{\mathcal{S}}, \forall i, j \in \mathcal{S}, \forall \mu(s_k) \in A(s_k)$, and $N = |\mathcal{S}|$.

4.3.1 Simulation Results

After completing the learning process for each road grade, the POD cruise controller realizes the control policy (gas pedal position) to maintain the vehicle's speed at the desired set point. The vehicle model was initiated from zero speed. The driver model, following the driving cycle, accelerated the vehicle up to 40 mph and at 10 sec

activated the POD cruise controller. The desired and actual vehicle speeds for three different road grades as well as the gas pedal rates of the POD controller are illustrated in Figure 4.7. The small discrepancy between the desired and actual vehicle speed before the cruise controller activation is due to the steady-state error of the driver's model. However, since the desired driving cycle set the vehicle's speed at 40 mph, activation of the POD cruise controller helps to correct this error and, afterwards, maintains the vehicle's actual speed at the set point. The accelerator pedal position is at different values because, in the case of road grades 2º and 6º, the selected transmission gear is 2, shown in Figure 4.8, while in case of road grade 10º the selected transmission gear is 1. So, at different selected gears, the accelerator pedal position varies to maintain constant vehicle speed. In Figure 4.9, the performance of the POD cruise controller is evaluated on a severe driving scenario where the road grade changes from 0° to 10°, while the POD cruise controller is active. In this scenario, the POD is activated again at 10 sec when the road grade is 0°, and at 14 sec the road grade becomes 10°. The engine speed and the selected transmission gear for this scenario are shown in Figure 4.10. While the vehicle is cruising at constant speed and the road grade changes from 0º to 10º, the vehicle's speed starts to decrease after some time. Once this occurs, the self-learning cruise controller senses the discrepancy between the desired and actual vehicle speed and commands the accelerator pedal so as to correct the error. Consequently, there is a small time delay in the acceleration pedal command, shown in Figure 4.9, which depends on vehicle inertia.

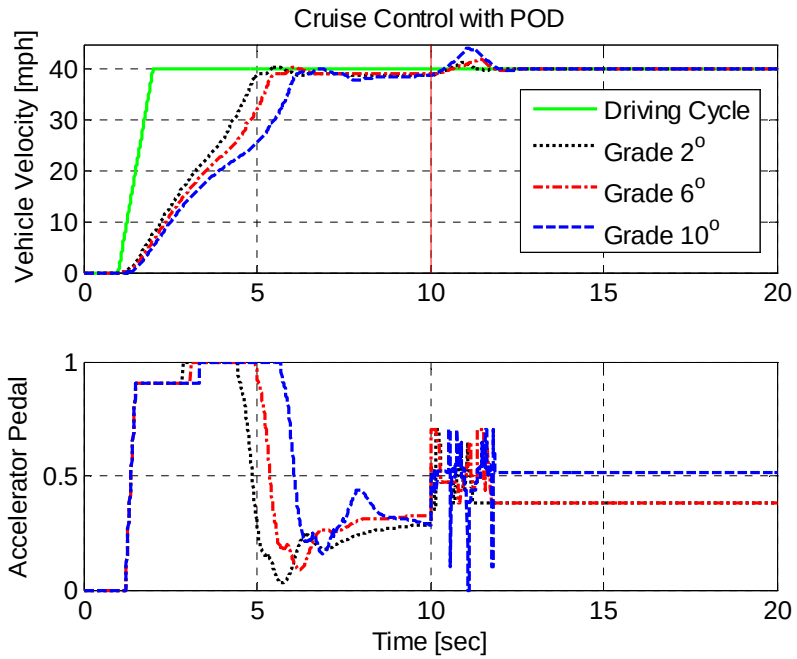

Figure 4.7 − Vehicle speed and accelerator pedal rate for different road grades by self-learning cruise control with POD.

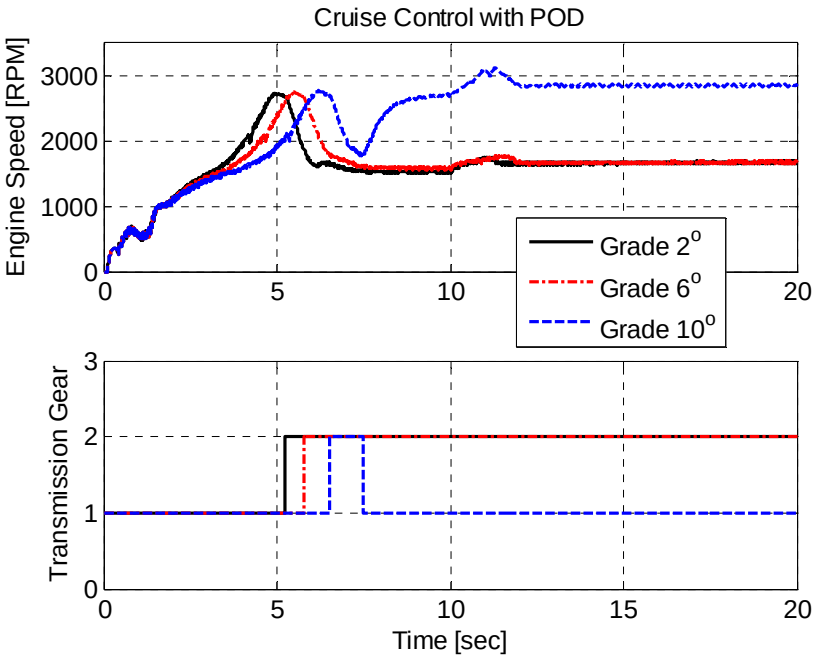

Figure 4.8 − Engine speed and transmission gear selection for different road grades by self-learning cruise control with POD.

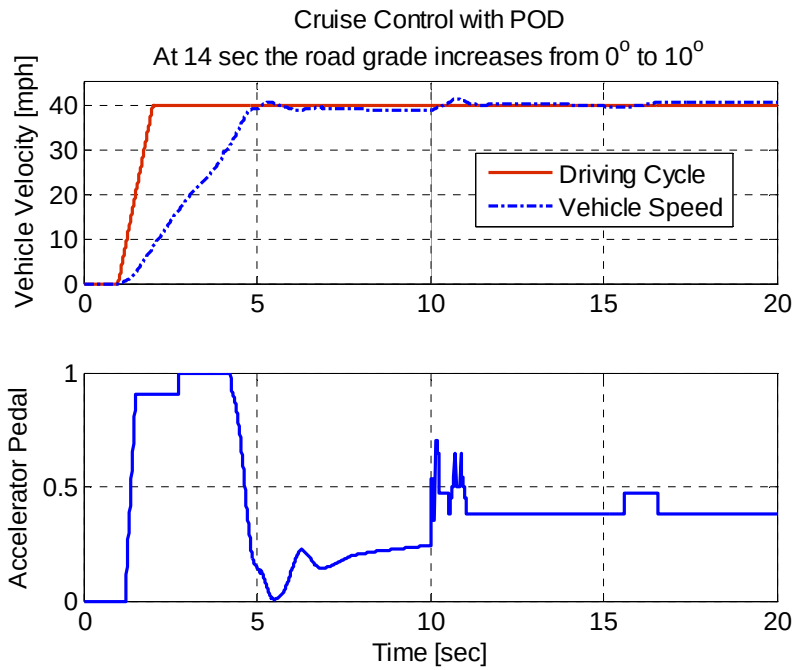

Figure 4.9 − Vehicle speed and accelerator pedal rate for a road grade increase from 0° to 10°.

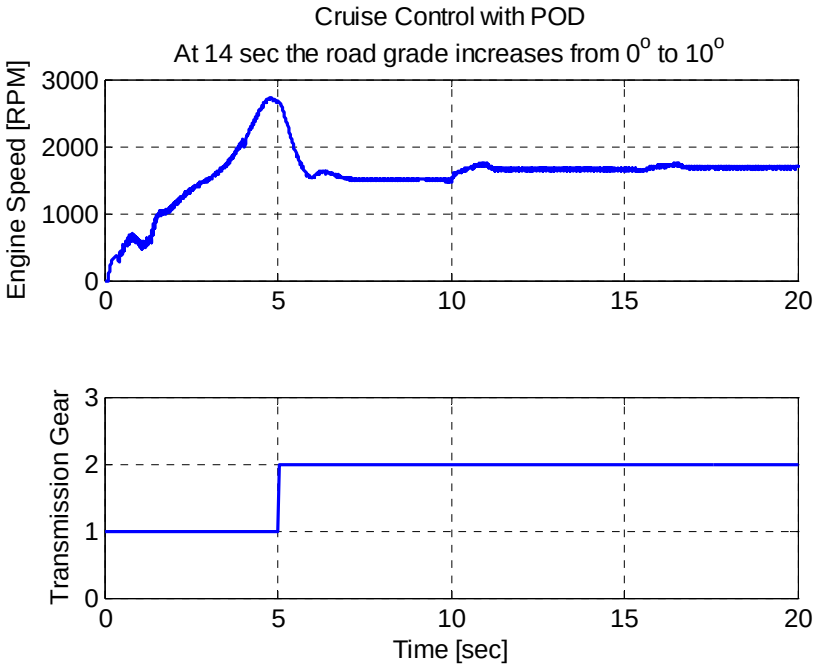

Figure 4.10 − Engine speed and transmission gear selection for a road grade increase from 0° to 10°.

4.4 Real-Time, Self-Learning Optimization of Engine Calibration

In this section, the POD model and POSCA are applied to make the engine an autonomous intelligent system that can learn the values of the controllable variables in real time for each engine operating point transition that optimize specified engine performance criteria, e.g., engine power, fuel economy, or pollutant emissions. The learning process transpires while the engine is running the vehicle and interacting with the driver. Taken in conjunction with assigning values of the controllable variables from the feasible action space, $\mathcal{A}$, this interaction portrays the progressive enhancement of the controller's "knowledge" of the driver's driving style with respect to the controllable variables. This property arises due to the learning process required by the POD state representation to capture the stationary distribution of the engine operation with respect to the driver's driving style. More precisely, at each of a sequence of decision epochs $k = 0,1,2,...,M$, the driver introduces a state $s_k = i \in \mathcal{S}$ (engine operating point) to the engine's self-learning controller, and on that basis the controller selects an action, $\alpha_k \in A(s_k)$ (values of the controllable variables). This state arises as a result of the driver's driving style corresponding to particular engine operating points. One decision epoch later, as a consequence of its action, the engine receives a numerical cost, $R_{k+1} \in \mathbb{R}$, and transits to a new state $s_{k+1} = j, j \in \mathcal{S}$ as illustrated in Figure 4.11.

The POD state-space representation $\tilde{\mathcal{S}}$, is implemented by a mapping $\mathcal{H}$ from the Cartesian product of the finite state space and action space of the Markov chain

$$\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \tilde{\mathcal{S}}, \tag{4.38}$$

where $\mathcal{S} = \{1,2,...,N\}$, $N \in \mathbb{N}$ denotes the engine operating domain, and $\mathcal{A} = \bigcup_{s_k \in \mathcal{S}} A(s_k), \forall s_k = i \in \mathcal{S}$ stands for the values of the controllable variables. Each state of the POD domain represents a Markov state transition from $s_k = i \in \mathcal{S}$ to $s_{k+1} = j \in \mathcal{S}$ for all $k \geq 0$, that is

$$\tilde{\mathcal{S}} := \left\{ \tilde{s}_{k+1}^{ij} \mid \tilde{s}_{k+1}^{ij} \equiv s_k = i \underset{\mu(s_k)\in A(s_k)}{\rightarrow} s_{k+1}=j,\ \sum_{j=1}^{N} p(s_{k+1}=j \mid s_k=i,\alpha_k)=1, N=|\mathcal{S}| \right\}, \tag{4.39}$$

$\forall i, j \in \mathcal{S},\ \forall \mu(s_k) \in A(s_k).$

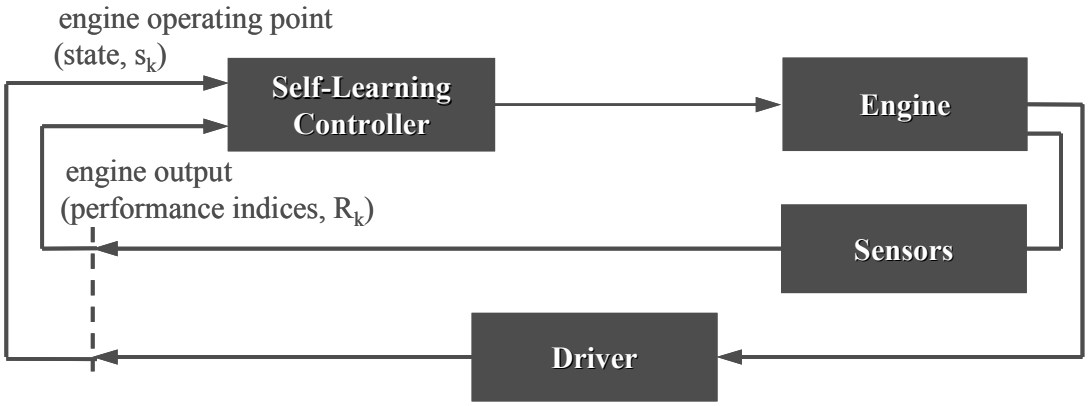

Figure 4.11 − The learning process during the interaction between the engine and the driver.

At each decision epoch, the controller implements a mapping from the Cartesian product of the state space and action space to the set of real numbers, $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, by means of the costs that it receives $R(s_{k+1}=j \mid s_k=i,\alpha_k)$. Similarly, another mapping from the Cartesian product of the state space and action space to the closed set [0,1] is executed, $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$, that provides the realization of the engine operating point transitions $p(s_{k+1}=j \mid s_k=i,\alpha_k)$. The implementation of these two mappings aims POSCA to compute the optimal control policy $\bar{\pi}$ of the self-learning controller

$$\bar{\pi}(s_k) = \underset{\bar{\mu}_k(s_k)\in A(s_k)}{\arg\min} \left\{ \max_{s_{k+1}\in\mathcal{S}} \left[R_{PRN}^{i}(\tilde{s}_{k+1}^{ij}, \mu(s_k)) \right] \right\}, \tag{4.40}$$

where

$$R^{i}_{PRN}(\tilde{s}^{ij}_{k+1}, \mu(s_k)) = \{ p(s_{k+1} = j \mid s_k = i, \mu(s_k)) \cdot R(s_{k+1} = j \mid s_k = i, \mu(s_k)) + \\ + \bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1}))\}, \tag{4.41}$$

where

$$\bar{R}_j(\tilde{s}^{jm}_{k+2} \mid \mu(s_{k+1})) := \\ := \min_{\mu(s_{k+1}) \in \mathcal{A}} \sum_{j=1}^{N} p(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})) \cdot R(s_{k+2} = m \mid s_{k+1} = j, \mu(s_{k+1})), \tag{4.42}$$

$\forall \tilde{s}^{jm}_{k+2} \in \tilde{\mathcal{S}}, \forall i, j \in \mathcal{S}, \forall \mu(s_k) \in A(s_k), \text{and } N = \mid \mathcal{S} \mid .$

4.5 Application: Self-Learning Spark Ignition in a Gasoline Engine

An example of real-time, self-learning optimization of the calibration with respect to spark ignition timing in a spark ignition engine is presented in this section. In spark ignition engines the fuel and air mixture is prepared in advance before it is ignited by the spark discharge. The major objectives for the spark ignition are to initiate a stable combustion and to ignite the air-fuel mixture at the crank angle resulting in maximum efficiency, while fulfilling emissions standards and preventing the engine from knocking. Simultaneous achievement of the aforementioned objectives is sometimes inconsistent; for instance, at high engine loads the ignition timing for maximum efficiency has to be abandoned in favor of prevention of engine destruction by way of engine knock. Two essential parameters are controlled with the spark ignition: ignition energy and ignition timing. Control of ignition energy is important for assuring combustion initiation, but the focus here is on the spark timing that maximizes engine efficiency. Ignition timing influences nearly all engine outputs and is essential for efficiency, drivability, and emissions. The optimum spark ignition timing generating the maximum engine brake torque is defined as Maximum Brake Torque (MBT) timing [11]. Any ignition timing

that deviates from MBT lowers the engine's output torque as illustrated in Figure 4.12. A useful parameter for evaluating fuel consumption of an engine is the Brake-Specific Fuel Consumption (BSFC), defined as the fuel flow rate per unit power output. This parameter evaluates how efficiently an engine is utilizing the fuel supplied to produce work

$$bsfc(g/kW \cdot h) = \frac{\dot{m}_f(g/h)}{P(kW)}, \tag{4.43}$$

where $\dot{m}_f$ is the fuel mass flow rate per unit time and P is engine's power output. Continuous engine operation at MBT ensures optimum fuel economy with respect to the spark ignition timing.

For a successful engine calibration with respect to spark ignition timing, the engine should realize the MBT timing for each engine operating point (steady-state and transient) dictated by the driving style of a driver. Consequently, by achieving MBT timing for all steady-state and transient operating points an overall improvement of the BSFC is expected. Aspects of preventing knocking are not considered in this example; however, they can be easily incorporated by defining the spark ignition space to include the maximum allowable values.

The software package enDYNA is employed for the implementation or real-time, self-learning optimization of engine calibration. The software utilizes thermodynamic models of the gas path and is well suited for testing and development of Electronic Control Units (ECUs). In the example, a four-cylinder gasoline engine is used from the enDYNA model database. The software's static correlation involving spark ignition timing and engine operating points is bypassed to incorporate the self-learning controller. This correlation is designated by the baseline calibration that enDYNA model is accompanied by, and is included in, the engine's ECU.

The control actions, a_k, correspond to the spark ignition timing that can take values from the feasible set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k = i) = [5°, 35°], \tag{4.44}$$

where $i = 1, 2, ..., N$, $N = |\mathcal{S}|$.

The engine model is run repeatedly over the same driving style represented by the pedal position. Every run over this driving style constitutes one complete simulation. To evaluate the efficiency of our approach in both steady-state and transient engine operation, the pedal position rate is chosen to represent an aggressive acceleration, as illustrated in Figure 4.13.

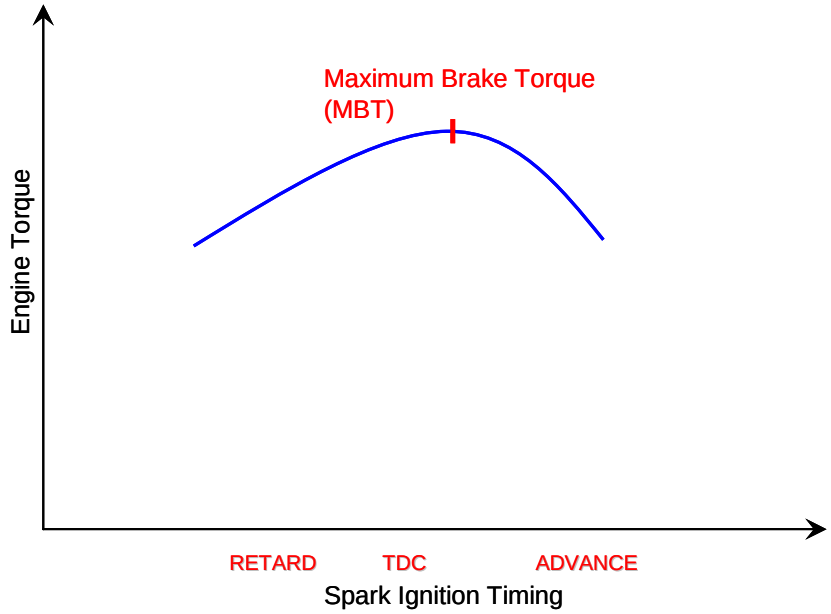

Figure 4.12 − Effect of spark ignition timing on the engine brake torque at constant engine speed.

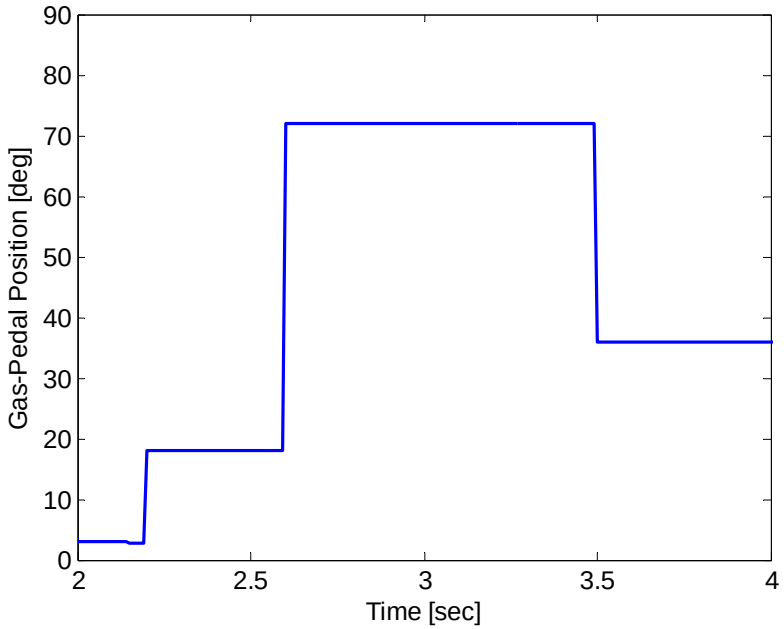

Figure 4.13 − Gas-pedal position rate representing a driver's driving style.

4.5.1 Simulation Results

After completing the learning process, the self-learning controller specified the optimal policy in terms of the spark ignition timing, as shown in Figure 4.14, and compared with the spark ignition timing designated by the baseline calibration of the enDYNA model. The optimal policy resulted in higher engine brake torque compared to the baseline calibration as shown in Figure 4.15 and Figure 4.16. This improvement indicates that the engine with self-learning calibration was able to operate closer to MBT timing. Having the engine operate at MBT timing resulted in an overall minimization of the BSFC, illustrated in Figure 4.17. Figure 4.18 compares the velocity of the two vehicles, one carrying the engine with the baseline calibration and the other with the self-calibrated one.

The two vehicles were simulated for the same driving style, namely, the same pedal-position rate. The vehicle carrying the engine with the self-learning calibration demonstrated higher velocity, since the engine produced higher brake torque for the same

gas-pedal position rate. Consequently, if the driver wishes to follow a specific vehicle's speed profile, this can now be achieved by stepping on the gas-pedal more lightly than required in the engine with the baseline calibration and, therefore, directly enabling in additional benefits in fuel economy.

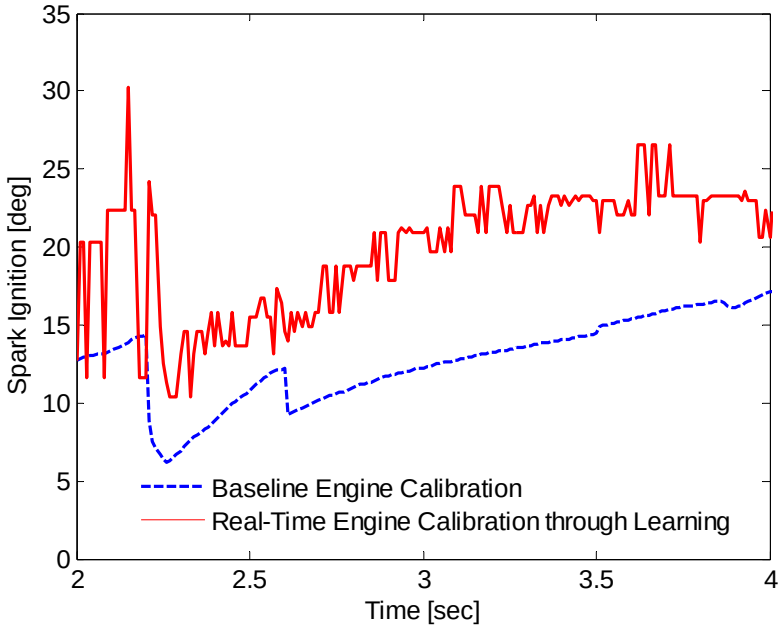

Figure 4.14 − Spark ignition timing over the driving style.

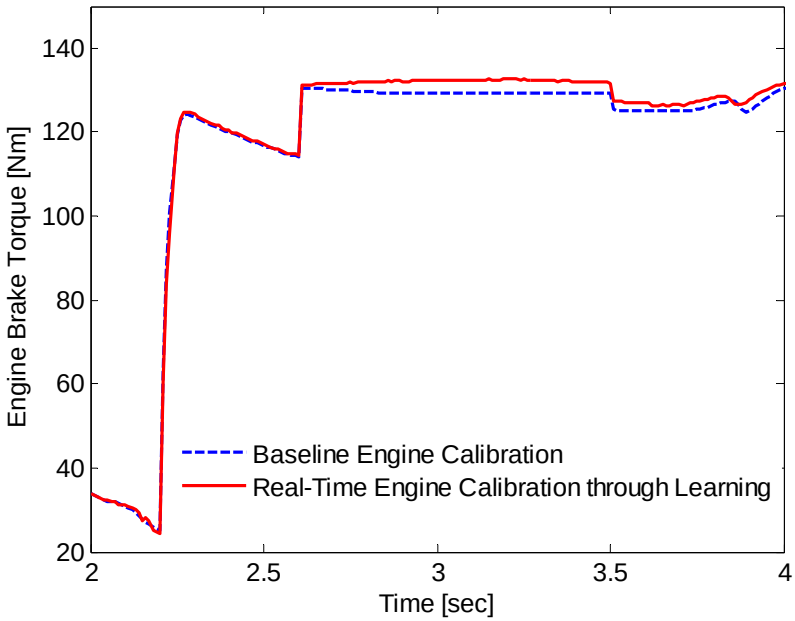

Figure 4.15 − Engine brake torque.

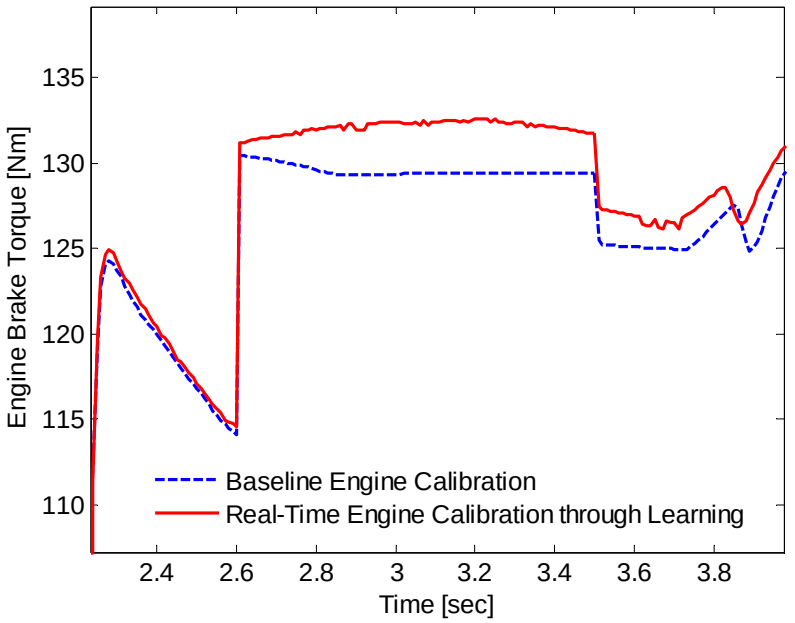

Figure 4.16 − Engine brake torque (zoom-in).

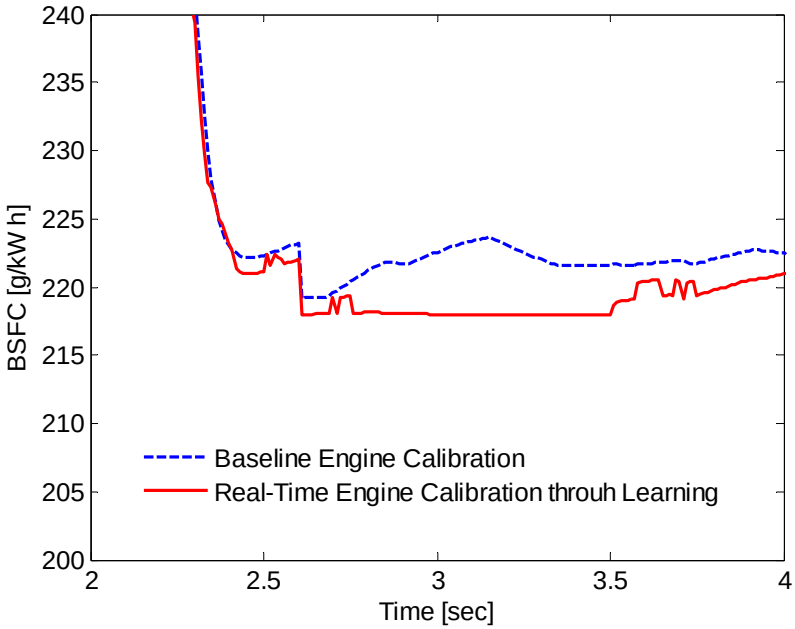

Figure 4.17 − BSFC comparison between the baseline and self-learning calibration.

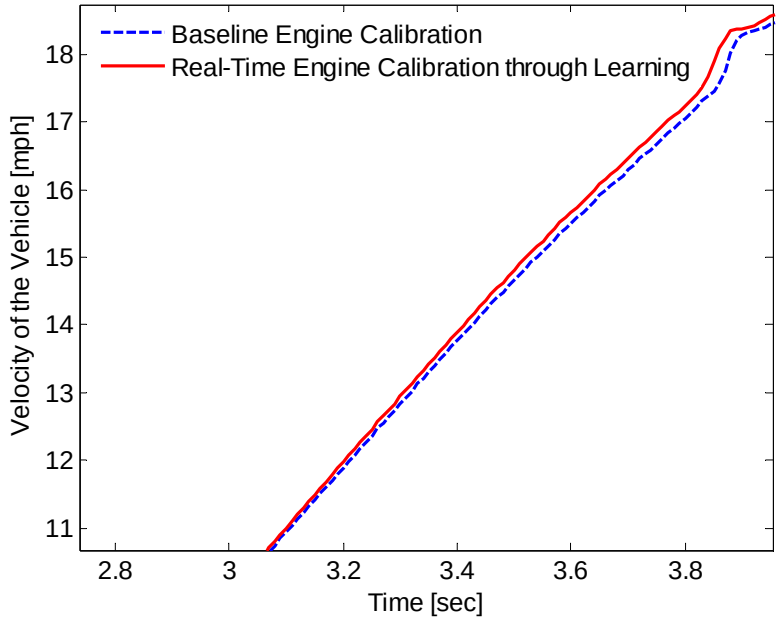

Figure 4.18 − Velocity of the two vehicles carrying the engine with baseline and self-learning calibration.

To evaluate the efficiency of the algorithm in learning, the vehicles were simulated for three additional acceleration profiles, shown in Figure 4.19. The algorithm specified successfully the optimal policy in terms of the spark ignition timing minimizing the BSFC compared to the baseline calibration, as illustrated in Figures 4.20 - 4.22.

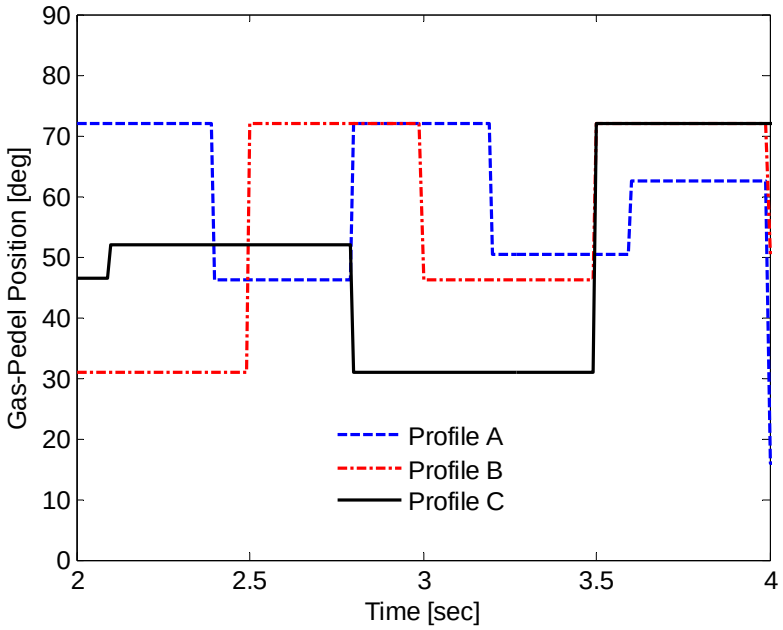

Figure 4.19 − Three different acceleration profiles.

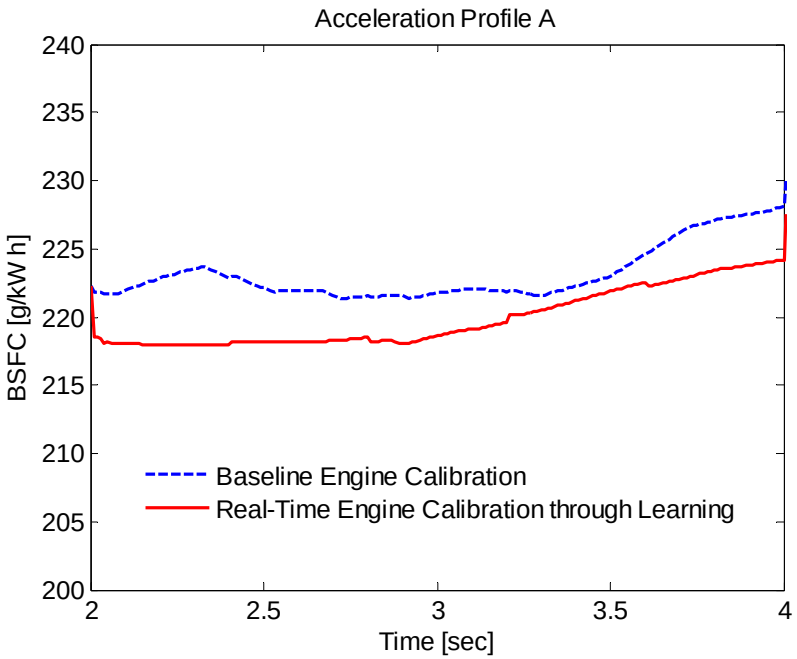

Figure 4.20 − BSFC comparison between the baseline and self-learning calibration (Acceleration profile A).

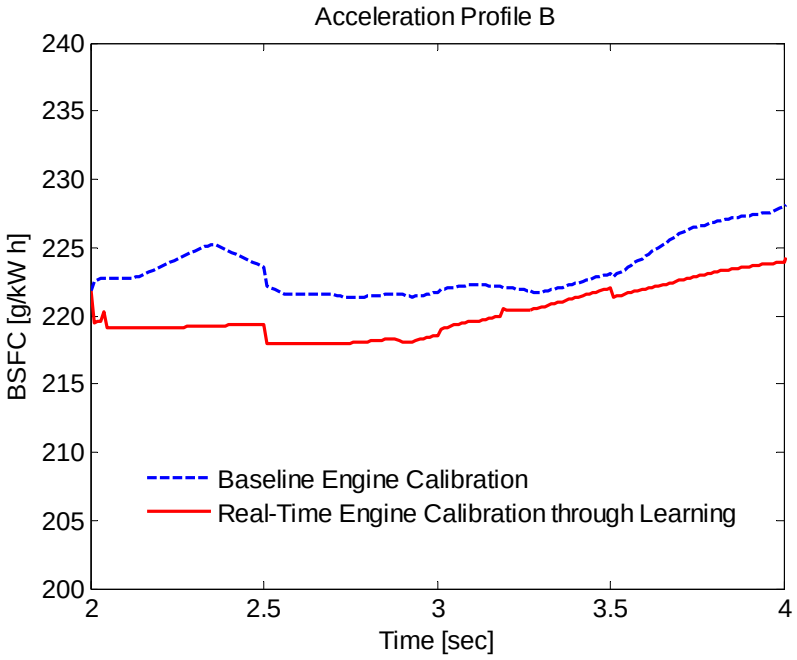

Figure 4.21 − BSFC comparison between the baseline and self-learning calibration (Acceleration profile B).

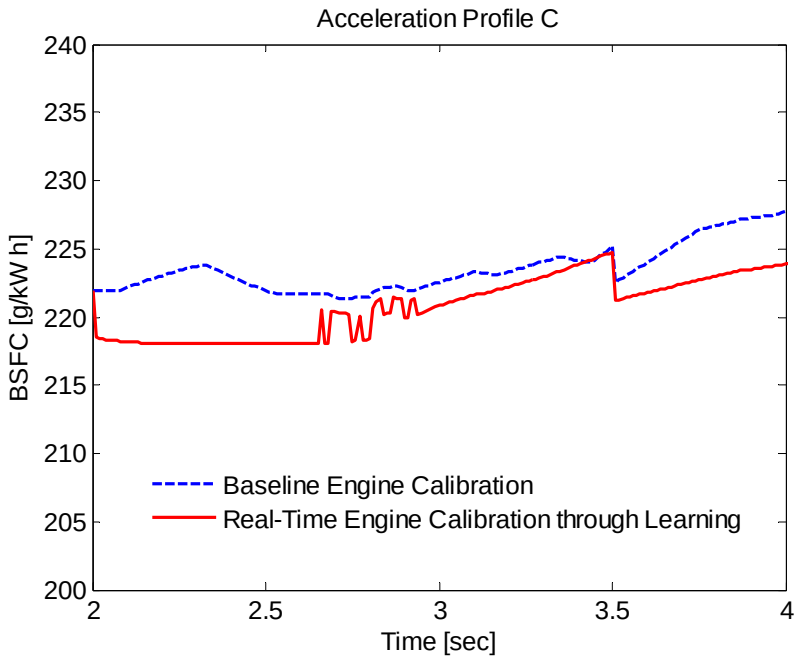

Figure 4.22 − BSFC comparison between the baseline and self-learning calibration (Acceleration profile C).

4.6 Application: Self-Learning Injection Timing in a Diesel engine

The objective of this study is to evaluate the efficiency of the self-learning controller in deriving the optimal control policy (injection timing) during transient engine operation. A desired speed profile, shown in Figure 4.23, including an acceleration and deceleration segment designated by a hypothetical driver, was selected. The model with the baseline ECU and the model with the self-learning controller are run repeatedly over the same profile. The first model incorporates a static calibration map for injection timing corresponding to steady-state operating points. Before initiating the first simulation, the model with the self-learning controller has no knowledge regarding the particular transient engine operation and injection timing associated with it.

The control actions, a_k, correspond to the injection timing that can take values from the feasible set $\mathcal{A}$, defined as

$$\mathcal{A} = A(s_k = i) = [-2°, 18°], \tag{4.45}$$

where $i = 1, 2, ..., N,\ N = |\mathcal{S}|$.

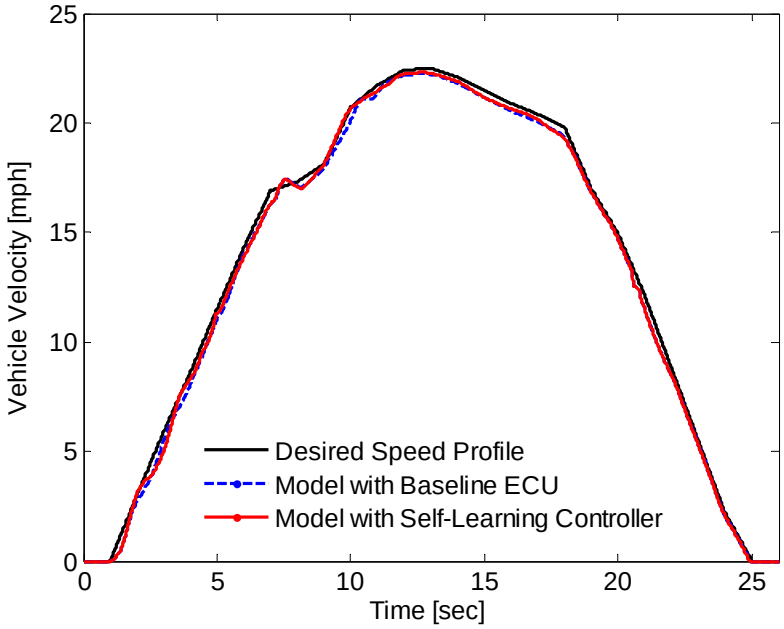

Figure 4.23 − Desired speed profile.

4.6.1 Simulation Results

After completing the exploration phase, the self-learning controller derived the values of injection timing, shown in Figure 4.24. The significant variation of injection timing is attributed to the engine behavior during the transient period before steady-state operation occurs. During this period, the maximum brake torque (MBT), and thus, brake-specific fuel consumption and emissions, are varied for the same engine operating point [12]. These values, corresponding to a particular operating point, highly depend on the previous operating points from which they have been arrived. Consequently, start of injection at steady-state operating points is not optimal for the same points when transiting one from another.

The injection timing computed by the self-learning controller maximized engine torque during transient operation, and the desired speed profile was achieved requiring

lower pedal position rates for the same engine speed as illustrated in Figure 4.25 and Figure 4.26. The implication is that injection timing altered the brake mean effective pressure (BMEP) for this range of engine speed, and engine operation was modified as shown in Figure 4.27. Lower pedal position rates required less injected fuel mass into the cylinders since injection duration was reduced (Figure 4.28), resulting in minimization of fuel consumption as illustrated in Figure 4.29.

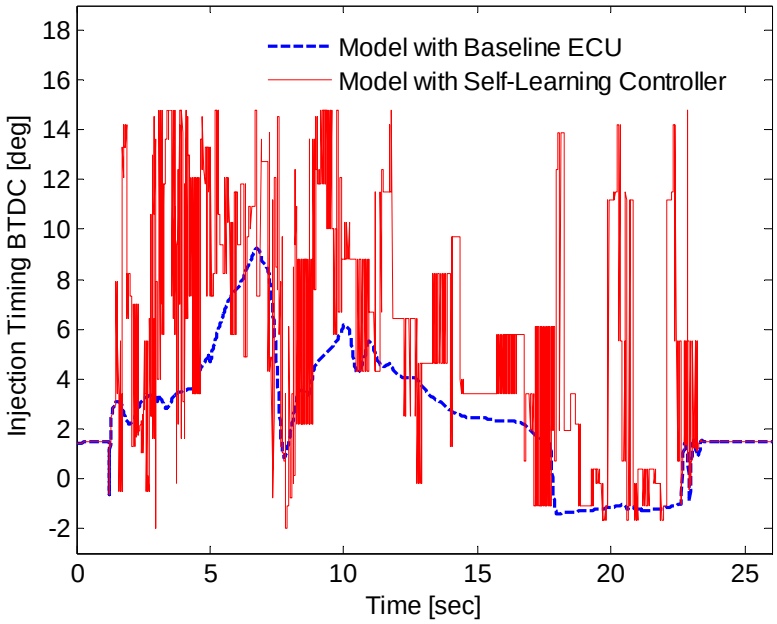

Figure 4.24 − Injection timing.

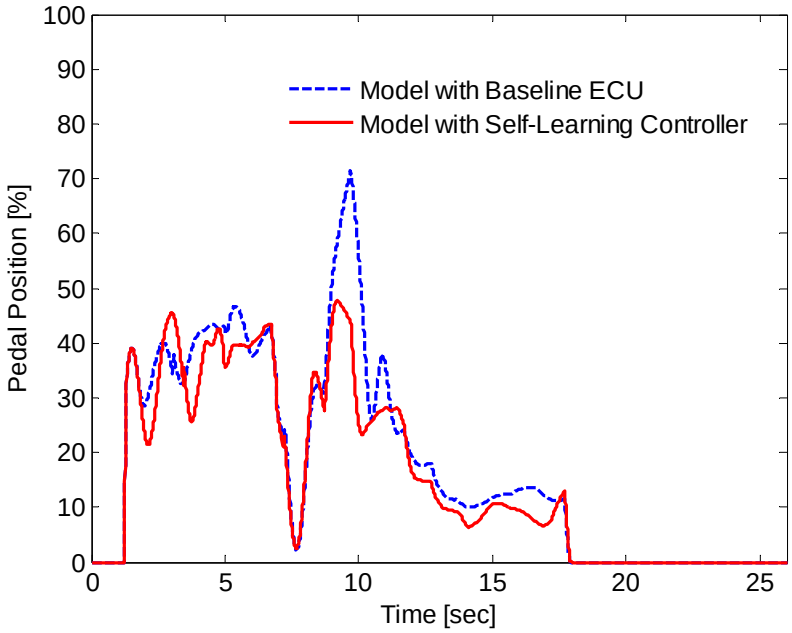

Figure 4.25 − Pedal position rate.

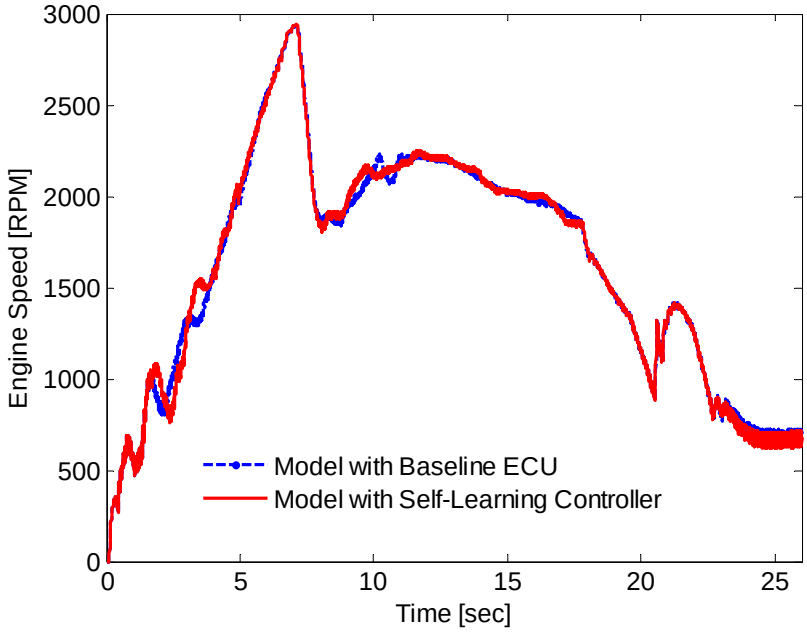

Figure 4.26 − Engine speed.

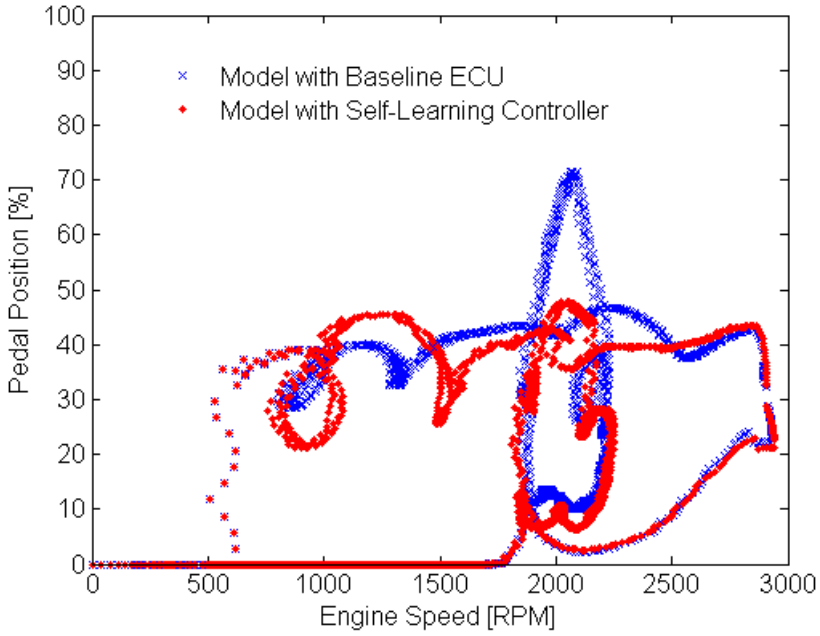

Figure 4.27 − Engine operating point transitions.

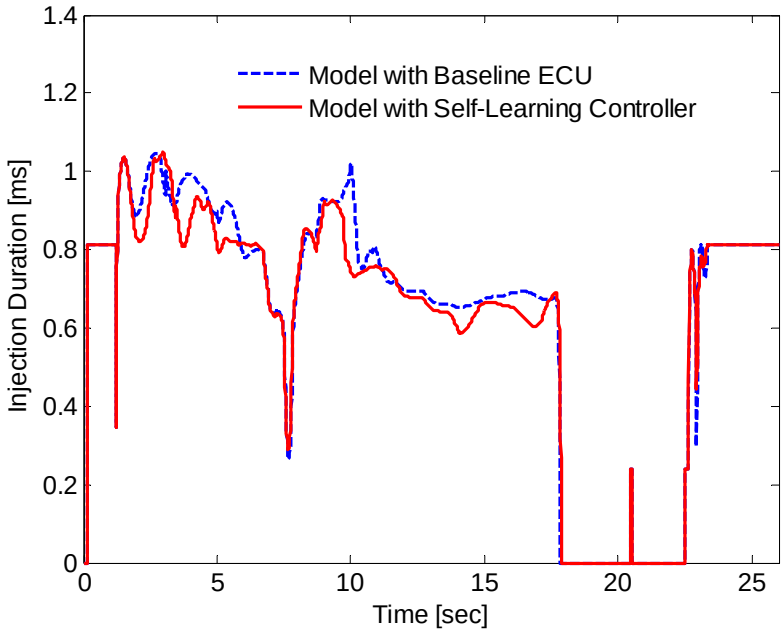

Figure 4.28 − Injection duration.

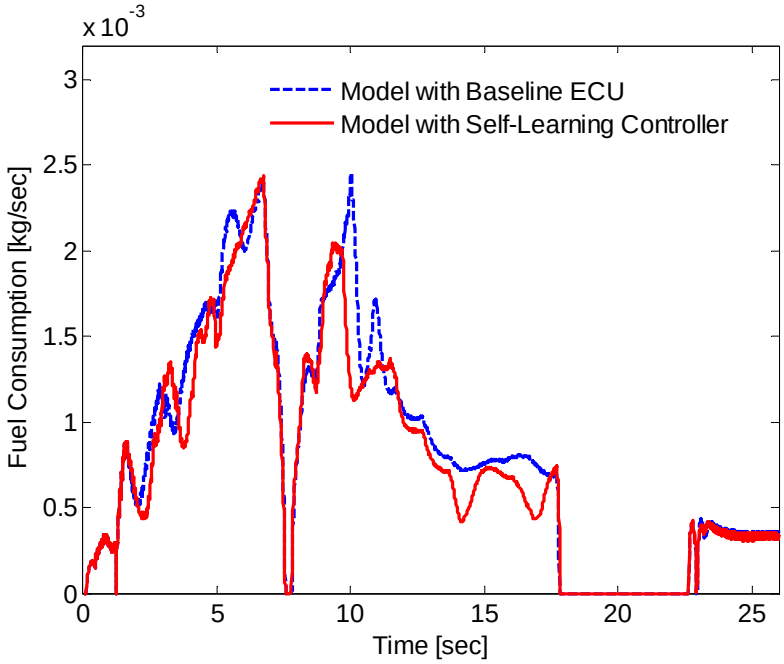

Figure 4.29 − Fuel consumption.

While the fuel mass injected into the cylinders is reduced, the mass air flow was kept almost constant (Figure 4.30) providing excess of air. These conditions degraded the formation of HC, CO and PM as illustrated in Figures 4.31 - 4.33. The injection timing of the baseline ECU provided higher emission temperatures in the exhaust manifold, shown in Figure 4.34, and consequently, NOx formation was progressed much faster as depicted in Figure 4.35.

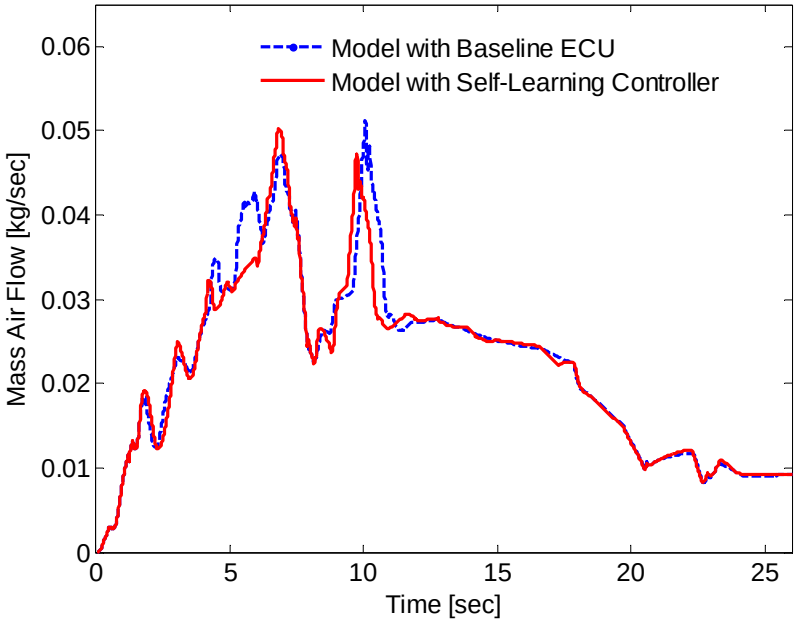

Figure 4.30 − Mass air flow into the cylinders.

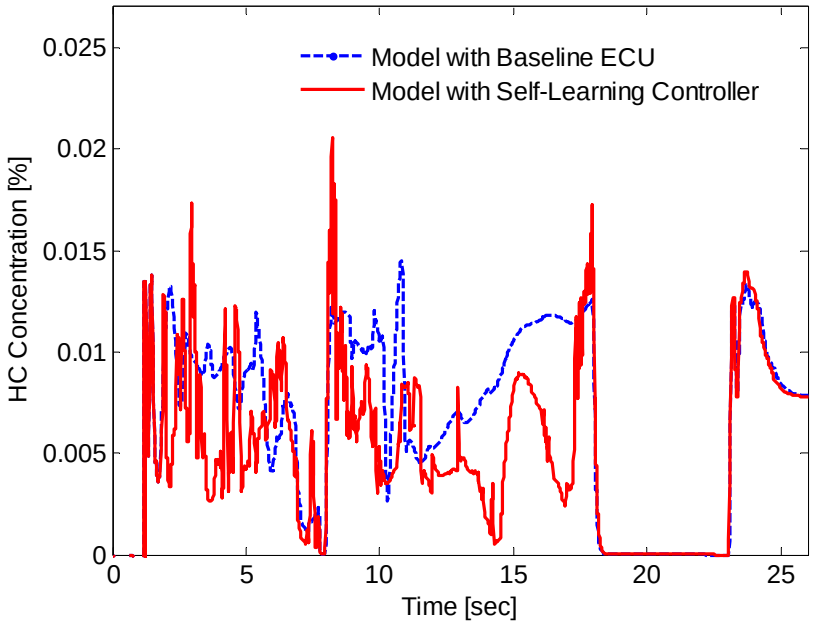

Figure 4.31 − HC concentration of emissions.

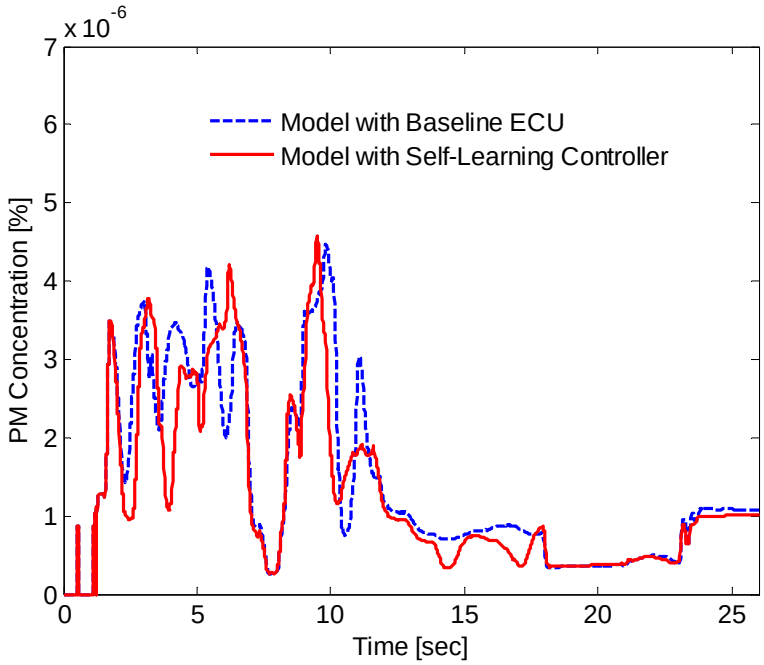

Figure 4.32 − PM Concentration.

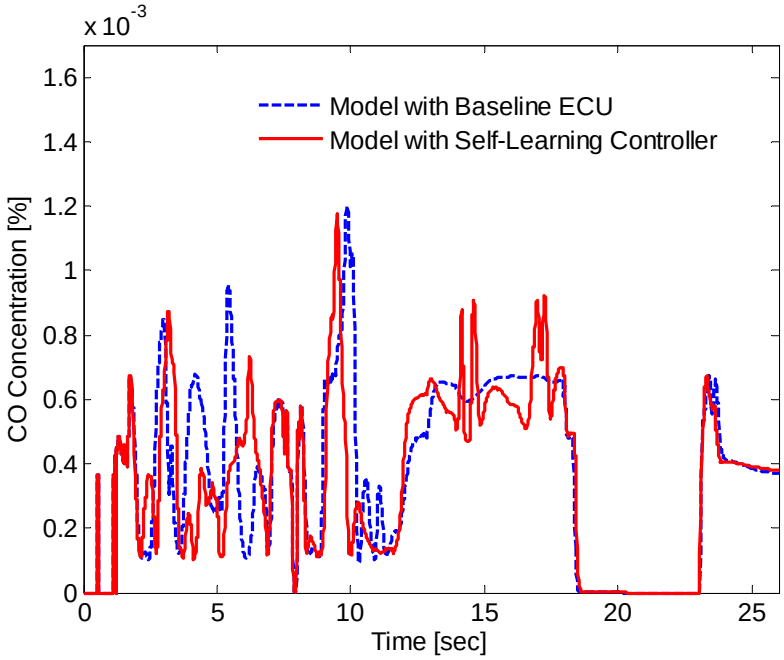

Figure 4.33 − CO concentration of emissions.

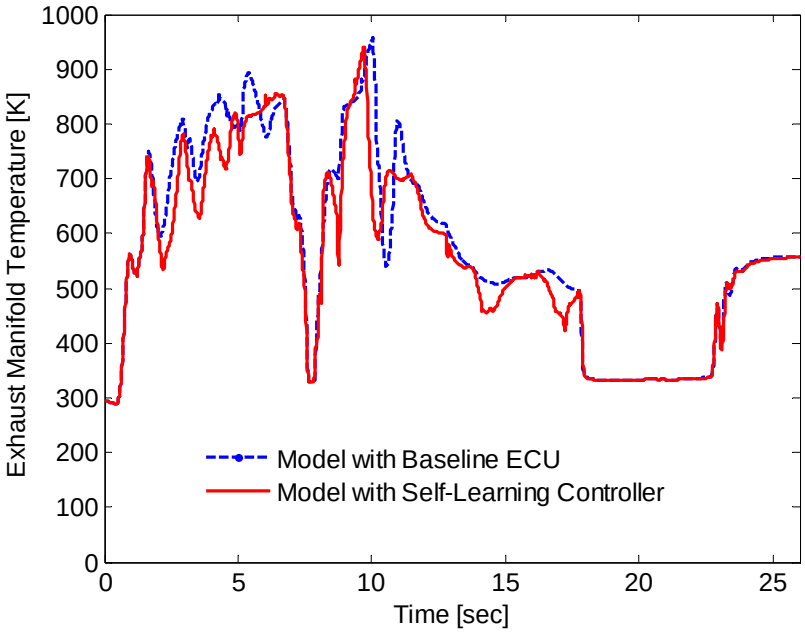

Figure 4.34 − Exhaust manifold temperature.

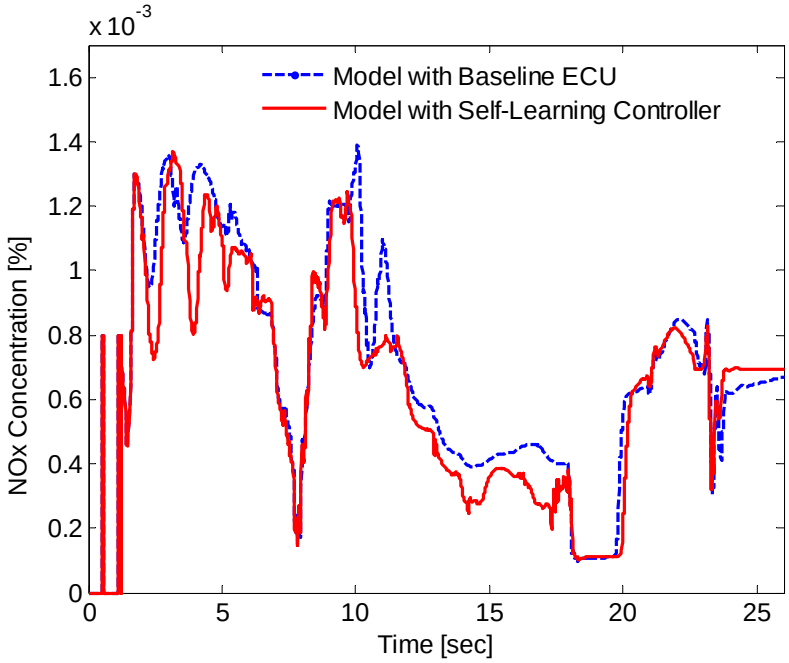

Figure 4.35 − NOx concentration of emissions.

Table 1 summarizes the quantitative assessment of the improvement of fuel consumption, and emissions, by employing the self-learning controller in ECU development.

Table 1: Quantification assessment of benefits in fuel consumption and emissions compared to baseline ECU.

Engine Performance Indices	Improvement [%]
Fuel consumption	8.4
NOx	7.7
HC	32.9
CO	5.0
PM	9.8

4.7 Concluding Remarks

This chapter has presented the algorithmic implementation that provides the decision-making mechanism suitable for real-time implementation. The algorithm solves the stochastic control sub-problem by utilizing accumulated data acquired over the learning process of the POD model. The solution of the algorithm exhibits performance bound that is superior compared to the solution provided by the minimax control policy of the dynamic programming algorithm (Theorem 4.1).

The overall performance of the POD model and POSCA in deriving an optimal control policy was evaluated through application to several examples. In the cart-pole balancing problem, POD and POSCA realized the balancing control policy for an inverted pendulum when the pendulum was released from any angle between 3° and -3°. In implementing the real-time cruise controller, POD and POSCA maintained the desired

vehicle's speed at any road grade between 0° and 10°. The engine calibration problem demonstrated that the POD model and POSCA can make the engine of a vehicle an autonomous intelligent system. The engine can learn by means of a self-learning controller the optimal values of the controllable variables in real time while the driver drives the vehicle. The longer the engine runs during a particular driving style, the better the engine's specified performance indices will be. This property arises due to the learning process required by the state representation to capture the stationary distribution of the engine operation with respect to the driver's driving style. The engine's ability to learn its optimum calibration is not limited, however, to a particular driving style. The engine can learn to operate optimally for different drivers by assigning the transition probability $\mathbf{P}(\cdot,\cdot)$, and cost matrices $\mathbf{R}(\cdot,\cdot)$ for each driver. The drivers should indicate their identities before starting the vehicle to denote the pair of these matrices that the engine should employ. The engine can then adjust its operation to be optimal for a particular driver based on what it has learned in the past regarding his/her driving style.

It is left for future research to explore the impact of traffic patterns, and terrain, on the general applicability of having the engine learn its optimal calibration for an individual driving style. Future research should also investigate the potential of advancing the POD model to accommodate more than one decision maker in sequential decision-making problems under uncertainty, known as multi-agent systems [13]. These problems are found in systems in which many intelligent decision makers (agents) interact with each other. The agents are considered to be autonomous entities. Their interactions can be either cooperative or selfish, i.e., the agents can share a common goal, e.g., control of vehicles operating in platoons to improve throughput on congested highways by allowing groups of vehicles to travel together in a tightly spaced platoon at high speeds. Alternatively, the agents can pursue their own interests.

4.8 References

[1] Bertsekas, D. P., Dynamic Programming and Optimal Control (Volumes 1 and 2), Athena Scientific, September 2001.

[2] Anderson, C. W., "Learning to Control an Inverted Pendulum Using Neural Networks," IEEE Control Systems Magazine, vol. 9, pp. 31-7, 1989.

[3] Williams, V. and Matsuoka, K., "Learning to Balance the Inverted Pendulum Using Neural Networks," Proceedings of the 1991 IEEE International Joint Conference on Neural Networks (Cat. No.91CH3065-0), pp. 214-19, Singapore, 1991.

[4] Zhidong, D., Zaixing, Z., and Peifa, J., "A Neural-Fuzzy BOXES Control System with Reinforcement Learning and its Applications to Inverted Pendulum," Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century (Cat. No.95CH3576-7), pp. 1250-4, Vancouver, BC, Canada, 1995.

[5] Jeen-Shing, W. and McLaren, R., "A Modified Defuzzifier for Control of the Inverted Pendulum Using Learning," Proceedings of the 1997 Annual Meeting of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.97TH8297), pp. 118-23, Syracuse, NY, USA, 1997.

[6] Mustapha, S. M. and Lachiver, G., "A Modified Actor-Critic Reinforcement Learning Algorithm," Proceedings of the 2000 Canadian Conference on Electrical and Computer Engineering, pp. 605-9, Halifax, NS, Canada, 2000.

[7] Khamsi, M. A. and Kirk, W. A., An Introduction to Metric Spaces and Fixed Point Theory, 1st edition, Wiley-Interscience, March 6, 2001.

[8] Zhang, B. S., Leigh, I., and Leigh, J. R., "Learning Control Based on Pattern Recognition Applied to Vehicle Cruise Control Systems," Proceedings of the the American Control Conference, pp. 3101-3105, Seattle, WA, USA, 1995.

[9] Shahdi, S. A. and Shouraki, S. B., "Use of Active Learning Method to Develop an Intelligent Stop and Go Cruise Control," Proceedings of the the IASTED International Conference on Intelligent Systems and Control, pp. 87-90, Salzburg, Austria, 2003.

[10] TESIS, <http://www.tesis.de/en/>.

[11] Heywood, J., Internal Combustion Engine Fundamentals, 1 edition, McGraw-Hill Science/Engineering/Math, April 1988.

[12] Malikopoulos, A. A., Papalambros, P. Y., and Assanis, D. N., "A Learning Algorithm for Optimal Internal Combustion Engine Calibration in Real Time," Proceedings of the ASME 2007 International Design Engineering Technical

Conferences Computers and Information in Engineering Conference, Las Vegas, Nevada, September 4-7, 2007.

[13] Panait, L. and Luke, S., "Cooperative Multi-Agent Learning: The State of the Art," Autonomous Agents and Multi-Agent Systems, vol. 11, pp. 387-434, 2005.

CHAPTER 5

DECENTRALIZED LEARNING

This chapter proposes a decentralized learning control scheme in finite Markov chains that aims to address the problem of dimensionality, when more than one decision makers are engaged. This scheme draws from multi-agent learning research in a range of areas including reinforcement learning, and game theory to coordinate optimal behavior among the decision makers. The solution of the decentralized scheme attempts to provide a Nash equilibrium coordinated control policy.

In applying this scheme to the engine calibration problem, the engine is treated as a cooperative multi-agent system, in which the subsystems, i.e., controllable variables, are considered autonomous intelligent agents who strive interactively and jointly to optimize engine performance criteria.

5.1 Decentralized Learning in Finite Markov Chains

Decentralized decision making requiring limited information is a highly desirable feature of large complex systems. It is necessary when complete information among decision makers, which is required in centralized decision making, is impractical due to the increase of the problem's dimensionality. Moreover, decentralized decision making can often be useful in complex systems with uncertainties regarding their behavior and the nature of external events [1]. Even in the absence of such uncertainties the coordination of decentralized decision makers is still a formidable problem; local

optimality and global optimality are often inconsistent. Uncertainty adds to the difficulty of an identification problem, the feature that motivates the use of a learning approach. Mathematical learning theory has been developed in systems to address the modeling and control aspects of sequential decision making under uncertainty [2-4]. Learning automata have been applied to network routing in which decentralization is attractive and large uncertainties are present [5, 6]. The resulting system performance has demonstrated that decentralized learning schemes can be successful while the problem's dimensionality remains tractable.

The problem of decentralized iterative control for a class of large scale interconnected dynamic systems in continuous time domain was studied by Wu [7]. In this off-line approach, it is assumed that the considered systems are linear time varying, and the interconnections between each subsystem are unknown. Szer *et al.* [8] proposed a model-free distributed reinforcement learning algorithm that utilizes communication to improve learning among the decision makers in a Markov decision process formalism. Scherre *et al.* [9] developed a general iterative heuristic approach in which at each decision epoch the focus is on a sub-group of decision makers and their policies given the rest of the decision makers have fixed plans. Beynier *et al.* [10] introduced the notion of expected opportunity cost to better assess the influence of a local decision of an agent on the others. An iterative version of the algorithm was implemented to incrementally improve the policies of agents leading to higher quality solutions in some settings. Yagan *et al.* [11] implemented a model-free coordinated reinforcement learning for decentralized optimal control assuming that each decision maker can partially observe the state condition. This decentralized scheme is suited for partially observable Markov decision processes. Shen *et al.* [12] developed a decentralized Markov game model to estimate the belief among the decision makers. In the proposed model, the model-free Q-learning algorithm was employed to adjust dynamically the payoff function of each player.

Although many of these algorithms addressed the decentralized learning problem their use of the accumulated data acquired over the learning process is inefficient, and they require a significant amount of experience to achieve acceptable performance. This requirement arises due to the formation of these algorithms in deriving optimal policies without learning the system models *en route*; that is, they do not solve the state estimation and system identification problem simultaneously.

The study of interacting decision makers inevitably entails game theory [13-16]. The use of learning schemes by players does not circumvent the basic complexities of *N*-player games. In general, rational behavior is not well defined even when the payoff structure is known to all players. Wheeler *et al*. [1] employed a game-theoretic approach and developed a decentralized learning control scheme in finite Markov chains with unknown transition probabilities and costs. In this scheme, the decision makers demonstrate a myopic behavior, namely, they are unaware of the surrounding world. In attempting to improve his/her performance, each decision maker selects a control action, observes the corresponding cost associated with the occupied state, and then updates the action.

The decentralized learning control scheme proposed in this dissertation differs from Wheeler's scheme: Here the decision makers do not demonstrate myopic behavior explicitly. On the contrary, a random hierarchy among them is assumed, based on which each one observes the control actions of the other. In particular, POSCA is employed to derive the control actions of the first member in the hierarchy of decision makers with respect to the sequence of state transitions. At the same time, the algorithm is engaged separately to derive the control actions of the second member in the hierarchy of decision makers with respect to the optimal policy as being learned from the first one. Similarly, the algorithm is employed to derive the control actions of the third decision maker with respect to the second one and so forth.

This decentralized learning scheme entails a game-theoretic approach. In particular, the interaction among the controllers is modeled as an identical payoff game. The game involves $\Sigma \in \mathbb{N}$ players (controllers) interacting through a stationary random environment. At each decision epoch k, the environment presents a state $s_k = i \in \mathcal{S}$ to the players, and on that basis each player selects a strategy (control action) from his/her feasible set of strategies $A^r(s_k), \{r = 1, ..., \Sigma\}$. The players seek a Nash equilibrium strategy that exists under certain conditions.

5.2 Game Theory

Game theory is defined as the study of mathematical models of conflict and cooperation between intelligent rational decision makers. Game theory provides the mathematical framework for analyzing situations in which two or more individuals make decisions that will influence one another's welfare. A *game* refers to any situation involving two or more decision makers who are called *players*.

There is a main assumption that game theorists generally make about the players: they are rational and intelligent. A decision maker is rational if he makes decisions consistently in pursuit of his own objectives. In game theory, building on the fundamental results of decision theory, it is assumed that each player's objective is to maximize the expected value of his/her own payoff, which is measured in some utility scale.

Formally, a game Γ in the strategic form is represented by

$$\Gamma := \left(\mathbf{\Sigma}, (A^r)_{r\in\Sigma}, (R^r)_{r\in\Sigma}\right), \tag{5.1}$$

where $\mathbf{\Sigma}$ is the set of players, A^r is the set of feasible strategies for each player r, and R^r is the payoff function of each player that implements a mapping from the Cartesian product of the state space $\mathcal{S}$ (stationary environment) and the set of feasible strategies of each player A^r to the set of real numbers, $R^r : \mathcal{S} \times A^1 \times ... \times A^\Sigma \to \mathbb{R}$.

5.3 The Decentralized Learning Control Scheme in POD Domain

In the proposed decentralized learning scheme, the interaction among the controllers is modeled as an identical payoff game. First, a random hierarchy among the controllers is assumed, based on which each one observes the control actions (strategies) of the other. More precisely, the first member in the hierarchy of the decision makers observes the sequence of state transitions. At the same time, the second member in the hierarchy of the decision makers observes the control actions as being learned from the first one and so forth. The game between the first controller and the environment is defined

$$\Gamma_1 = \left(\left(\mathcal{S}, a^1 \right), A^1(i), R^r(i, a^1, a^2, ..., a^\Sigma) \right), \forall i \in \mathcal{S}, \tag{5.2}$$

where $\mathcal{S}$ is the state space, a^1 is the action of the first controller, $A^1(i)$ is its the feasible set of strategies (control actions), and $R^r(i, a^1, a^2, ..., a^\Sigma)$ is the common payoff that all controllers receive when they select control actions at a particular state $i \in \mathcal{S}$. The second game between the second controller in the hierarchy and the first one is defined as

$$\Gamma_2 = \left(\left(a^1, a^2 \right), A^2(a^1), R^r(i, a^1, a^2, ..., a^\Sigma) \right), \forall i \in \mathcal{S}, \tag{5.3}$$

where a^2 denotes the action of the second controller, and $A^2(a^1)$ is its feasible set of control actions with respect to the control actions of the first controller a^1. Similarly, the game between the last two in the hierarchy of controllers is defined as

$$\Gamma_\Sigma = \left(\left(a^{\Sigma-1}, a^\Sigma \right), A^\Sigma(a^{\Sigma-1}), R^r(i, a^1, a^2, ..., a^\Sigma) \right), \forall i \in \mathcal{S}. \tag{5.4}$$

These games are played simultaneously at each decision epoch k, while the chain $\{s_k, k \geq 0\}$ visits a state $s_k = i \in \mathcal{S}$. The POD state representation is employed to provide the realization of the state transitions that occurred in the Markov domain. The different

sequences of the state transitions are captured by the POD states and evaluated through the expected evaluation functions that correspond to the common payoff $R^r(i,a^1,a^2,...,a^\Sigma)$. POSCA selects the lookahead policy that determines the control actions at each decision epoch.

Definition 5. 1 ("Nash Equilibrium") [14]. In a strategic game, an action a^* is called Nash equilibrium if no player r has an action yielding an outcome that he/she prefers to that generated when he/she chooses a^{*r}, given that every other player $l \in \Sigma$ chooses his/her equilibrium actions a^{*l}. Briefly, no player can profitably deviate, given the actions of the other players.

The decentralized learning control scheme seeks to reach a Nash equilibrium strategy $(a^{*1},a^{*2},...,a^{*\Sigma})$ for all players that is guaranteed the maximum common payoff $R^r(i,a^1,a^2,...,a^\Sigma)$.

Definition 5. 2. For each controller r, the function F^r is defined as

$$F^r = \max\left\{0, R^r(i,a^1,a^2,...,a^r,...,a^\Sigma) - R^{*r}(i,a^{*1},a^{*2},...,a^{*r},...,a^{*\Sigma})\right\}, \forall i \in \mathcal{S}, \tag{5.5}$$

where $a^r \in A(a^{r-1})$ is the action of the controller r from the feasible set of actions, and R^{*r} is the maximum common payoff at the current decision epoch when all controllers believe that their Nash equilibrium control actions is $(a^{*1},a^{*2},...,a^{*r},...,a^{*\Sigma})$.

Definition 5. 3. For each controller r, the function R_r' is defined as the mapping $R_r' : \mathbb{R} \to \mathbb{R}$, namely,

$$R_r' = \frac{R^{*r}(i,a^{*1},a^{*2},...,a^{*r},...,a^{*\Sigma}) + F^r \cdot R^r(i,a^1,a^2,...,a^r,...,a^\Sigma)}{1+F^r}, \forall i \in \mathcal{S}. \tag{5.6}$$

The decentralized learning scheme seeks the control actions that provide the fixed point of the mapping $R_r' : \mathbb{R} \to \mathbb{R}$.

Suppose that during the learning process the belief of the controllers for the Nash equilibrium control actions is $(a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$, which results in a common payoff equal to $R^{*r}(i, a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$. Suppose that at the next decision epoch, they switch to any $\Sigma-$tuple $(a^1, a^2, ..., a^r, ..., a^\Sigma)$ with a payoff $R^r(i, a^1, a^2, ..., a^r, ..., a^\Sigma)$. If this payoff is less than $R^{*r}(i, a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$, then the function F^r in Eq. (5.5) becomes zero. This means that no controller can improve the payoff by changing its control action, and thus the Nash equilibrium control actions $(a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$ vanishes all functions F^r for each controller, and makes $(a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$ fixed under the mapping $R'_r : \mathbb{R} \to \mathbb{R}$. On the other hand, if the $\Sigma-$tuple $(a^1, a^2, ..., a^r, ..., a^\Sigma)$ yields a payoff higher than $R^{*r}(i, a^{*1}, a^{*2}, ..., a^{*r}, ..., a^{*\Sigma})$, then the controller change their belief and make this tuple to be the Nash equilibrium control actions for that decision epoch and on.

5.3.1 Existence of a Nash Equilibrium

The conditions under which a Nash equilibrium of a game exists have been extensively reported in the literature. An existence result has two purposes. First, if a game satisfies the hypothesis of the result, then it is known that our effort to find an equilibrium will meet with success. Second, the existence of an equilibrium shows that the game is consistent with a steady state solution. Nash [17] proved that every finite game that has at least on equilibrium strategy.

Kakutani's Fixed Point Theorem gives conditions on the mapping $R'_r : \mathbb{R} \to \mathbb{R}$ under which there indeed exists a Nash equilibrium.

Theorem 5. 1 ("Kakutani's Fixed Point Theorem") [18]. Let X be a compact convex subset of $\mathbb{R}^n$ and let $f : X \to X$ be a set-valued function for which

1. For all $x \in X$ the set $f(x)$ is nonempty and convex,
2. the graph of f is closed.

Then there exists $x^* \in X$ such that $x^* \in f(x^*)$, or $x^* = f(x^*)$.

Proof. The proof is provided by Khamsi and Kirk [18].

□

If the mapping $R'_r : \mathbb{R} \to \mathbb{R}$ in Eq. (5.6) satisfies the conditions imposed in Theorem 5. 1, then it is guaranteed that a Nash equilibrium control action for the controllers exists.

The decentralized learning scheme is illustrated with a simple example including a state space with two states $\mathcal{S} = \{1, 2\}$, and two controllers a^1 and a^2. Each controller has available two control actions, that is, $a^1 = \{a_1^1, a_2^1\}$ and $a^2 = \{a_1^2, a_2^2\}$. In the centralized form the game with the corresponding common payoffs is formalized as follows

States / *Actions*	a_1^1, a_1^2	a_1^1, a_2^2	a_2^1, a_1^2	a_2^1, a_2^2
1	10	8	2	40
2	9	7	20	3

. (5.7)

Obviously, at state 1 the optimal control actions are the pair a_2^1, a_2^2, whereas at state 2 the optimal actions are a_2^1, a_1^2 that maximize the common payoffs. In the proposed decentralized learning scheme, two games are formed. The first one is between the environment and controller 1,

States / *Controller* 1	a_1^1	a_2^1
1	?	?
2	?	?

(5.8)

and the second game is between the two controllers.

Controller 1 / *Controller* 2	a_1^2	a_1^2
a_1^1	?	?
a_2^1	?	?

(5.9)

The states appear with an unknown probability distribution that is captured by the POD state representation. The controllers should explore their action space to learn the corresponding payoffs, and eventually, find the control actions resulting in the maximum payoffs.

The decentralized scheme is applied for one thousand decision epochs. At state 1, the controllers converge to their Nash equilibrium control actions after 356 decision epochs, illustrated in Figure 5.1. At state 2, the controllers converge to their Nash equilibrium after 227 decision epochs as depicted in Figure 5.2.

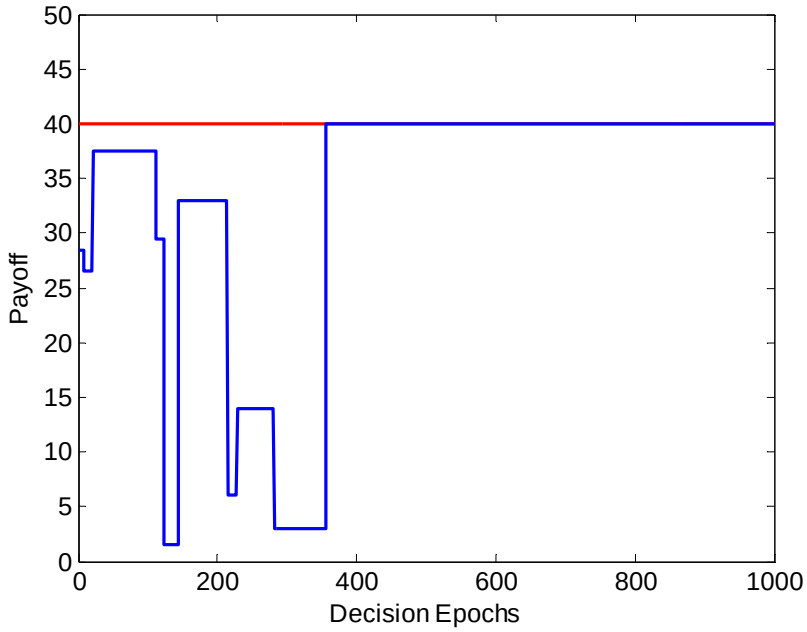

Figure 5.1 − Common payoff at state 1 with respect to decision epochs.

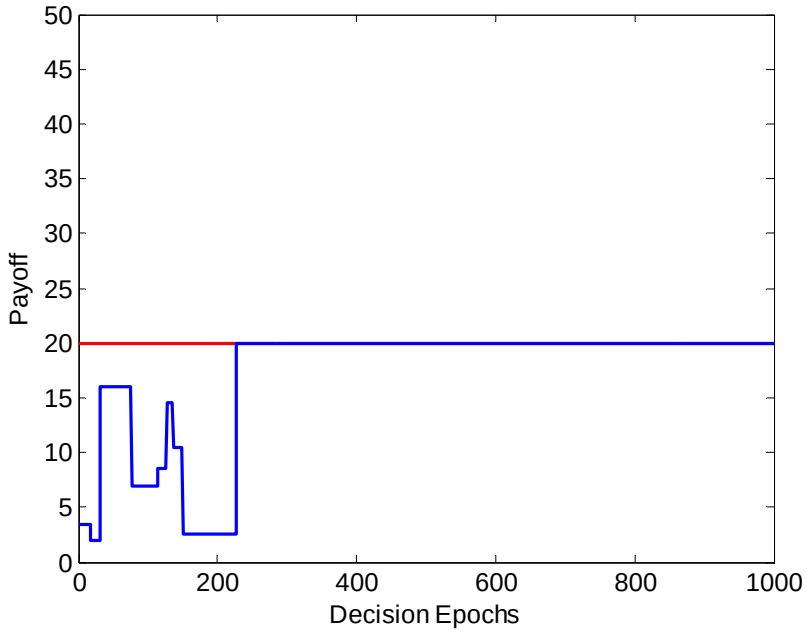

Figure 5.2 − Common payoff at state 2 with respect to decision epochs.

5.4 Decentralized Learning in Engine Calibration

The decentralized learning method [19] establishes a learning process that enables the derivation of the optimal values of the controllable variables to occur in parallel phases. The algorithm is employed to derive the optimal policy of one controllable variable with respect to the sequence of state transitions imposed by the driver's driving style. Concurrently, the algorithm is also engaged separately to derive the optimal policy of the second controllable variable with respect to the optimal policy being learned for the first one. In case of more than two controllable variables, the algorithm is employed in a similar fashion, namely, the third variable with respect to the second one and so forth.

For instance, in implementing a diesel engine calibration with respect to the injection timing, α, and VGT vane position, β, a feasible set of values, $\mathcal{A}$ and $\mathcal{B}$, for each controllable variable is defined. The decentralized learning enables the engine to implement two different mappings in parallel. In the first, injection timing is mapped to

the states as a result of the correspondence of the driver's driving style to particular engine operating points, i.e., $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In the second, VGT is mapped to the injection timing, i.e., $\mathcal{A} \times \mathcal{B} \to \mathbb{R}$. The learning algorithm utilizes these two mappings to derive the optimal policies, $\pi_\alpha^* \in \mathcal{A}$, and $\pi_\beta^* \in \mathcal{B}$ (optimal values of injection timing and VGT) for the driver's driving style as expressed by the incidence in which particular states or particular sequences of states arise.

The decentralized learning process of the engine transpires at each stage k in conjunction with the injection timing $\alpha_k \in \mathcal{A}$ taken for each state $s_k = i \in \mathcal{S}$, and VGT vane position $\beta_k \in \mathcal{B}$ for each $\alpha_k \in \mathcal{A}$. At the early stages, and until full exploration of the feasible sets $\mathcal{A}$ and $\mathcal{B}$, occurs, the mapping from states to probabilities of selecting a particular value of injection timing $\alpha_k \in \mathcal{A}$, and the mapping from $\alpha_k \in \mathcal{A}$ to probabilities of selecting VGT $\beta_k \in \mathcal{B}$ are constant; namely, the values of each controllable variable are selected randomly with the same probability

$$p(\alpha_k \mid s_k = i) = \frac{1}{|\mathcal{A}|}, \forall \alpha_k \in \mathcal{A}, \forall i \in \mathcal{S}, \text{ and} \tag{5.10}$$

$$p(\beta_k \mid \alpha_k) = \frac{1}{|\mathcal{B}|}, \forall a \in \mathcal{A}, \forall \beta \in \mathcal{B}, \tag{5.11}$$

$i = 1, 2, ..., N, N = |\mathcal{S}|$.

Exploration of the entire feasible set for each variable is important to evade sub-optimal solutions. POSCA is thus used after the exploration phase to realize the optimal policies, π_α^*, and π_β^* by means of the expected costs, $V(s_{k+1} \mid s_k, a_k)$ and $V(\alpha_{k+1} \mid \alpha_k, \beta_k)$, generated by the mappings $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and $\mathcal{A} \times \mathcal{B} \to \mathbb{R}$, respectively. The expected costs are defined to be

$$V(s_{k+1}=j \mid s_k=i,a_k)=p(s_{k+1}=j \mid s_k=i,a_k)\cdot R(s_{k+1}=j \mid s_k=i,a_k)+$$

$$+\min_{a_{k+1}\in \mathcal{A}}\left[\sum_{l=1}^{N} p(s_{k+2}=l \mid s_{k+1}=j,a_{k+1})\cdot R(s_{k+1}=l \mid s_k=j,a_{k+1})\right], and \tag{5.12}$$

$$V(\alpha_{k+1}=m \mid \alpha_k=n,\beta_k)=p(\alpha_{k+1}=m \mid \alpha_k=n,\beta_k)\cdot R(\alpha_{k+1}=m \mid \alpha_k=n,\beta_k)+$$

$$+\max_{\beta_{k+1}\in \mathcal{B}}\left[\sum_{p=1}^{\Lambda} p(\alpha_{k+2}=p \mid \alpha_{k+1}=m,\beta_{k+1})\cdot R(\alpha_{k+2}=p \mid \alpha_{k+1}=m,\beta_{k+1})\right], \tag{5.13}$$

$i,j=1,2,...,N,N=|\mathcal{S}|, and$
$m,n=1,2,...,\Lambda,\Lambda=|\mathcal{A}|.$

In deriving the optimal policies of the injection timing and VGT in self-learning calibration, which is treated in a stochastic framework, all uncertain quantities are described by probability distributions. The optimal policies, π_α^*, and π_β^* are based on the minimax control approach, whereby the worst possible values of the uncertain quantities within the given set are assumed to occur. This is a pessimistic point of view that essentially assures the optimal policies will result in at least a minimum overall cost value. Consequently, at state $s_k=i,$ the algorithm predicts the optimal policy π_α^* in terms of the values of injection timing α as

$$\pi_\alpha^*(s_k)=\underset{\bar{\mu}_k(s_k)\in A(s_k)}{\arg\min}\ \max_{s_{k+1}\in \mathcal{S}}\left[V(s_{k+1}=j \mid s_k=i,a_k)\right], \tag{5.14}$$

$\forall i,j\in \mathcal{S}.$

For this optimal policy π_α^* the algorithm predicts the optimal policy π_β^* in terms of the values of the VGT vane position β as

$$\pi_{\beta}^{*}(\alpha_k) = \underset{\beta_k(a_k)\in \mathcal{B}(a_k)}{\arg\min} \max_{a_{k+1}\in \mathcal{A}} \left[V(a_{k+1} = m \mid a_k = n, \beta_k) \right], \tag{5.15}$$

$\forall m, n \in \mathcal{A}$.

Employing decentralized learning, the derivation of the optimal values of more than one controllable variable can be achieved while the problem's dimensionality remains tractable.

5.5 Application: Decentralized Learning in a Diesel Engine

The decentralized learning introduced in the previous section is now applied to a four-cylinder, 1.9-liter turbocharged diesel engine. The objective is to find the optimal injection timing and VGT vane position, while the engine is running the vehicle, that maximize the engine brake torque. Injection timing is an important controllable variable in the combustion process, and affects performance and emissions [20]. The major objective of injection timing is to initiate the start of the fuel injection at the crank angle resulting in the maximum brake torque (MBT). It designates the ignition delay defined to be the crank angle between the start of injection (SOI) and the start of combustion (SOC). The VGT technology was originally considered to increase engine brake torque at tip-ins and reduce turbo-lag. VGT has a system of movable guide vanes located on the turbine stator. By adjusting the guide vanes, the exhaust gas energy to the turbocharger can be regulated, and thus the compressor mass airflow and exhaust manifold pressure can be controlled.

The software package enDYNA Themos CRTD by TESIS [21] suitable for real-time simulation of diesel engines is employed. In the example, the existing static correlation involving injection timing and VGT is bypassed to incorporate the learning method and is used as a baseline comparison. The engine models with the baseline and self-learning calibration are run repeatedly over the same driving style represented by a

segment of the FTP-75 driving cycle, illustrated in Figure 5.3. Every run over this driving style constitutes one complete simulation. Before initiating the first simulation of the engine model, the elements of the transition probability and cost matrix are assigned to be zero. That is, the engine at the beginning has no knowledge regarding the particular driving style and the values of the costs associated with the controllable variables (injection timing and VGT).

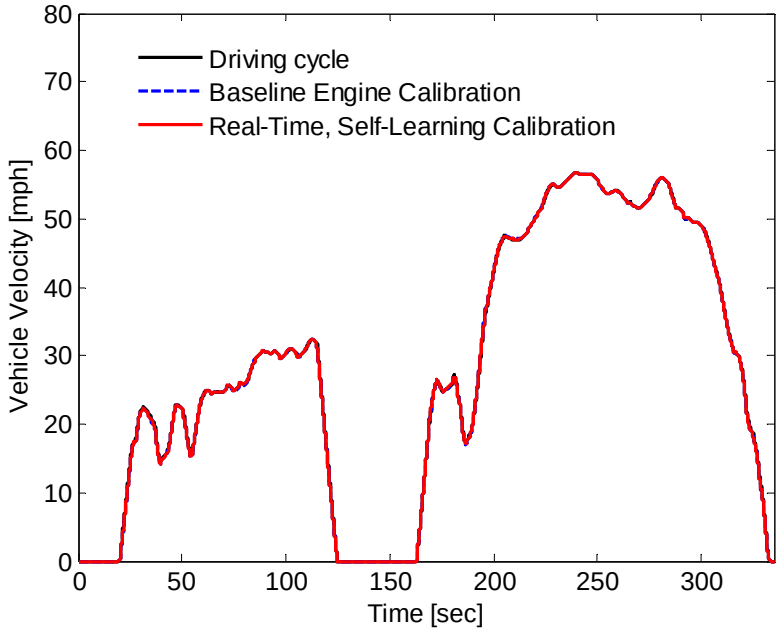

Figure 5.3 − Segment of the FTP-75 driving cycle.

5.5.1 Simulation Results

Applying the decentralized learning method, the vehicle with the self- learning calibration was able to follow the segment of the driving cycle requiring lower gas pedal position rates for the same engine speed, as illustrated in Figures 5.4 – 5.6. The implication is that the derived policy of injection timing and VGT resulted in higher engine torque compared to the baseline calibration. The injection timing (before top dead center BTDC) for both vehicles is illustrated in Figures 5.7 and 5.8. While the baseline

calibration interpolates values of the injection timing of steady-state operating points, the injection timing derived by the learning algorithm corresponded to the engine operating point transitions imposed by the driver's driving style, and thus, self-learning calibration was able to capture transient engine operation. Lower gas pedal position rates resulted in reducing the fuel mass injection duration, shown in Figure 5.9, and consequently, less fuel mass was injected into the cylinders, as illustrated in Figure 5.10 (in zoom-in for clarity). In the decentralized learning of the engine, the injection timing was mapped to the engine operating points (states) while the VGT vane position was mapped to the optimal injection timing. The derived VGT policy is illustrated in Figure 5.11 – 5.12.

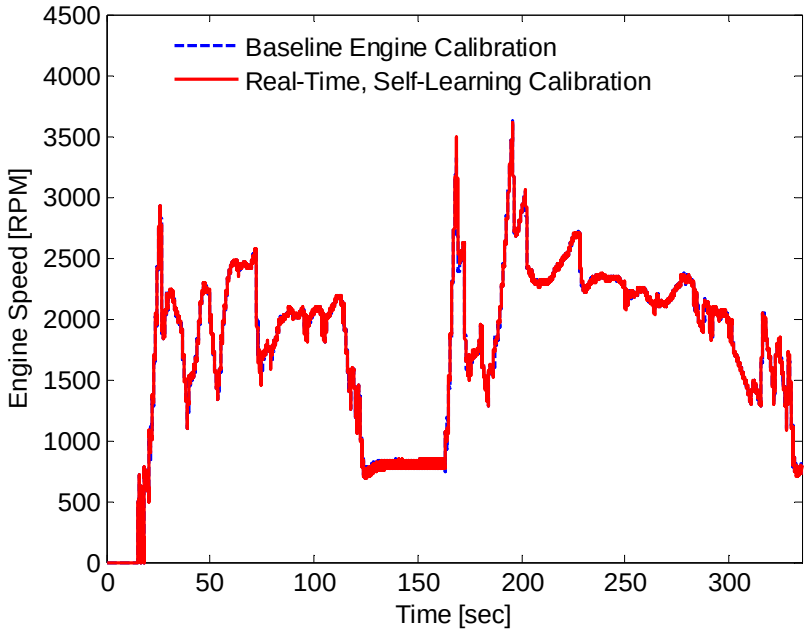

Figure 5.4 − Engine speed.

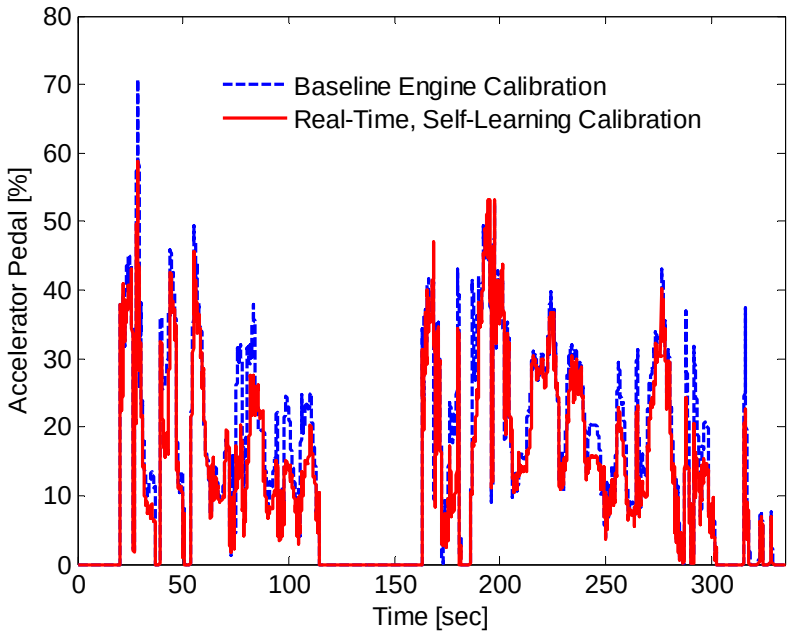

Figure 5.5 − Gas-pedal position rate representing a driver's driving style.

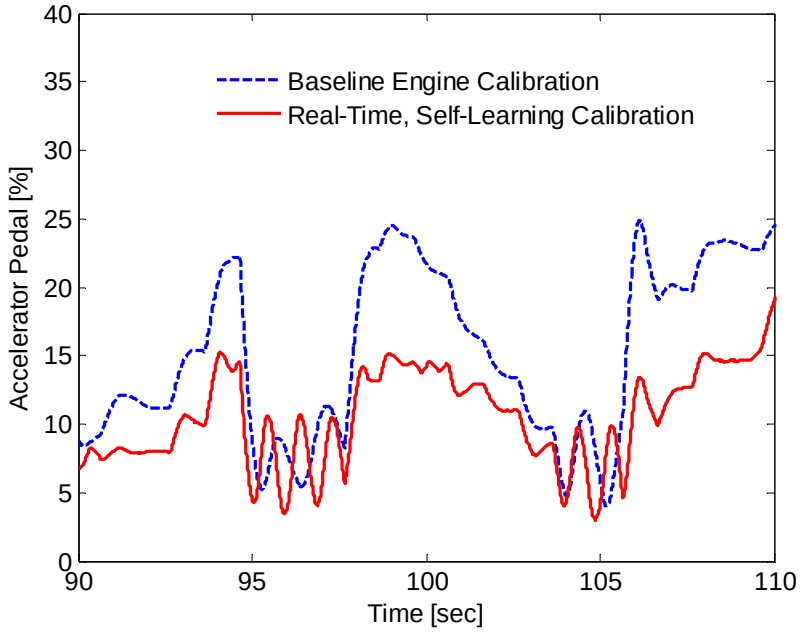

Figure 5.6 − Gas-pedal position rate representing a driver's driving style (zoom-in).

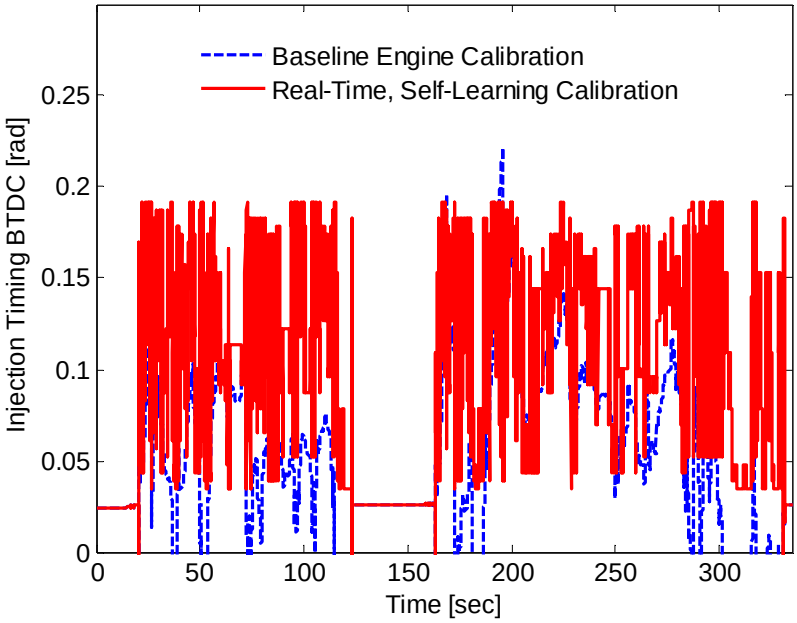

Figure 5.7 − Injection timing.

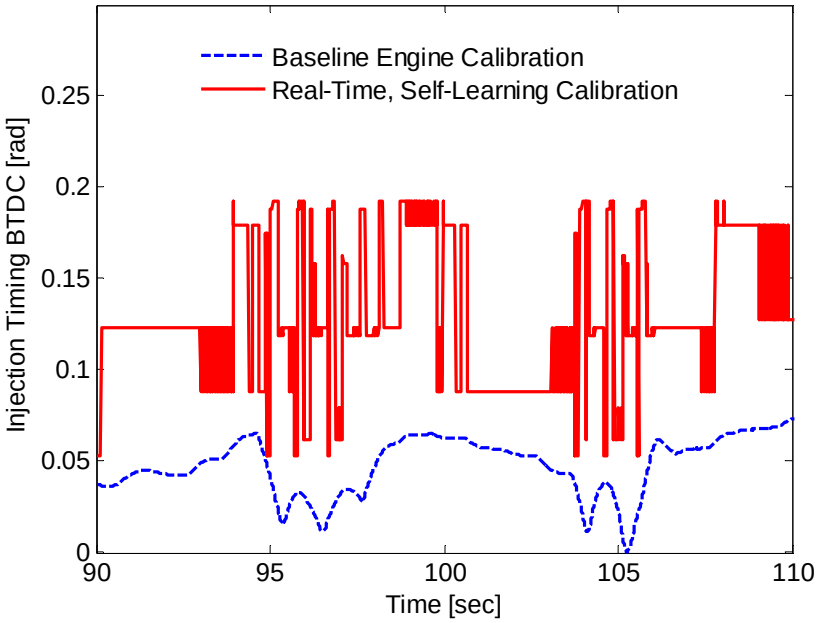

Figure 5.8 − Injection timing (zoom-in).

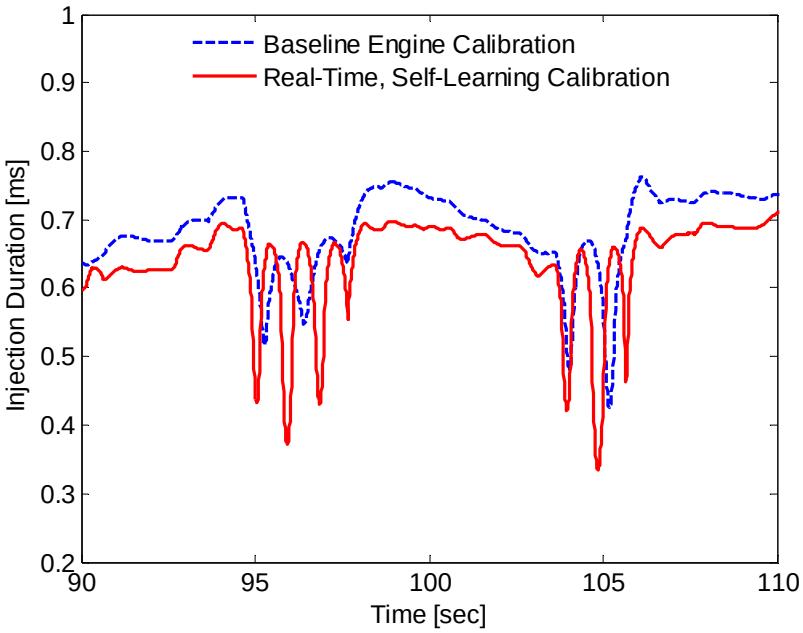

Figure 5.9 − Fuel mass injection duration (zoom-in).

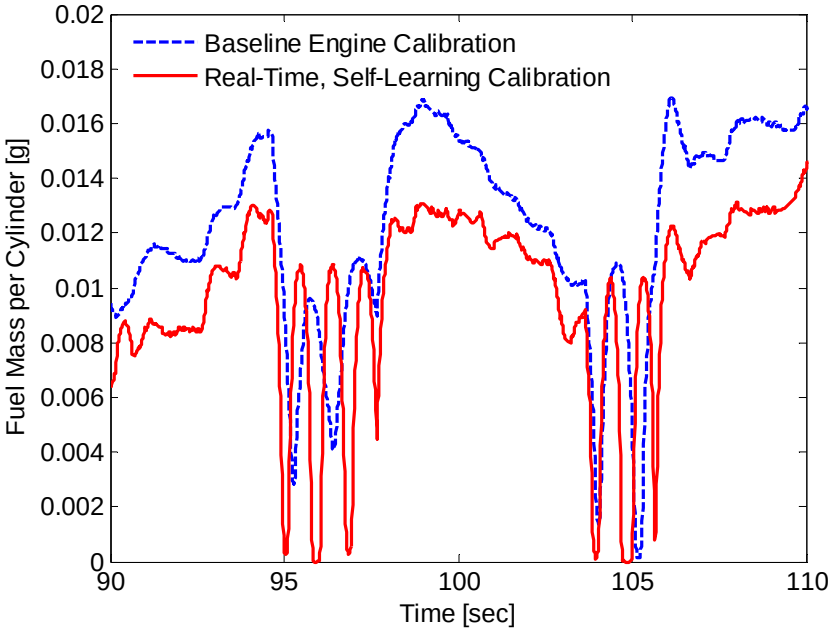

Figure 5.10 − Fuel mass injected per cylinder (zoom-in).

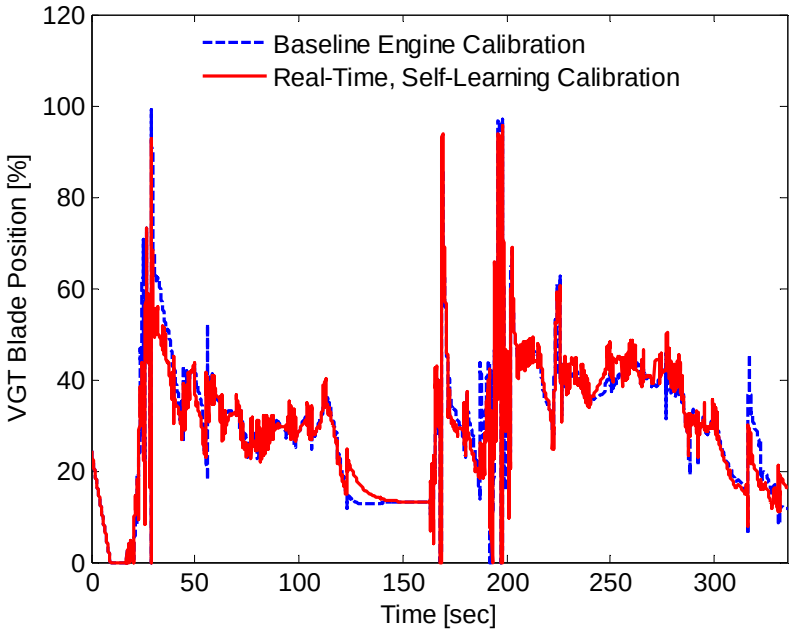

Figure 5.11 − VGT vane position.

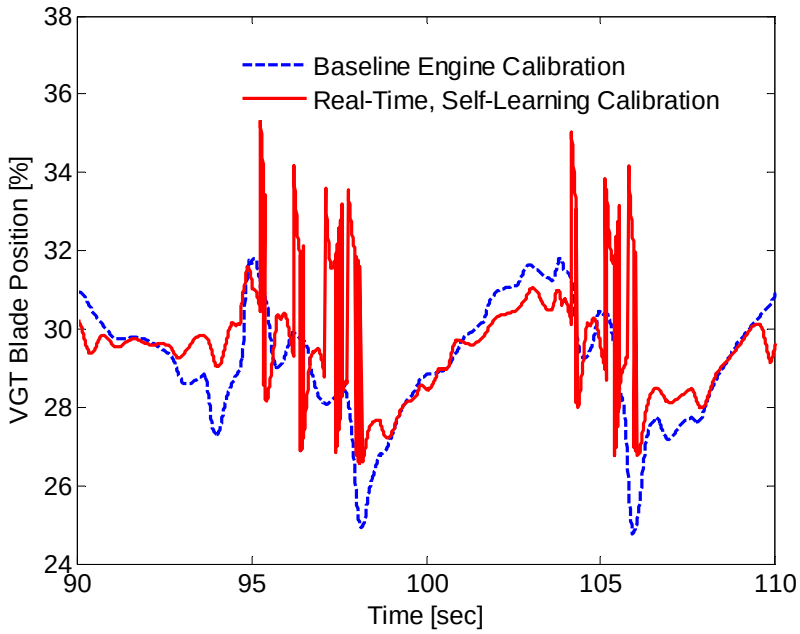

Figure 5.12 − VGT vane position (zoom-in).

Having the engine operate at the maximum brake torque, a 9.3% overall improvement of fuel economy was accomplished, as illustrated in Figure 5.13, compared to the baseline calibration. Figures 5.14 and 5.15 show a decrease in the temperature and NOx concentration of the exhaust gas; this is due to the earlier injection determined for the engine operating transitions of the particular driver's driving style. Table 2 summarizes the quantitative assessment of the improvement of fuel economy, and NOx, by employing the self-learning controller in ECU development.

Table 2: Quantification assessment of benefits with self-learning controller compared to baseline ECU.

Engine Performance Indices	**Improvement [%]**
Fuel consumption	9.1
NOx	8.6

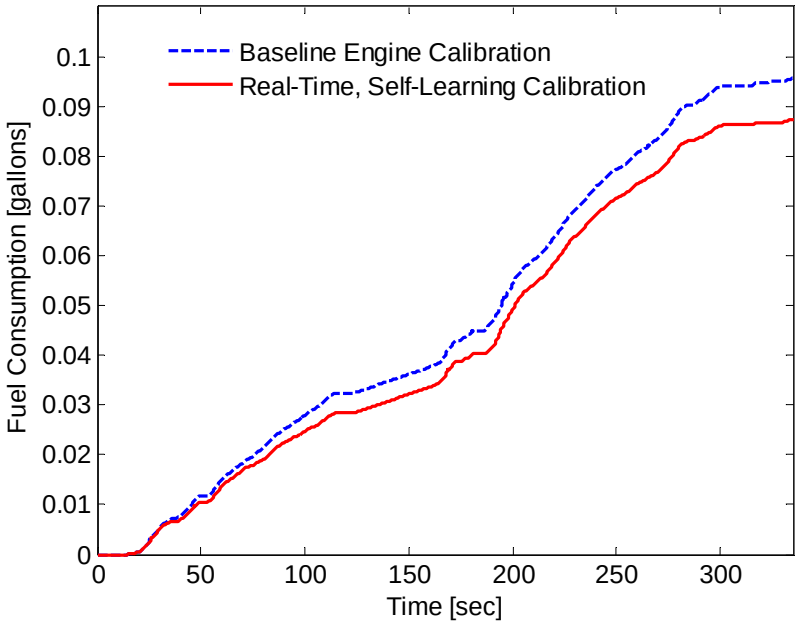

Figure 5.13 − Fuel consumption for the driving cycle.

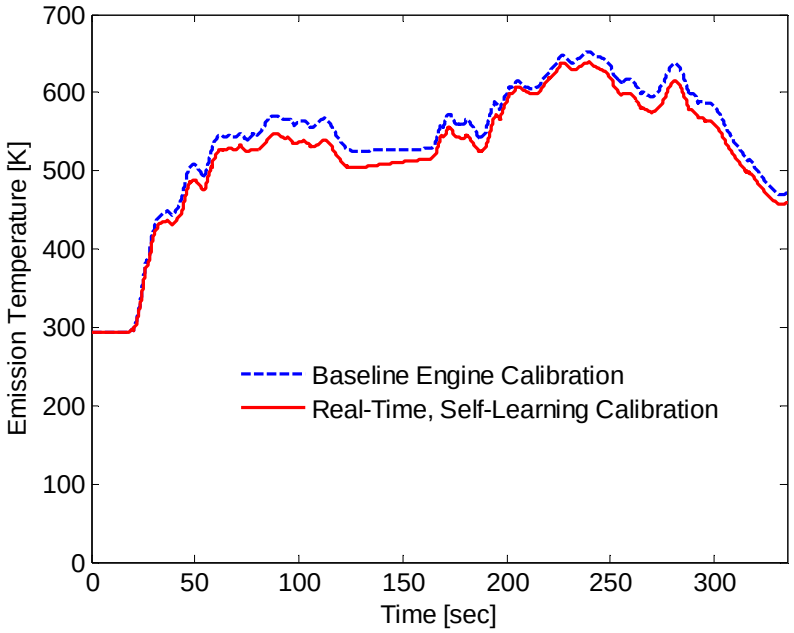

Figure 5.14 − Emission temperature in the exhaust manifold.

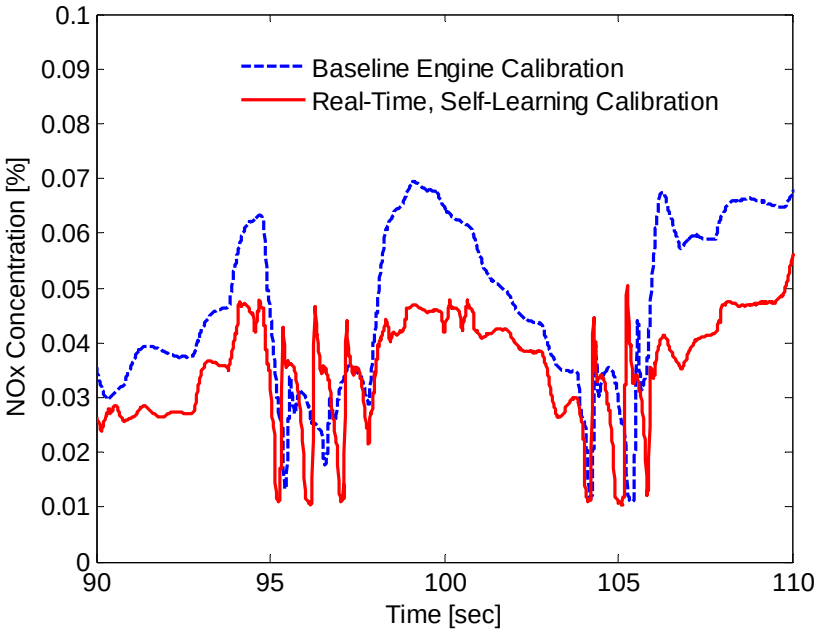

Figure 5.15 − NOx concentration of emissions (zoom-in).

5.6 Concluding Remarks

This chapter has proposed a decentralized learning scheme suitable for finite Markov chains. In this scheme, the decision makers do not demonstrate myopic behavior explicitly. Instead, a random hierarchy among them is assumed, based on which each one observes the control actions of the other while attempting to select a Nash equilibrium coordinated control policy. Decentralization is a common and often necessary aspect of large sequential decision-making problems. It is necessary when complete information among decision makers is impractical due to the increase of the problem's dimensionality.

In applying the proposed decentralized scheme to the engine calibration problem, a learning process was established that enables the derivation of the values of the controllable variables to occur in parallel phases. The values for more than one controllable variable can thus be determined while keeping the problem's dimensionality tractable. The example presented an application of this scheme in real-time, self-learning calibration of a diesel engine with respect to injection timing and VGT vane position. The engine was able to realize the optimal values of injection timing and VGT for a driving style represented by a segment of the FTP-75 driving cycle, thus, optimizing fuel economy. Future research should validate this method to more than two controllable variables and the implications for the required learning time.

The proposed method, in conjunction with the POD model and POSCA, can guarantee optimal calibration for steady-state and transient engine operating points resulting from the driver's driving style. This capability can be valuable in engines utilized in hybrid-electric powertrain configurations when real-time optimization of the power management is considered.

5.7 References

[1] Wheeler, R. and Narenda, K., "Decentralized Learning in Finite Markov Chains," IEEE Transactions on Automatic Control, vol. 31(6), pp. 373-376, 1986.

[2] Tsypkin, Y. Z., Adaptation and Learning in Automatic Systems. New York, Academic Press, 1971.

[3] Moore, A. W. and Atkinson, C. G., "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time," Machine Learning, vol. 13, pp. 103-30, 1993.

[4] Bush, R. R. and Mosteler, F., Stochastic Models for Learning. New York, John Wiley, 1958.

[5] Narendra, K. S. and Wheeler, R. M., Jr., "N-Player Sequential Stochastic Game with Identical Payoffs," IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-13, pp. 1154-1158, 1983.

[6] Srikantakumar, P. R. and Narendra, K. S., "A Learning Model for Routing in Telephone Networks," SIAM Journal on Control and Optimization, vol. 20, pp. 34-57, 1982.

[7] Zheng, Y., Luo, S., Lv, Z., and Wu, L., "Control Parallel Double Inverted Pendulum by Hierarchical Reinforcement Learning," Proceedings of the 2004 7th International Conference on Signal Processing Proceedings (IEEE Cat. No.04TH8739), pp. 1614-17, Beijing, China, 2004.

[8] Szer, D. and Charpillet, F., "Improving Coordination with Communication in Multi-Agent Reinforcement Learning," Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004, pp. 436-440, Boca Raton, FL, United States, 2004.

[9] Scherrer, B. and Charpillet, F., "Cooperative Co-Learning: a Model-Based Approach for Solving Multi-Agent Reinforcement Problems," Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, pp. 463-8, Washington, DC, USA, 2002.

[10] Beynier, A. and Mouaddib, A.-I., "An Iterative Algorithm for Solving Constrained Decentralized Markov Decision Processes," Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI-06/IAAI-06, pp. 1089-1094, Boston, MA, United States, 2006.

[11] Yagan, D. and Chen-Khong, T., "Coordinated Reinforcement Learning for Decentralized Optimal Control," Proceedings of the 2007 First IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (IEEE Cat. No.07EX1572), pp. 7 pp., Honolulu, HI, USA, 2007.

[12] Shen, D., Chen, G., Cruz, J. B., Jr., Kwan, C., and Kruger, M., "An Adaptive Markov Game Model for Threat Intent Inference," Proceedings of the IEEE Aerospace Conference, pp. 4161613, Big Sky, MT, United States, 2007.

[13] Myerson, R. B., Game Theory: Analysis of Conflict, Harvard University Press, September 15, 1997.

[14] Osborne, M. J. and Rubinstein, A., A Course in Game Theory, The MIT Press, July 12, 1994.

[15] Fudenberg, D. and Tirole, J., Game Theory, The MIT Press, August 29, 1991.

[16] Neumann, J. v., Morgenstern, O., Rubinstein, A., and Kuhn, H. W., Theory of Games and Economic Behavior, Princeton University Press, March 2007.

[17] Nash, J. F., "Equilibrium Points in N-Person Games," Proceedings of the National Academy of Sciences, 18, 1950.

[18] Khamsi, M. A. and Kirk, W. A., An Introduction to Metric Spaces and Fixed Point Theory, 1st edition, Wiley-Interscience, March 6, 2001.

[19] Malikopoulos, A. A., Assanis, D. N., and Papalambros, P. Y., "Real-Time, Self-Learning Optimization of Diesel Engine Calibration," Proceedings of the 2007 Fall Technical Conference of the ASME Internal Combustion Engine Division, Charleston, South Carolina, October 14-17, 2007.

[20] Heywood, J., Internal Combustion Engine Fundamentals, 1 edition, McGraw-Hill Science/Engineering/Math, April 1988.

[21] TESIS, <http://www.tesis.de/en/>.

CHAPTER 6

CONCLUSIONS

6.1 Dissertation Summary

This dissertation has proposed the theory and algorithms toward making the engine of a vehicle an autonomous intelligent system that can learn the optimal values of various controllable variables in real time while the driver drives the vehicle. Through this approach, engine calibration is optimized with respect to both steady-state and transient operation designated by the driver's driving style. Consequently, every driver can realize optimal fuel economy and pollutant emissions as fully as possible.

The engine was treated as a controlled stochastic system, and engine calibration was formulated as a sequential decision-making problem under uncertainty. This problem involved two major sub-problems: (a) the state estimation and system identification problem, and (b) the stochastic control problem. In Chapter 2, the underlying theory for building computational models suitable for sequential decision-making under uncertainty was reviewed. These models constitute an essential framework for making intelligent systems able to learn the control actions that optimize their long-term performance.

In Chapter 3, a real-time computational learning model was implemented suitable for solution of the state estimation and system identification sub-problem. A state-space representation was constructed through a learning mechanism that can be employed simultaneously with a lookahead control algorithm in solving the stochastic control problem. The model allows decision making based on gradually enhanced knowledge of

system response as it transitions from one state to another, in conjunction with control actions taken at each state.

To enable the engine to select the optimal values of various controllable variables in real time, a lookahead control algorithm was developed in Chapter 4. The algorithm solves the stochastic control sub-problem by utilizing accumulated data acquired over the learning process of the state-space representation. The combination of the state-space representation and control algorithm make the engine progressively perceive the driver's driving style and eventually learn its optimal calibration for this driving style. The longer the engine runs during a particular driving style, the better the engine's specified performance indices will be. This property arises due to the learning process required by the state representation to capture the stationary distribution of the engine operation with respect to the driver's driving style. The engine can learn its optimal calibration for any other driver who indicates his or her identity before starting the vehicle by assigning the transition probability $\mathbf{P}(\cdot,\cdot)$, and cost (or reward) matrices $\mathbf{R}(\cdot,\cdot)$ for each driver.

The enhancement of the problem's dimensionality, when more than one controllable variable is considered, was addressed by the development of a decentralized learning control scheme, presented in Chapter 5. This scheme draws from multi-agent learning research in a range of areas, including reinforcement learning, and game theory, to coordinate optimal behavior among the various controllable variables. The engine was modeled as a cooperative multi-agent system, in which the subsystems, i.e., controllable variables, were treated as autonomous intelligent agents who strive interactively and jointly to optimize engine performance criteria.

In summary, the research reported in this dissertation has taken steps toward development engine calibration that can capture transient engine operation designated by the driver's driving style. Each individual driving style is different and rarely meets those driving conditions of testing for which the engine is calibrated to operate optimally by means of the state-of-the-art calibration methods. The implementation of the proposed

approach in a vehicle is expected to significantly reduce the discrepancy between the gas mileage estimate displayed on the window sticker or featured in advertisements and the actual gas mileage of the vehicle.

6.2 Summary of Contributions

Three distinct steps were taken toward making the engine an autonomous intelligent system:

1. A computational model suitable for real-time sequential decision-making under uncertainty was implemented. A state-space representation was constructed through a learning mechanism and utilized in solving the state estimation and system identification sub-problem. The model accumulates gradually enhanced knowledge of system response as it transitions from one state to another, in conjunction with actions taken at each state. As the system interacts with its environment, the state representation realizes the sequences of state transitions that occurred in the Markov domain. This realization converges to the stationary distribution of the Markov chain (Theorem 3.3).

2. A lookahead control algorithm was developed that addresses the stochastic control sub-problem in real time by utilizing accumulated data acquired over the learning process of the state-space representation. The principle of the algorithm is founded on the theory of stochastic control problems with unknown disturbance distribution, also known as games against nature. The solution of the algorithm exhibits performance bounds that are

better than the solution of any minimax control policy provided by dynamic programming (Theorem 4.1).

3. A decentralized learning control scheme was proposed to coordinate optimal behavior among various decision makers for mutual benefit. The solution of the decentralized scheme provides a Nash equilibrium coordinated control policy.

6.3 Future Research

The proposed approach toward making the engine of a vehicle an autonomous intelligent system assumes implicitly that engine output can be perfectly observed. In this context, engine operation was modeled as a completely observable Markov decision process. Future research should examine the potential of having limited capabilities in observing engine output, and thus, should consider modeling engine operation as a partially observable Markov decision process.

Future research should also investigate the potential of advancing the POD model to accommodate more than one decision maker with non-cooperative interactions. These problems are found in systems in which many intelligent decision makers interact with each other to pursue their own interests, e.g., subsystems of a hybrid-electric vehicle.

Sequential decision-making under uncertainly is a fundamental problem faced by autonomous intelligent or rational systems, e.g., physical systems, robots, automated manufacturing systems, etc, embedded in complex environments that choose actions to achieve long-term goals efficiently. Computational rationality can be achieved by modeling a system and the interaction with its environment through actions, perceptions, and costs (or rewards). A widely adopted paradigm for modeling this interaction is the completely or partially observable Markov decision process. Theory and algorithms

related to these problems have been extensively reported in the literature. However, no research has been reported addressing the computational cost associated with deriving these optimal policies. Future research should address the impact of the computational time required in deriving optimal control policies and the coupled tradeoffs. The latter could result in a quantitative assessment of optimal policies with respect to required computational time. This assessment would provide an essential treatment in selecting control policies suitable for real-time stochastic control implementation.