Distributed Supervisory Controller Design for Battery Swapping Modularity in Plug-in Hybrid Electric Vehicles

by

Shifang Li

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2011

Doctoral Committee:

Professor A. Galip Ulsoy, Co-Chair
Professor Ilya V. Kolmanovsky, Co-Chair
Professor Jessy W. Grizzle
Professor Panos Y. Papalambros

© Shifang Li

2011

To my loving family

ACKNOWLEDGMENTS

I would like to convey my deepest gratitude to my advisers Professor Galip Ulsoy and Professor Ilya Kolmanovsky for all their encouragement and guidance. I sincerely believe that this work would not be possible without their inspiration and guidance. Their passion and preciseness for research will always influence me in my future career.

I would like to thank my committee members, Professor Panos Papalambros and Professor Jessy Grizzle. Their insightful comments during our meetings have led this dissertation to be more thorough and complete.

I am grateful to all our research group members, Sun, Yongseob, Rachael, Rob, William, Diane and Shiming. I will always remember your help and happy times together. My special thanks goes to Dr. Melih Cakmakci for his kind help at the beginning of my research. I would also like to thank all my friends in Michigan, who have made my graduate school time so colorful.

I also gratefully acknowledge the financial support from our department of Mechanical Engineering and the Ford Motor Company.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

As industry strives to standardize engineering design, manufacturing, and maintenance processes, the focus on achieving component modularity is increasing. Component swapping modularity (CSM) in control systems allows component change without redesign of the system level controller, while achieving the required system performance. Opportunities to achieve CSM are emerging in control systems consisting of smart components connected by bidirectional communication networks. By distributing a part of the controller into the component module, controller recalibration can be limited to only the component module when the component changes.

In this dissertation, a novel Direct Method is proposed to generate the distributed controller with CSM through a bi-level optimization. The distributed controller enables CSM and provides required system performance for each component variant. The Direct Method is applied to throttle actuator CSM design in engine idle speed control. The results demonstrate that the new Direct Method improves the CSM results compared to the previous 3-Step Method. In addition, the Direct Method permits the designer to trade off desired system performance versus achievable CSM.

The Direct Method is then applied to design a distributed supervisory controller for battery CSM in plug-in hybrid electric vehicles. A novel feedback based controller for the charge sustaining mode is proposed. For effective controller distribution, a method based

on sensitivity analysis of the control signals with respect to the battery hardware parameter is introduced. The bi-level optimization problem for the distributed controller gains is solved using the Augmented Lagrangian Decomposition method. The results demonstrate that battery CSM can be achieved without compromising fuel economy compared to the centralized control case.

CHAPTER I

INTRODUCTION

In the broadest terms, modularity (or modularization) is an approach for organizing complex products and processes efficiently, by decomposing complex tasks into simpler portions so they can be managed independently and yet operate together as a whole [1]. Ulrich and Tung define modularity in terms of two characteristics of product design: (1) similarity between the physical and functional architecture of the design and (2) minimization of incidental interactions between physical components [2]. This definition accounts for the functional aspects of a product but ignores all other life-cycle characteristics [3]. Gershenson *et al.* state that modular design also aims to develop product architecture consisting of physically detachable units for rapid development, ease of assembly, re-use, and other life-cycle objectives [4].

Besides the component modularity properties of physical and functional independence, component swapping modularity (CSM) allows two or more alternative basic components to be paired with the same modular components creating different product variants belonging to the same product family [2]. The benefits of CSM include [5-7]: 1) Economy of scale; 2) Ease of product updating; 3) Increased product variety; 4) Decreased order lead-time; 5) Decoupled design; 6) Ease of product diagnosis, maintenance and disposal.

Changing a component in a control system may require a complete redesign of the entire controller for performance and stability requirements, with tremendous cost involved. Consumers cannot easily customize or upgrade electro-mechanical products, such as automobiles. However, if the control system is designed such that the component has CSM, the system can be easily customized and upgraded.

A component in a control system has CSM if the component change can be accommodated by only recalibrating (i.e., retuning) the component controller built inside the component module, without redesign of the system level controller, so that the system performance meets a defined performance metric subject to specified constraints. Thus the swappable component becomes a plug-and-play component.

1.1. Control Methods for Component Change

Existing control techniques that can deal with component change include robust control, adaptive control, and gain scheduled control. However, to some extent, they are not applicable for large component change. For instance, typical robust controllers are designed to withstand parameter variations caused by manufacturing tolerances, changes in operating conditions or aging; they may not generally be able to cope with large changes in parameters associated with swappable components. Due to a performance-robustness trade-off, even if such a controller can be made robust for stability, the performance must necessarily be compromised. An adaptive controller relies on on-line adaptation and intricate implementation steps to assure stability, robustness to noise, and persistence of excitation. In many industrial applications degraded transient performance during the adaptation phase may not be acceptable. Certification and validation of adaptive controllers remains a challenge. Gain scheduled controllers may require large

memory and computing power of the microcontroller for implementation, which is not cost-efficient in realistic applications.

One simple approach for controller implementation to accommodate component change is to reflash the controller gains when the component changes. This approach requires involvement of the original equipment manufacturer (OEM) (which owns the controller) every time component update occurs, and may preclude customers making changes independently.

Recently, plug-and-play control is proposed to accommodate component change. However, it considers only system stability, without considering system performance for each component variant. A high level model predictive control for plug and play process control [8], and a hierarchical model predictive control to accommodate load variations on the grid [9], have been investigated only considering system stability.

These considerations create the opportunity for further research on distributed controller architecture that achieves CSM, which avoids the above drawbacks.

1.2. Networked Control Systems

With the proliferation of low cost electronics, many control system components, such as sensors and actuators, can now incorporate on-board computers (i.e. CPU, memory, I/O interface), which have communication interfaces and can perform component specific control functions. These components are referred to as “smart components” [10-12]. As an example, today’s automobile has up to 80 microprocessors, and the software in those microprocessors provides 500 – 600 customer visible features [13]. An overview of the design considerations of smart components to integrate sensing, computing, and communication with limited packaging constraints is given in [14].

Networked Control Systems (NCSs) are spatially distributed systems for which the communication between controllers and intelligent I/O devices (e.g., smart sensors and smart actuators) is supported by a shared communication network [15]. The defining feature of NCSs is that the feedback signals are exchanged in the form of information packages through a network as shown in Figure 1.1. NCSs enable bidirectional communication among the smart components and the system level controller. The additional information exchange on the bidirectional communication network has been shown to improve CSM [16].

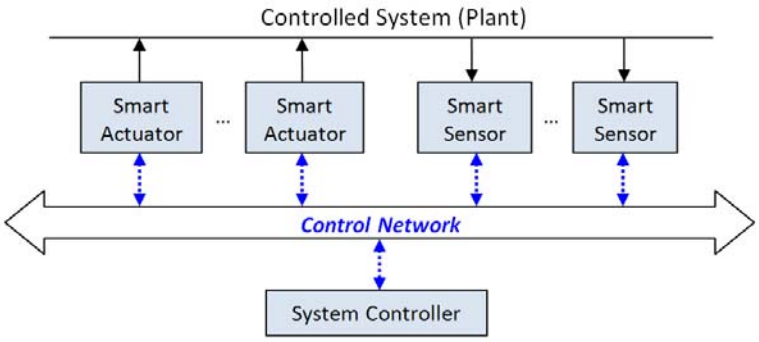


Figure 1.1: NCS with bidirectional communication

A wide variety of different networks are commercially available (e.g., ControlNet, DeviceNet, Ethernet, Profibus, Sercos, WorldFIP) for the implementation of a NCS architecture [17]. Murray *et al.* identify control over networks as one of the key future directions for control [18]. Although NCSs provide many benefits such as flexible architectures, reduced system wiring, ease of system maintenance and upgrade at lower cost [15, 19], there are some disadvantages such as communication delays, bandwidth limitations and packet loss [19-22]. Current research on NCSs primarily focuses on

understanding the effects of network delays and packet loss [15, 23]. The effects of the network-induced delay will not be considered in this dissertation.

1.3. Distributed Control with CSM

Systems with smart components connected by bidirectional communication networks facilitate distributed controller implementation to achieve CSM [24]. This approach to achieve CSM relies on distributing part of the control function, which is dependent on the component hardware parameters, to the component module. When component change occurs, only the controller implemented in the component module needs to be redesigned or recalibrated to achieve the required system performance. Such a distributed architecture is designed to achieve the required system performance for each system configuration and to maintain the controller in the swappable component as simple as possible. A simple controller in the swappable component is suitable for implementation in a microcontroller within the component module, which often has very limited computing power, and a simple controller in the swappable component also results in reduced engineering recalibration time and effort when the component changes.

Figure 1.2 shows a control system in which many different types of actuators may be deployed. The original centralized controller is distributed into two parts, the base controller and the actuator controller, where the actuator controller is implemented in the actuator module, making the actuator a smart component. Bidirectional communication is introduced between the base controller and the actuator controller. The distribution ensures that only the actuator controller is dependent on the actuator hardware parameters, while the base controller remains the same for different actuator variants. Thus, when the actuator changes, rework in terms of controller redesign or recalibration can be limited to

the smart actuator only, making it a modularly swappable component, or a plug-and-play component, while providing the required system performance.

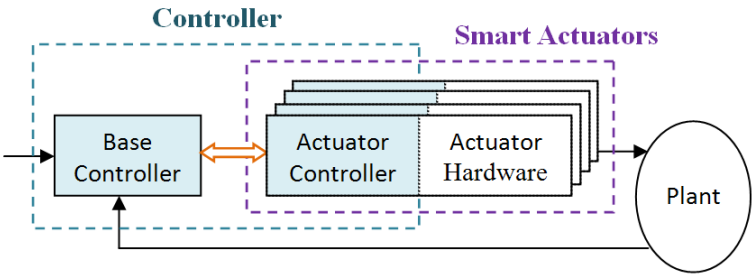


Figure 1.2: Control system with modularly swappable actuator component.

Note that a controller implemented in the component module, which uses pole-zero cancellation (assume the component dynamic model is stable) to cancel the dynamics of the new component and maintain the dynamics of the original/nominal component, can trivially provide CSM. The shortcoming of such an approach is that the overall system response remains the same as for the original component, and potential performance improvements that can be achieved with a better component are not realized. It is, thus, of interest to understand if a distributed control system can be designed to achieve CSM as well as improved system performance for a better component variant.

The 3-Step Method for distributed controller design to achieve CSM has been developed in [24]. In this approach, the centralized controller for each system configuration with different component variant is designed first. Then, the order and structure of the distributed controller is assumed, such that only the component controller that is built into the swappable component is dependent on the component hardware

parameters. Finally, the CSM metric is maximized by exact (or approximate) matching of the transfer function of the distributed controller with that of the centralized controller.

The 3-Step Method has been applied to design CSM of the VCT actuator and the EGO sensor in a VCT engine [25]. A case study for a distributed idle speed control (ISC) design with throttle actuator CSM has been considered in [16].

In [16], it was shown that the 3-Step Method can achieve throttle actuator CSM by distributing a ISC between a base controller and an actuator controller, where only the actuator controller is dependent on the actuator hardware parameter. However, no swapping modularity could be achieved when the actuator controller transfer function is relatively simple, such as first order or just gains. This could be a significant deficiency when computing power of the smart actuator is limited. From both computation and recalibration perspectives, a simple controller in the smart actuator is more amenable to implementation. Moreover, the current 3-Step Method is based on model matching of controller transfer functions, it is limited to linear controller design. Thus, in this dissertation, we propose a new improved Direct Method for distributed controller design with CSM.

The Direct Method developed in this dissertation generates the distributed controllers directly by solving a bi-level optimization problem. The Direct Method is applicable to nonlinear, as well as linear, control systems. Moreover, unlike the 3-Step Method, the Direct Method simultaneously addresses the two design objectives, system performance and CSM, through a nonlinear optimization formulation. For multi-objective optimization, a simultaneous approach will deliver a better, or at least the same, solution compared to

that of a sequential approach [16]. Therefore, the Direct Method is expected to outperform the 3-Step Method.

A solution algorithm using the Collaborative Optimization (CO) [26, 27] and Augmented Lagrangian Decomposition (ALD) methods [28] is implemented for the bi-level optimization in the Direct Method. CO is a multidisciplinary design method that preserves disciplinary-level design autonomy while providing a coordinating mechanism that ensures progress toward an optimum and compatibility between the disciplinary designs [27, 29-31]. CO possesses analytical features that lead to computational difficulties when conventional nonlinear programming algorithms are applied to the system-level problem [26]. The ALD method has been investigated to relax the system consistency constraints for sub-problem feasibility in CO [28]. The ALD algorithm converges to the Karush-Kuhn-Tucker points of the original problem under mild assumptions [28]. Compared to the general bi-level formulation, the application of CO and ALD methods is more complex, but provides more design freedom for each of the inner stage problems [32].

To illustrate our approach, we have investigated throttle actuator CSM with respect to engine idle speed control (ISC). Both the 3-Step Method and the novel Direct Method have been applied to this case study. The results demonstrate that the Direct Method can significantly improve the CSM metric compared to the 3-Step Method. In addition, the Direct Method allows the designer to trade off between desired system performance and achievable CSM.

1.4. Battery CSM in PHEVs

The new Direct Method is applied to design a distributed supervisory controller for CSM of the batteries with different energy capacities in plug-in hybrid electric vehicles (PHEVs).

PHEVs enable the transportation energy sector to access lower-cost, cleaner, and renewable energy from the electric grid [33]. One main barrier to the commercialization of PHEVs is the cost and reliability of the batteries. Increasing the battery energy capacity from 20 to 40 miles of all electric range provides an extra 15% reduction in fuel consumption, but it also nearly doubles the cost [34]. It is beneficial to investigate battery CSM in PHEVs, such that different batteries can be applied to the same vehicle depending on the customer driving pattern. Moreover, as battery technology advances, the vehicle can be easily upgraded with newer batteries (e.g., batteries with higher energy capacity). Thus, battery CSM in PHEVs is investigated in this dissertation.

Current supervisory controllers for PHEVs have centralized architecture, see [35]. A typical PHEV operates in a charge depleting (CD) or electric vehicle (EV) mode until the battery state of charge decreases to a certain value, then it switches to a charge sustaining (CS) mode and operates like a conventional hybrid electric vehicle (HEV). The control strategies proposed for HEVs can be applied to the CS mode controller design for PHEVs. Various control design methods for HEVs are available, such as rule-based control [36-39] and optimization-based control, (e.g., equivalent consumption minimization strategy [40-42], stochastic and deterministic dynamic programming [43-45]). The rule-based controllers are simple to implement but sub-optimal, while the optimization-based controllers are computationally expensive or non-causal for real-time application. A

machine learning framework has been investigated to find an algebraic function that emulates the optimal solutions generated by Dynamic Programming for various roadway type and traffic conditions [46]. Feedback–based supervisory controllers have also been developed for load following while smoothing engine power [47]. The feedback controller synthesized from model predictive control was experimentally evaluated and showed improved fuel economy compared to two baseline strategies [48].

In this dissertation, we propose a novel feedback-based controller for the CS mode. The controller gains are generated though optimization to achieve optimal fuel economy and driving performance, while satisfying the constraints on closed loop system stability, battery charge sustainability and component reliability. The controller is designed with respect to the EPA US06 cycle, but the simulation results demonstrate that the controller also achieves good fuel economy, good driving performance and charge sustainability over other driving cycles (e.g., the EPA UDDS and HWFET cycles). The feedback-based controller for the CS mode facilitates controller distribution for battery CSM.

After the centralized controller is obtained through optimization, the method based on sensitivity analysis of the control signals with respect to the battery parameter is applied to determine effective controller distribution. The distributed controller gains are obtained by solving a bi-level optimization using the ALD method. The simulation results demonstrate that the proposed distributed controller achieves battery CSM without compromising fuel economy compared to the centralized control case.

1.5. Original Contributions

A. Methodology for CSM in Control Systems

1. A new Direct Method for the design of distributed controllers with CSM using smart components and bidirectional communication is proposed. The aim of the method is to design a distributed controller such that the component change can be accommodated by only recalibrating the component controller inside the component module so that the system performance meets a defined performance metric subject to specified constraints. This work is published in [49].
2. The bi-level optimization problem for the distributed controller gains in the Direct Method is formulated and solved using the multidisciplinary design optimization algorithms (e.g., Augmented Lagrangian Decomposition method). This work is published in [50, 51].
3. A method based on sensitivity analysis of the control signals with respect to the component hardware parameters for effective controller distribution is introduced. This work is published in [50].

B. Case Studies and Applications

1. The Direct Method is applied to throttle actuator swapping modularity design in engine idle speed control (ISC) and is compared to the previous 3-Step Method [24, 25]. It is shown that: 1) bidirectional communication among smart components can improve CSM compared to conventional unidirectional communication. 2) The Direct Method improves the CSM metric significantly compared to the previous 3-Step Method. The Direct Method can provide actuator CSM, while the 3-Step Method fails, when the actuator controller is just gains or first order, and the Direct Method can achieve full range of actuator CSM, while the 3-Step Method can only achieve partial range of actuator CSM, when the actuator controller is second order or third order. 3)

The Direct Method allows the designer to trade off desired system performance and achievable CSM when the controller distribution is of a certain structure. For instance, by increasing the settling time design target for ISC from 1.5 sec to 4.1 sec, the actuator CSM can be improved by 20% when the actuator controller is just gains. This work is published in [16, 49, 52].

2. The Direct Method is applied to achieve battery CSM in PHEVs. A novel feedback based controller for the charge sustaining mode for PHEVs is proposed to facilitate battery CSM. The controller provides good fuel economy, good driving performance in terms of power tracking error, and charge sustainability over three standard driving cycles. The method based on sensitivity analysis of the control signals with respect to the battery parameter is applied to define the controller distribution architecture. The distributed controller gains are obtained by solving a bi-level optimization using the ALD method. The results demonstrate that the proposed distributed supervisory controller achieves battery CSM without compromising fuel economy compared to the centralized control case. This work is published in [50, 51].

1.6. Outline of the Dissertation

This dissertation is organized as follows. After the introduction in Chapter I, the general Direct Method for distributed control system design with CSM and the definition of CSM metric is presented in Chapter II. In Chapter III, the 3-Step Method and the Direct Method are applied to design a distributed ISC that achieves throttle actuator CSM. Then Chapter IV and Chapter V present the application of the Direct Method to battery CSM in PHEVs. In Chapter IV, the centralized supervisory controller is developed and evaluated, while in Chapter V, the distributed controller which enables battery CSM is

designed. Finally Chapter VI gives the summary and conclusions, and lays out some directions for future work.

CHAPTER II

THE DIRECT METHOD FOR CONTROL SYSTEM DESIGN WITH CSM

In this Chapter, the Direct Method for designing control systems with component swapping modularity (CSM) is introduced. We consider CSM for one component in a control system, and we assume that the component is a smart component, i.e., it includes a microcontroller that can perform control functions and has network communication capability. The conventional centralized controller, thus, can be distributed into two parts, the system controller and the component controller, where the component controller is built into the smart component. By distributing the component related control functions into the component controller, and by introducing bidirectional communication between the system controller and the component controller, CSM can be achieved.

The component in a control system is said to have CSM if the component change can be accommodated by only recalibrating (i.e., retuning) the component controller inside the component module so that the system performance meets a defined performance metric subject to specified constraints.

In this dissertation we focus on the case when the component hardware depends on a parameter vector that takes values within a given continuous parameter set and the

number of component variants of interest, i.e., values of the parameters within the specified parameter set, is finite.

As an example, consider a system where the swappable component is an actuator, and the actuator variants differ by the value of a parameter which is the actuator time constant. The system may be configured with a finite number of different actuator components, each corresponding to a particular value of the time constant. Suppose that the system performance metric is the settling time of the closed-loop system response, and the constraints are imposed on the magnitude of the control signal (in response to specified reference commands), on internal system stability, and on gain and phase margins. CSM of the actuator is achieved if the controller can be distributed between a system controller and an actuator controller (component controller built in the actuator component), in such a way that the actuator change can be accommodated by only changing the actuator controller without changing the system controller, and the system performance meets the performance (settling time) requirement under the specified constraints (control magnitude limits, internal system stability, gain and phase margins).

Clearly, CSM is dependent on the chosen distribution architecture between the system controller and the component controller. For instance, consider the centralized controller designed for each component variant without imposing any constraints on the structure of this controller and suppose these controllers are feasible and can achieve the required performance under all the constraints of the problem. If the entire centralized controller is distributed into the smart component, then CSM can always be achieved. But this solution is not very appealing in terms of typical industry practice, as the controller recalibration effort is not reduced, but shifted to the component controller. Furthermore,

from the standpoint of the limited computing power of the component microcontroller, it is often desirable to maintain the component controller to be as simple as possible. While controller simplicity can be measured in a variety of ways, in this dissertation we generally relate controller simplicity to controller order and the number of gains. Therefore, the design of distributed controller with CSM must address the inherent tradeoffs between achievable system performance and simplicity of the component controller implementation.

In what follows, we first discuss a CSM metric to quantify the degree of CSM. We then introduce a method to guide the controller distribution based on sensitivity analysis of the control signals with respect to the component hardware parameters. With this approach, some of the centralized controller gains that result in relatively high sensitivity of the control signals, along with the corresponding calculations, are distributed into the component controller. Finally, once the controller distribution architecture is determined using the sensitivity analysis, we present a bilevel optimization problem formulation to solve for the distributed controller gains.

The bi-level optimization formulation ensures that the resulting system controller gains are the same for different component variants, while the component controller gains can change for each component. Thus, the bilevel optimization problem addresses the two design objectives, system performance and CSM, simultaneously. We demonstrate that this bilevel optimization problem can be solved using Multidisciplinary Design Optimization (MDO) algorithms. We refer to this approach as the Direct Method to contrast it with the previous 3-Step Method [24], which is based on exact (or approximate)

matching of the transfer function of the distributed controller with that of the centralized controller.

2.1. CSM Metric

Consider a finite number of component variants for CSM design. Let $N > 1$ be the number of the possible component variants, and let $M \geq 1$ be the number of the components that can be made to satisfy the CSM property. The CSM metric, M_C, is defined as,

$$M_C = \frac{M-1}{N-1} \tag{2.1}$$

When $M = 1$, only one component that satisfies the CSM property can be applied to the system. In this case, $M_C = 0$, and the component is actually not swappable with other components. When $M = N$, $M_C = 1$, and the full range of CSM is achieved. When $N > M > 1$, M_C indicates the degree of CSM over the N considered component variants.

Note that the CSM metric defined here is related to, and consistent with, the earlier definition in [24]. However, we consider a finite number of components that can potentially be swapped. The previous work considered the components to be continuously dependent on a parameter vector.

2.2. Controller Distribution

Suppose the component has N variants, and the corresponding component parameter vector $\mathbf{p} = [p_1, p_2, ..., p_{n_p}]^T \in R^{n_p}$, takes on the following values, $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\} = P_D \subset P_C$, where P_C is a continuous set of the component parameter vector, and n_p is the dimension of $\mathbf{p}$.

Figure 2.1 shows the block diagram of the centralized controller. For the system configuration with component parameter vector $\mathbf{p} \in P_D$, denote the plant model including the component as $\mathbf{G}_p(\mathbf{p})$, and the centralized controller as $\mathbf{G}_c(\mathbf{x}_c(\mathbf{p}))$, with controller gains $\mathbf{x}_c \in R^n$, where n is the dimension of $\mathbf{x}_c$. The inputs to the controller are the feedback signal vector $\mathbf{y}$, and input signal vector $\mathbf{q}$. The output of the controller is the control vector $\mathbf{u}(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c) = [u_1(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c), \quad u_2(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c), ..., u_{n_u}(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c)]^T \in R^{n_u}$, where n_u is the dimension of $\mathbf{u}$.

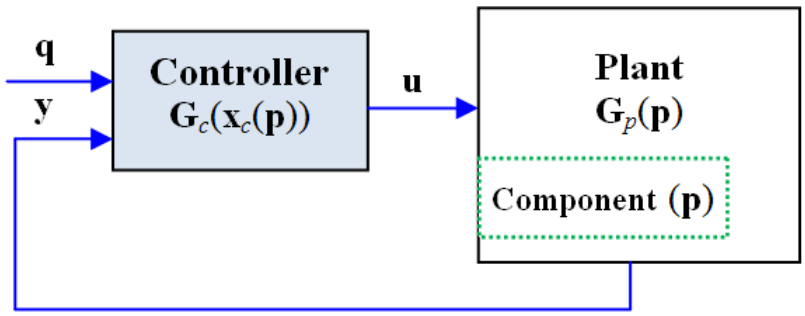


Figure 2.1: Block diagram of the centralized controller

Assume the design objective for the centralized controller is to minimize a cost function that is related to a specific performance metric F, with respect to the controller gains $\mathbf{x}_c$, for a given component parameter vector $\mathbf{p}$,

$$F(\mathbf{p}, \mathbf{x}_c(\mathbf{p})) \tag{2.2}$$

and to satisfy the imposed constraints:

$$\mathbf{g}(\mathbf{p}, \mathbf{x}_c(\mathbf{p})) \leq \mathbf{0}, \tag{2.3}$$

$$\mathbf{h}(\mathbf{p}, \mathbf{x}_c(\mathbf{p})) = \mathbf{0}. \tag{2.4}$$

The centralized controller for each system configuration with component parameter vector $\mathbf{p} \in P_D$ can be designed by conventional methods. Our approach here is first to assume the controller structure (e.g., state feedback control, feed-forward control, etc.), then to solve for the controller gains, $\mathbf{x}_c$, using an optimization problem of the following form:

$$\min_{\mathbf{x}_c} F(\mathbf{p}, \mathbf{x}_c(\mathbf{p}))$$

Subject to:

$$\mathbf{g}(\mathbf{p}, \mathbf{x}_c(\mathbf{p})) \leq \mathbf{0},$$

$$\mathbf{h}(\mathbf{p}, \mathbf{x}_c(\mathbf{p})) = \mathbf{0}.$$

We assume that for each system configuration with component parameter vector $\mathbf{p} \in P_D$, the above optimization is feasible, and the optimal system performance, $F^*(\mathbf{p}, \mathbf{x}_c^*(\mathbf{p}))$, satisfies the required performance metric.

A diagram of the plant with the distributed controller is illustrated in Figure 2.2. The conventional centralized controller is distributed into two parts, the system controller and the component controller. Denote the system controller as $\mathbf{G}_s(\mathbf{x}_s)$, with controller gains, $\mathbf{x}_s \in R^{m_1}$, and the component controller as $\mathbf{G}_m(\mathbf{x}_m(\mathbf{p}))$, with controller gains, $\mathbf{x}_m \in R^{m_2}$, where only the component controller gains, $\mathbf{x}_m$, are dependent on the component parameter, $\mathbf{p}$. Bidirectional communication is introduced between the system controller and the component controller. Denote the signals on the communication network as $\mathbf{s}_1$ and $\mathbf{s}_2$. Depending on the controller distribution structure, $\mathbf{q}_1$ and $\mathbf{q}_2$ are vectors formed from the input signal vector $\mathbf{q}$; $\mathbf{y}_1$ and $\mathbf{y}_2$ are vectors formed from the feedback signal vector $\mathbf{y}$; $\mathbf{u}_1$ and $\mathbf{u}_2$ are vectors formed from the control vector $\mathbf{u}$, compared to the centralized controller in Figure 2.1.

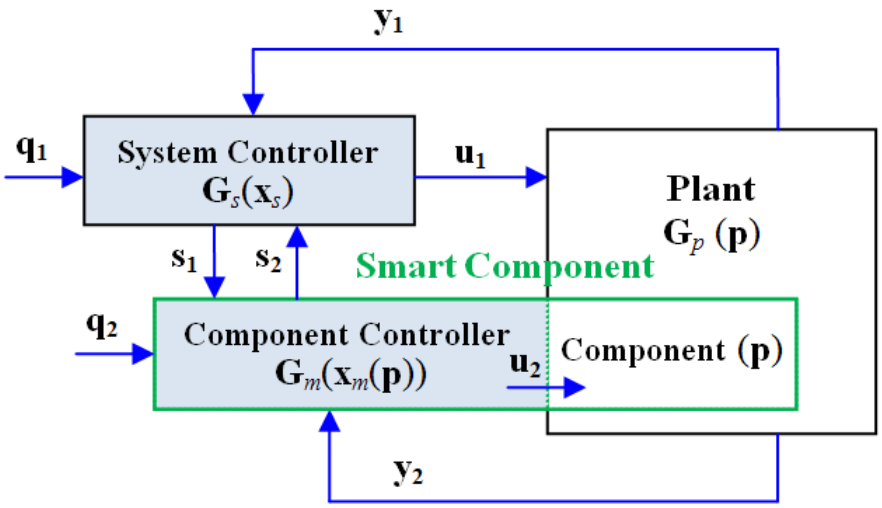


Figure 2.2: Diagram of the distributed controller.

We now introduce a method based on sensitivity analysis of the control signals with respect to the component parameters to guide the definition of the distributed controller architecture. Consider the optimal values of the centralized controller gains, $\mathbf{x}_c^*(\mathbf{p}) = [x_{c,1}^*(\mathbf{p}),\ x_{c,2}^*(\mathbf{p}),\ \ldots\ x_{c,n}^*(\mathbf{p})]$, where $\mathbf{p} \in P_P$, and assume that these values can be regressed (curve fitted) so that $\mathbf{x}_c^*(\mathbf{p})$ is defined for all $\mathbf{p} \in P_C$.

We define the normalized sensitivity of the control vector, **u**, with respect to the component parameter vector, **p**, through the controller gain, $x_{c,j}$, where $j \in \{1, 2, \ldots, n\}$, as,

$$S_n(x_{c,j}) = \frac{S(x_{c,j})}{\sum_{j=1}^{n} S(x_{c,j})} \tag{2.5}$$

$S(x_{c,j})$ is the sensitivity of the control vector, **u**, with respect to the component parameter vector, **p**, through the controller gain, $x_{c,j}$:

$$S(x_{c,j}) = \sum_{i=1}^{n_u} \sum_{k=1}^{n_p} \left| \frac{\partial u_i(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c)}{\partial x_{c,j}} \frac{\partial x_{c,j}^{*}(\mathbf{p})}{\partial p_k} \right| \tag{2.6}$$

The control signal $u_i(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c)$ is the i^{th} component of $\mathbf{u}(\mathbf{p}, \mathbf{q}, \mathbf{y}, \mathbf{x}_c)$ as in Figure 2.1, $p_k \in \mathbf{p}$ is a component parameter. The sensitivity of the control signals, $S(x_{c,j})$, through the controller gain, $x_{c,j}$, is defined using the chain rule. $S(x_{c,j})$ depends on the input signals to the controller, **q** and **y**, that are multiplied by the controller gains, and the sensitivity of the optimal values of the controller gains with respect to the component parameters. If each sensitivity of the control signals, $S(x_{c,j})$, for $j \in \{1, 2, \ldots, n\}$, is zero, then there is no need to distribute the controller, and the centralized controller can achieve CSM.

Here we use the same weight for each control signal and for each component parameter. For specific applications, different weights can be used depending on the magnitude of the control signals and the magnitude of the component parameter.

The sensitivity, $S(x_{c,j})$, depends on the values of **q**, **y** and **p**. In our case study, we used the mean value over the operating range for **q** and **y,** and middle-of-the-range values for **p**.

The normalized sensitivity, $S_n(x_{c,j})$, is used to define the controller distribution. The basic approach is to distribute the controller gains that result in the highest sensitivity of the control signals, along with the corresponding calculations, into the component controller. If the system performance requirements cannot be satisfied with such a distributed controller architecture, where the component controller is the simplest in terms of the number of the controller gains, more controller gains that results in relatively high sensitivity of the control signals, along with the corresponding calculations, may need to be distributed into the component controller. Examples can be found in Chapter V for battery CSM design in PHEVs.

2.3. The Direct Method

The sensitivity analysis in the preceding subsection is used to guide the distributed controller architecture decisions. In this section, we demonstrate how the distributed controller gains, i.e., the system controller gains, $\mathbf{x}_s$, and the component controller gains, $\mathbf{x}_m$, are determined.

First we consider an All-in-One optimization for the distributed controller gains. In this case, the cost function is the sum of the cost function for each component variant. The design variables are the system controller gains, $\mathbf{x}_s$, and the component controller gains, $\mathbf{x}_{m,i}$, which correspond to the component variant with parameters $\mathbf{p}_i$, where $i \in \{1, 2, \ldots, N\}$.

$$\min_{\mathbf{x}_s,\ \mathbf{x}_{m,1},\ \mathbf{x}_{m,2},\ \ldots,\ \mathbf{x}_{m,N}} \quad \sum_{i=1}^{N} (F(\mathbf{p}_i,\ \mathbf{x}_s,\ \mathbf{x}_{m,i}))^2$$

Subject to:

$$\mathbf{g}\ (\mathbf{p}_i,\ \mathbf{x}_s,\ \mathbf{x}_{m,i}) \leq \mathbf{0},$$

$$\mathbf{h}\ (\mathbf{p}_i,\ \mathbf{x}_s,\ \mathbf{x}_{m,i}) = \mathbf{0},\ i \in \{1, 2, \ldots, N\}.$$

where F, $\mathbf{g}$ and $\mathbf{h}$ are defined in equations (2.2), (2.3) and (2.4).

The All-in-One formulation is conceptually straightforward, but it is hard, if not impossible, to handle numerically, due to the large number of design variables and constraints. In order to facilitate the numerical solution, the All-in-One optimization problem is re-formulated as a bilevel optimization problem.

The bilevel optimization problem has two iterative stages, as illustrated in Figure 2.3. The master problem in the outer stage generates the system controller gains, $\mathbf{x}_s^*$, while the sub-problems in the inner stage, corresponding to each system configuration with

different component variant, generates the component controller gains, $\mathbf{x}_{m,i}^*$, for the system configuration with component parameter vector $\mathbf{p}_i$, where $i \in \{1, 2, \ldots, N\}$. At each iteration, the system controller gains generated by the outer stage optimization problem are fixed as parameters for each of the inner stage optimization problem. Consequently, the inner stage problems are independent of each other and can be solved in parallel.

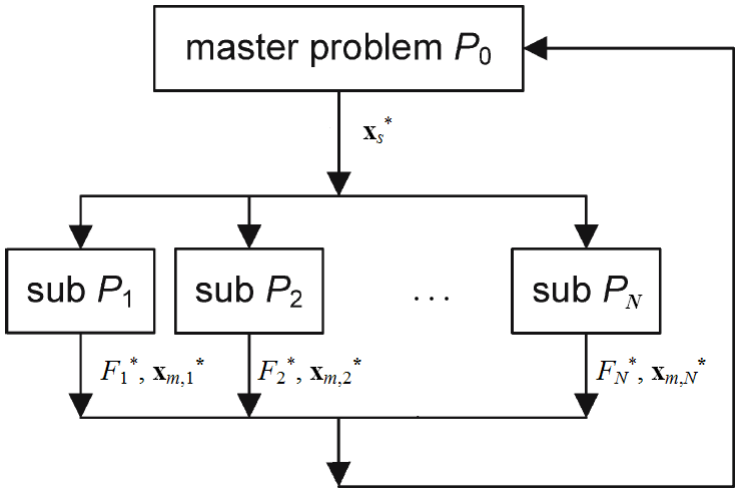


Figure 2.3: Flowchart of a bilevel optimization problem

The i^{th} inner stage optimization problem P_i searches for the optimal values of $\mathbf{x}_{m,i}$ that minimizes the cost function for the system configuration with component parameter vector $\mathbf{p}_i$, where $i \in \{1, 2, \ldots, N\}$.

$$F_i^* = \min_{\mathbf{x}_{m,i}} F(\mathbf{p}_i, \mathbf{x}_s, \mathbf{x}_{m,i})$$

Subject to:

$$\mathbf{g}(\mathbf{p}_i, \mathbf{x}_s, \mathbf{x}_{m,i}) \leq \mathbf{0},$$

$$\mathbf{h}(\mathbf{p}_i, \mathbf{x}_s, \mathbf{x}_{m,i}) = \mathbf{0}.$$

The outer stage optimization searches for the optimal values of $\mathbf{x}_s$ that minimizes the square of the 2-norm of the cost function values computed from the inner stage, without constraints.

$$\max_{\mathbf{x}_s} H(\mathbf{x}_s) = \| [F_1^*, F_2^*, ..., F_N^*] \|_2^2$$

The above general bilevel optimization formulation provides an approach to coordinating the design of the system controller gains and the component controller gains. Bilevel optimization problems are studied in Multidisciplinary Design Optimization [26, 53]. It was shown that the bilevel optimization of the kind we consider here converges locally at a super-linear rate assuming that the problem minimizer satisfies the strong linear independence constraint qualification condition [54]. A major difficulty with the general bilevel optimization approach is that the numerical optimization algorithm breaks down when one of the sub-problems in the inner stage is infeasible at one of the master problem iterates. To overcome this difficulty, Braun [55] allows the global variables (the system controller gains in our case) to take different values with each of the sub-problems. Then, the global variables for all the problems in the inner stage are forced to converge to the same value by using penalty functions. In the sequel, the Augmented Lagrangian Decomposition method [28] is applied to reformulate and solve the bilevel optimization problem.

Specifically, we introduce auxiliary variables $\mathbf{x}_{s,i}$, to serve as local copies of the shared system controller gains, $\mathbf{x}_s$, for the system configuration with component parameter vector, $\mathbf{p}_i$, where $i \in \{1, 2, \ldots, N\}$. The design variables $\mathbf{x}_s$ and $\mathbf{x}_{s,i}$ are forced to be equal by the consistency constraint $\mathbf{c}: R^{(N+1)\cdot m_1} \mapsto R^{N \cdot m_1}$, which is defined as $\mathbf{c}(\mathbf{x}_s, \mathbf{x}_{s,1}, \mathbf{x}_{s,2}, ..., \mathbf{x}_{s,N}) = [\mathbf{c}_1^T, \mathbf{c}_2^T, ..., \mathbf{c}_N^T]^T = \mathbf{0}$, with $\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i}) = \mathbf{x}_s - \mathbf{x}_{s,i}$, where $\mathbf{c}_i \in R^{m_1}$ is the vector of

consistency constraints for the system configuration with the component parameter vector $\mathbf{p}_i$, and m_1 is the dimension of $\mathbf{x}_s$.

The consistency constraints are relaxed by an augmented Lagrangian penalty function $\phi : R^{N \cdot m_1} \mapsto R$.

$$\phi(\mathbf{c}) = \mathbf{v}^T \mathbf{c} + \left\| \mathbf{w} \circ \mathbf{c} \right\|_2^2 = \sum_{i=1}^{N} \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) \tag{2.7}$$

with the penalty function for each system configuration with component parameter vector $\mathbf{p}_i$, $\phi_i : R^{m_1} \mapsto R$, defined by:

$$\phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) = \mathbf{v}_i^T (\mathbf{x}_s - \mathbf{x}_{s,i}) + \left\| \mathbf{w}_i \circ (\mathbf{x}_s - \mathbf{x}_{s,i}) \right\|_2^2 \tag{2.8}$$

where: $\mathbf{v} = [\mathbf{v}_1^T, \mathbf{v}_2^T, ..., \mathbf{v}_N^T]^T \in R^{N \cdot m_1}$ is the vector of Lagrange multiplier estimates for the consistency constraints, and $\mathbf{w} = [\mathbf{w}_1^T, \mathbf{w}_2^T, ..., \mathbf{w}_N^T]^T \in R^{N \cdot m_1}$ is the vector of penalty weights, with $\mathbf{v}_i \in R^{m_1}$, $\mathbf{w}_i \in R^{m_1}$. The symbol $\circ$ represents the Hadamard product: an entry-wise product of two vectors, such that $a \circ b = [a_1, ..., a_n]^{\mathrm{T}} \circ [b_1, \ldots, b_n]^{\mathrm{T}} = [a_1 b_1, \ldots, a_n b_n]^{\mathrm{T}}$.

The outer stage optimization minimizes the penalty function with respect to the system controller gains, $\mathbf{x}_s$:

$$\phi(\mathbf{c}) = \sum_{i=1}^{N} \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) \tag{2.9}$$

The outer stage problem has no constraints, and it can be solved analytically:

$$\mathbf{x}_s^* = \arg \min_{\mathbf{x}_s} \phi(\mathbf{c}) = \frac{\sum_{i=1}^{N} (\mathbf{w}_i \circ \mathbf{w}_i \circ \mathbf{x}_{s,i}) - \frac{1}{2} \sum_{i=1}^{N} \mathbf{v}_i}{\sum_{i=1}^{N} (\mathbf{w}_i \circ \mathbf{w}_i)} \tag{2.10}$$

In the inner stage optimization for each system configuration with component parameter vector $\mathbf{p}_i$, where $i \in \{1, 2, \ldots, N\}$, the objective function includes two terms,

the cost function F as defined in equation (2.2), and the penalty function as defined in equation (2.8).

$$\min_{\mathbf{x}_{s,i},\mathbf{x}_{m,i}} \; F(\mathbf{p}_i, \mathbf{x}_{s,i}, \mathbf{x}_{m,i}) + \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i}))$$

Subject to:

$$\mathbf{g}\,(\mathbf{p}_i,\, \mathbf{x}_{s,i},\, \mathbf{x}_{m,i}) \le \mathbf{0},$$

$$\mathbf{h}\,(\mathbf{p}_i,\, \mathbf{x}_{s,i},\, \mathbf{x}_{m,i}) = \mathbf{0}.$$

The design variables for each inner stage optimization are the auxiliary variables, $\mathbf{x}_{s,i}$, and the component controller gains, $\mathbf{x}_{m,i}$. The relaxation error between $\mathbf{x}_{s,i}$ and $\mathbf{x}_s$ are driven to zero by the penalty function in the objective function.

The flowchart for the solution algorithm is illustrated in Figure 2.4. At each iteration of the outer stage problem, a new estimate of $\mathbf{x}_s$ is generated and each of the inner stage problems is solved using $\mathbf{x}_s$ as a parameter; the penalty weights $\mathbf{v}$ and $\mathbf{w}$ are updated using the method of multipliers [56] to gradually drive the penalty function to zero for the consistency of $\mathbf{x}_s$ and $\mathbf{x}_{s,i}$. This procedure is repeated until a feasible solution that satisfies the consistency constraints $\mathbf{c} < \boldsymbol{\varepsilon}$, for each inner stage optimization, is found, or until the maximum number of function evaluations is reached, where $\boldsymbol{\varepsilon}$ is the error tolerance.

A good initial guess is important for the numerical solution of this bilevel optimization problem. For our problem, such an initial guess can be easily generated using the optimal values of the centralized controller gains. The solver "fmincon" in MATLAB can be used to solve the inner stage nonlinear constrained optimization problems.

Once the distributed controller and the corresponding system performance over the distributed controller architecture for each system configuration with different component variant by the Direct Method are obtained, the CSM metric, M_C, can be calculated based

on the system performance requirements. Only the components that satisfy the defined system performance requirements under the specified constraints are considered to have CSM.

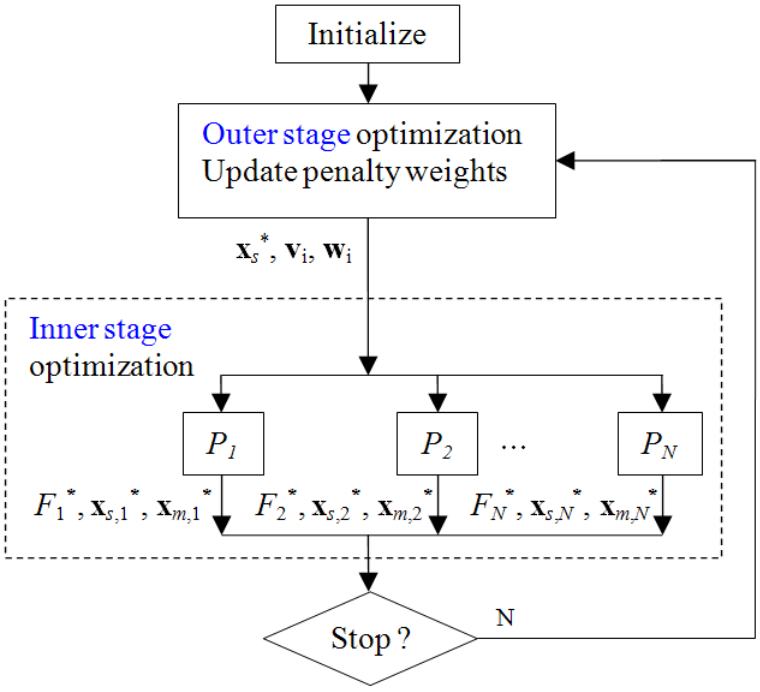


Figure 2.4: Flowchart of the solution algorithm for the bilevel optimization

2.4. Summary and Remarks

In this Chapter, the definition of CSM in control systems was introduced; the control system design for CSM was considered and the CSM metric was defined quantitatively to measure the degree of CSM. A method based on sensitivity analysis of the control signals with respect to the component parameters has been introduced to guide controller distribution between the system level controller and the component level controller. The

Direct Method was introduced to generate the distributed controller gains with CSM by solving a bilevel optimization problem.

The bilevel optimization formulation in the Direct Method ensures that the system level controller gains are independent of the component hardware parameters, while the component controller gains can be dependent on the component hardware parameters. In this way, the change in the component can be accommodated by only recalibrating the relatively simple component controller that is built into the component module, without the redesign or recalibration of the system level controller. With this approach, we make the component a plug-and-play component, and each component variant meets the defined performance metric under specified constraints.

CHAPTER III

THROTTLE ACTUATOR CSM FOR ENGINE ISC

In this Chapter, a case study of distributed controller design to achieve throttle actuator component swapping modularity (CSM) from the perspective of engine Idle Speed Control (ISC) is presented.

The primary objective of an engine ISC system is to regulate the engine speed to a set-point despite torque disturbances, due to accessory loads (e.g., air conditioning, power steering, alternator, etc.) and due to engagement of the transmission. A typical ISC strategy includes a PID control for the air loop, a proportional feedback control for the spark loop, and several feed forward controls realizing compensations for accessory loads, engine temperature, ambient temperature and barometric pressure [57]. Approaches to ISC based on modern control theory have also been considered, including LQ based optimization [58], H_∞ control [59], and l_1 control [60]. Additional ISC improvements are made possible through the preview and feed forward control of known or measurable disturbances, such as, for example, air conditioning or power steering load. A design technique, which includes lead compensation, feed forward and a disturbance observer, is presented for ISC systems with minimal spark reserve levels in [61].

The focus of this study is to analyze the swapping modularity of the ISC design for the air path of the engine with respect to the throttle actuator time constant. Specifically, we seek to distribute a centralized ISC into a base controller and an actuator controller, where the actuator controller is built into the actuator component, such that the actuator change can be accommodated by only recalibrating the actuator controller, and the system performance meets the performance requirement under specified constraints. Both the 3-Step Method and the Direct Method are applied to this case study. In addition, we will demonstrate that bidirectional communication between the base controller and the actuator controller improves CSM compared to unidirectional communication.

For the ISC case study in this Chapter, the controller structure is assumed in transfer function forms. The controller distribution is based on order assumptions of the distributed controllers. All the potential controller distribution cases are considered to compare the new Direct Method with the previous 3-Step Method.

In what follows, first, the engine model and the throttle actuator model are given; the performance requirement for controller design and the distributed controller architecture are presented. Second, the 3-Sptep Method and the Direct Method are applied to design the distributed controller with actuator CSM. Finally, the two methods are compared, and conclusions are given.

3.1. System Description

The throttle actuator is modeled as a first order system with a time constant τ. The nominal value for the time constant is $\tau_0 = 0.05$, which may refer to the original choice of the throttle actuator. The actuator is simplified to have unit DC gain. A nonzero DC gain

can be easily accommodated by the controller. The actuator transfer function from the reference throttle position (deg) to actual throttle position (deg) is as follows,

$$G_a = \frac{1}{\tau s + 1} \tag{3.1}$$

A Ford F-150 engine model [9] linearized around an idle speed operating point with the nominal throttle position, load torque and engine speed set, respectively, as $u_{th,0}$ = 3.15 (deg), M_L = 31.15 (Nm) and N = 800 (rpm), is used to obtain the engine transfer function from the deviation in the throttle position (deg) to the deviation in engine speed (rpm) ,

$$G_e = \frac{572.2997}{s^2 + 1.545s + 2.228} e^{-t_d s} \tag{3.2}$$

The engine transfer function from the deviation in the disturbance torque (Nm) to the deviation in the engine speed (rpm) is given as,

$$G_t = \frac{-37.04s - 57.22}{s^2 + 1.545s + 2.228} \tag{3.3}$$

The delay t_d is between the intake stroke of the engine and torque production, and corresponds approximately to 360 degree of crankshaft revolution. Consequently, it is calculated as,

$$t_d = \frac{60}{N} \approx 0.075 \text{ (sec)} \tag{3.4}$$

A first order Padé approximation of the delay $t_d = 0.075$ is applied,

$$e^{-t_d s} = \frac{e^{-s\frac{t_d}{2}}}{e^{s\frac{t_d}{2}}} \approx \frac{1 - \frac{t_d}{2}s}{1 + \frac{t_d}{2}s} = \frac{1 - 0.0375s}{1 + 0.0375s} \tag{3.5}$$

With this approximation, a pole-zero pair is added to the delay-free transfer function, thereby permitting the resulting plant model to be treated with conventional control methods.

The closed-loop system is illustrated in Figure 3.1. In the Figure, G_c is the centralized controller, r is the reference engine speed, u is the control signal to the throttle actuator, y is the actual engine speed, and d is the disturbance torque.

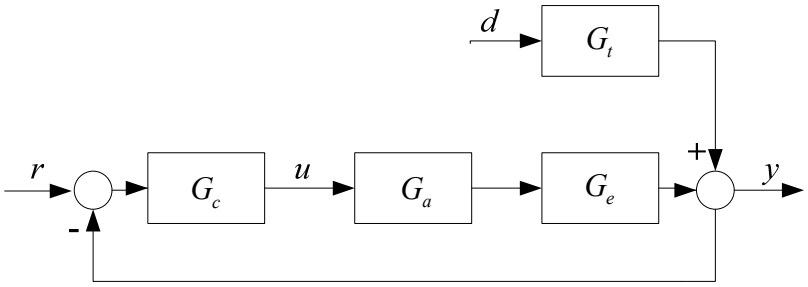


Figure 3.1: Engine ISC closed loop system.

The performance requirement for controller design is that the engine speed converges to the set-point within a specified time despite torque disturbances, under constraints on the magnitude of the control signal (in response to prescribed disturbance torque), and on the maximum excursion of engine speed from the set-point. To be specific, a step torque disturbance $d(t) = 10$ (Nm) is applied at time 0, while the set-point for the engine speed deviation is maintained at $r(t) = 0$ (rpm). Note that the engine model is linearized around the idle speed operating point. At $r(t) = 0$ (rpm), the actual engine speed is 800 rpm. The engine speed converges to the set-point with settling time $t_s \leq t_{s,target} = 1.5$ (sec). The magnitude of the control signal $u^{step}(t)$ corresponding to a 10 (Nm) step torque

disturbance, is limited to the range $[u_{min}, u_{max}] = [-3, 13]$ (deg) around the linearization position of 3.15 (deg). The maximum excursion of engine speed, M_p, is constrained to ± 10% of the speed set-point value (i.e., 800 rpm).

Consider N (= 21) throttle actuators with parameter $\tau \in \{\tau_1, \tau_2, \ldots, \tau_N\} = \{0.01, 0.02, \ldots, 0.21\} \subset [0.01, 0.21]$ for actuator CSM design.

The centralized controller G_c as shown in Figure 3.1 is distributed into two parts, the base controller, $\mathbf{C}_{BC}(\mathbf{x}_{BC})$, with controller gains, $\mathbf{x}_{BC}$, and the actuator controller, $\mathbf{C}_A(\mathbf{x}_A(\tau))$, with controller gains, $\mathbf{x}_A$, where only the actuator controller gains, $\mathbf{x}_A$, are dependent on the actuator hardware parameter, τ.

Bidirectional communication is introduced between the base controller and the actuator controller. The signals on the communication network are illustrated in Figure 3.2, where all the dynamics of the actuator and the plant are grouped into $P(\tau)$. The input to the base controller is the error signal, e; the control signal from the actuator controller is, u; and u_{ca} and y_{ac} denote the signals on the bidirectional communication network between the base controller and the actuator controller.

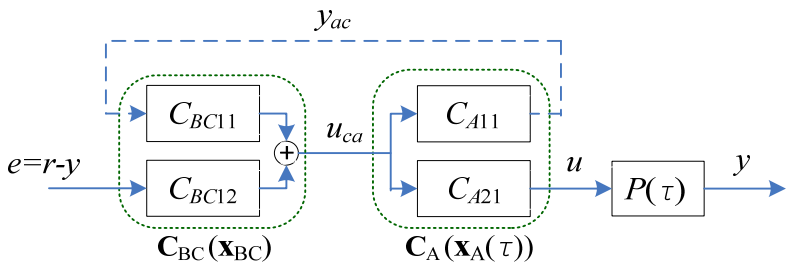


Figure 3.2: Bidirectional communication between the base controller and the actuator controller.

With bidirectional communication, not only can the base controller, $\mathbf{C}_{\mathrm{BC}}(\mathbf{x}_{\mathrm{BC}})$, send signal u_{ca} to the actuator controller, but the actuator controller, $\mathbf{C}_{\mathrm{A}}(\mathbf{x}_{\mathrm{A}}(\tau))$, can also send back signal y_{ac} to the base controller. The base controller and the actuator controller become MIMO controllers and are defined as,

$$\mathbf{C}_{BC}(\mathbf{x}_{BC}) = \begin{bmatrix} C_{BC11} & C_{BC12} \end{bmatrix} \tag{3.6}$$

$$\mathbf{C}_{A}(\mathbf{x}_{A}) = \begin{bmatrix} C_{A11} \\ C_{A21} \end{bmatrix} \tag{3.7}$$

By analyzing Figure 3.2 and using the notation presented in equations (3.6)-(3.7), the equations representing individual signals are

$$u_{ca} = C_{BC11} y_{ac} + C_{BC12} e \tag{3.8}$$

$$y_{ac} = C_{A11} u_{ca} \tag{3.9}$$

$$u = C_{A21} u_{ca} \tag{3.10}$$

Equations (3.8)-(3.10) can be rewritten in matrix form as

$$\begin{bmatrix} u_{ca} \\ y_{ac} \end{bmatrix} = \begin{bmatrix} 1 & -C_{BC11} \\ -C_{A11} & 1 \end{bmatrix}^{-1} \begin{bmatrix} C_{BC12} \\ 0 \end{bmatrix} e \tag{3.11}$$

$$u = \begin{bmatrix} C_{A21} & 0 \end{bmatrix} \begin{bmatrix} u_{ca} \\ y_{ac} \end{bmatrix} \tag{3.12}$$

Therefore, the equivalent overall distributed controller with e as input, u as output is given by:

$$G_{c,\,dis}(\tau, \mathbf{x}_{BC}, \mathbf{x}_{A}(\tau)) = \begin{bmatrix} C_{A21} & 0 \end{bmatrix} \begin{bmatrix} 1 & -C_{BC11} \\ -C_{A11} & 1 \end{bmatrix}^{-1} \begin{bmatrix} C_{BC12} \\ 0 \end{bmatrix} = \frac{C_{A21} C_{BC12}}{1 - C_{BC11} C_{A11}} \tag{3.13}$$

In order to illustrate the advantage of bidirectional communication as shown in Figure 3.2, conventional unidirectional communication is also considered. The distributed

controller architecture with unidirectional communication is given in Figure 3.3. There is a one directional connection between the base controller and the actuator controller. Both the base controller and the actuator controller are single input single output systems.

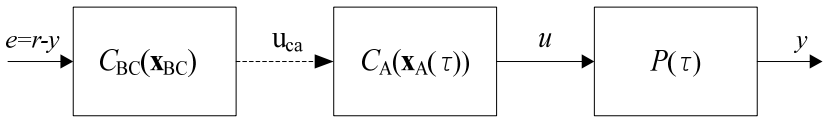


Figure 3.3: Distributed Controller architecture with unidirectional communication.

The overall distributed controller with e as the input and u as the output is then given by:

$$G_{c,\,dis}(\tau,\,\mathbf{x}_{BC},\,\mathbf{x}_A(\tau)) = C_{BC}(\mathbf{x}_{BC})C_A(\mathbf{x}_A(\tau)) \tag{3.14}$$

where $\mathbf{x}_{BC}$, $\mathbf{x}_A$ are controller gains of the distributed controllers C_{BC} and C_A respectively. Only $\mathbf{x}_A$ can depend on the actuator parameter, τ.

3.2. Design with the 3-Step Method

The 3-Step Method was developed in [24]. As its name suggests, this method includes three steps, the centralized controller design, controller distribution, and swapping modularity optimization. The main idea is to obtain the distributed controllers such that they match exactly, or approximately, certain pre-computed centralized controllers.

A. Centralized Controller Design

In order to arbitrarily assign the closed-loop pole locations, the centralized controller G_c is assumed to be fourth order, which equals the order of the plant (including the

throttle actuator and the engine model with delay approximation). The controller transfer function in general pole zero form is assumed as,

$$G_c = \frac{k(s+z_1)(s+z_2)(s+z_3)(s+z_4)}{s(s+p_1)(s+p_2)(s+p_3)} \tag{3.15}$$

The controller gains are represented using a vector $\mathbf{x}_c = [k, z_1, z_2, z_3, z_4, p_1, p_2, p_3]$. The centralized controller $G_c(\tau, \mathbf{x}_c(\tau))$ for each system configuration with actuator parameter, $\tau \in \{0.01, 0.02, \ldots, 0.21\}$ can be designed by an optimization formulation of the following form.

$$\min_{\mathbf{x}_c} t_s(\tau, \mathbf{x}_c(\tau))$$

Subject to:

g_1: $M_p - M_{p,\max} \le 0$

g_2: $u^{step}(t) - u_{\max} \le 0$

g_3: $u_{\min} - u^{step}(t) \le 0$

$\mathbf{g}_4$: Real (poles($G_c(\tau, \mathbf{x}_c(\tau))$)) $\le \mathbf{0}$

$\mathbf{g}_5$: Real (zeros($G_c(\tau, \mathbf{x}_c(\tau))$)) + $\boldsymbol{\varepsilon}_1 \le \mathbf{0}$

$\mathbf{g}_6$: Real (poles ($G_{cl}(\tau, \mathbf{x}_c(\tau))$)) + $\boldsymbol{\varepsilon}_2 \le \mathbf{0}$

where t_s is the settling time of the engine speed; M_P is the maximum excursion of engine speed; $u^{step}(t)$ is the control input; $G_{cl}(\tau, \mathbf{x}_c(\tau))$ represents the closed loop system. The constraint g_1 is imposed to limit the maximum excursion of engine speed; g_2 and g_3 are introduced to limit the magnitude of the control input corresponding to the 10 (Nm) disturbance torque; $\mathbf{g}_4$ enforces non-positive real parts of the controller poles. The controller may have a pole at the origin for integral control; $\mathbf{g}_5$ enforces minimum phase

zeros for the controller; $\mathbf{g}_6$ enforces negative real parts of the closed loop poles for system stability, where $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2$ are vectors of small positive values for robustness.

In numerical implementation, the solver "fmincon" in MATLAB® is used to solve the optimization problem. For each system configuration with actuator parameter, $\tau \in \{0.01, 0.02, \ldots, 0.21\}$, and corresponding obtained optimal centralized controllers, the system responses are given in Figure 3.4. The 2% settling time for each configuration satisfies the design target, $t_s \leq t_{s,target} = 1.5$ (sec). The maximum excursion of the engine speed is within ± 80 rpm (± 10% of 800 rpm). The control input signals, which are plotted in Figure 3.5, is within [-3, 13] (deg). Note that the throttle position is linearized around 3.15 (deg), so the actual throttle position limit is [0.15, 16.15] (deg).

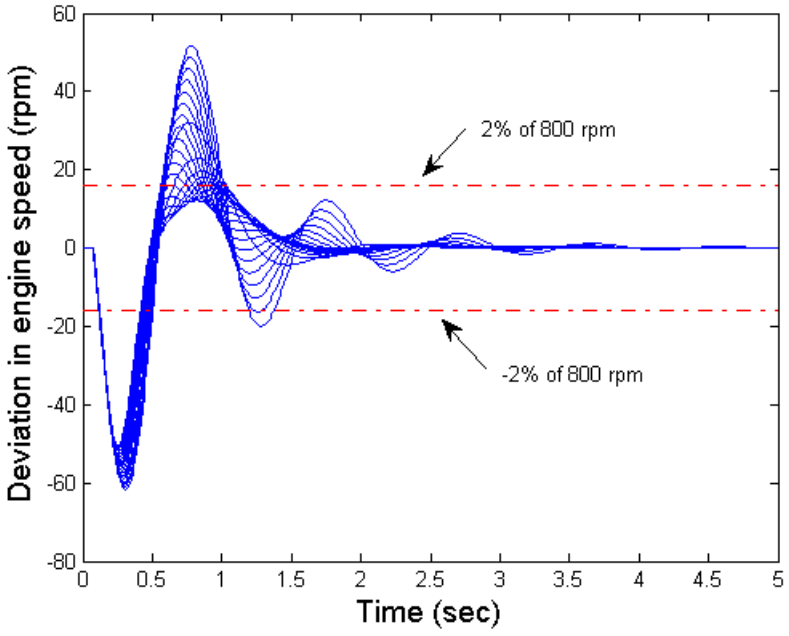


Figure 3.4: Closed-loop disturbance rejection responses for each actuator application with corresponding optimal centralized controller.

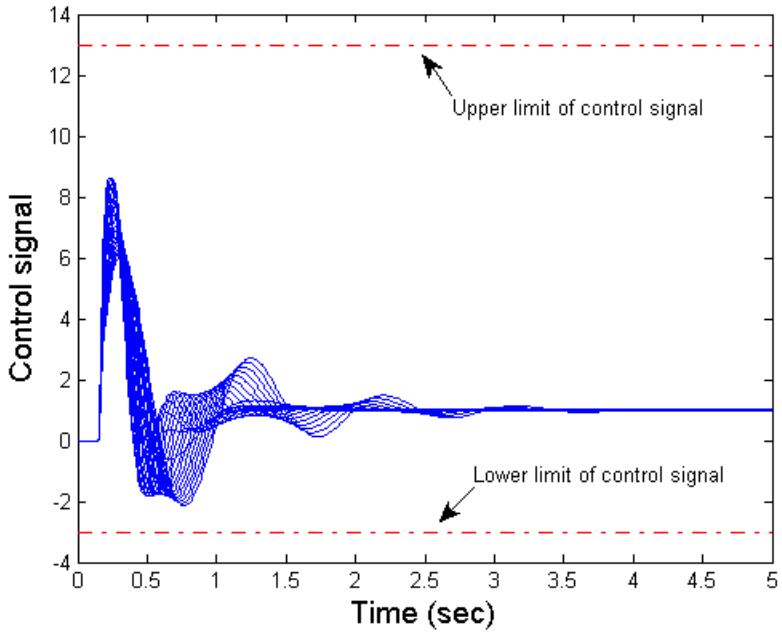


Figure 3.5: Control signals for each actuator application with corresponding optimal centralized controller.

To facilitate the existence of the swapping modularity solution, we employ a heuristic approach developed by [25]. This approach entails replacing an optimal centralized controller by an approximate optimal centralized controller using the following analysis: suppose that as τ varies away from nominal value of $\tau = \tau_0 = 0.05$, certain poles or zeros are essentially unchanged (specifically, suppose they remain within 5% of the nominal pole or zero magnitude for all or a range of τ, where the nominal poles and zeros are those which correspond to $\tau = \tau_0$). We then maintain such poles and zeros at the nominal values in the approximate optimal centralized controller.

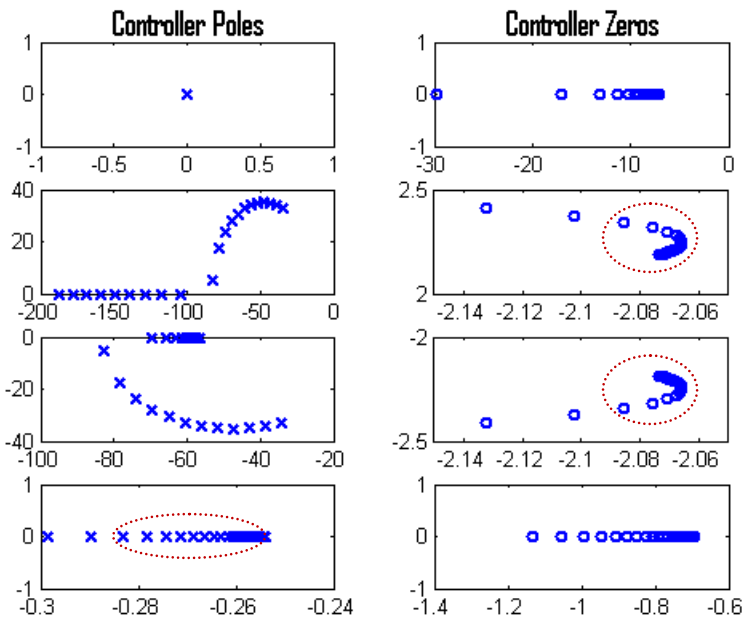


Figure 3.6: Poles and zeros of the optimal centralized controllers in the complex plane for each actuator application

The poles and zeros of the centralized controller for $\tau \in \{0.01, 0.02, \ldots, 0.21\}$ are plotted in Figure 3.6. We observe the pole $p = 0$ is essentially unchanged for $\tau \in \{0.01, 0.02, \ldots, 0.21\}$ the pole $p = -0.27$ is essentially unchanged for $\tau \in \{0.03, 0.04, \ldots, 0.18\}$ and the zeros $z_{1,2} = -2.07 \pm 2.3i$ are essentially unchanged for $\tau \in \{0.03, 0.04, \ldots, 0.21\}$. The poles and zeros of the approximate centralized controller for $\tau \in \{0.01, 0.02, \ldots, 0.21\}$ are plotted in Figure 3.7.

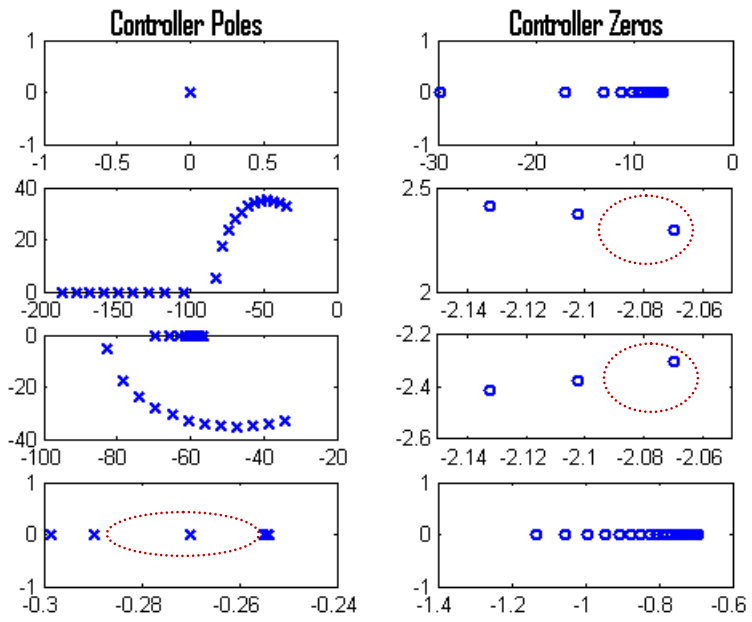


Figure 3.7: Poles and zeros of the approximate optimal centralized controllers in the complex plane for each actuator application.

The closed loop responses and the control input signals for the systems with different throttle actuator parameter $\tau \in \{0.01, 0.02, \ldots, 0.21\}$, and corresponding approximate optimal centralized controllers, are plotted in Figure 3.8 and 3.9. These responses are virtually indistinguishable from the closed-loop responses with the original optimal centralized controllers in Figure 3.4 and 3.5. Then we use the approximate optimal centralized controllers to obtain the distributed controllers in step 3 (i.e., optimization of CSM).

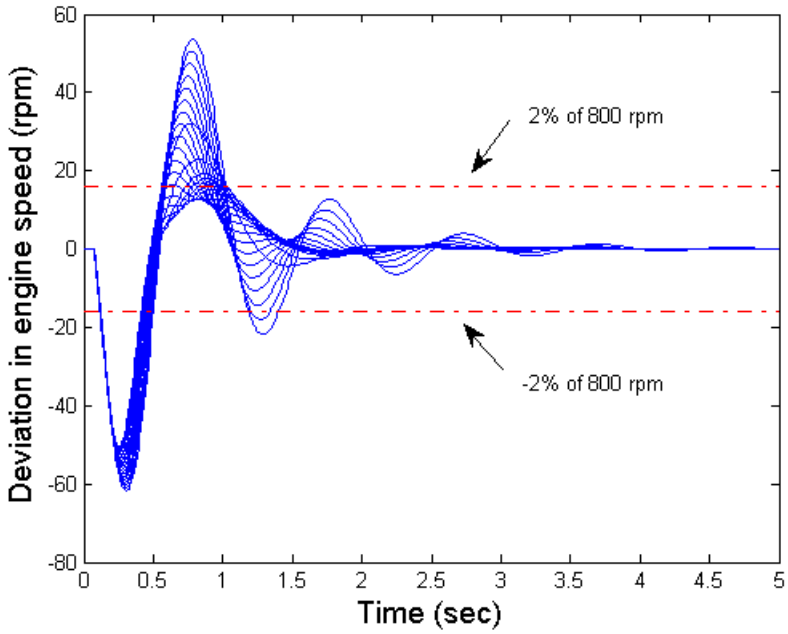


Figure 3.8: Closed-loop disturbance rejection responses for each actuator application with corresponding approximate optimal centralized controller.

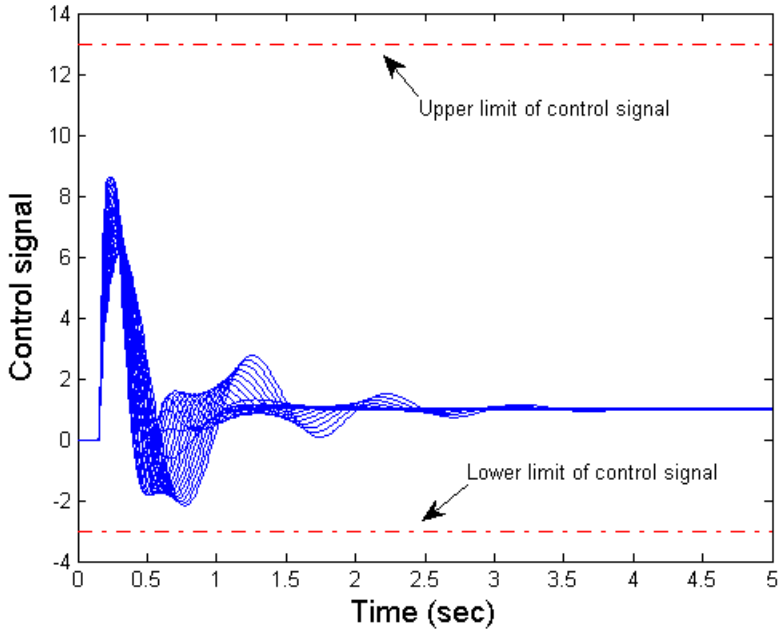


Figure 3.9: Control signals for each actuator application with corresponding approximate optimal centralized controller.

B. Order assumption of the distributed controller

Table 3.1 summarizes all the cases considered to determine the effect of the controller distribution, where the notation B*n*A*m* refers to a distributed controller with an n^{th} order base controller, and an m^{th} order actuator controller. In order to assign the closed-loop pole locations arbitrarily, we have $n+m = 4$. Note that in case B4A0, most of the control algorithm resides in the base controller, while the actuator controller is just gains. A simple actuator controller is desirable as it entails low computing effort as well as low recalibration effort when the actuator changes. However, the simplicity of the actuator controller may make it more difficult to achieve the required system performance, because only the actuator controller is changeable when the actuator changes.

Table 3.1: Case descriptions for controller distribution

Cases	B4A0	B3A1	B2A2	B1A3	B0A4
Order of $\mathbf{C}_{BC}$ ($\mathbf{x}_{BC}$)	4	3	2	1	0
Order of $\mathbf{C}_{A}$ ($\mathbf{x}_{A}$)	0	1	2	3	4

C. Optimization of CSM

The cost function is the CSM metric of the actuator, M_C, as defined in equation (2.1).

$$\max_{\mathbf{x}_{BC},\ \mathbf{x}_{A,1},\ \mathbf{x}_{A,2},\ \ldots,\ \mathbf{x}_{A,N}} \left\{ M_C = \frac{M-1}{N-1} \right\} \tag{3.18}$$

$$M = cardinality(P_C) \tag{3.19}$$

where $P_C = \{\tau_i \mid$ this optimization problem has a feasible solution for τ_i, $i \in \{1, 2, \ldots, N\}\}$,

The design variables are the controller gains of the base controller $\mathbf{x}_{BC}$, and the controller gains of the actuator controller $\mathbf{x}_{A,i}$, for each actuator parameter τ_i, $i \in \{1, 2, \ldots, N\}$. The constraints include: the equality constraints, as in equations $\mathbf{h}_1$, to match the optimal solution of the centralized controller (obtained in step 1) with the distributed controller, defined by equation (3.13) for bidirectional communication case, and defined by equation (3.14) for unidirectional communication case; stability and minimum phase requirements for the base controller and the actuator controller. The controllers may have a pole at the origin for integral control.

$\mathbf{h}_1$: $G_c^{\,*}(\tau_i, \mathbf{x}_c(\tau_i)) - G_{c,dis}(\tau_i, \mathbf{x}_{BC}, \mathbf{x}_{A,i}) = \mathbf{0}$

$\mathbf{g}_7$: Real(poles($\mathbf{C}_{BC}(\mathbf{x}_{BC})$)) $\leq \mathbf{0}$

$\mathbf{g}_8$: Real(zeros($\mathbf{C}_{BC}(\mathbf{x}_{BC})$)) + $\boldsymbol{\varepsilon}_3 \leq \mathbf{0}$

$\mathbf{g}_9$: Real(poles($\mathbf{C}_A(\mathbf{x}_{A,i})$)) $\leq \mathbf{0}$

$\mathbf{g}_{10}$: Real(zeros($\mathbf{C}_A(\mathbf{x}_{A,i})$)) + $\boldsymbol{\varepsilon}_4 \leq \mathbf{0}$

where $G_c^{\,*}(\tau_i, \mathbf{x}_c(\tau_i))$ is the optimal solution for the centralized controller for each actuator variant with τ_i, $i \in \{1, 2, \ldots, N\}$, and $\boldsymbol{\varepsilon}_3$ and $\boldsymbol{\varepsilon}_4$ are vectors of small positive values.

The actuator CSM metric obtained using the 3-Step Method for both the unidirectional communication case and the bidirectional communication case are given in Figure 3.10.

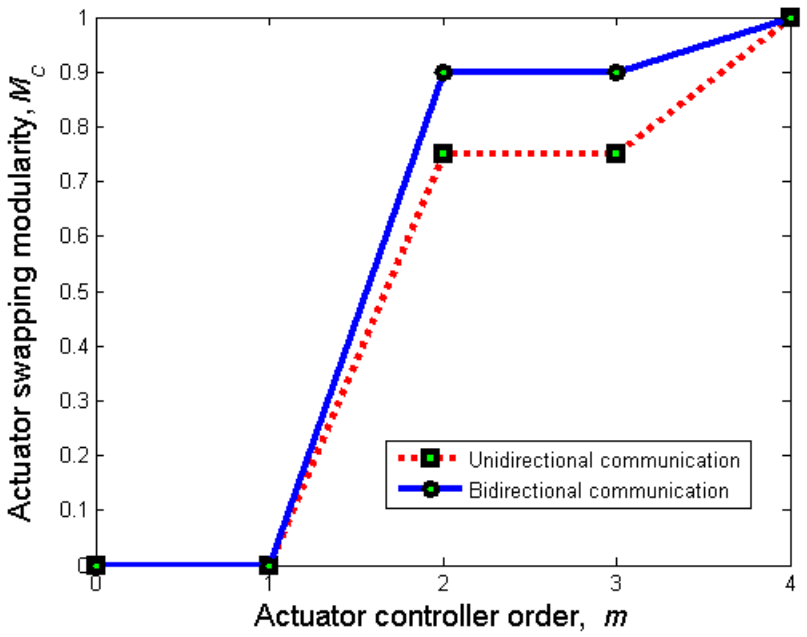


Figure 3.10: CSM metric values for the unidirectional communication case and the bidirectional communication case by the 3-Step Method.

We observe that the swapping modularity improves, as the order of the actuator controller increases. When the actuator controller is first order or just gains, no swapping modularity can be achieved by the 3-Step Method for both the unidirectional communication case and the bidirectional communication case. In other words, it is impossible to achieve the target system performance ($t_s \leq t_{s,target}$) by changing only the throttle actuator gains when the throttle actuator changes. The full range of swapping modularity (i.e., $M_C = 1$) is achieved only when the actuator controller is fourth order for both communication cases. This is reasonable, since in this case the controller is moved entirely to the actuator. The bidirectional communication improves the actuator CSM metric compared to the unidirectional communication when the actuator controller is second or third order.

The design results by the 3-Step Method have demonstrated the advantage of the bidirectional communication over the unidirectional communication for CSM. However, no CSM could be achieved by the 3-Step Method when the smart actuator comprises a simple built-in controller, which is a first order transfer function or just gains. This could be a deficiency when component computing power and cost are limited. From both computation and calibration perspectives, simple controllers are generally more amenable to implementation in the smart component.

3.3. Design with the Direct Method

The controller distribution cases as in Table 3.1 are all considered by the Direct Method. For each controller distribution case, the distributed controller gains with actuator CSM are obtained by solving a general bilevel optimization problem in the following form. The notations are the same as for the 3-Step Method in the preceding Section.

The i^{th} inner stage optimization P_i minimizes the settling time for the system configuration with actuator parameter τ_i, with respect to the actuator controller gains $x_{A,i}$, where $i \in \{1, 2, \ldots, N\}$, and $N = 21$. The design constraints for each inner stage optimization include: limit on engine speed excursion, limit on the magnitude of the control input signal, stability of the closed loop system and, stability and minimum phase of the actuator controller and the base controller. The controllers may have a pole at the origin for integral control. $\boldsymbol{\varepsilon_3}$ and $\boldsymbol{\varepsilon_4}$ are vectors of small positive values.

$$h_i^* = \min_{\mathbf{x}_{A,i}} t_s(\mathbf{x}_{BC}, \mathbf{x}_{A,i}, \tau_i)$$

Subject to:

q_1: $M_p - M_{p,\max} \le 0$

q_2: $u^{step}(t) - u_{\max} \le 0$

q_3: $u_{\min} - u^{step}(t) \le 0$

$\mathbf{q}_4$: Real(poles(G_{cl}($\mathbf{x}_{BC}$, $\mathbf{x}_{Ai}$, τ_i))) + $\boldsymbol{\varepsilon}_2 \le \mathbf{0}$

$\mathbf{q}_5$: Real(poles($\mathbf{C}_A$ ($\mathbf{x}_{A,i}$))) $\le \mathbf{0}$

$\mathbf{q}_6$: Real(zeros($\mathbf{C}_A$ ($\mathbf{x}_{A,i}$))) + $\boldsymbol{\varepsilon}_4 \le \mathbf{0}$

$\mathbf{q}_7$: Real (poles($\mathbf{C}_{BC}$ ($\mathbf{x}_{BC}$))) ≤ 0

$\mathbf{q}_8$: Real(zeros($\mathbf{C}_{BC}$ ($\mathbf{x}_{BC}$))) + $\boldsymbol{\varepsilon}_3 \le 0$

The outer stage optimization minimizes the square of the 2-norm of the objective function values computed from the inner stage, with respect to the base controller gains $\mathbf{x}_{BC}$, with no constraints.

$$\min_{\mathbf{x}_{BC}} H(\mathbf{x}_{BC}) = \| [h_1^*, h_2^*, \cdots, h_N^*] \|_2^2$$

A flowchart of the solution algorithm for this general bilevel optimization problem is illustrated in Figure 3.11. At each iteration, a new estimate of $\mathbf{x}_{BC}$ is generated and each of the inner stage problems is solved using $\mathbf{x}_{BC}$ as a parameter. This procedure is repeated until a feasible solution that satisfies the goal attainment, $t_s \le t_{s,target}$, for each inner stage optimization is found. It is possible that there does not exist a feasible solution for some $i \in \{1, 2, \ldots, N\}$, i.e., the constraints cannot be satisfied for the component with parameter $\mathbf{p}_i$. In this case, the inner stage optimization stops when the maximum number of function evaluations is reached.

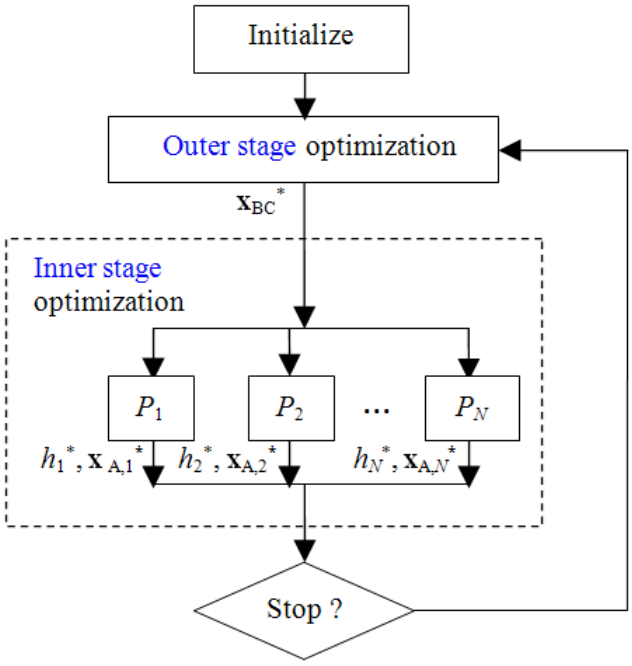


Figure 3.11: Flowchart for the bi-level optimization.

The bi-level formulation ensures that the resulting base controller gains $\mathbf{x}_{BC}$ are the same for each actuator variant, while the actuator controller gains $\mathbf{x}_A$ can be different for each actuator variant.

In numerical implementation, we use the solver "fmincon" and "fminsearch" in MATLAB® for the inner stage and the outer stage optimization problems, respectively. The solution and feasibility information of each inner stage optimization are recorded for CSM calculation. A good initial condition for the bilevel optimization can be obtained by distributing an optimal centralized controller for the nominal value of τ, $\tau_0 = 0.05$, using equations $\mathbf{h}_1$ in Section 3.2.

After solving the bilevel optimization problem, the CSM metric, M_C, can be calculated from equation (2.1), where M is calculated using equation (3.20).

$$M = cardinality(P_C) \tag{3.20}$$

where $P_C = \{\tau_i |$ this optimization problem has a feasible solution for τ_i, $i \in \{1, 2, \ldots, N\}$ and $t_s \leq t_{s,target}\}$.

Here we start with the case B3A1 as described in Table 3.1.

Figure 3.12 and 3.13 illustrate the closed-loop disturbance rejection responses and the control signals for the systems with different throttle actuators with parameter $\tau \in \{0.01, 0.02, \ldots, 0.21\}$ and corresponding B3A1 type distributed controllers obtained by the Direct Method.

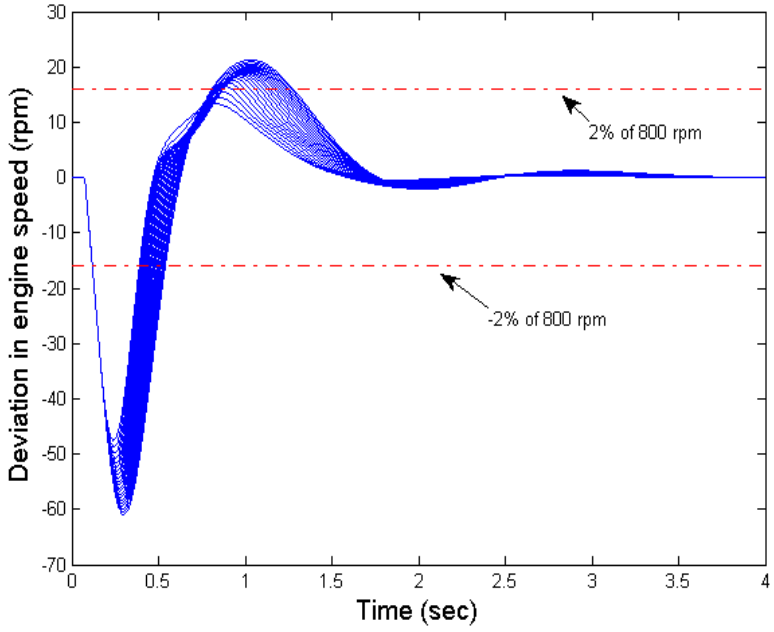


Figure 3.12: Closed-loop disturbance rejection responses for each actuator application with corresponding B3A1 type distributed controllers obtained by the Direct Method.

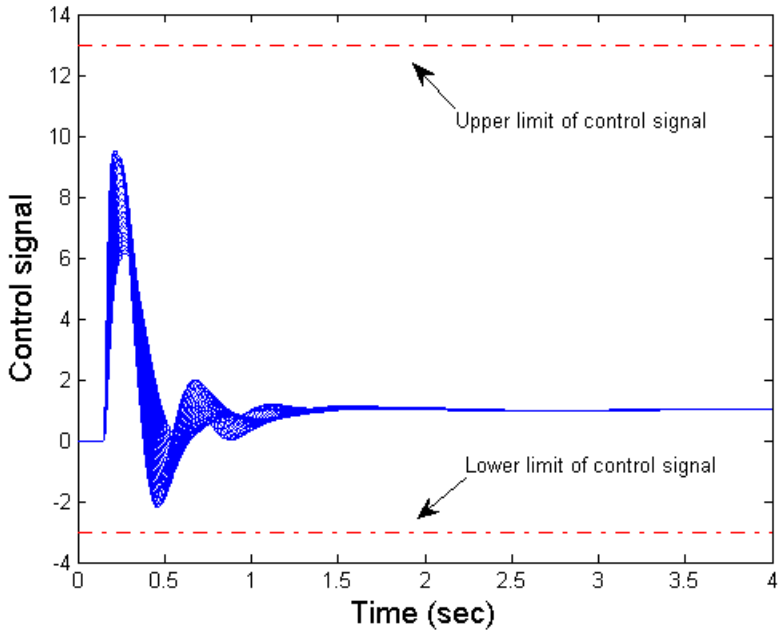


Figure 3.13: Control signals for each actuator application with corresponding B3A1 type distributed controllers obtained by the Direct Method.

The settling time for each configuration satisfies the design target, $t_s \leq t_{s,target} = 1.5$ (sec). In this case, when the throttle actuator changes, only the first order actuator controller needs to be recalibrated, but the third order base controller remains the same. So we can achieve the full range of CSM by only distributing a first order controller ($m = 1$) into the actuator module.

To see if we can further simplify the actuator controller, the case B4A0 is also considered, for the same design target, $t_s \leq t_{s,target} = 1.5$ (sec). Figure 3.14 and 3.15 illustrate the closed-loop disturbance rejection responses and the control signals for the systems with different throttle actuators with parameter $\tau \in \{0.01, 0.02, \ldots, 0.13\}$ and corresponding B4A0 type distributed controllers obtained by the Direct Method. The distributed controller gains, that satisfy the design target, exist only for $\tau \in \{0.01,$

0.02,…,0.13}. Comparing this result to the case B3A1, we see the tradeoff between the complexity of the actuator controller and the achievable CSM.

In the case B4A0, by only changing the actuator controller which is just gains ($m = 0$), actuator CSM can be achieved for $\tau \in \{0.01, 0.02,\ldots,0.13\}$ without redesign of the fourth order base controller. This saves a lot of recalibration effort when the actuator changes. A pure gain controller in the smart actuator is also very easy and cheap to implement.

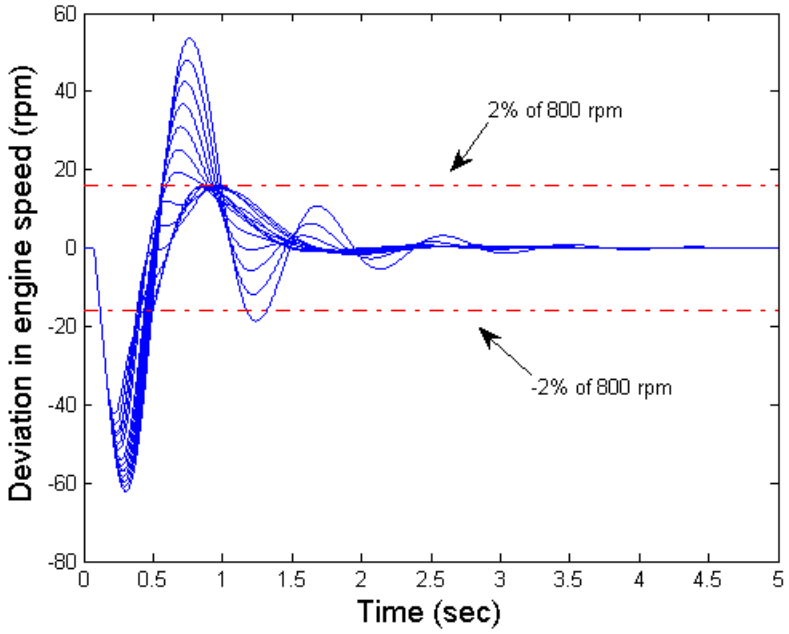


Figure 3.14: Closed-loop disturbance rejection responses for each actuator application with corresponding B4A0 type distributed controllers obtained by the Direct Method.

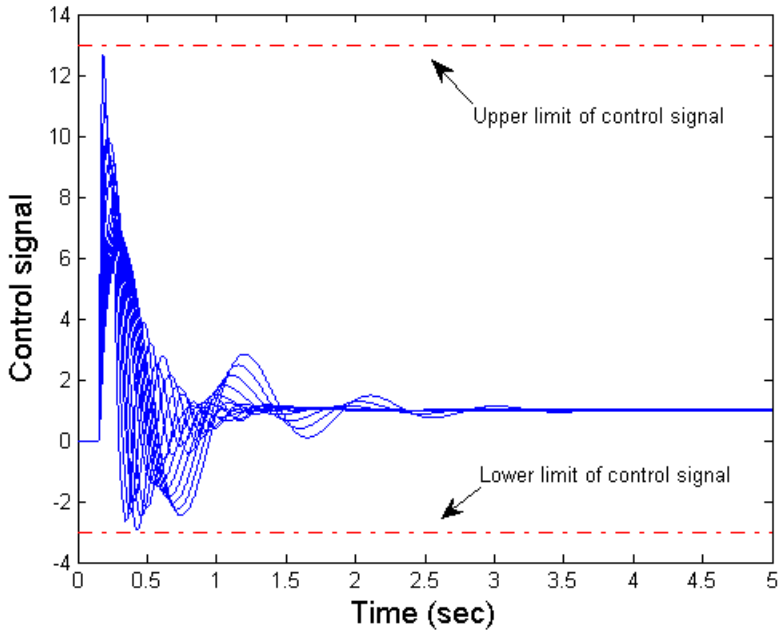


Figure 3.15: Control signals for each actuator application with corresponding B4A0 type distributed controllers obtained by the Direct Method.

Figure 3.16 compares the actuator CSM results using bidirectional communication network by the Direct Method and the 3-Step Method. The Direct Method provides equal or larger actuator swapping modularity compared to the 3-Step Method in all distribution cases considered. In the case B4A0, when the actuator controller is just gains, the Direct Method can provide partial range of CSM, $M_C = 0.6$. In the case B3A1, when the actuator controller is first order, the Direct Method achieves the full range of CSM, $M_C = 1$. In contrast, the 3-Step Method failed to achieve partial range CSM in either of these two cases. And full range of CSM can be achieved only when the actuator controller is fourth order. Therefore, the Direct Method improves CSM compared to the previous 3-Step Method.

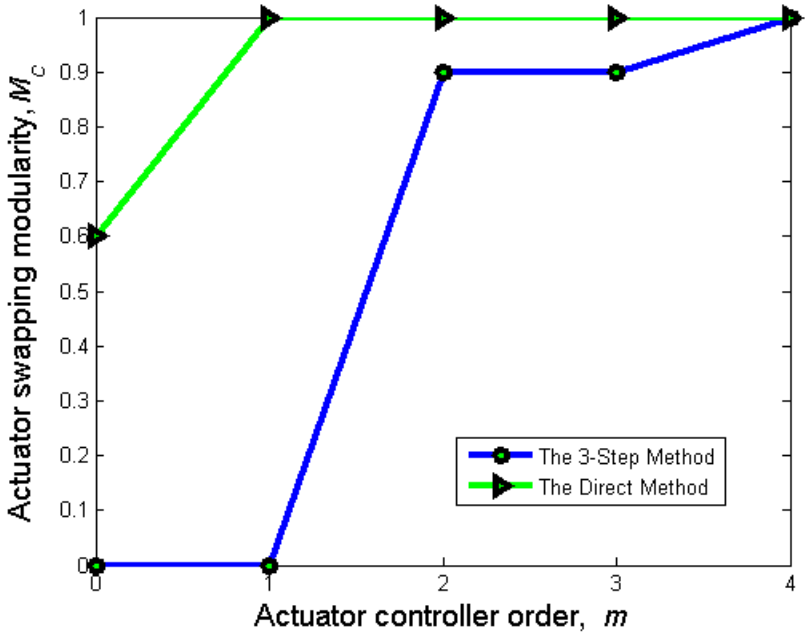


Figure 3.16: CSM metric values by the 3-Step Method and by the Direct Method.

Considering the case B4A0 further, if one relaxes the settling time requirement to $t_{s,target}$ = 4.1 (sec), feasible solutions for the distributed controller gains in a larger range, for $\tau \in \{0.01,0.02,\ldots,0.17\}$ can be achieved. See Figure 3.17 and 3.18 for the closed-loop disturbance rejection responses and the control signals for the systems with different throttle actuators with parameter $\tau \in \{0.01, 0.02, \ldots, 0.17\}$ and corresponding B4A0 type distributed controllers obtained by the Direct Method for the relaxed settling time target.

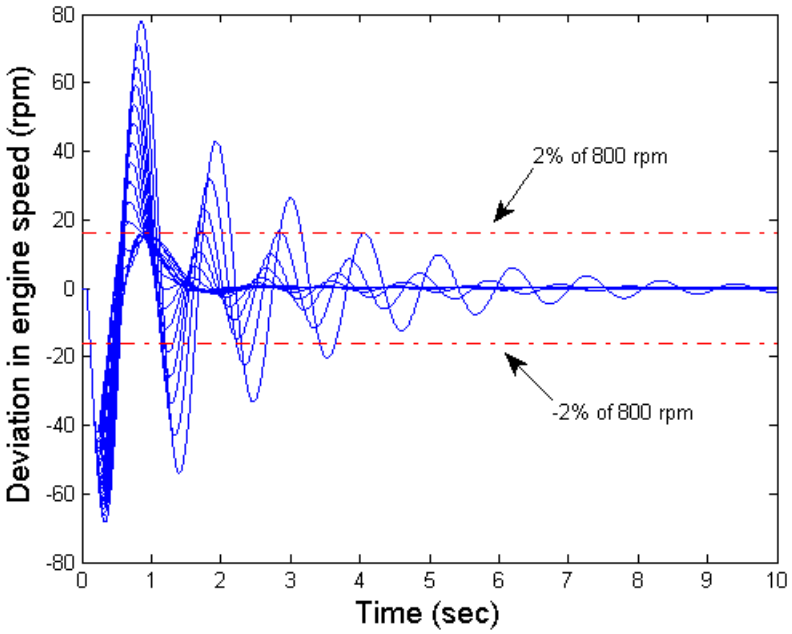


Figure 3.17: Closed-loop disturbance rejection responses for each actuator application with corresponding B4A0 type distributed controllers obtained by the Direct Method for the relaxed settling time target.

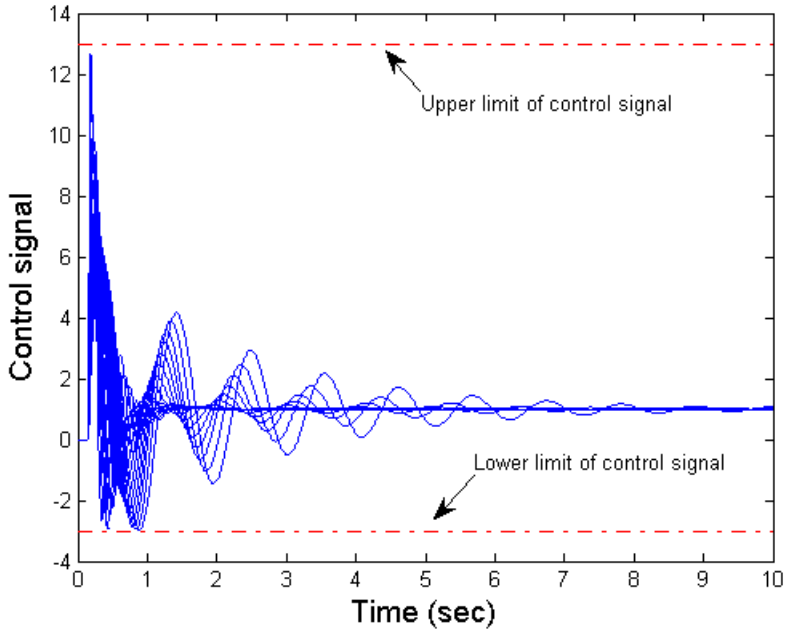


Figure 3.18: Control signals for each actuator application with corresponding B4A0 type distributed controllers obtained by the Direct Method for the relaxed settling time target.

By compromising some system performance, CSM can be improved. This shows another advantage of the Direct Method. The designer can make a tradeoff between the desired system performance and CSM when full range of swapping modularity is not achievable. As discussed above for case B4A0, if $t_{s,target}$ = 1.5 (sec), we obtain M_C = 0.6, but if $t_{s,target}$ = 4.1 (sec), we obtain M_C = 0.8. Thus for different system performance design targets, $t_{s,target}$, one obtains a different CSM metric, that corresponds to a different range of τ, which satisfies swapping modularity. Let the inverse of the settling time represent the system performance. The tradeoff between the two competing design objectives, system performance (i.e., $1/t_{s,target}$) and CSM, is illustrated in Figure 3.19. Using such curves, designers can balance desired system performance and CSM according to specific application scenarios.

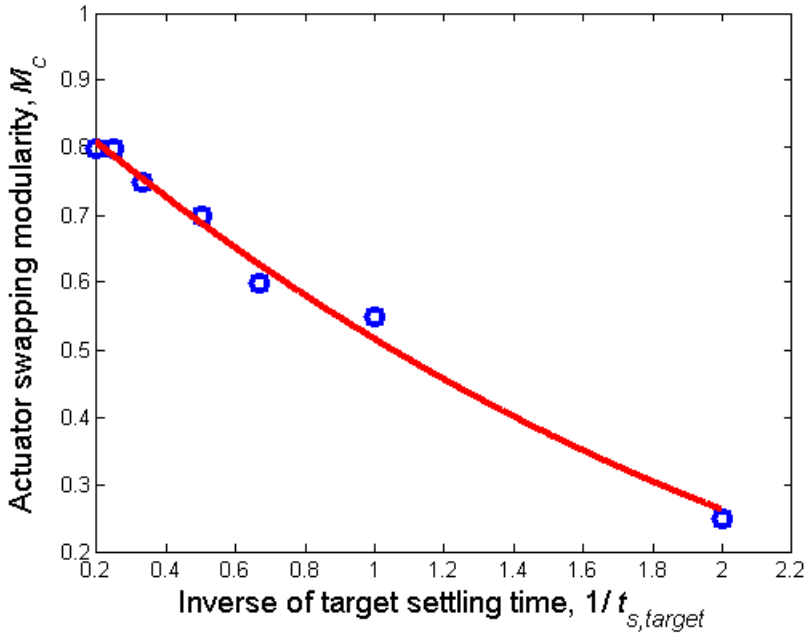


Figure 3.19: Illustration of the tradeoff between desired system performance and CSM for case B4A0.

3.4. Comparison of the 3-Step Method and the Direct Method

The 3-Step Method sequentially addresses the two design objectives, system performance and CSM, in two steps (step 1 and step 3), while the Direct Method combines the design for CSM and system performance into one bi-level optimization, and addresses the two design objectives simultaneously. For multi-objective optimization problem, a simultaneous approach generally delivers a solution which is at least as good as, or better than, that of a sequential approach [62]. Therefore, the Direct Method is expected to deliver better, or at least the same, results compared to the 3-Step Method. The results of throttle actuator CSM in engine ISC indicate that the Direct Method can improve CSM significantly compared to the 3-Step Method, as shown in Figure 3.16.

The 3-Step Method generates the distributed controllers by matching with certain pre-computed centralized controllers. The current method based on model matching of transfer functions is limited to the case of linear controller design. In contrast, the Direct Method relies on solving a nonlinear optimization problem to obtain the distributed controller gains directly. The nonlinearities of the controlled plant or the controller can be easily incorporated into the optimization formulation. Thus, it is a more general approach, which is applicable to the design of both linear and nonlinear controllers.

3.5. Summary

In this Chapter we examined two approaches to the controller design for CSM in the context of throttle actuator swapping modularity for engine idle speed control. Both the 3-Step Method and the Direct Method were implemented using nonlinear programming methods. The results demonstrate that: 1) bidirectional communication improves CSM compared to unidirectional communication by applying the 3-Step Method; 2) the Direct

Method provides improved CSM compared to the 3-Step Method. The 3-Step Method provides no swapping modularity for the case B3A1 or the case B4A0, and the actuator controller has to be fourth order to achieve the full range of swapping modularity. However, the Direct Method provides the full range of swapping modularity, $M_C = 1$, for the case B3A1. While for the case B4A0, it provides partial range of swapping modularity, $M_C = 0.6$; 3) unlike the 3-Step Method, the Direct Method permits one to simultaneously address and trade off the two design objectives, system performance and CSM.

CHAPTER IV

FEEDBACK BASED SUPERVISORY CONTROLLER FOR PHEV

Presently, supervisory controllers for PHEVs have centralized architectures, see [35]. A typical PHEV operates in a charge depleting (CD) or electric vehicle (EV) mode before the battery state of charge (SoC) decreases to a certain value, then it switches to a charge sustaining (CS) mode and operates like a conventional hybrid electric vehicle (HEV). The control strategies proposed for HEVs can be applied to the CS mode controller design for PHEVs. As reviewed in Chapter I, various control design methods for HEVs are available. For instance in [48], feedback controllers based on model predictive control have been experimentally evaluated and showed improved fuel economy compared to two baseline strategies.

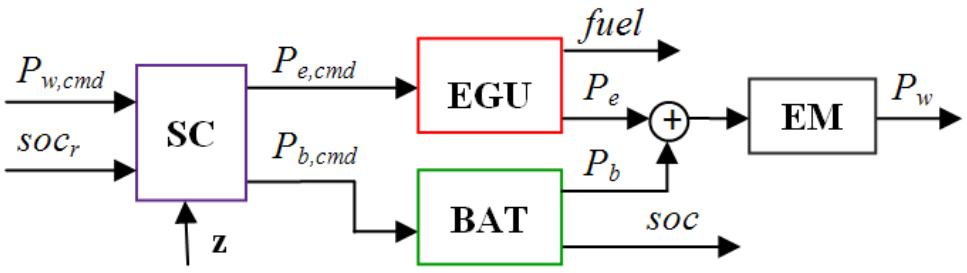
In this dissertation, we propose a novel feedback-based controller for the CS mode to facilitate distributed controller design for battery CSM. The controller is designed with respect to the EPA US06 cycle, but the simulation results demonstrate that the feedback based controller also achieves good fuel economy, good driving performance and charge sustainability over other driving cycles (e.g., the EPA UDDS and HWFET cycles).

In this Chapter, first, the control-oriented model of a PHEV used in this analysis is presented. The dynamics of the engine and the electric machines are much faster than that

of the battery. Thus, we use static models for the engine and the electric machines, while only the battery is modeled as a first order system with the battery SoC as the state. Second, the feedback-based centralized supervisory controller for the CS mode is introduced. The controller gains are obtained though optimization to achieve optimal fuel economy and optimal driving performance, while satisfying the constraints on closed loop system stability, battery charge sustainability and component reliability. Finally, the obtained controllers are evaluated over three standard driving cycles.

4.1. Vehicle Model

The control-oriented vehicle model is presented in this Section. The model inputs are the wheel power command, $P_{w,cmd}$, and the reference battery SoC, soc_r. The system outputs are the actual wheel power delivered, P_w, engine fuel consumption, *fuel*, and actual battery SoC, *soc*. A diagram of a series PHEV is shown in Figure 4.1.



SC – supervisory controller
EGU – internal combustion engine and generator unit
BAT – battery
EM – electric machine
z – feedback state vector

Figure 4.1: Diagram of the PHEV.

The supervisory controller generates the engine/generator power command and the battery power command. The wheels are driven by the electric motor. The battery is being charged, when the battery power P_b is negative.

We only consider the power flows in this system. Lower level controllers, which realize the power demand from the components, are not considered. We focus on a series HEV configuration due to its relevance to PHEVs and to a variety of other HEVs including fuel cell hybrids.

The component sizes of the nominal vehicle configuration used in this paper are listed in Table 4.1. This PHEV is representative of current designs, such as the 2011 Chevrolet Volt. In the sequel, the controller will be designed to enable CSM between four batteries with different energy capacity (and different all electric range capability) for the same vehicle, while delivering corresponding optimal fuel economy and driving performance.

Table 4.1: Nominal vehicle configuration

PHEV	All Electric Range (AER) (mile)	30
Engine	Engine Power (kW)	50
Generator	Generator Power (kW)	50
Battery	Battery Capacity (kWh)	12
	Battery Maximum Power (kW)	110
Motor	Motor Power (kW)	110
Vehicle	Vehicle Weight (kg)	1680

4.1.1. Engine

The engine is modeled using a static fuel consumption map from ADVISOR [63] as given in Figure 4.2.

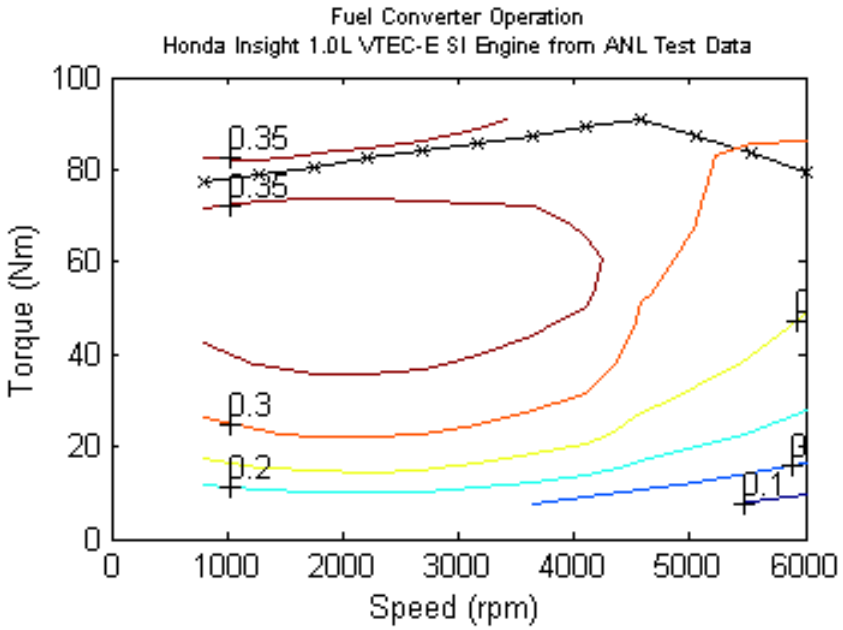


Figure 4.2: Engine fuel consumption map (g/W/h).

A combustion engine can achieve a required power (excluding the maximum power) with different combinations of torques and speeds. However, given a required power level, there is usually a unique pair of engine torque and speed which achieves minimum fuel consumption. The Optimal Operating Points Line (OOP-Line) can be defined as the curve on which the fuel consumption is minimized for each power level [47].

Using the data points of the fuel consumption map in Figure 4.2, the OOP-Line can be constructed as plotted in Figure 4.3. The blue dots represent the data points from the engine fuel consumption map, and the green line is the OOP-Line consisting of the most efficient points corresponding to each requested power level. In order for the engine to

operate along the OOP-Line, the engine should avoid large transients. This is based on the perspective that aggressive engine transients and engine operation away from the OOP Line may degrade fuel economy and emissions. The maximum rate of requested engine power output change can be constrained, e.g., to 3.5 kW/sec for a system considered in [47], or to 11 kW/sec for a system considered in [48]. If the rate of engine power output change is constrained, the engine can smoothly and efficiently operate along the OOP-Line responding to power commands.

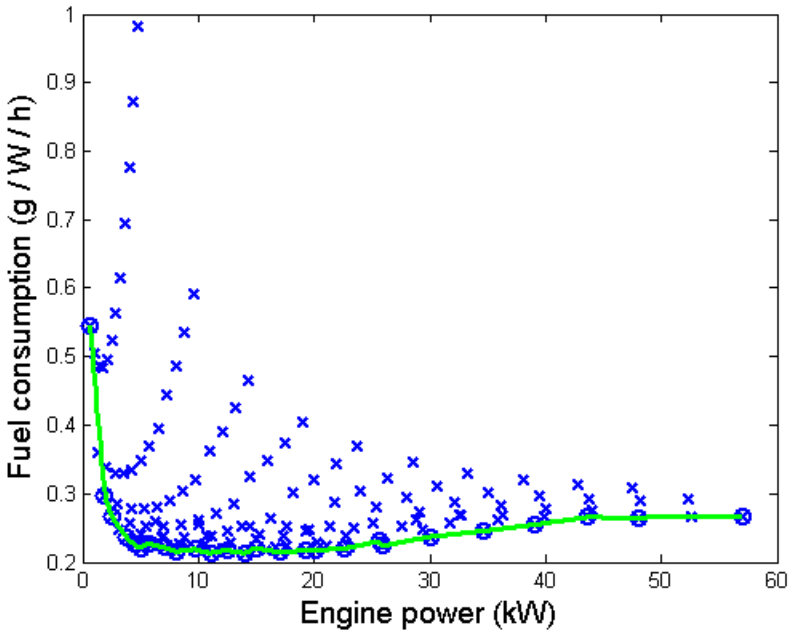


Figure 4.3: Fuel consumption v.s. engine power along engine OOP Line.

4.1.2. Battery

The battery efficiency is assumed to be $\eta_b = 0.9$. The battery is modeled as an integrator with parameter B_s [48].

$$\dot{\Delta soc} = -B_s P_{b,cmd} \tag{4.1}$$

where $\Delta soc = soc_r - soc$. The battery SoC is the only state of the powertrain system.

Here we consider four candidate batteries for CSM design. As a benchmark comparison, in the 2011 Chevrolet Volt, the energy capacity of the Lithium-ion battery is 16 kWh, which can provide roughly 40 mile all electric range (AER) with a battery weight of 175 kg. Four batteries with similar characteristics are scaled to have the parameters in Table 4.2.

Table 4.2: Battery parameters

#	Battery number	1	2	3	4
AER (mile)	All-electric range	60	45	30	15
E_b (kWh)	Battery energy capacity	24	18	12	6
B_s (e-5)	Battery parameter	1.29	1.71	2.57	5.14
W_b (kg)	Battery weight	263	197	131	66

4.1.3. Electric Machines

The motor and generator are modeled using a constant mean efficiency $\eta_m = \eta_g = 0.85$. This assumption can be easily relaxed by incorporating efficiency maps.

4.1.4. Vehicle Dynamics

The vehicle longitudinal dynamics includes the acceleration force, the rolling resistance force, and the aerodynamic force.

$$F = ma + fF_z + \frac{A\rho C_d v^2}{2} \tag{4.2}$$

where m is the vehicle mass, kg; a is the longitudinal acceleration, m/s^2; f is the rolling resistance coefficient; F_z is the normal force on the vehicle, N; A is the vehicle cross sectional area, m^2; ρ is the air density, kg/m^3; C_d is the drag coefficient; and v is the longitudinal velocity, m/s.

The vehicle longitudinal dynamics is used to calculate the wheel power command for the vehicle to follow the driving cycles that are specified with vehicle speed.

4.2. Feedback Based Supervisory Controller

Assume that the PHEV operates in two main modes, the charge depleting (CD) mode when battery SoC is larger than a certain reference value, soc_r, and the charge sustaining (CS) mode once the battery SoC reaches soc_r. Regenerative braking is activated when the wheel power command is negative. If the battery SoC goes outside the specified range [soc_{min}, soc_{max}], the priority of the control strategy is to drive the battery SoC back to the interval.

In the CD mode, the battery provides the propulsion energy. The engine is used to satisfy the transient load demand beyond the power capacity of the battery. In the CS mode, the control strategy is to optimize fuel economy and driving performance, while sustaining battery charge.

The CD mode and regenerative braking control are straight-forward, hence, we focus on designing the controller for the CS mode. First, a feedback-based control structure is proposed. Then the controller gains are obtained through optimization. In the optimization, the wheel power command is assumed to be non-negative and regenerative

braking is excluded. Finally, the obtained controller is tested in realistic simulations over three standard driving cycles, with negative wheel power command and regenerative braking included, to evaluate fuel economy, driving performance and battery charge sustainability.

4.2.1. Controller Structure

Two integrators are introduced to regulate battery SoC and to eliminate wheel power tracking error in steady-state. Recall that the powertrain system is modeled as first order with the battery SoC as the single state. The three states of the closed loop system are as follows,

$$z_1 = \Delta soc = soc_r - soc \tag{4.3}$$

$$z_2 = \int (\Delta soc)dt = \int (soc_r - soc)dt \tag{4.4}$$

$$z_3 = \int (P_{w,cmd} - P_w)dt \tag{4.5}$$

where, state z_1 represents the deviation of battery SoC, soc, from the reference value, soc_r; state z_2 represents the integral error of battery SoC; and state z_3 represents the integral error between power command, $P_{w,cmd}$, and actual power delivered, P_w. The actual power equals the engine power P_e and the battery power P_b.

The control algorithm includes state feedback control, feed-forward control and the terms representing information exchange between the engine power command $P_{e,cmd}$ and the battery power command $P_{b,cmd}$.

$$P_{e,cmd} = \mathbf{K_1}\,\mathbf{z} + n_1 P_{w,cmd} + k_e P_{b,cmd} \tag{4.6}$$

$$P_{b,cmd} = \mathbf{K}_2\,\mathbf{z} + n_2 P_{w,cmd} + k_b P_{e,cmd} \tag{4.7}$$

where $\mathbf{z} = [z_1\ z_2\ z_3]^T$ is the state vector; $\mathbf{K_1} = [k_1\ k_2\ k_3]$ and $\mathbf{K_2} = [k_4\ k_5\ k_6]$ are state feedback gain vectors; n_1 and n_2 are feed-forward gains; k_e and k_b are controller gains that represent the information exchange.

The regulation of battery SoC should be slow to allow the battery to augment the engine in transients, while the wheel power tracking should be fast and accurate for good driving performance and safety. In order to achieve different convergence rates for different states, we employ eigen-structure assignment [64] to decouple the state z_3 from the other two states, z_1 and z_2, which are related to battery SoC. Assume the desired closed loop poles are p_1, p_2 and p_3, we obtain equations (4.8a) - (4.8f) relating the controller gains and the desired poles:

$$k_1 = -\frac{(p_1 + p_2)\eta_b}{B_s \eta_g} \tag{4.8a}$$

$$k_2 = \frac{p_1 p_2 \eta_b}{B_s \eta_g} \tag{4.8b}$$

$$k_3 = -\frac{p_3}{\eta_m \eta_g} \tag{4.8c}$$

$$k_4 = -\frac{k_1 \eta_g}{\eta_b} \tag{4.8d}$$

$$k_5 = -\frac{k_2 \eta_g}{\eta_b} \tag{4.8e}$$

$$k_6 = 0 \tag{4.8f}$$

The six feedback gains are uniquely determined by the closed loop poles for a specific vehicle configuration with battery parameter B_s. Note that k_6 equals 0, k_4 and k_5

are determined by k_1 and k_2 respectively. Therefore, we have three independent feedback gains, k_1, k_2 and k_3 to optimize.

4.2.2. Optimization of the Controller Gains

The aggressive driving cycle of EPA US06 (see Figure 4.4) is chosen to generate the controller gains through optimization.

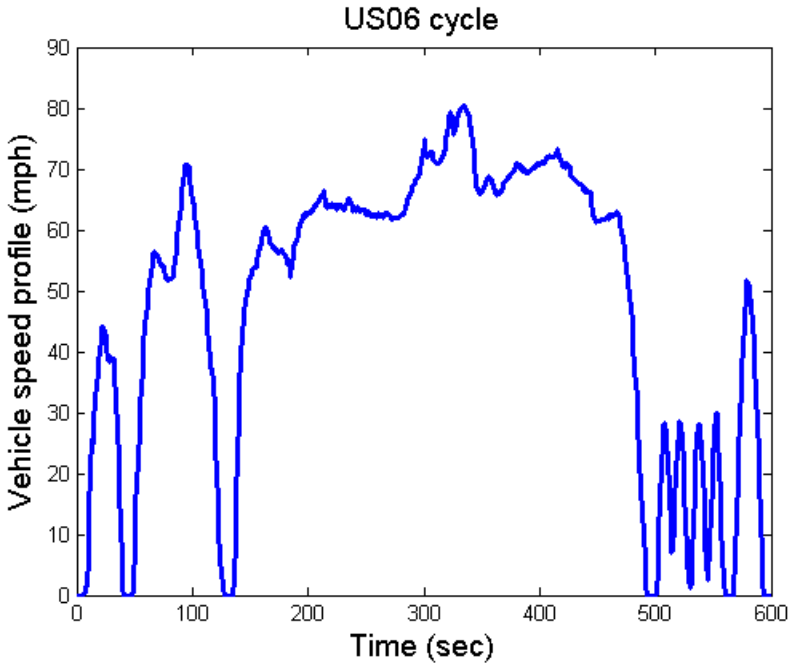


Figure 4.4: EPA US06 driving cycle.

The wheel power command for the vehicle to follow the US06 cycle is saturated to non-negative values and used for the controller gain optimization. The non-negative wheel power command for the vehicle with battery 3 (B_s = 2.57e-5) to follow the US06 cycle is plotted in Figure 4.5. The wheel power command for vehicles with the other

considered batteries to follow US06 cycle are similar to this figure, but a larger battery requires larger wheel power command due to larger battery weight.

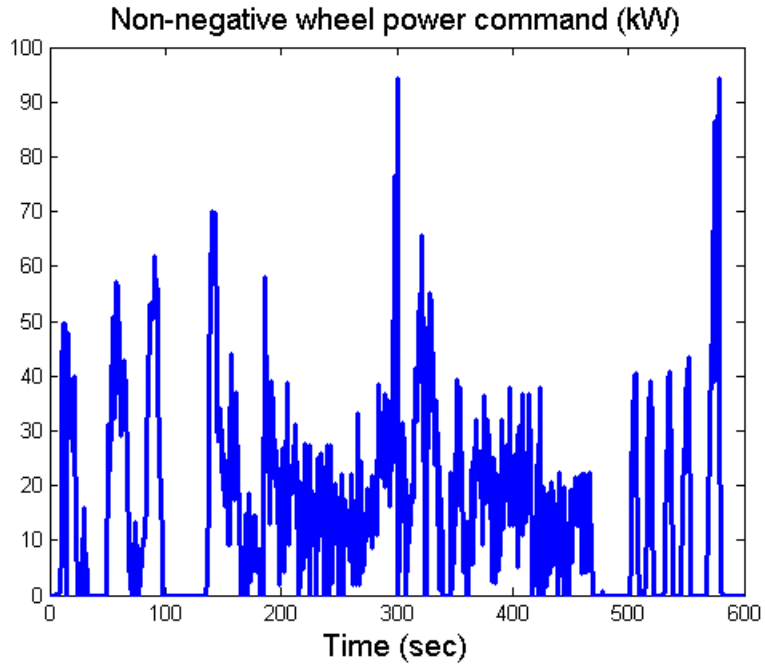


Figure 4.5: The saturated non-negative wheel power command for the vehicle with battery 3 (B_s = 2.57e-5) to follow the US06 cycle.

In order to check the battery charge sustainability, we set the initial battery SoC as the reference SoC, soc_r. The lowest value of soc_r is desirable since we would like to use the battery in the CD mode as much as possible. On the other side, soc_r should be large enough to satisfy the required battery energy availability for the CS mode. Thus, soc_r can be determined based on the energy needed for the CS mode. Since SoC varies between 0 and 1, a smaller battery needs to have a larger soc_r to satisfy the required battery energy availability for the CS mode. After determining the reference battery SoC and the amount of battery energy needed in the CS mode driving for the nominal vehicle configuration

(parameters illustrated in Table 4.1), the reference battery SoC for the other battery applications with different energy capacity can be scaled according to the nominal case. The reference battery SoC, soc_r, is chosen as {0.25, 0.27, 0.30, 0.40}, respectively, for the battery variant with parameter, B_s = {1.29e-5, 1.71 e-5, 2.57 e-5, 5.14 e-5}.

In the optimization for the controller gains, the cost function J includes three terms: engine fuel consumption, equivalent fuel consumption from the battery at the end of the driving cycle, and accumulated vehicle power tracking error.

$$J = \int_0^{T_f} \dot{m}_{ice}(t)dt + K_{eqf}(soc_r - soc_f) + \alpha \int_0^{T_f} \left| P_{w,cmd}(t) - P_w(t) \right| dt \tag{4.9}$$

where: $\dot{m}_{ice}(t)$ represents the engine fuel consumption; soc_f is the battery SoC at the end of the driving cycle; K_{eqf} is the equivalent fuel consumption factor from external charge [65]; and α is the penalty weight to drive the power tracking error to zero.

The controller gains are: k_1, k_2, k_3, k_4, k_5, k_6 , n_1, n_2, k_e and k_b. The feedback gains k_4, k_5 and k_6 are calculated from k_1, k_2 and k_3 using equations (4.8a) – (4.8f) from eigen-structure assignment. Thus, the design variables for the optimization are k_1, k_2, k_3, n_1, n_2, k_e and k_b.

The optimization constraints include: (1) stability of the closed loop system with the linear controller for the CS mode, which is enforced by the closed loop pole locations; (2) upper and lower power limits of the engine; (3) limit on engine power rate of change to smooth engine power during continuous engine operation; (4) upper and lower power limits of the battery; (5) upper and lower limits on battery SoC, $soc(t)$; (6) upper and lower bounds on battery SoC at the end of the driving cycle to enforce charge sustainability.

The controller gains for each battery application are optimized using the solver "fmincon" in MATLAB®, with the penalty weight α = 100, and the equivalent fuel consumption factor, K_{eqf}, as {747.96, 564.25, 375.44, 187.72} respectively, for each battery variant with parameter, B_s = {1.29e-5, 1.71 e-5, 2.57 e-5, 5.14 e-5}.

The vehicle performance for each battery application with corresponding optimal centralized controller is listed in Table 4.3. The fuel economy considering both fuel consumption from the engine and equivalent fuel consumption from the battery at the end of the driving cycle are evaluated in miles per gallon (MPG). The driving performance is evaluated using the maximum power tracking error, $err_{P,max}$, in kW. The battery charge sustaining performance is evaluated by the deviation of the final SoC from the reference SoC:

$$soc_{dev,p} = \frac{soc_f - soc_r}{soc_r} \cdot 100\% \tag{4.10}$$

From Table 4.3, the wheel power tracking error is very small, which ensures good driving performance. The deviation of battery SoC at the end of driving cycle is within ±2% as in the optimization constraints. We impose strict constraints on SoC deviation at the end of driving cycle, because we do not consider regenerative braking during the optimization. The SoC deviation at the end of driving cycle may be larger when regenerative braking is included. Actually regenerative braking can be considered as a disturbance to the feedback controlled system. Assume that the charge sustainability is satisfied if the SoC deviation at the end of driving cycle, $soc_{dev,p}$, is within ±10% when regenerative braking is considered.

Good fuel economy is also achieved for each battery application. Note that the fuel economy, MPG, is calculated considering both fuel consumption from the engine and equivalent fuel consumption from the battery at the end of the driving cycle.

In the simulation with initial battery SoC, $soc = soc_r$, a smaller battery with larger value of B_s provides higher fuel economy compared to that of a larger battery. This is due to the battery weight penalty as given in Table 4.2. Note that the four considered batteries are all large enough to meet the power demand, because the reference SoC is determined by the required battery energy availability. On the other hand, a larger battery with higher energy capacity can provide higher fuel economy through extended all electric range. Therefore, a larger battery will still provide larger overall fuel economy considering both all electric range and charge sustaining range [34].

Table 4.3: Vehicle performance for each battery application with corresponding optimal centralized controller to follow the saturated non-negative power command from the US06 cycle

B_s (e-5)	MPG	$err_{p,max}$	$soc_{dev,p}$
1.29	27.58	2.16e-7	-2%
1.71	28.17	8.10e-8	-2%
2.57	28.78	4.65e-8	-2%
5.14	29.40	3.61e-6	-2%

The SoC trajectories for each vehicle configuration, with the four batteries considered in Table 4.2 and corresponding optimal controllers, to follow the saturated non-negative

power command from the US06 cycle, are plotted in Figure 4.6. The initial battery SoC in the simulation is the reference SoC respectively for each battery application. We see that the obtained controller from the optimization problem tends to use the battery power as much as possible while satisfying the charge sustainability constraints.

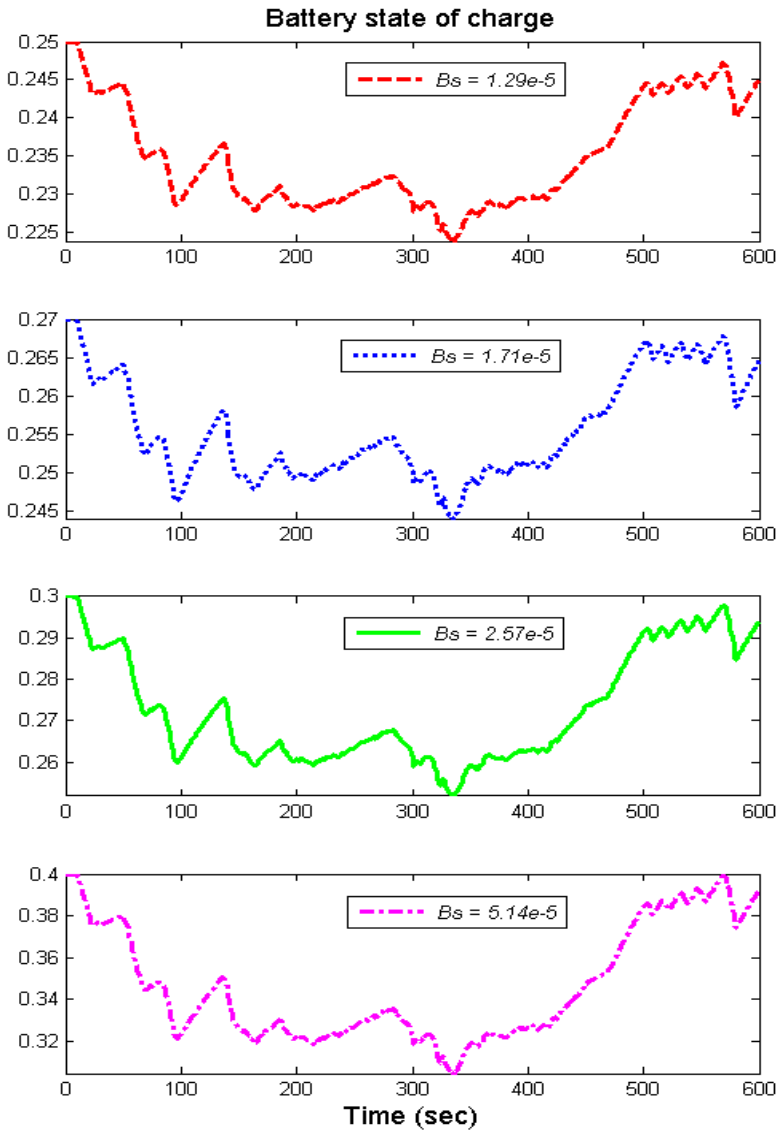


Figure 4.6: Battery SoC profiles for each battery application with corresponding optimal centralized controller to follow the saturated non-negative power command from the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) to follow the saturated non-negative power command from the US06 cycle is given in Figure 4.7. We see the engine provides the slowly changing power, which is roughly the moving average of the wheel power command, while the battery assists in handling the transients.

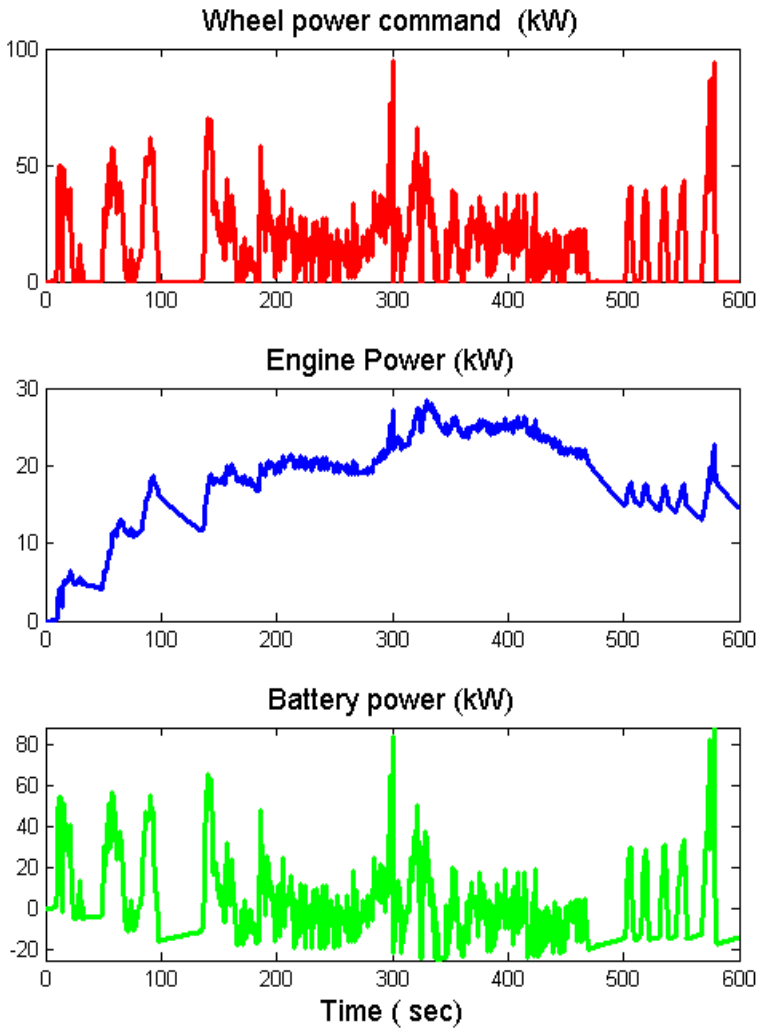


Figure 4.7: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal centralized controller to follow the saturated non-negative power command from the US06 cycle.

4.2.3. Controller Evaluation

The obtained controller is tested over three standard driving cycles to evaluate fuel economy, driving performance in terms of wheel power tracking error and battery charge sustainability. Now the original wheel power command is applied. Regenerative braking is activated when wheel power command is negative, and the engine is shut off without delivering power.

Although the controller gains are optimized over the EPA US06 driving cycle, the EPA UDDS driving cycle as given in Figure 4.8 and the EPA HWFET driving cycle as given in Figure 4.9 is also used to evaluate the proposed controller.

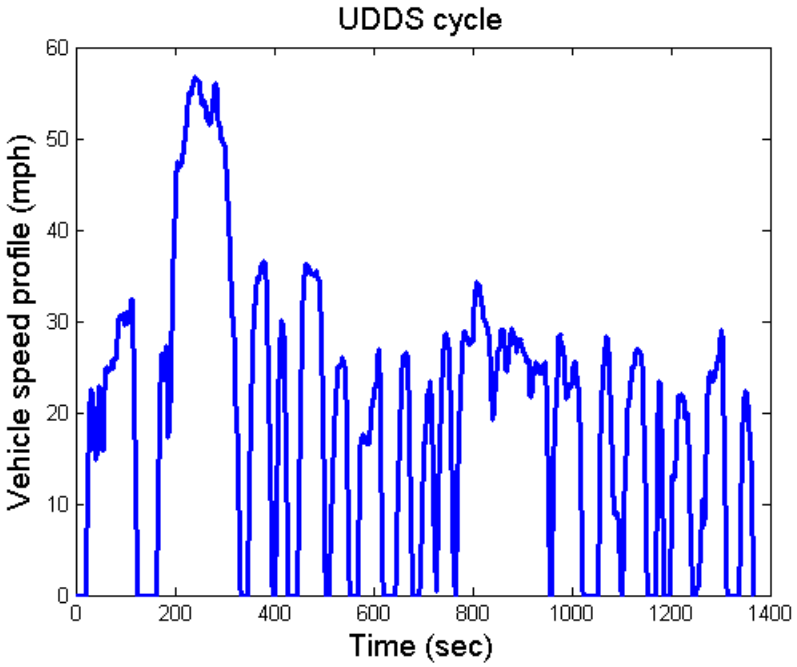


Figure 4.8: EPA UDDS driving cycle.

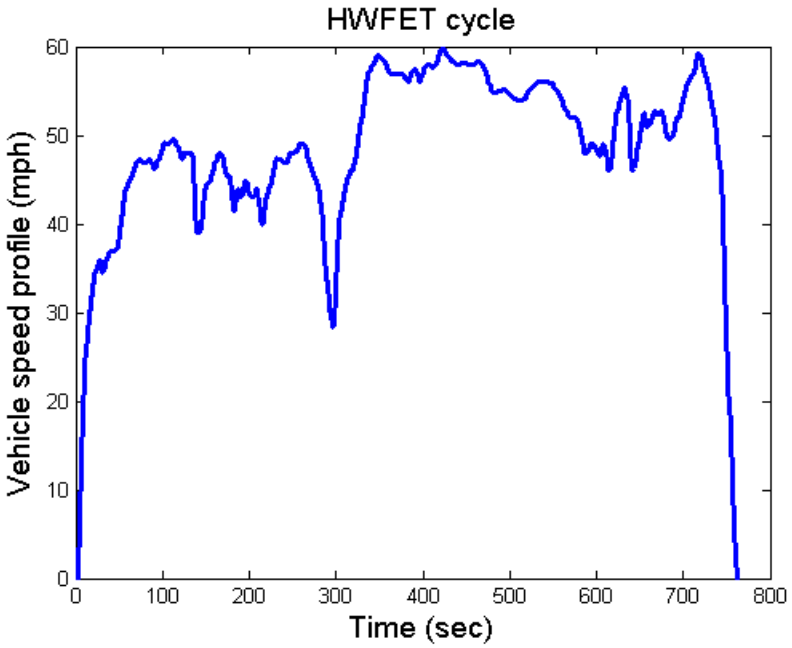


Figure 4.9: EPA HWFET driving cycle.

The vehicle performance for each battery variant with corresponding optimal centralized controller obtained in Section 4.2.2 to follow the three standard driving cycles are listed in Tables 4.4 – 4.7. The fuel economy in MPG considering both fuel consumption from the engine and equivalent fuel consumption from the battery at the end of the driving cycle, the driving performance evaluated by the maximum power tracking error, $err_{P,max}$, in kW, and the battery charge sustaining performance evaluated by the deviation of the final SoC from the reference SoC, $soc_{dev,p}$, are defined the same as in Table 4.3.

Table 4.4: Performance results for the vehicle with battery 1 (B_s = 1.29e-5) and corresponding optimal centralized controller

Driving Cycle	MPG	$err_{P,max}$	$soc_{dev,p}$
EPA US06	34.93	1.45e-6	-3.99%
EPA HWFET	53.78	3.24e-7	-0.87%
EPA UDDS	55.88	4.43e-7	-1.52%

Table 4.5: Performance results for the vehicle with battery 2 (B_s = 1.71e-5) and corresponding optimal centralized controller

Driving Cycle	MPG	$err_{P,max}$	$soc_{dev,p}$
EPA US06	34.92	1.04e-6	-3.29%
EPA HWFET	54.24	2.31e-7	-0.52%
EPA UDDS	56.47	3.20e-7	-1.25%

Table 4.6: Performance results for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal centralized controller

Driving Cycle	MPG	$err_{P,max}$	$soc_{dev,p}$
EPA US06	36.05	2.97e-8	-5.33%
EPA HWFET	55.19	4.67e-9	-0.79%
EPA UDDS	57.88	7.56e-9	-1.30%

Table 4.7: Performance results for the vehicle with battery 4 (B_s = 5.14e-5) and corresponding optimal centralized controller

Driving Cycle	MPG	$err_{P,max}$	$soc_{dev,p}$
EPA US06	36.73	2.15e-6	-7.41%
EPA HWFET	55.92	4.98e-7	-0.76%
EPA UDDS	59.12	7.28e-7	-1.57%

From Tables 4.4 – 4.7, the proposed controller provides good fuel economy for each battery application over different driving cycles. The wheel power tracking error, $err_{P,max}$, is less than 2.15e-6 kW for all simulation scenarios, which ensures good driving performance. The maximum deviation of battery SoC at the end of the driving cycles, $soc_{dev,p}$, is -7.41% for all simulation scenarios. For the mild EPA UDDS and HWFET cycles, the maximum $soc_{dev,p}$ for all battery applications is -1.52%.

For the US06 driving cycle, an example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal centralized controller obtained in Section 4.2.2 to follow the US06 cycle is given in Figure 4.10. When the wheel power command is positive, the engine provides the slowly changing power, while the battery provides the transient power command. When the wheel power command is negative, the engine is shut off without delivering power, and the battery is charged by regenerative braking.

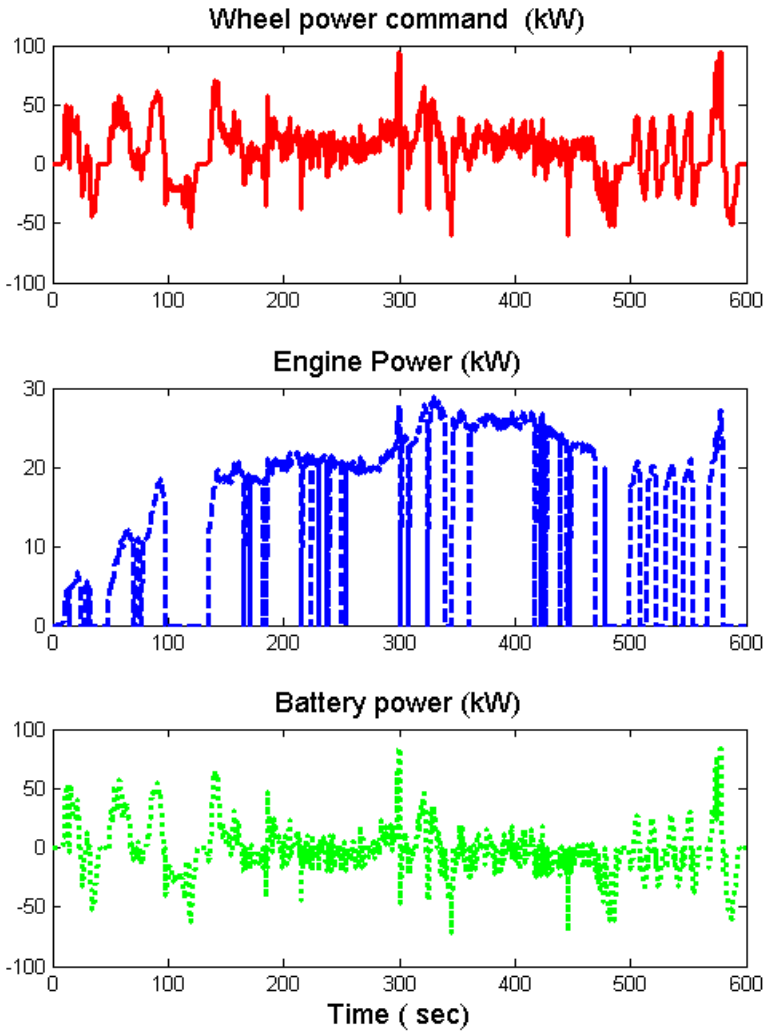


Figure 4.10: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal controller over the US06 cycle.

The SoC trajectories for each vehicle configuration with the four batteries and corresponding optimal centralized controllers obtained in Section 4.2.2 to follow the US06 cycle, are plotted in Figure 4.11. The initial battery SoC in the simulation is the reference SoC respectively for each battery application.

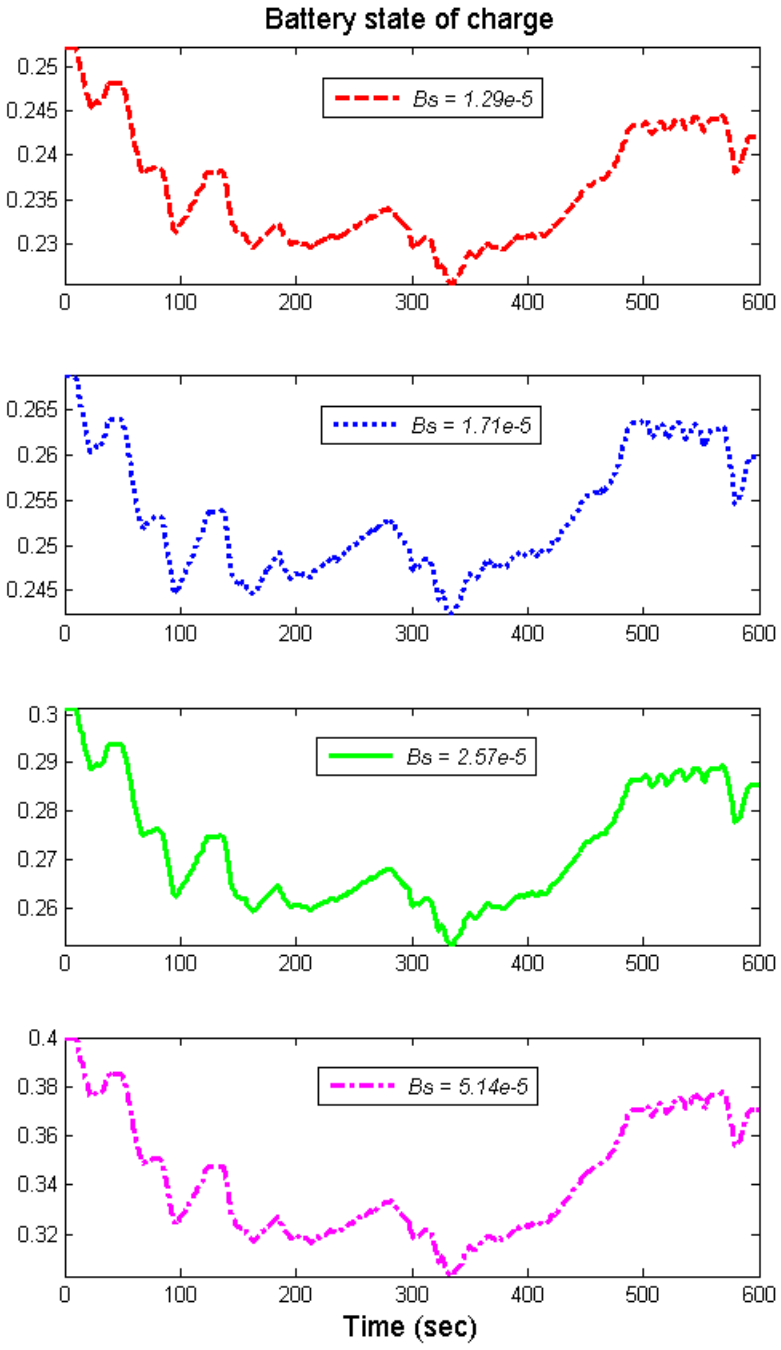


Figure 4.11: Battery SoC profiles for each vehicle configuration with different batteries and corresponding optimal controllers over the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal centralized controller obtained in section 4.2.2 to follow the UDDS cycle is shown in Figure 4.12.

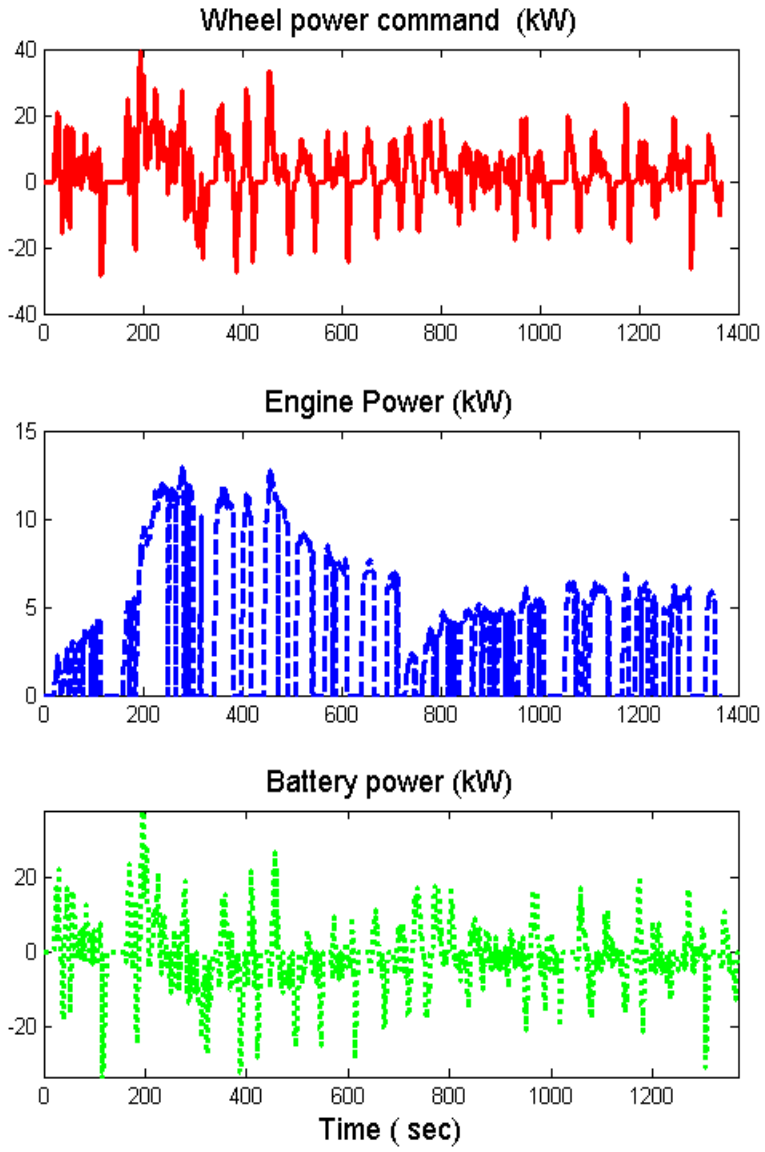


Figure 4.12: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal controller over the UDDS cycle.

The SoC trajectories for each vehicle configuration with the four batteries and corresponding optimal controllers obtained in Section 4.2.2 to follow the UDDS cycle are plotted in Figure 4.13.

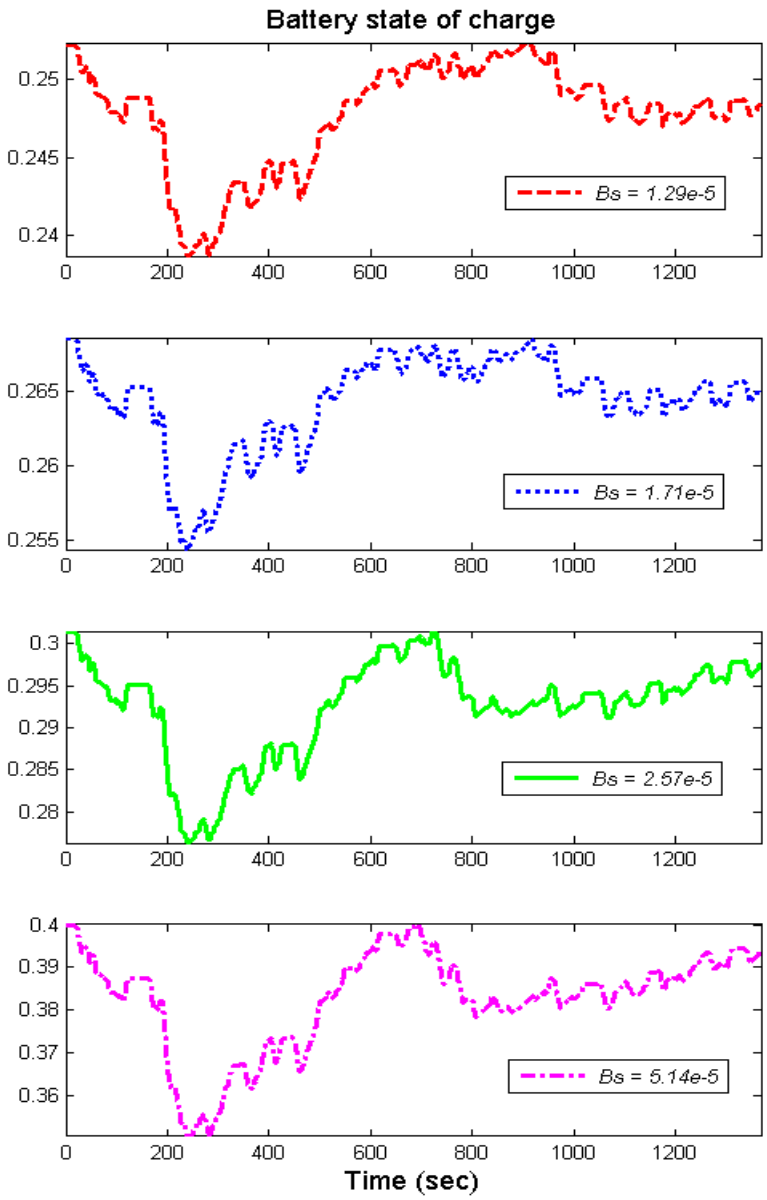


Figure 4.13: Battery SoC profiles for each vehicle configuration with different batteries and corresponding optimal controller over the UDDS cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal centralized controller obtained in Section 4.2.2 to follow the HWFET cycle is shown in Figure 4.14.

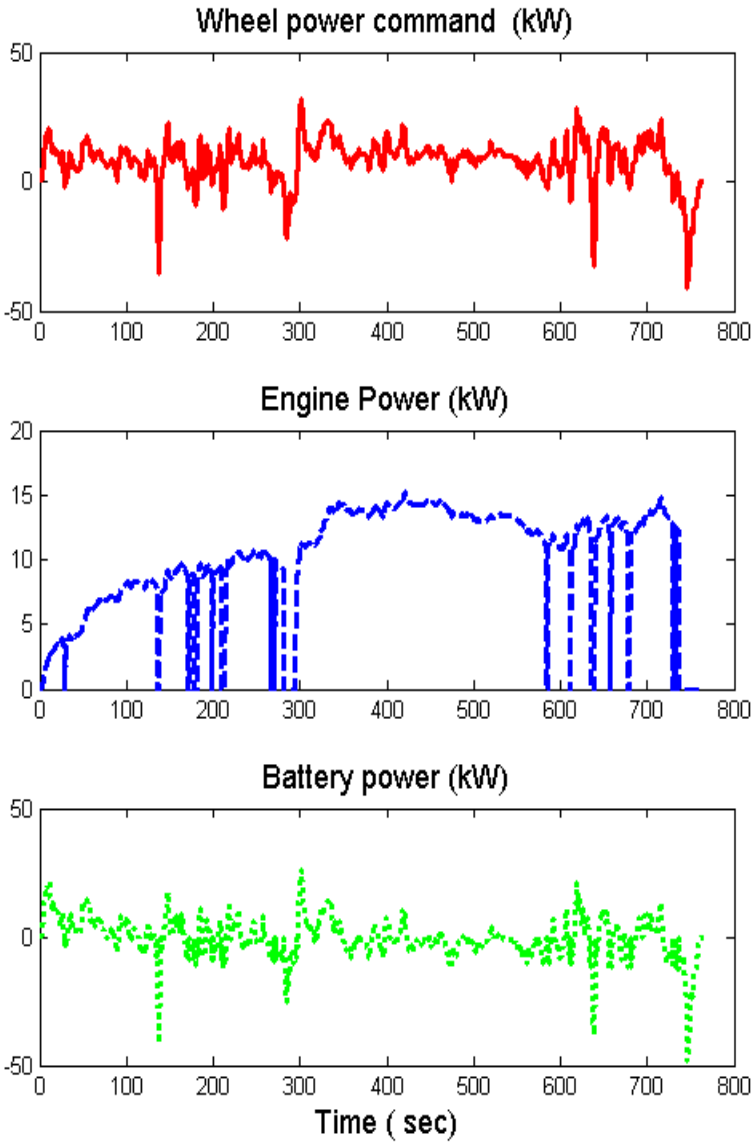


Figure 4.14: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding optimal controller over the HWFET cycle.

The SoC trajectories for each vehicle configuration with the four batteries and corresponding optimal controllers obtained in Section 4.2.2 to follow the HWFET cycle are plotted in Figure 4.15.

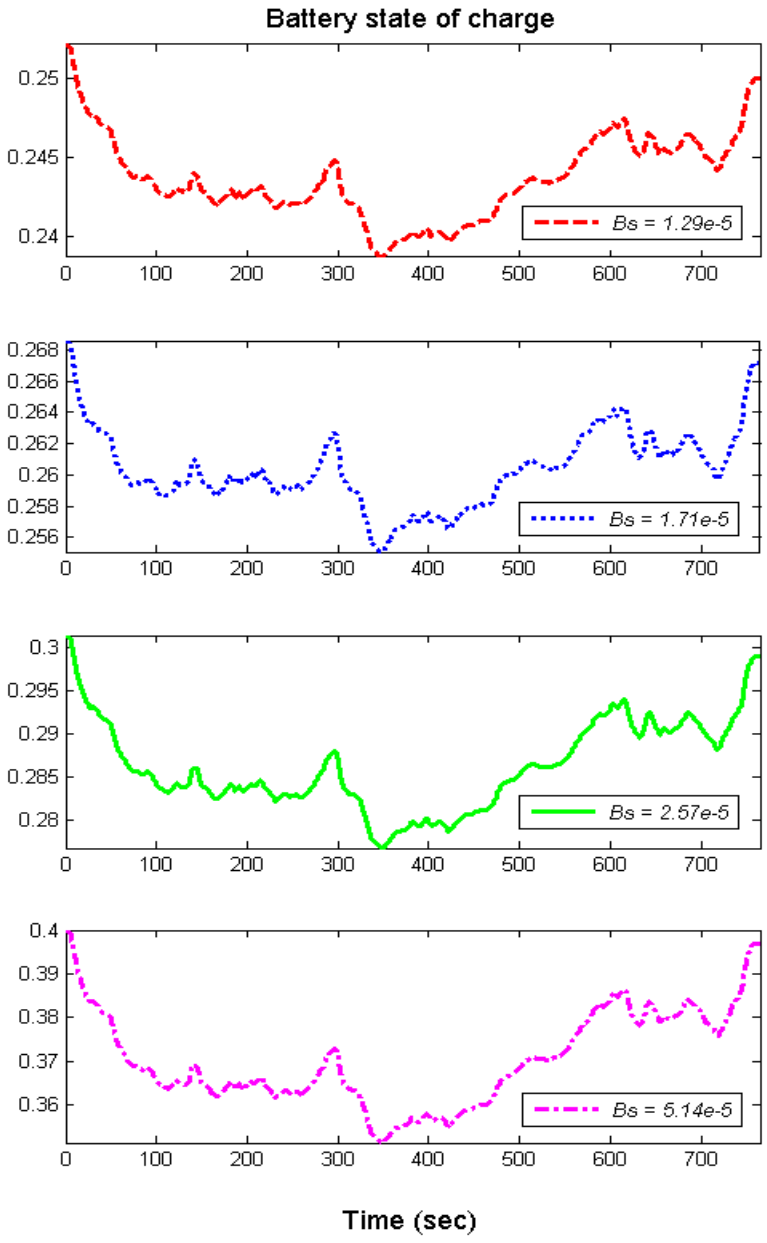


Figure 4.15: Battery SoC profiles for each vehicle configuration with different batteries and corresponding optimal controller over the HWFET cycle.

4.3. Summary

In this Chapter, we presented a control-oriented model for a PHEV. The component sizes of the nominal vehicle configuration are typical of current designs such as the 2011 Chevrolet Volt. Four candidate batteries with different energy capacities (and different all electric range capabilities) for the same vehicle are chosen for CSM design. The battery parameters are scaled based on the parameters of the battery in the 2011 Chevrolet Volt. Then a novel feedback-based controller for the CS mode to facilitate distributed controller design for battery CSM is introduced. The controller gains are obtained though optimization to achieve optimal fuel economy and driving performance, while satisfying the constraints on closed loop system stability, battery charge sustainability and component reliability. Although the controller is designed with respect to the EPA US06 cycle, the simulation results demonstrate that the feedback based controller also achieves good fuel economy over other driving cycles (e.g., the EPA UDDS and HWFET cycles). In the meantime, the maximum wheel power tracking error is less than 2.15e-6 kW for all simulation scenarios, which ensures good driving performance. The maximum deviation of the battery SoC at the end of the driving cycles is -7.41% of the reference SoC for all simulation scenarios, which ensures charge sustainability.

Here we use a practical approach to solve for the controller gains that reduces computation time. The wheel power command is saturated to non-negative values and used for the controller gain optimization, thus, regenerative braking is excluded. The obtained controller is then evaluated over the driving cycles with regenerative braking included. The results show that this approach delivers good fuel economy, good driving

performance in terms of power tracking error, and battery charge sustainability (the final SoC is within ±7.14% of the reference value).

CHAPTER V

BATTERY CSM FOR PHEV

As discussed in Chapter I, extended all electric range of PHEVs further reduces fuel consumption and provides higher fuel economy. A big hurdle for PHEV commercialization is the cost and reliability of batteries. Thus, it is beneficial to investigate a decoupled design of the vehicle and the battery component. The battery component becomes a swappable module if the battery change can be accommodated by only recalibrating the controller built inside the battery module so that the vehicle performance meets the performance that is achievable by redesigning the entire centralized controller.

Some potential benefits from battery CSM in PHEVs are as follows:

1) Consumer oriented vehicles can be developed. For the same size vehicle, the consumers can choose the battery size for their vehicle according to their daily driving patterns and budget.
2) As the battery technology advances, the vehicle can be easily upgraded by simply plugging in a new battery with higher energy capacity to extend the all electric range for higher efficiencies.
3) Having a range of battery alternatives for use will increase the flexibility to choose battery components from different suppliers for lower cost.

4) From a manufacturing point of view, a modularized battery component lowers the coupling risk of the vehicle system, and enables parallel design to decrease the lead-time.
5) From the battery supplier point of view, the battery component can be easily customized by deploying different battery controllers for different vehicle companies.

In this Chapter, first, using the centralized controller obtained in Chapter IV, a controller distribution architecture is proposed based on the sensitivity analysis of the control signals with respect to the battery hardware parameter. Second, the distributed controller, which achieves battery CSM and optimizes fuel economy, is obtained by solving a bi-level optimization problem. The Augmented Lagrangian Decomposition method is employed to solve the bi-level optimization problem. As was the case for the centralized controller design in Chapter IV, a non-negative wheel power command is applied in the optimization and regenerative braking is excluded. Third, the obtained distributed controllers are evaluated in terms of fuel economy, driving performance and charge sustainability over the US06 cycle with regenerative braking included. Finally, the design results of the centralized controller and that of the distributed controller which provides battery CSM are compared.

5.1. Controller Distribution Architecture

A distributed controller architecture for the PHEV is introduced in Figure 5.1. Compared to the vehicle diagram with a centralized supervisory controller in Figure 4.1, the centralized supervisory controller is distributed into two parts: the vehicle system controller (VSC), which is fixed with the vehicle, and the battery system controller (BSC),

which resides in the battery module and thus is swappable along with the battery. Such an implementation assumes that the battery is a smart component, which has an embedded microcontroller to perform control functions and to communicate with the VSC over a network, see the dashed line in Figure 5.1. Other physical implementations are also possible, e.g., the VSC and the BCU can be physically implemented in the same microprocessor and the BCU software and calibration can be "reflashed'' when the battery changes.

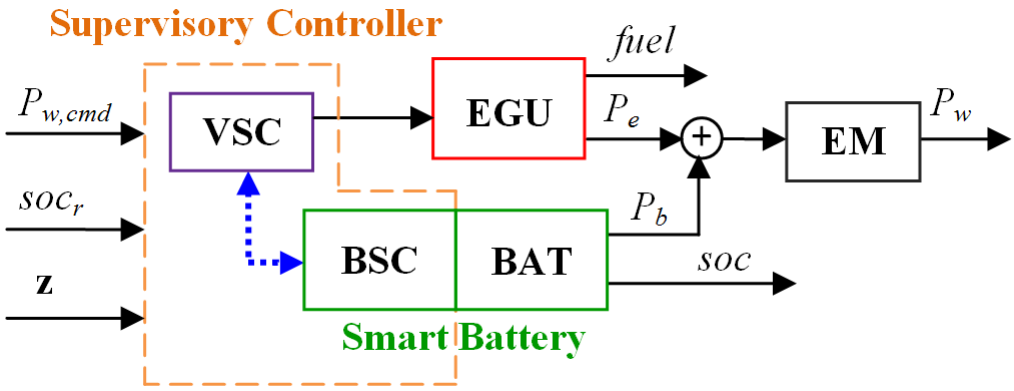


Figure 5.1: Diagram of the vehicle components with distributed supervisory controller.

The controller distribution between the VSC and the BSC addresses the tradeoff between performance (generally highest when the controller is entirely within the BSC) and simplicity of the BSC implementation (desirable in terms of computing and calibration effort). Here we relate the controller simplicity to controller order and the number of gains.

We determine the effective controller distribution between the VSC and the BSC using the sensitivity analysis of the control signals with respect to the battery parameter. To

facilitate the sensitivity analysis, the optimal values of the controller gains obtained in Chapter IV are fitted using fourth order polynomials, see Figure 5.2.

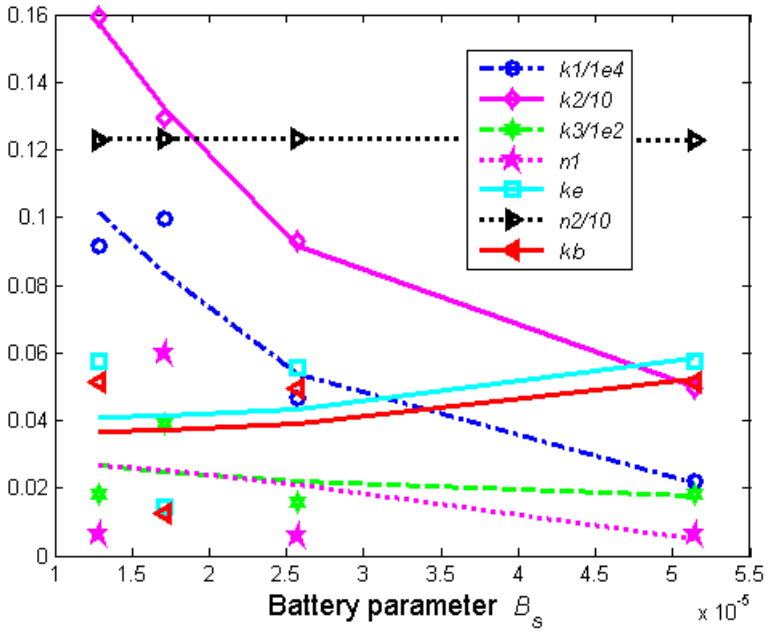


Figure 5.2: The fourth order polynomial fit of the centralized controller gains.

Let $x_i \in \{k_1, k_2, k_3, n_1, n_2, k_e, k_b\}$, $i = 1{:}7$, denote the controller gains, thus we get

$$x_i(B_s) \approx c_{0,i} + c_{1,i}B_s + c_{2,i}B_s^2 + c_{3,i}B_s^3 + c_{4,i}B_s^4 \tag{5.1}$$

Recall that the control signals, $P_{e,cmd}$ and $P_{b,cmd}$, are functions of the controller gains, the states and the wheel power command as in equations (4.6) and (4.7). The normalized gain sensitivity with respect to the battery parameter as defined in equation (2.5) is,

$$S_n(x_i) = \frac{\left|\frac{\partial P_{e,cmd}}{\partial x_i}\frac{\partial x_i^*}{\partial B_s}\right| + \left|\frac{\partial P_{b,cmd}}{\partial x_i}\frac{\partial x_i^*}{\partial B_s}\right|}{\sum_{i=1}^{7}\left|\frac{\partial P_{e,cmd}}{\partial x_i}\frac{\partial x_i^*}{\partial B_s}\right| + \left|\frac{\partial P_{b,cmd}}{\partial x_i}\frac{\partial x_i^*}{\partial B_s}\right|} \tag{5.2}$$

Figure 5.3 shows the normalized sensitivity calculated using the mean value over the operating range of $\mathbf{z}$ and $P_{w,cmd}$, and middle-of-the-range value of B_s. We see that k_1 is the most sensitive gain, k_2 is the second most sensitive gain, while the other gains are less sensitive in comparison.

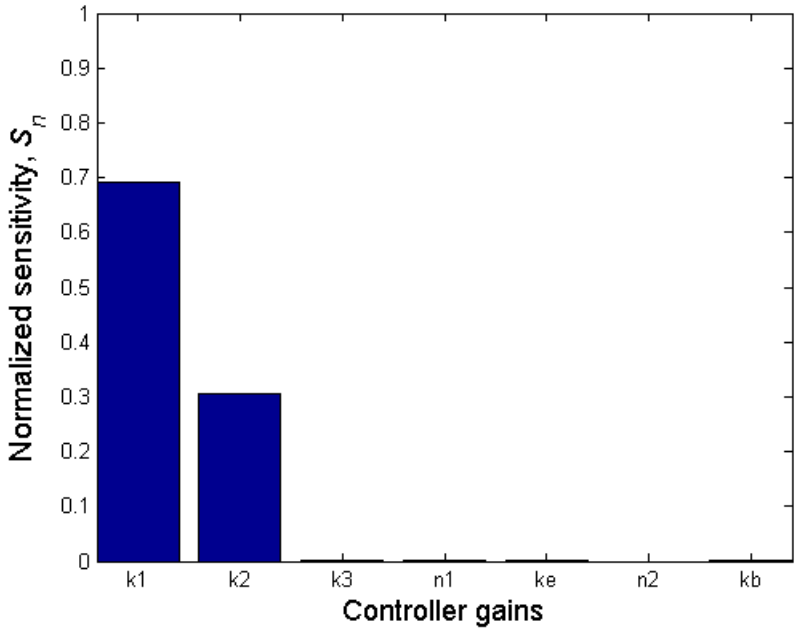


Figure 5.3: Sensitivity of controller gains with respect to the battery parameter.

Based on this sensitivity analysis, two distribution cases are considered. In Case 1, k_1 and k_2, along with the related calculations, are distributed into the BSC. In Case 2, k_1 along with the related calculations, are distributed into the BSC. Note that k_4 and k_5 are dependent on k_1 and k_2, respectively, from eigen-structure assignment, therefore, k_4 and k_5 follow the distribution pattern of k_1 and k_2, respectively.

To be specific, in Case 1, the linear controller for the CS mode in the VSC is:

$$z_3 = \int (P_{w,cmd} - P_w)dt \tag{4.5}$$

$$P_{e,cmd} = k_3 z_3 + n_1 P_{w,cmd} + k_e P_{b,cmd} + P_{be} \tag{5.3}$$

$$P_{eb} = k_6 z_3 + n_2 P_{w,cmd} + k_b P_{e,cmd} \tag{5.4}$$

The corresponding linear controller in the BSC in Case 1 is

$$z_1 = \Delta soc = soc_r - soc \tag{4.3}$$

$$z_2 = \int (\Delta soc)dt = \int (soc_r - soc)dt \tag{4.4}$$

$$P_{b,cmd} = k_4 z_1 + k_5 z_2 + P_{eb} \tag{5.5}$$

$$P_{be} = k_1 z_1 + k_2 z_2 \tag{5.6}$$

In Case 2, the linear controller for the CS mode in the VSC is:

$$z_2 = \int (\Delta soc)dt = \int (soc_r - soc)dt \tag{4.4}$$

$$z_3 = \int (P_{w,cmd} - P_w)dt \tag{4.5}$$

$$P_{e,cmd} = k_2 z_2 + k_3 z_3 + n_1 P_{w,cmd} + k_e P_{b,cmd} + P_{be} \tag{5.7}$$

$$P_{eb} = k_5 z_2 + k_6 z_3 + n_2 P_{w,cmd} + k_b P_{e,cmd} \tag{5.8}$$

The corresponding linear controller in the BSC in Case 2 is

$$z_1 = \Delta soc = soc_r - soc \tag{4.3}$$

$$P_{b,cmd} = k_4 z_1 + P_{eb} \tag{5.9}$$

$$P_{be} = k_1 z_1 \tag{5.10}$$

The signal paths on the network between the VSC and the BSC are detailed in Figure 5.4. The CD mode control and regenerative braking (RB) control reside in the VSC. The controller for the CS mode is distributed between the VSC and the BSC as described above in Case 1 and Case 2.

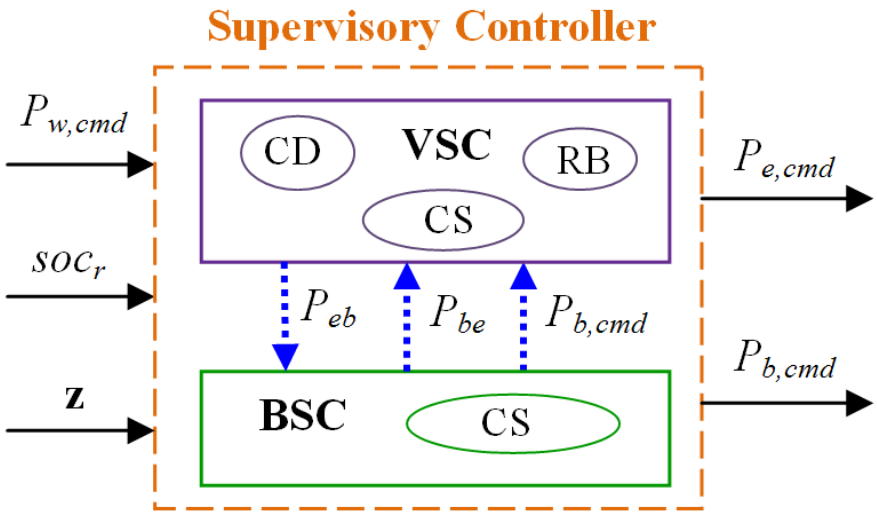


Figure 5.4: The signal paths on the network between the VSC and the BSC.

5.2. Optimization of the Distributed Controller Gains

The sensitivity analysis in the preceding subsection has been used to guide the distributed controller architecture decisions. In this section we demonstrate how the distributed controller gains can be obtained by solving a bi-level optimization problem using the Augmented Lagrangian Decomposition method [28], as discussed in Chapter II.

Specifically, consider N (= 4) batteries as given in Table 4.2. Denote the controller gains in the VSC as $\mathbf{x}_s$, and the controller gains in the BSC as $\mathbf{x}_{m,i}$, for each vehicle configuration with battery parameter $B_{s,i}$, $i \in \{1, \ldots, N\}$. Introduce auxiliary variables $\mathbf{x}_{s,i}$, to serve as local copies of the shared controller gains $\mathbf{x}_s$ for the vehicle configuration with battery parameter $B_{s,i}$. The design variables $\mathbf{x}_s$ and $\mathbf{x}_{s,i}$ are forced to be equal by the consistency constraint $\mathbf{c}: R^{(N+1)\cdot m_1} \mapsto R^{N\cdot m_1}$, which is defined as $\mathbf{c}(\mathbf{x}_s, \mathbf{x}_{s,1}, \mathbf{x}_{s,2}, ..., \mathbf{x}_{s,N}) =$

$[\mathbf{c}_1^T, \mathbf{c}_2^T, ..., \mathbf{c}_N^T]^T = \mathbf{0}$, with $\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i}) = \mathbf{x}_s - \mathbf{x}_{s,i}$, where $\mathbf{c}_i \in R^{m_1}$ is the vector of consistency constraints for the system configuration with battery parameter $B_{s,i}$.

The consistency constraints are relaxed by an augmented Lagrangian penalty function $\phi : R^{Nm_1} \mapsto R$.

$$\phi(\mathbf{c}) = \mathbf{v}^T\mathbf{c} + \left\|\mathbf{w} \circ \mathbf{c}\right\|_2^2 = \sum_{i=1}^{N} \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) \tag{5.11}$$

with the penalty function for each system configuration with component parameter $B_{s,i}$, $\phi_i : R^{m_1} \mapsto R$ defined by:

$$\phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) = \mathbf{v}_i^T(\mathbf{x}_s - \mathbf{x}_{s,i}) + \left\|\mathbf{w}_i \circ (\mathbf{x}_s - \mathbf{x}_{s,i})\right\|_2^2 \tag{5.12}$$

where: $\mathbf{v} = [\mathbf{v}_1^T, \mathbf{v}_2^T, ..., \mathbf{v}_N^T]^T \in R^{N \cdot m_1}$ is the vector of Lagrange multiplier estimates for the consistency constraints, and $\mathbf{w} = [\mathbf{w}_1^T, \mathbf{w}_2^T, ..., \mathbf{w}_N^T]^T \in R^{N \cdot m_1}$ is the vector of penalty weights, with $\mathbf{v}_i \in R^{m_1}$, $\mathbf{w}_i \in R^{m_1}$. The symbol ◦ represents the Hadamard product.

The outer stage optimization minimizes the penalty function with respect to the controller gains in the VSC, $\mathbf{x}_s$:

$$\phi(\mathbf{c}) = \sum_{i=1}^{N} \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i})) \tag{5.13}$$

The outer stage problem has no constraints. It can be solved analytically:

$$\mathbf{x}_s^* = \arg\min_{\mathbf{x}_s} \phi(\mathbf{c}) = \frac{\sum_{i=1}^{N}(\mathbf{w}_i \circ \mathbf{w}_i \circ \mathbf{x}_{s,i}) - \frac{1}{2}\sum_{i=1}^{N}\mathbf{v}_i}{\sum_{i=1}^{N}(\mathbf{w}_i \circ \mathbf{w}_i)} \tag{5.14}$$

In the inner stage optimization for each vehicle configuration with battery parameter $B_{s,i}$, the objective function includes two terms, the cost function J, which is defined in equation (4.9), and the penalty function as defined in equation (5.12).

$$\min \; J(\mathbf{x}_{s,i}, \mathbf{x}_{m,i}, B_{s,i}) + \phi_i(\mathbf{c}_i(\mathbf{x}_s, \mathbf{x}_{s,i}))$$

The design variables of each inner stage optimization are the auxiliary variables, $\mathbf{x}_{s,i}$, and the controller gains in the BSC, $\mathbf{x}_{m,i}$. The relaxation error between $\mathbf{x}_{s,i}$ and $\mathbf{x}_s$ are driven to zero by the penalty function in the objective function. The constraints for each battery application are the same as described in Chapter IV Section 4.2.2 for centralized controller design. The inner stage problems can be solved using nonlinear programming methods.

At each iteration of the outer stage problem, a new estimate of $\mathbf{x}_s$ is generated and each of the inner stage problems is solved using $\mathbf{x}_s$ as a parameter; the penalty weights $\mathbf{v}$ and $\mathbf{w}$ are updated using the method of multipliers [56] to gradually drive the penalty function to zero for the consistency of $\mathbf{x}_s$ and $\mathbf{x}_{s,i}$. This procedure is repeated until a feasible solution that satisfies the consistency constraints $\mathbf{c} < \boldsymbol{\varepsilon}$, for each inner stage optimization is found, or until the maximum number of function evaluations is reached, where $\boldsymbol{\varepsilon}$ is the error tolerance.

A good initial guess is very important for this bi-level optimization problem, and the design results from the centralized controller can be employed. The solver "fmincon" in MATLAB is used to solve the inner stage optimization problems, while the outer stage problem is solved analytically by equation (5.14).

The distributed controller gains for the CS mode are obtained by the above bi-level optimization over the US06 driving cycle.

For Case 1, the resulting controller gains for the VSC are:

$k_3 = 1.8164$, $n_1 = 0.0062$, $k_e = 0.0574$, $n_2 = 1.2305$, $k_b = 0.0512$.

The controller gains for the BSC in Case 1 for each battery application are listed in Table 5.1. The controller gains k_4, k_5 and k_6 can be calculated from k_1, k_2 and k_3 using equations (4.8a) – (4.8f) from eigen-structure assignment.

The vehicle performance metric for each battery application with corresponding distributed controller in Case 1 to follow the saturated non-negative power command from the US06 cycle is also listed in Table 5.1. The fuel economy in MPG considering both fuel consumption from the engine and equivalent fuel consumption from the battery at the end of the driving cycle, the driving performance evaluated by the maximum power tracking error, $err_{P,max}$, in kW, and the battery charge sustaining performance evaluated by the deviation of the final SoC from the reference SoC, $soc_{dev,p}$, are defined the same as in Table 4.3 in Chapter IV. We see the fuel economy of the distributed control in Case 1 is almost the same as that of the centralized control case given in Table 4.3. The maximum power tracking error in all simulation scenarios is less than 6.18e-4 kW. The battery charge sustainability evaluated by the deviation of the final SoC from the reference SoC is within ± 2% as in the constraints.

Table 5.1: Performance results for each battery application with corresponding distributed controller in Case 1 to follow the saturated non-negative power command from the US06 cycle

B_s (e-5)	k_1	k_2	MPG	$err_{p,max}$	$soc_{dev,p}$
1.29	917.42	1.59	27.58	6.18e-4	-2%
1.71	1000	1.34	28.16	6.02e-4	-2%
2.57	466.87	0.93	28.78	5.82e-4	-2%
5.14	222.23	0.50	29.40	5.54e-4	-2%

The SoC trajectories for each vehicle configuration, with the four batteries and corresponding distributed controllers in Case 1, to follow the saturated non-negative power command from the US06 cycle, are plotted in Figure 5.5. The initial battery SoC in the simulation is the reference SoC respectively for each battery application.

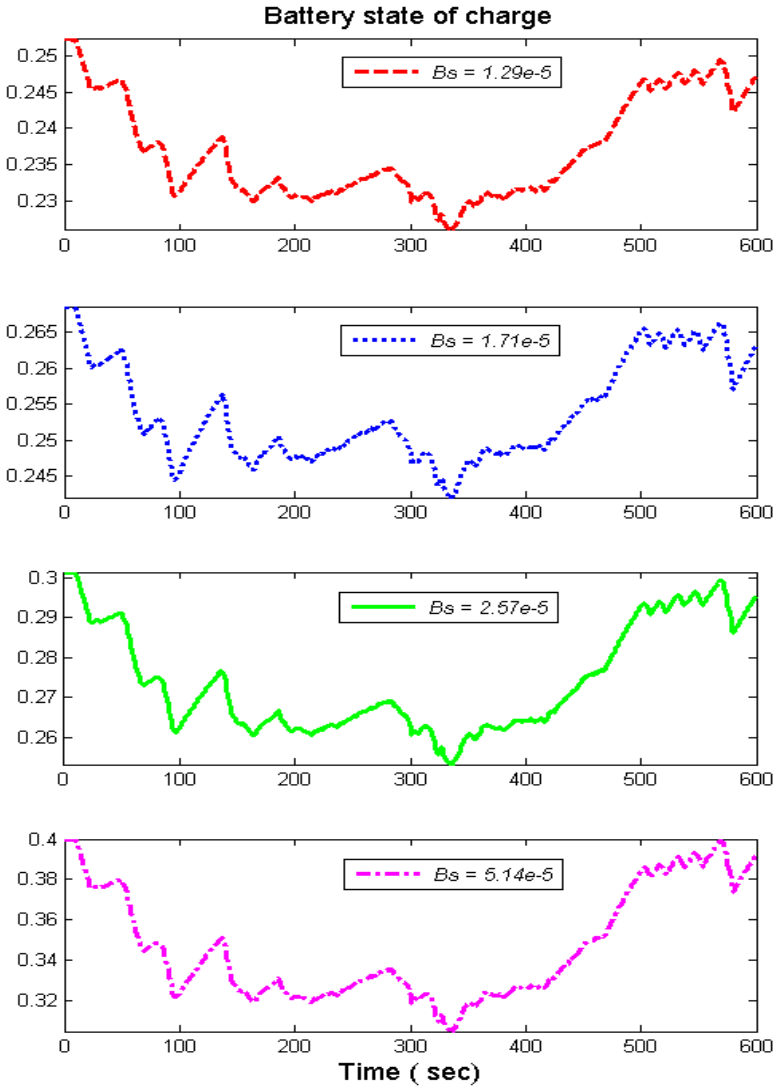


Figure 5.5: Battery SoC profiles for each battery application with corresponding distributed controller in Case 1 to follow the saturated non-negative power command from the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 1 to follow the saturated non-negative power command from the US06 cycle is given in Figure 5.6. We see the engine provides the slowly changing power, which is roughly the moving average of the wheel power command, while the battery provides the transients.

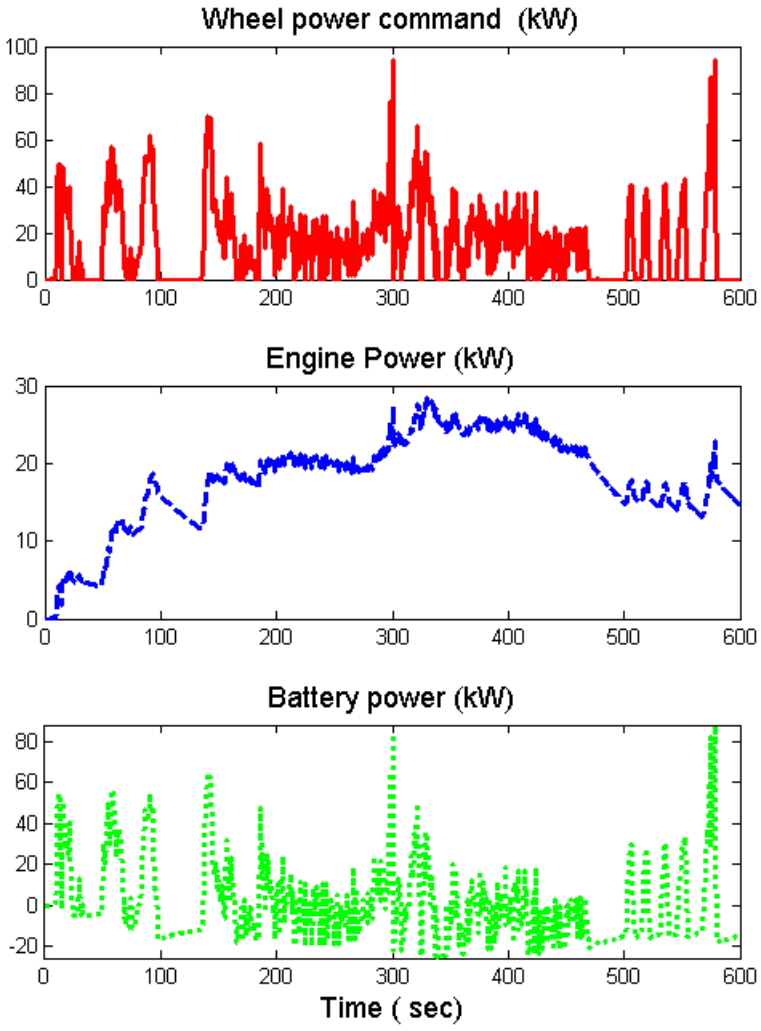


Figure 5.6: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 1 to follow the saturated non-negative power command from the US06 cycle.

For Case 2, the controller gains for the VSC are:

$k_2 = 1.3315$, $k_3 = 1.8164$, $n_1 = 0.0062$, $k_e = 0.0574$, $n_2 = 1.2305$, $k_b = 0.0512$.

The controller gains for the BSC in Case 2 and the vehicle performance metric for each battery application with corresponding distributed controller in Case 2 to follow the saturated non-negative power command from the US06 cycle are listed in Table 5.2, where MPG, $err_{P,max}$ and $soc_{dev,p}$, are defined the same as in Table 4.3 in Chapter IV. The fuel economy of the distributed control in Case 2 is not as good as that of the centralized control case. It will be compared in detail in next Section after including the regenerative braking. The maximum power tracking error in all simulation scenarios is less than 2.10e-3 kW. The battery charge sustainability evaluated by the deviation of the final SoC from the reference SoC is within ± 2%.

Table 5.2: Performance results for each battery application with corresponding distributed controller in Case 2 to follow the saturated non-negative power command from the US06 cycle

B_s (e-5)	k_1	MPG	$err_{p,max}$	$soc_{dev,p}$
1.29	1978.70	27.56	2.10e-3	-2%
1.71	1355.19	28.15	2.10e-3	-2%
2.57	1052.21	28.58	2.00e-3	-2%
5.14	674.10	28.92	1.90e-3	-2%

The SoC trajectories for each vehicle configuration, with the four batteries and corresponding distributed controllers in Case 2, to follow the saturated non-negative

power command from the US06 cycle, are plotted in Figure 5.7. The initial battery SoC in the simulation is the reference SoC respectively for each battery application.

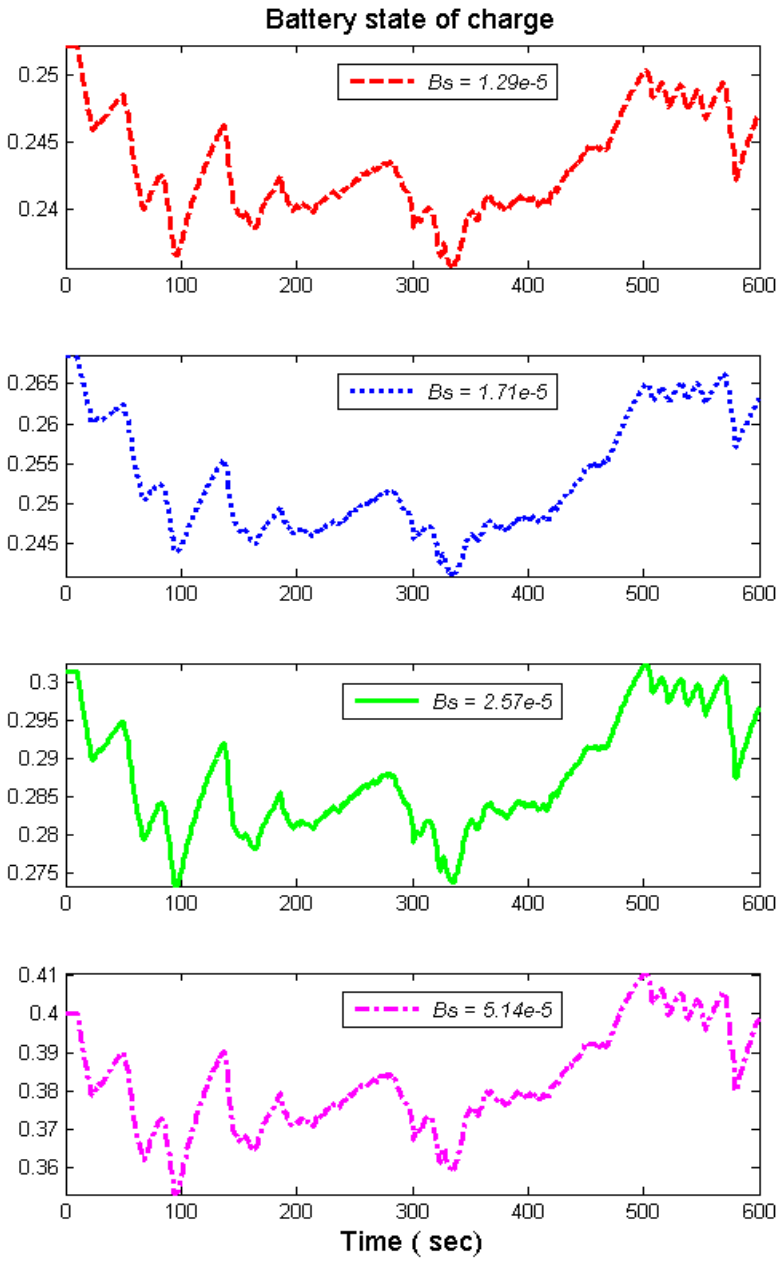


Figure 5.7: Battery SoC profiles for each battery application with corresponding distributed controller in Case 2 to follow the saturated non-negative power command from the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 2 to follow the saturated non-negative power command from the US06 cycle is given in Figure 5.8.

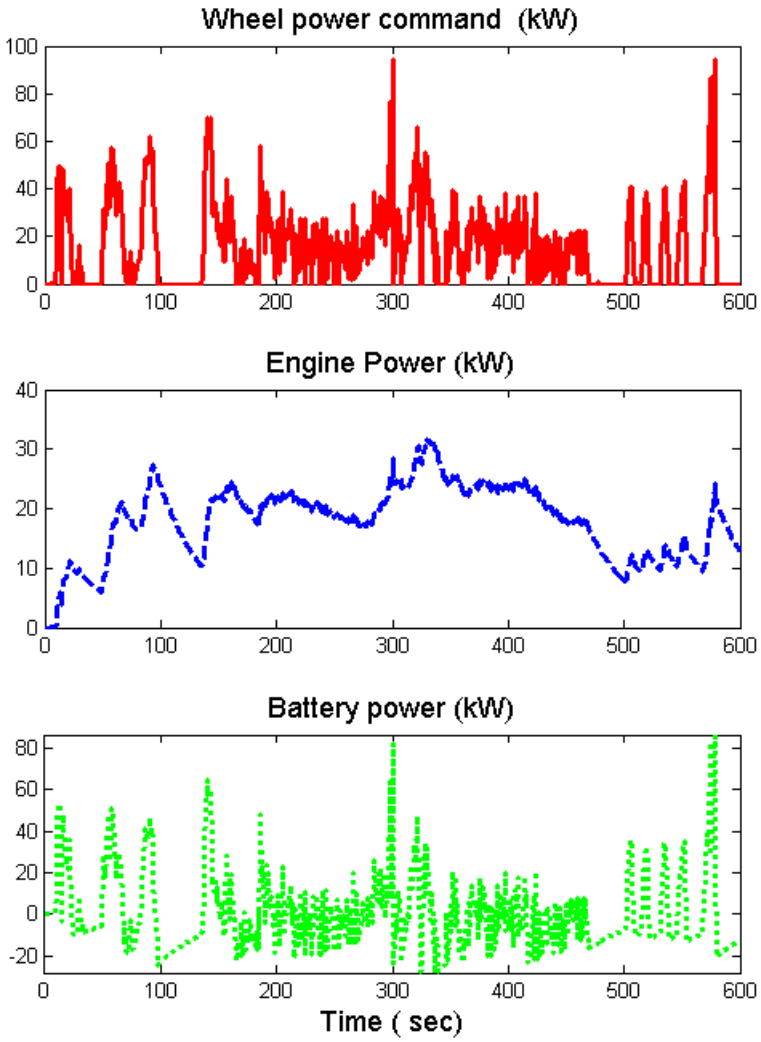


Figure 5.8: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 2 to follow the saturated non-negative power command from the US06 cycle.

5.3. Distributed Controller Evaluation

In this Section, the distributed controllers in Case 1 and Case 2 are evaluated according to the original power command from US06 cycle. Regenerative braking is activated when wheel power command is negative, and the engine is shut off without delivering power.

The maximum power tracking error and the deviation of the final battery SoC from the reference SoC for each battery application with corresponding distributed controllers in Case 1 and Case 2 to follow the US06 cycle is given in Table 5.3, where $err_{P,max}$ and $soc_{dev,p}$ are defined the same as in Table 4.3 in Chapter IV.

Table 5.3: Performance results for each battery application with the distributed controller in Case 1 and Case 2 over the US06 cycle

	Case 1		**Case 2**	
B_s (e-5)	$err_{P,max}$	$soc_{dev,p}$	$err_{P,max}$	$soc_{dev,p}$
1.29	3.76e-4	-3.99%	1.29e-3	-2.65%
1.71	3.64e-4	-3.37%	1.26e-3	-3.52%
2.57	3.55e-4	-5.30%	1.21e-3	-4.77%
5.14	3.35e-4	-7.40%	1.18e-3	-5.38%

The maximum power tracking error for each vehicle configuration to follow the US06 cycle is 3.76e-4 kW for Case 1, and 1.29e-3 kW for Case 2. The maximum deviation of the final battery SoC from the reference SoC for each vehicle configuration to follow the US06 cycle is -7.4% for Case 1, and -5.38% for Case 2. Thus, the driving performance

and charge sustainability are satisfied for both distribution cases. The fuel economy in terms of MPG is compared with that of the centralized control case in Section 5.4.

The SoC trajectories for each vehicle configuration with the four batteries and corresponding distributed controllers in Case 1 to follow the US06 cycle, are plotted in Figure 5.9.

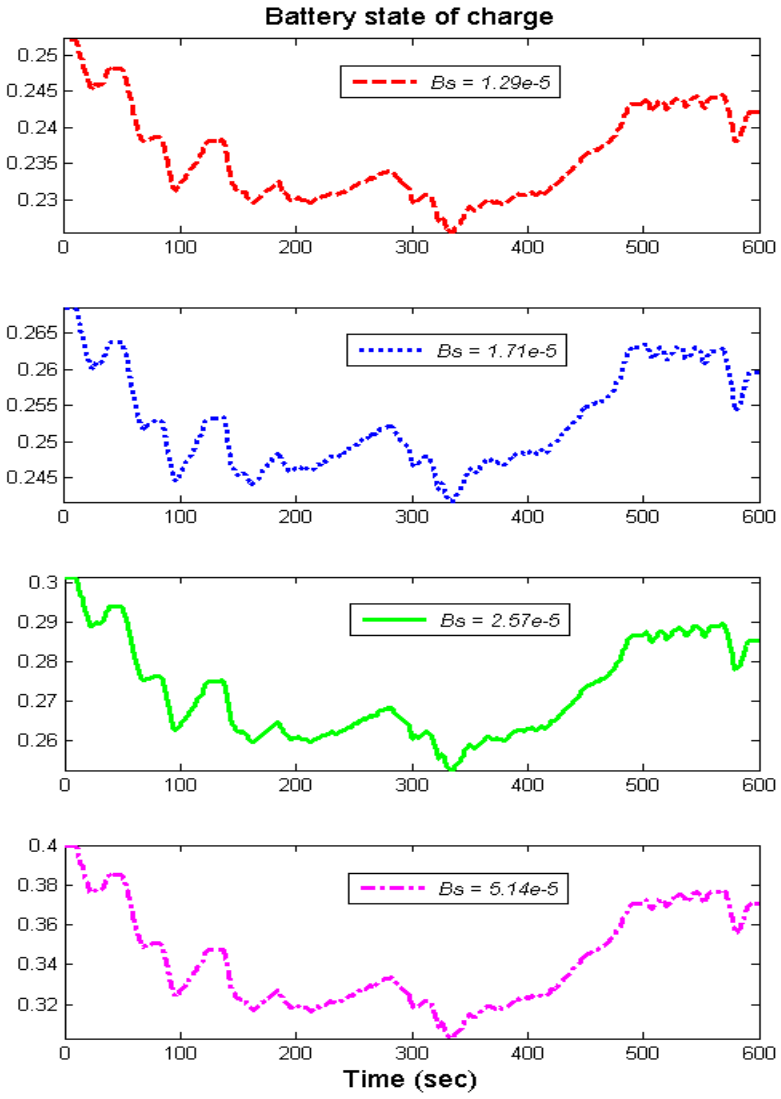


Figure 5.9: Battery SoC profiles for each battery application with corresponding distributed controller in Case 1 over the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 1 to follow the US06 cycle is given in Figure 5.10.

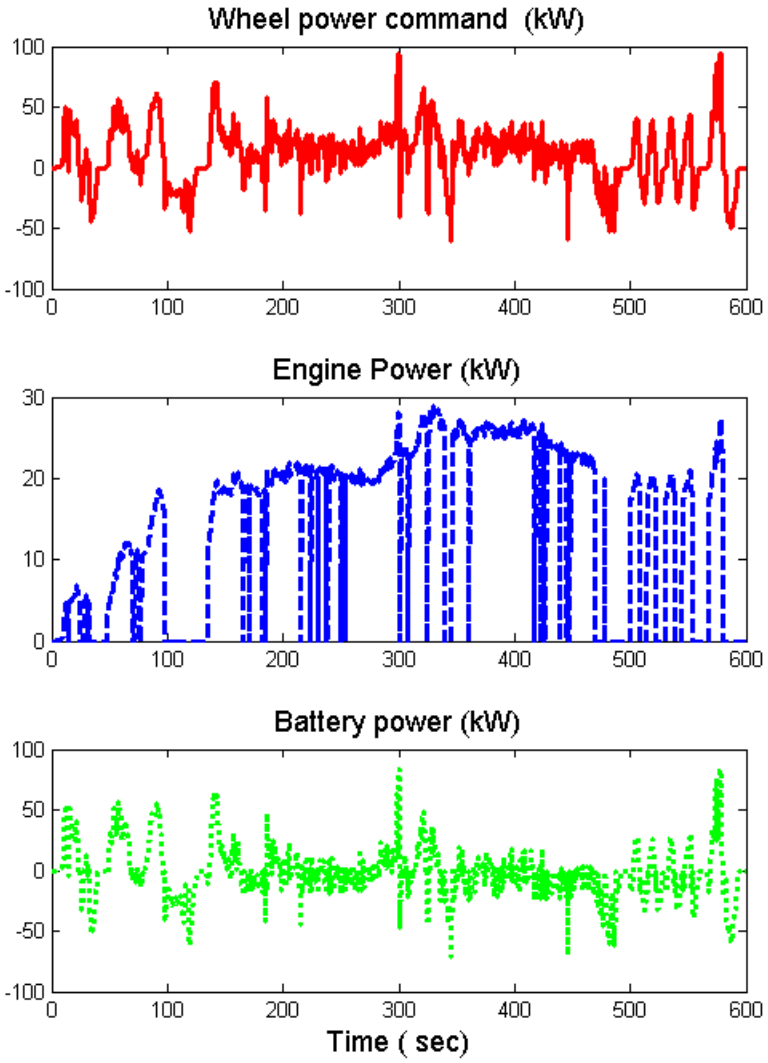


Figure 5.10: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 1 over the US06 cycle.

The SoC trajectories for each vehicle configuration with the four batteries and corresponding distributed controllers in Case 2 to follow the US06 cycle, are plotted in Figure 5.11.

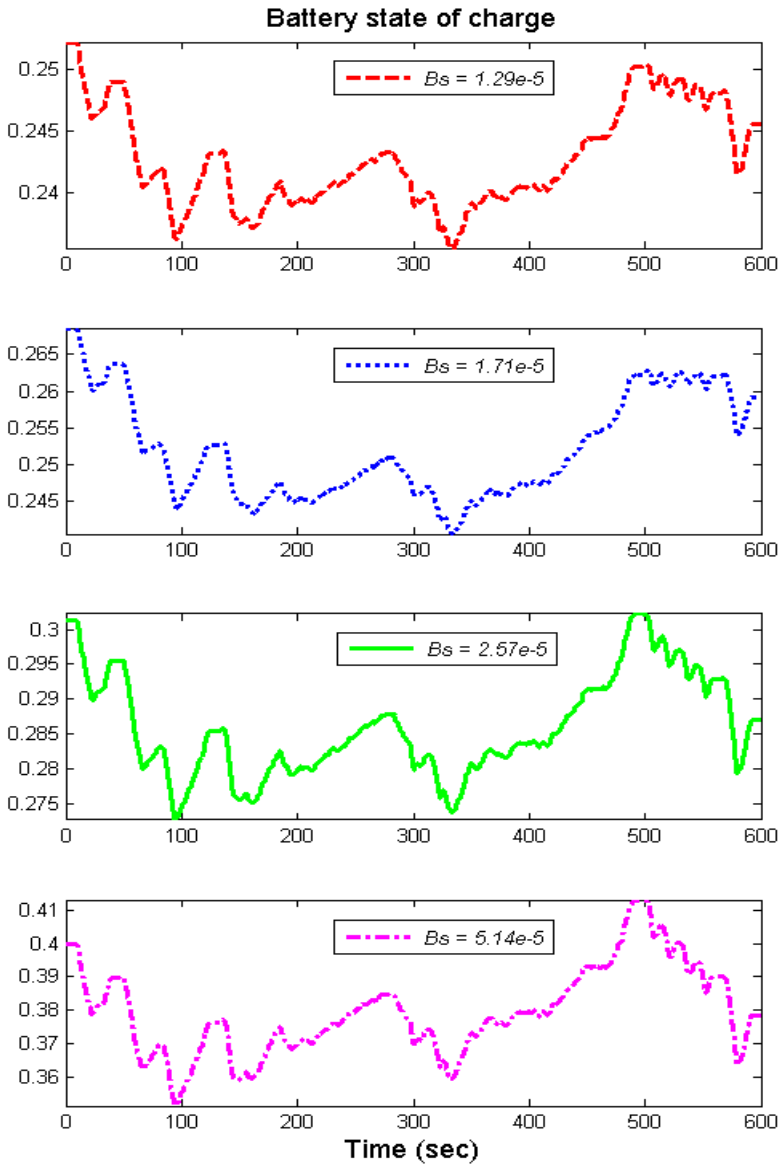


Figure 5.11: Battery SoC profiles for each battery application with corresponding distributed controller in Case 2 over the US06 cycle.

An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 2 to follow the US06 cycle is given in Figure 5.12.

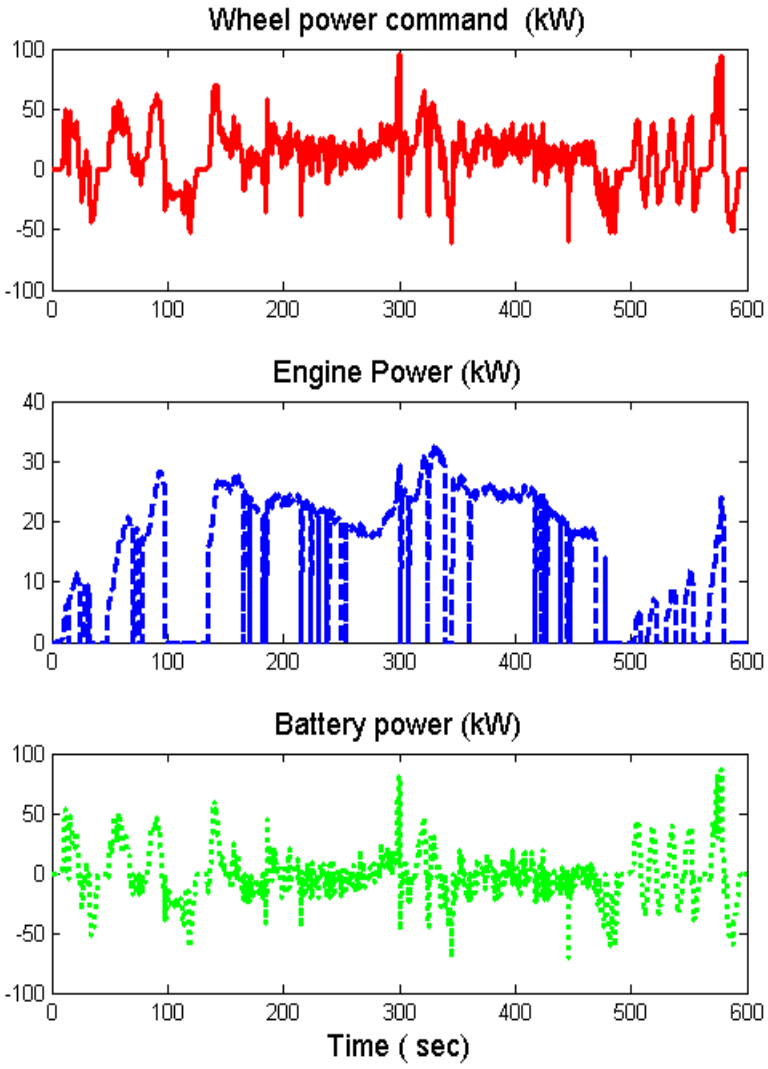


Figure 5.12: An example power split between the engine and the battery for the vehicle with battery 3 (B_s = 2.57e-5) and corresponding distributed controller in Case 2 over the US06 cycle.

5.4. Controller Comparison and Discussion

The driving performance and charge sustaining objectives are satisfied with the centralized control, and the distributed control in Case 1 and Case 2, as shown in Tables 4.4 - 4.7 and Table 5.3. The fuel economy results over the US06 cycle are compared in Figure 5.13. We assume that the centralized controller set the benchmark for fuel economy and are normalized as 1 respectively for each battery application.

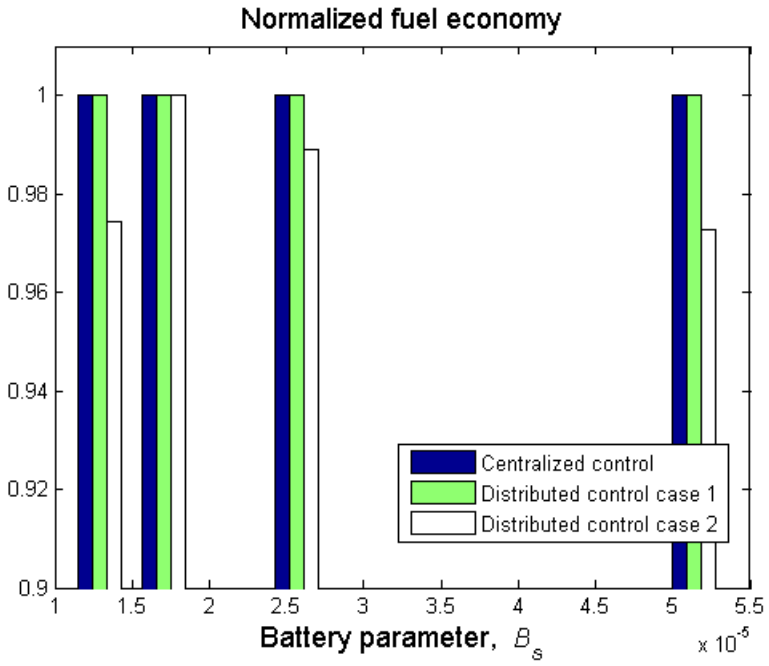


Figure 5.13: Normalized fuel economy comparison between the PHEV with centralized controller and the PHEV with distributed controllers that provide battery CSM.

From Figure 5.13, in the distributed control Case 1, battery CSM can be essentially achieved without compromising fuel economy compared to the centralized control case. In the distributed control Case 2, battery CSM is achieved by compromising some fuel economy (less than 3%). This shows a tradeoff between the simplicity of BSC and the

achievable fuel economy. As more controller gains are moved to the BSC, more design freedom is available to achieve better fuel economy, since only the BSC can be recalibrated when the battery changes. However, a simple BSC implementation is desirable as it entails low computing effort as well as low recalibration effort.

Based on the above results, the distributed controller in Case 1 is preferred. In this case, four gains of the feedback-based controller, along with the related calculations, are distributed into the BSC, while the rest of the controller for the CS mode, together with the CD mode and regenerative braking control, remain in the VSC. With this approach, battery CSM is achieved without compromising fuel economy versus the centralized control case. At the same time, the BSC is reasonably simple. As the controller functionality related to the battery SoC are confined to the BSC, the estimation of battery SoC (not considered in this paper) can be confined to the BSC as well, making the battery module functionally independent both in hardware and software.

From Figure 5.13, in the distribution control Case 2, one can make a tradeoff between desired fuel economy and achievable battery CSM. If the desired fuel economy is the same as that of the centralized control, only one battery with parameter, B_s = 1.71e-5, satisfies the performance requirement and can be applied to the PHEV. If the desired fuel economy is within 2% of degradation of the fuel economy that is achievable by the centralized controller, then, two batteries satisfy the performance requirement and can be swapped with each other to create PHEV product variant. If the desired fuel economy is within 3% of degradation of the fuel economy that is achievable by the centralized controller, then, all the four batteries satisfy the performance requirement and can be swapped with each other to create PHEV product variant.

5.5. Summary

Battery CSM in PHEVs is achieved by the proposed distributed supervisory controller. A novel feedback-based controller for the CS mode is proposed to facilitate battery CSM design. A sensitivity analysis of the control signals with respect to the battery parameter is introduced for effective controller distribution. The distributed controller which enables battery CSM is then obtained by solving a bi-level optimization problem using the numerical algorithm of the Augmented Lagrangian Decomposition method. The bi-level formulation ensures that only the controller gains in the BSC depend on the battery parameters, while the VSC remains the same for different battery applications. With such a distributed controller implementation, the battery module can be swapped without redesign or recalibration of the VSC, so that the vehicle performance meets the performance achievable by redesigning the entire centralized controller.

Two distributed control cases are considered. In Case 1, four controller gains of the linear controller for the CS mode, along with the related calculations, are distributed into the BSC, while the rest of the controller for the CS mode, together with the CD mode and regenerative braking control, remain in the VSC. Battery CSM is achieved for the four considered batteries, without sacrificing fuel economy. While in Case 2, only two controller gains of the linear controller for the CS mode, along with the related calculations, are distributed into the BSC. Battery CSM is achieved for the considered batteries, while compromising some fuel economy (less than 3%) compared to the centralized control case. This shows a tradeoff between the simplicity of the BSC and the achievable fuel economy. In order to maintain the fuel economy of the vehicle with centralized control, the distributed control architecture in Case 1 is preferred.

In the distribution control Case 2, one can make a tradeoff between desired fuel economy and achievable battery CSM. If the desired fuel economy is the same as that of the centralized control, only one battery with parameter, B_s = 1.71e-5, satisfies the performance requirement and can be applied to the PHEV. If the desired fuel economy is within 3% of degradation of the fuel economy that is achievable by the centralized controller, then, all the four batteries satisfy the performance requirement and can be swapped with each other to create PHEV product variant.

At present, the battery is modeled as an integrator with only one parameter, which represents the energy capacity. If the current battery is replaced with a battery of a different type, instead of only changing the controller gains in the BSC, the controller structure in the BSC can also be redesigned to accommodate the new dynamics of the battery hardware. The BSC can then communicate with the VSC to achieve optimal fuel economy. This can be a topic for continuing research.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this dissertation, distributed controller design to achieve component swapping modularity (CSM) in control systems is investigated. With the proliferation of low cost electronics, many control system components, can now incorporate microcontrollers, which have communication interfaces and can perform component specific control functions. These components are referred to as "smart components". Bidirectional communication among the smart components can be used to facilitate CSM design. The networked control system is a natural setting to realize bidirectional communication. By distributing the centralized controller among a smart component and a system level controller, which are connected by a bidirectional communication network, swapping modularity of the smart component can be achieved. The distribution ensures that only the controller in the swappable component is dependent on the hardware parameters of this component and needs to be recalibrated if the component changes, while the system level controller remains the same for different component applications. Thus, whenever the component changes, simply plugging in a new component, which has component related control within its module, without redesign of the system controller, will provide

the required system performance. Such an approach can lead to significant savings in engineering time, as well as other economic benefits.

The previous 3-Step Method for control system design with CSM is reviewed. The 3-Step Method generates the distributed controllers by matching with a certain pre-computed centralized controller for each component application. The 3-Step Method sequentially addresses the two design objectives, system performance and CSM. Meanwhile, the model matching of transfer functions is limited to linear controller design.

The novel Direct Method for control system design with CSM, introduced in this dissertation, employs a bi-level optimization formulation to generate the distributed controllers directly. The bi-level optimization ensures that the resulting distributed controller gains will satisfy the CSM property, and the Direct Method enables the designer to address the two design objectives simultaneously. For multi-objective optimization, a simultaneous approach generally delivers a solution which is at least as good as, or better than, that of a sequential approach. Therefore, the Direct Method is expected to outperform the 3-Step Method. Moreover, the Direct Method relies on solving a nonlinear optimization problem to obtain the distributed controller gains. The nonlinearities of the controlled plant or the controller can be easily incorporated into the optimization formulation. Thus, compared to the precious 3-Step Method, it is a more general approach, which is applicable to the design of both linear and nonlinear controllers.

To illustrate the developments and conclusions, both the 3-Step Method and the Direct Method have been applied to the problem of throttle actuator CSM from the perspective of engine Idle Speed Control (ISC). For the 3-Step Method, both the unidirectional

communication case and the bidirectional communication case are studied. The results demonstrate that bidirectional communication improves CSM compared to unidirectional communication.

When solving the ISC problem using the 3-Step Method, we have also employed a heuristic approach to approximate the optimal centralized controller to facilitate the existence of the solution for actuator CSM as proposed in [25]. The results show that the 3-Step Method fails to provide actuator CSM when the actuator controller is first order or just gains, and only partial range of actuator CSM can be achieved when the actuator controller is second order or higher order. This could be a deficiency when component computing power and cost are limited. From both computation and calibration perspectives, simple controllers are generally more amenable to implementation in the smart component. In contrast, the Direct Method shows a significant improvement over the 3-Step Method in generating actuator CSM, with the same system performance requirement. The Direct Method can achieve partial actuator CSM when the actuator controller is just gains, and full range of CSM when the actuator controller is first order. In addition, the Direct Method permits the designer to trade-off the two design objectives, desired system performance and CSM, according to specific application scenarios.

The Direct Method is then applied to the important engineering problem of achieving battery CSM for PHEVs. A novel feedback-based controller for the CS mode is proposed to facilitate battery CSM design. The controller gains are obtained though optimization to achieve optimal fuel economy and optimal driving performance, while satisfying the constraints on closed loop system stability, battery charge sustainability and component reliability. The controller is designed with respect to the aggressive EPA US06 cycle, but

the simulation results demonstrate that the feedback based controller also achieves good fuel economy, good driving performance and charge sustainability over other driving cycles (e.g., the EPA UDDS and HWFET cycles).

The centralized supervisory controller for the PHEV is distributed into two parts: 1) the vehicle system controller (VSC), which is fixed with the vehicle, and 2) the battery control unit (BSC), which resides in the battery module and, thus, is swappable along with the battery. The controller distribution between the VSC and the BSC addresses the tradeoff between performance (generally highest when the controller is entirely within the BSC) and simplicity of the BSC implementation (desirable in terms of computing and calibration effort). For effective controller distribution, we proposed a method based on sensitivity analysis of the control signals with respect to the battery parameter. Only the controller gains that result in high sensitivity of the control signals, along with the corresponding calculations, are distributed into the BSC. Two distributed control cases are considered. In Case 1, four controller gains of the linear controller for the CS mode, along with related calculations, are distributed into the BSC. The rest of the controller for the CS mode, together with the CD mode and regenerative braking control, remain in the VSC. While in Case 2, only two controller gains of the linear controller for the CS mode, along with the related calculations, are distributed into the BSC.

The distributed controller gains are then obtained by solving a bi-level optimization problem. The bi-level formulation ensures that only the controller gains in the BSC depend on the battery parameters, while the VSC remains the same for different battery applications. The bi-level optimization is solved using the Augmented Lagrangian Decomposition (ALD) method. Compared to the general bi-level formulation, the

application of the ALD method is more complex, but provides more design freedom for each of the inner stage problems.

The results show that battery CSM is achieved for the four batteries considered, without compromising fuel economy compared to the centralized case, by the distributed controller in Case 1. While for the distributed controller in Case 2, battery CSM is achieved for the four batteries considered, while compromising some fuel economy (less than 3%) compared to the centralized control case. This shows a tradeoff between the simplicity of the BSC and the achievable fuel economy. In order to maintain the optimal fuel economy of the vehicle with centralized control, the distributed control architecture in Case 1 is preferred.

Some potential directions for future research are:

1. Distribution of adaptive and robust controllers to achieve CSM.
2. New optimization formulation that facilitates solution algorithms and exploits linear matrix inequalities (LMI) and bi-linear matrix inequalities (BMI) techniques.
3. Development of CSM design techniques and formulation for nonlinear controller design.
4. The treatment of the delay and data loss in the control network.
5. Swapping modularity design for multiple components concurrently, e.g., the engine and the battery at the same time in PHEVs.

APPENDICES

Appendix A: MATLAB Codes Used in Chapter IV and V

```
% generate the wheel power command from the driving cycles that are
% specified by vehicle speed
% driving profile (speed mph)
drivingcycle = 1; %1:US06 acceleration %2:EPA UDDS cycle %3:EPA Highway cycle
switch (drivingcycle)
      case 1
         %load US06 cycle
         load CYC_US06.mat;
      case 2
         %load EPA urban cycle
         load CYC_UDDS.mat;
      case 3
         %load EPA highway cycle
         load CYC_HWFET.mat;
end
time = cyc_mph(:,1);
time_step = cyc_mph(2,1)-cyc_mph(1,1);
time_final = 1*length(cyc_mph(:,2));
speed = cyc_mph(:,2)*1609.35/3600; %m/s
t = time;u = speed;
% distance
distance = sum(cyc_mph(:,2)*time_step/3600)
% plot the speed profile
figure;
plot(t,cyc_mph(:,2),'LineWidth',2.5);
ylabel('Vehicle speed profile (mph)','FontSize',14);
xlabel('Time (sec)','FontSize',14);
title('US06 cycle','FontSize',14);
% calculate wheel power command
Bs0 = [1.29 1.71 2.57 5.14]*1e-5;
m0 = [263 197 131 66]; % kg
```

```
n = length(Bs0);
Data_Pw(:,1) = t;
Data_Pwp(:,1) = t;
for i = 1:n
      m = m0(i)+1550+86; g = 9.81; f = 0.009;
      Fz = m*g; FA = 2; ro = 1.2; Cd = 0.335;
      v = u;
      nl = length(v);
      a(1) = 0;
      for j = 1:nl-1
         a(j+1) = (v(j+1)-v(j))/time_step;
      end
      F = m*a'+f.*Fz+FA*ro*Cd/2*v.^2;
      Pwc = F.*v/1000;  % unit from W to kW
      % nonnegative power command
      Pwcp = Pwc;
      nl = length(Pwcp);
      for k = 1:nl
         if Pwcp(k)<0
            Pwcp(k)=0;
         end
   end
   Data_Pw(:,i+1) = Pwc;
   Data_Pwp(:,i+1) = Pwcp;
   Data_Pw_sum(i) = sum(Pwc)/3600; % kWs -> kWh
end
% save data
save  Pw_US6 Data_Pw
save  Pwp_US6 Data_Pwp
```

```
% Centralized optimal supervisory controller for the CS mode for the PHEV
% it includes three files: main file, objective function, and constraint function
% main file
clear all; clc; close all
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl max_Ped con_max soca Pwa
global Bs b_scale d_scale soc_d soc_i socfmin t Pwc Pe_o fc_o soc_f cost_driv cost_bat
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb MPG E_factor belta iter Pe_chg_max
global eff_e eff_b eff_m Pe_c soc_max_deviation socfmax socfmin
% parameters
eff_m = 0.85; eff_b = 0.9; eff_e = 0.85;
Bs0 = [1.29 1.71 2.57 5.14]*1e-5;
n = length(Bs0);
soc_d0 = [0.2522 0.2686 0.30140.4];
```

```
soc_i0 = soc_d0;
socfmin0 = soc_d0*(1-0.02);
socfmax0 = soc_d0*(1+0.02);
Pe_chg_max = 5;
Pemax = 50; Pemin = 0; Pbmax = 110; Pbmin = -Pbmax;
socmax = 0.9; socmin = 0.2;

% % penalty weights in the objective function
% Bsw = Bs0*1e-3; % battery parameter w.r.t. power with unit Watt
% PUC = 1000*3.8*0.75; % petroleum unit change 1 gallon = 1000*3.8*0.75 gram = 2850 gram
% PEF = 82049*3600; % petroleum-equivalency factor is 82,049 Watt-hours per gallon charged
by external source
% E_factor = PUC/PEF./Bsw % equivalent factor bost_bat -> fuel in gram
E_factor = [747.961694157487,564.251804364420,375.436025472046,187.718012736023;];
belta = 100;
% simulation input
load Pwp_US6
t = Data_Pwp(:,1); Pwcp = Data_Pwp(:,2:n+1);
% engine OOP Line
Peo1 = [-1e50 0
636.074561681784,1908.22368504535,2532.93977241139,3805.08889577496,4441.1634574567
4,5077.23801913853,5713.31258082031,7067.00135167805,8079.15499795419,9687.28779157
751,11074.7254292459,12462.1630669143,14076.6424080615,14938.9635800201,17078.55938
68772,19218.1551937343,20080.4763656929,22596.1512571444,25598.7355463260,25967.005
6798283,30098.2531655800,34588.2580625944,39087.7756818483,43587.2933011022,48086.8
109203560,57076.3334366244,1e50];
efupro1 = [-1e5 0
0.544464471404623,0.297323633726156,0.267625786993999,0.236336134222391,0.227372851
657720,0.220655402755786,0.227468737552148,0.221746101637281,0.215755236833728,0.21
8996281068938,0.213339826354583,0.217609092854815,0.213136052833294,0.220593616307
321,0.214732397324911,0.217125939401324,0.218002796361895,0.219494016638424,0.23062
2328564479,0.223539058433262,0.237290847435876,0.245393103770643,0.254804982536398,
0.265585658646697,0.265267747140208,0.267803467385868,1e5];
Pe_o = Peo1;
fc_o_gWh = efupro1;
fc_o = fc_o_gWh/3600.*Pe_o; % unit: g/W/h --> g/s

% initial conditions
% --------- (1)
% k1 = 978; k2 = 3;k3 = 2;
% n1 = 0/eff_e/eff_m; ke = 0; n2 = 1/eff_b/eff_m; kb = 0;
% x0 = [k1 k2 k3 n1 ke n2 kb];
% --------- (2)
result_i = [1.29E-05       1.71E-05          2.57E-05          5.14E-05
```

```
27.58036999     28.163226       28.78317646     29.39853694
0.005043952     0.005372        0.006028001     0.008000004
0.038664425     0.03754035      0.036244665     0.034351765
917.4164857     1000    466.8684947     222.2322524
1.590840229     1.343633326     0.930707782     0.496792457
1.816356715     1.816356715     1.816356715     1.816356555
0.006166201     0.006166201     0.006166201     0.006166204
0.057388405     0.057388405     0.057388405     0.057388405
1.23050642      1.23050642      1.23050642      1.23050659
0.051198011     0.051198011     0.051198011     0.051198011];
ij = 2;
k1 = result_i(5,ij);
k2 = result_i(6,ij);
k3 = result_i(7,ij);
n1 = result_i(8,ij);
ke = result_i(9,ij);
n2 = result_i(10,ij);
kb = result_i(11,ij);
x0 = [k1 k2 k3 n1 ke n2 kb];
% variable bounds
k1l = 0; k2l = 0; k3l = 0;
n1l = 0; kel = 0; n2l = 1; kbl = 0;
k1u = 1000; k2u = 5; k3u = 10;
n1u = 1; keu = 1; n2u = 2; kbu = 1;
lb = [k1l k2l k3l n1l kel n2l kbl];
ub = [k1u k2u k3u n1u keu n2u kbu];
% scale to [-1, 1]
d_scale = 2./(ub-lb);
b_scale = -(ub+lb)./(ub-lb);
% new scale
lb = -ones(1,7);
ub = ones(1,7);
options = optimset('TolX',1e-6,'TolFun',1e-6, 'MaxFunEvals',1e6,'Display','iter','LargeScale','off');
clr = {'r--','g-', 'b:', 'm-.'};
for iter = 1:n
   x0 = result_i(5:11,iter)';
   x0 = d_scale.*x0+b_scale;
   Bs = Bs0(iter);
   Pwc = Pwcp(:,iter);
   soc_d = soc_d0(iter); soc_i = soc_i0(iter);
   socfmin = socfmin0(iter); socfmax = socfmax0(iter);
   [xi, h, exitflag] = fmincon(@objfun_clr_nw,x0,[],[],[],[],lb,ub,@confun_clr_nw,options);
   x = (xi-b_scale)./d_scale;
   result_data(:,iter) = [Bs, MPG, cost_bat, cost_driv, x];
```

```
    con_max_all(iter) = con_max;
    soc_max_dev(iter) = soc_max_deviation;
    figure(10+3);
    plot(t,soca,clr[66],'LineWidth',2.5)
    ylabel('soc','FontSize',12.5)
    xlabel('Time (sec)','FontSize',12.5);
    legend('Bs = 1.29e-5','Bs = 1.71e-5', 'Bs = 2.57e-5', 'Bs = 5.14e-5');
    hold on;
    axis tight;
    figure(10+4);
    plot(t,Pwc-Pwa,'b-o');
    xlabel('Time (sec)','FontSize',12.5);
    hold on;
    axis tight;
    figure(10+5);
    plot(t,Pb,'g--',t,Pe,'b-',t,Pwc,'r:','LineWidth',2.5)
    legend('P_{b}','P_{e}','P_{w,cmd}')
    xlabel('Time (sec)','FontSize',12.5);
    hold on;
    axis tight;
    xo = xi;
end
```

```
% Centralized optimal supervisory controller for the CS mode for the PHEV
% objective function
function f = objfun_clr_nw(xi)
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl fuel max_Ped soca Pwa
global Bs b_scale d_scale soc_d soc_i t Pwc Pe_o fc_o soc_f cost_driv cost_bat
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb MPG E_factor belta iter
global eff_e eff_b eff_m Pe_c soc_max_deviation
% scale of the variables
x = (xi-b_scale)./d_scale;
% x0 = [k1 k2 k3 n1 ke n2 kb];
k1 = x(1); k2 = x(2); k3 = x(3); n1 = x(4); ke = x(5); n2 = x(6); kb = x(7);
k4 = -k1/eff_b*eff_e;
k5 = -k2/eff_b*eff_e;
k6 = 0;
K = [k1 k2 k3;k4 k5 k6];
K1 = K(1,:);
K2 = K(2,:);
% closed loop system
Adiv = (1-kb*ke)+((1-kb*ke)==0)*1e-8;
```

```
Acl = [Bs*(K2+kb*K1)/Adiv; 1 0 0; -
eff_m*eff_e*(K1+ke*K2)/Adiveff_m*eff_b*(K2+kb*K1)/Adiv];
Bcl = [Bs*(n2+kb*n1)/Adiv; 0; 1-eff_m*eff_e*(n1+ke*n2)/Adiv-
eff_m*eff_b*(n2+kb*n1)/Adiv];
Ccl = [1 0 0; eff_e*(K1+ke*K2)/Adiv; eff_b*(K2+kb*K1)/Adiv];
Dcl = [0; eff_e*(n1+ke*n2)/Adiv; eff_b*(n2+kb*n1)/Adiv];
sys1 = ss(Acl,Bcl,Ccl,Dcl);
% simulation
y = lsim(sys1,Pwc,t);
soca = soc_d*ones(size(y(:,1)))-y(:,1);
Pe = y(:,2);
Pb = y(:,3);
Pwa = eff_m*(Pb+Pe);
figure(1);
plot(t,soca,'LineWidth',2.5)
ylabel('soc','FontSize',12.5)
xlabel('time (sec)','FontSize',12.5)
figure(2);
plot(t,Pb,'b--',t,Pe,'g-','LineWidth',2.5)
legend('Pb','Pe')
xlabel('time (sec)','FontSize',12.5)
% calculate constraints related variables
poles_cl = eig(Acl);
% soc limit
soc_min = min(soca); soc_max = max(soca);
soc_f = soca(length(soca));
soc_max_deviation = min(soca)-soc_d;
% power limit
Pe_min = min(Pe); Pe_max = max(Pe);
Pb_min = min(Pb); Pb_max = max(Pb);
% Pe_dot
for i = 1:length(t)-1
   Ped(i) = Pe(i+1)-Pe(i);
end
max_Ped = max(Ped);
% objective function
% % engine OOP Line
Pe_c = Pe/eff_e*1e3;
fuel0 = interp1(Pe_o,fc_o,Pe_c,'linear');
fuel = sum(fuel0);
cost_bat = (soc_d - soc_f);
cost_driv = sum(abs(Pwc-Pwa));
alpha = E_factor(iter);
% MPG
```

```
% equivalent fuel consumption gram -> gallon
FUEL = (fuel+alpha*cost_bat)/(1000*3.8*0.75);
distance_US6 = 8.008; % US06
MPG = distance_US6./FUEL; % mile per gallon
f = fuel+alpha*cost_bat+belta*cost_driv;
```

```
% Centralized optimal supervisory controller for the CS mode for the PHEV
% constraint function
function [c,ceq] = confun_clr_nw(x)
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl max_Ped Pe_chg_max
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb con_max soc_f iter socfmin socfmax
% inequalities c <= 0
c(1) = Pe_max - Pemax;
c(2) = Pemin - Pe_min;
c(3) = Pb_max - Pbmax;
c(4) = Pbmin - Pb_min;
c(5) = soc_max - socmax;
c(6) = socmin - soc_min;
c(7) = max_Ped - Pe_chg_max;
np = length(poles_cl);
for i = 1:np
   c(7+i) = real(poles_cl(i));
end
c(8+np) = socfmin - soc_f;
c(9+np) = soc_f - socfmax;
con_max = max(c);
% equality constraints
ceq = [];
```

```
% Polynomial fit of controller gains
Bs = [1.29 1.71 2.57 5.14]*1e-5;  % battery parameter w.r.t. power with unit Watt
n = length(Bs)
E_factor = [747.961694157487,564.251804364420,375.436025472046,187.718012736023;];
% input
load Pwp_US6
t = Data_Pwp(:,1);
Pwcp = Data_Pwp(:,2:n+1);
soc_d0 = [0.25 0.27 0.3 0.4];
soc_i0 = soc_d0;
val = 1:4;
```

```
eff_m = 0.85; eff_b = 0.9; eff_e = 0.85;
result_data = [1.29E-05 1.71E-05 2.57E-05 5.14E-05
27.58038771 28.17052855 28.78202658 29.39853584
0.005044 0.005371527 0.006027976 0.008
2.06E-05 7.15E-06 2.19E-06 0.000218312
917.5031731 996.9141767 465.4384124 222.1987975
1.590831582 1.295020006 0.929342823 0.496768346
1.815835659 3.922687567 1.571820202 1.818680108
0.006180712 0.059858397 0.005963246 0.006166723
0.057398264 0.014262721 0.055456965 0.057387406
1.230490123 1.233048302 1.233092198 1.230516879
0.051198195 0.012722171 0.049466248 0.051187345];
% fit the controller gains
Bs0 = result_data(1,:);
k1 = result_data(5,:); k2 = result_data(6,:); k3 = result_data(7,:);
n1 = result_data(8,:); ke = result_data(9,:);
n2 = result_data(10,:); kb = result_data(11,:);
% parameters from linear fitting
polyn = 4;
f_k1 = polyfit(Bs0(val),k1(val),polyn);
f_k2 = polyfit(Bs0(val),k2(val),polyn);
f_k3 = polyfit(Bs0(val),k3(val),polyn);
f_n1 = polyfit(Bs0(val),n1(val),polyn);
f_ke = polyfit(Bs0(val),ke(val),polyn);
f_n2 = polyfit(Bs0(val),n2(val),polyn);
f_kb = polyfit(Bs0(val),kb(val),polyn);
% unscaled linear fitted values
fk1 = polyval(f_k1,Bs0);
fk2 = polyval(f_k2,Bs0);
fk3 = polyval(f_k3,Bs0);
fn1 = polyval(f_n1,Bs0);
fke = polyval(f_ke,Bs0);
fn2 = polyval(f_n2,Bs0);
fkb = polyval(f_kb,Bs0);
% scalling factor
k1_sca = 10000; k2_sca = 10; k3_sca = 100; n1_sca = 1;
ke_sca = 1; n2_sca = 10; kb_sca = 1;
% scalled optimal value
k1_s = k1/k1_sca;
k2_s = k2/k2_sca;
k3_s = k3/k3_sca;
n1_s = n1/n1_sca;
ke_s = ke/ke_sca;
n2_s = n2/n2_sca;
```

```
kb_s = kb/kb_sca;
% scalled fitted value
fk1_s = fk1/k1_sca;
fk2_s = fk2/k2_sca;
fk3_s = fk3/k3_sca;
fn1_s = fn1/n1_sca;
fke_s = fke/ke_sca;
fn2_s = fn2/n2_sca;
fkb_s = fkb/kb_sca;
% plot actual and fitted scaled controller gains
figure('Name','fitting controller gains')
plot(Bs0,k1_s,'bo', 'LineWidth',2.5);
hold on;
plot(Bs0,fk1_s,'b-.', 'LineWidth',2.5);
hold on;
plot(Bs0,k2_s,'md', 'LineWidth',2.5);
hold on;
plot(Bs0,fk2_s,'m-', 'LineWidth',2.5);
hold on;
plot(Bs0,k3_s,'gh', 'LineWidth',2.5);
hold on;
plot(Bs0,fk3_s,'g--', 'LineWidth',2.5);
hold on;
plot(Bs0,n1_s,'mp', 'LineWidth',2.5);
hold on;
plot(Bs0,fn1_s,'m:', 'LineWidth',2.5);
hold on;
plot(Bs0,ke_s,'cs', 'LineWidth',2.5);
hold on;
plot(Bs0,fke_s,'c-', 'LineWidth',2.5);
hold on;
plot(Bs0,n2_s,'k>', 'LineWidth',2.5);
hold on;
plot(Bs0,fn2_s,'k:', 'LineWidth',2.5);
hold on;
plot(Bs0,kb_s,'r<', 'LineWidth',2.5);
hold on;
plot(Bs0,fkb_s,'r-', 'LineWidth',2.5);
hold on;
xlabel('\rm Battery parameter \it B_{s}','FontSize',12.5);
% title('1^{st} order polynomial fit','FontSize',12.5);
title('4^{th} order polynomial fit','FontSize',12.5);
figure('Name','controller gains')
plot(Bs0,k1_s,'-.bo','LineWidth',2.5);
```

```
hold on;
plot(Bs0,k2_s,'-md', 'LineWidth',2.5);
hold on;
plot(Bs0,k3_s,'--gh', 'LineWidth',2.5);
hold on;
plot(Bs0,n1_s,':mp', 'LineWidth',2.5);
hold on;
plot(Bs0,ke_s,'-cs', 'LineWidth',2.5);
hold on;
plot(Bs0,n2_s,':k>', 'LineWidth',2.5);
hold on;
plot(Bs0,kb_s,'-r<', 'LineWidth',2.5);
legend('\it k1/1e4','\it k2/10','\it k3/1e2','\it n1','\it ke','\it n2/10','\it kb')
xlabel('\rm Battery parameter \it B_{s}','FontSize',12.5);

for i = 1:n
   Bs = Bs0(i);
   soc_d = soc_d0(i); soc_i = soc_i0(i);
   k1 = fk1(i); k2 = fk2(i); k3 = fk3(i); n1 = fn1(i);
   ke = fke(i); n2 = fn2(i); kb = fkb(i);
   k4 = -k1/eff_b*eff_e; k5 = -k2/eff_b*eff_e; k6 = 0;
   K = [k1 k2 k3;k4 k5 k6];
   K1 = K(1,:); K2 = K(2,:);
%    closed loop system
   Adiv = (1-kb*ke)+((1-kb*ke)==0)*1e-6;
   Acl = [Bs*(K2+kb*K1)/Adiv; 1 0 0; -eff_m*eff_e*(K1+ke*K2)/Adiv-
eff_m*eff_b*(K2+kb*K1)/Adiv];
   Bcl = [Bs*(n2+kb*n1)/Adiv; 0; 1-eff_m*eff_e*(n1+ke*n2)/Adiv-
eff_m*eff_b*(n2+kb*n1)/Adiv];
   Ccl = [1 0 0; 0 1 0; 0 0 1];
   Dcl = [0; 0; 0];
   sys1 = ss(Acl,Bcl,Ccl,Dcl);
   y = lsim(sys1,Pwc,t);
   z1 = y(:,1); z2 = y(:,2); z3 = y(:,3);
   z1_data(:,i) = z1; z2_data(:,i) = z2; z3_data(:,i) = z3;
   % mean
   z1m(i) = mean(mean(z1_data));
   z2m(i) = mean(mean(z2_data));
   z3m(i) = mean(mean(z3_data));
   Pwm(i) = mean(mean(Pwc));
end
```

```
% Sensitivity analysis of the obtained centralized controller gains
```

```
syms bs z1 z2 z3 Pw_cmd
syms k1 k2 k3 k4 k5 k6 n1 ke n2 kb
K1 = [k1, k2, k3]; K2 = [k4, k5, k6]; z = [z1 z2 z3]';
% control signal
Pe_cmd = (K1+ke*K2)/(1-kb*ke)*z+(n1+ke*n2)/(1-kb*ke)*Pw_cmd;
Pb_cmd = (K2+kb*K1)/(1-kb*ke)*z+(n2+kb*n1)/(1-kb*ke)*Pw_cmd;
% different Pe_cmd and Pb_cmd w.r.t. x
diff_Pe_k1 = diff(Pe_cmd,'k1')
diff_Pe_k2 = diff(Pe_cmd,'k2')
diff_Pe_k3 = diff(Pe_cmd,'k3')
diff_Pe_k4 = diff(Pe_cmd,'k4')
diff_Pe_k5 = diff(Pe_cmd,'k5')
diff_Pe_k6 = diff(Pe_cmd,'k6')
diff_Pe_n1 = diff(Pe_cmd,'n1')
diff_Pe_ke = diff(Pe_cmd,'ke')
diff_Pe_n2 = diff(Pe_cmd,'n2')
diff_Pe_kb = diff(Pe_cmd,'kb')
diff_Pb_k1 = diff(Pb_cmd,'k1')
diff_Pb_k2 = diff(Pb_cmd,'k2')
diff_Pb_k3 = diff(Pb_cmd,'k3')
diff_Pb_k4 = diff(Pb_cmd,'k4')
diff_Pb_k5 = diff(Pb_cmd,'k5')
diff_Pb_k6 = diff(Pb_cmd,'k6')
diff_Pb_n1 = diff(Pb_cmd,'n1')
diff_Pb_ke = diff(Pb_cmd,'ke')
diff_Pb_n2 = diff(Pb_cmd,'n2')
diff_Pb_kb = diff(Pb_cmd,'kb')
% fitted controller gains
% from fitted_central_network_controller_11_20.m (run this file first)
% Linearized controller gains w.r.t. battery paramter Bs (bs = delt_Bs)
% 4th order poly
k1 = f_k1(1)*bs^4+f_k1(2)*bs^3+f_k1(3)*bs^2+f_k1(4)*bs+f_k1(5);
k2 = f_k2(1)*bs^4+f_k2(2)*bs^3+f_k2(3)*bs^2+f_k2(4)*bs+f_k2(5);
k3 = f_k3(1)*bs^4+f_k3(2)*bs^3+f_k3(3)*bs^2+f_k3(4)*bs+f_k3(5);
n1 = f_n1(1)*bs^4+f_n1(2)*bs^3+f_n1(3)*bs^2+f_n1(4)*bs+f_n1(5);
ke = f_ke(1)*bs^4+f_ke(2)*bs^3+f_ke(3)*bs^2+f_ke(4)*bs+f_ke(5);
n2 = f_n2(1)*bs^4+f_n2(2)*bs^3+f_n2(3)*bs^2+f_n2(4)*bs+f_n2(5);
kb = f_kb(1)*bs^4+f_kb(2)*bs^3+f_kb(3)*bs^2+f_kb(4)*bs+f_kb(5);
% %linear
% k1 = f_k1(1)*bs+f_k1(2);
% k2 = f_k2(1)*bs+f_k2(2);
% k3 = f_k3(1)*bs+f_k3(2);
% n1 = f_n1(1)*bs+f_n1(2);
% ke = f_ke(1)*bs+f_ke(2);
```

```
% n2 = f_n2(1)*bs+f_n2(2);
% kb = f_kb(1)*bs+f_kb(2);
k4 = -k1/eff_b*eff_e; k5 = -k2/eff_b*eff_e; k6 = 0;
% substitute the fitted controller gains
diff_Pe_k1 = subs(diff_Pe_k1);
diff_Pe_k2 = subs(diff_Pe_k2);
diff_Pe_k3 = subs(diff_Pe_k3);
diff_Pe_n1 = subs(diff_Pe_n1);
diff_Pe_ke = subs(diff_Pe_ke);
diff_Pe_n2 = subs(diff_Pe_n2);
diff_Pe_kb = subs(diff_Pe_kb);
diff_Pb_k1 = subs(diff_Pb_k1);
diff_Pb_k2 = subs(diff_Pb_k2);
diff_Pb_k3 = subs(diff_Pb_k3);
diff_Pb_n1 = subs(diff_Pb_n1);
diff_Pb_ke = subs(diff_Pb_ke);
diff_Pb_n2 = subs(diff_Pb_n2);
diff_Pb_kb = subs(diff_Pb_kb);
% polyn = polynomial order
% use the first order derivative
x_bs = [f_k1(polyn), f_k2(polyn), f_k3(polyn), f_n1(polyn), f_ke(polyn), f_n2(1), f_kb(polyn)];
% diffrenciate above differentials w.r.t. bs
k1_stv_Pe = diff_Pe_k1*x_bs(1);
k2_stv_Pe = diff_Pe_k2*x_bs(2);
k3_stv_Pe = diff_Pe_k3*x_bs(3);
n1_stv_Pe = diff_Pe_n1*x_bs(4);
ke_stv_Pe = diff_Pe_ke*x_bs(5);
n2_stv_Pe = diff_Pe_n2*x_bs(6);
kb_stv_Pe = diff_Pe_kb*x_bs(7);
k1_stv_Pb = diff_Pb_k1*x_bs(1);
k2_stv_Pb = diff_Pb_k2*x_bs(2);
k3_stv_Pb = diff_Pb_k3*x_bs(3);
n1_stv_Pb = diff_Pb_n1*x_bs(4);
ke_stv_Pb = diff_Pb_ke*x_bs(5);
n2_stv_Pb = diff_Pb_n2*x_bs(6);
kb_stv_Pb = diff_Pb_kb*x_bs(7);
% substitute the values for bs0, mean value of z and Pw_cmd
Bs0 = [1.29 1.71 2.57 5.14]*1e-5;
z1 = mean(z1m); z2 = mean(z2m); z3 = mean(z3m); Pw_cmd = mean(Pwm);
bs = mean(Bs0);
% substitute the values
diff_Pe_k1 = subs(diff_Pe_k1);
diff_Pe_k2 = subs(diff_Pe_k2);
diff_Pe_k3 = subs(diff_Pe_k3);
```

```
diff_Pe_n1 = subs(diff_Pe_n1);
diff_Pe_ke = subs(diff_Pe_ke);
diff_Pe_n2 = subs(diff_Pe_n2);
diff_Pe_kb = subs(diff_Pe_kb);
diff_Pb_k1 = subs(diff_Pb_k1);
diff_Pb_k2 = subs(diff_Pb_k2);
diff_Pb_k3 = subs(diff_Pb_k3);
diff_Pb_n1 = subs(diff_Pb_n1);
diff_Pb_ke = subs(diff_Pb_ke);
diff_Pb_n2 = subs(diff_Pb_n2);
diff_Pb_kb = subs(diff_Pb_kb);
k1_stv_Pe = subs(k1_stv_Pe);
k2_stv_Pe = subs(k2_stv_Pe);
k3_stv_Pe = subs(k3_stv_Pe);
n1_stv_Pe = subs(n1_stv_Pe);
ke_stv_Pe = subs(ke_stv_Pe);
n2_stv_Pe = subs(n2_stv_Pe);
kb_stv_Pe = subs(kb_stv_Pe);
k1_stv_Pb = subs(k1_stv_Pb);
k2_stv_Pb = subs(k2_stv_Pb);
k3_stv_Pb = subs(k3_stv_Pb);
n1_stv_Pb = subs(n1_stv_Pb);
ke_stv_Pb = subs(ke_stv_Pb);
n2_stv_Pb = subs(n2_stv_Pb);
kb_stv_Pb = subs(kb_stv_Pb);
x_stv = [k1_stv_Pe, k2_stv_Pe, k3_stv_Pe, n1_stv_Pe, ke_stv_Pe, n2_stv_Pe, kb_stv_Pe;
k1_stv_Pb, k2_stv_Pb, k3_stv_Pb, n1_stv_Pb, ke_stv_Pb, n2_stv_Pb, kb_stv_Pb];
P_x_stv = [diff_Pe_k1, diff_Pe_k2, diff_Pe_k3, diff_Pe_n1, diff_Pe_ke, diff_Pe_n2, diff_Pe_kb;
   diff_Pb_k1, diff_Pb_k2, diff_Pb_k3, diff_Pb_n1, diff_Pb_ke, diff_Pb_n2, diff_Pb_kb];
% plot the gain sensitivity using bars
xplot = 1:7
figure('Name','dP/dx');
bar(xplot,abs(P_x_stv)', 'group');
legend('\delta\it P_{e} / \delta\it x', '\delta\it P_{b} / \delta\it x')
axis tight
set(gca,'XTickLabel',{'k1','k2', 'k3','n1','ke', 'n2','kb'})
figure('Name','dP/dBs');
bar(xplot,abs(x_stv)', 'group');
% xlabel('Controller gains','FontSize',12.5);
legend('\it x_{stv, Pe}', '\it x_{stv, Pb}')
axis tight
set(gca,'XTickLabel',{'k1','k2', 'k3','n1','ke', 'n2','kb'})
x_stv = abs(x_stv(1,:))+abs(x_stv(2,:));
figure('Name','sum dP/dBs');
```

```
bar(xplot,x_stv');
% xlabel('Controller gains','FontSize',12.5);
ylabel('\it x_{stv}','FontSize',12.5);
axis tight
set(gca,'XTickLabel',{'k1','k2', 'k3','n1','ke', 'n2','kb'})
figure('Name','dx/dBs');
bar([1:7],abs(x_bs), 'group');
% xlabel('Controller gains','FontSize',12.5);
ylabel('\delta\it x / \delta\it B_{s}','FontSize',12.5)
axis tight
set(gca,'XTickLabel',{'k1','k2', 'k3','n1','ke', 'n2','kb'})
```

```
% Distributed controller for CS mode for the PHEV in distribution Case 1
% it includes three files: main file, objective function, and constraint function
% main file
clear all;clc; close all
% bi-level optimization using augmented lagrangian method
% outter stage (analytical solution)
% inner stage (fmincon)
global v w cc cc0 V W CC Data1 n m1 m2 iteri itero pf gama beta con_max
% v w (penalty weights); c c0 (discrepancy of the consistency constraints)
% n m (number of considered components, number of gains of the base controller)
% itero (iteration of the outer stage opt); iteri (inter # of inner stage)
% pf (vector of the penalty functions)
global d_scale b_scale yt k1 k2 k3 k4 k5 k6 n1 n2 ke kb soca Pe Pb Pwa
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl max_Ped fuel_e_g E_factor
belta
global Bs b_scale d_scale soc_d soc_i t Pwc Pe_o fc_o soc_f cost_driv cost_bat
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb iteri socfmin socfmax Pe_chg_max
global eff_e eff_b eff_m Pe_c MPG
% parameters
eff_m = 0.85; eff_b = 0.9; eff_e = 0.85;
Bs0 = [1.29 1.71 2.57 5.14]*1e-5;
n = length(Bs0);
soc_d0 = [0.2522 0.2686          0.3014  0.4];
soc_i0 = soc_d0;
socfmin0 = soc_d0*(1-0.02);
socfmax0 = soc_d0*(1+0.02);
Pe_chg_max = 5;
Pemax = 50; Pemin = 0;
Pbmax = 110; Pbmin = -Pbmax;
socmax = 0.9; socmin = 0.2;
```

```
% penalty weights in the objective function
E_factor = [747.961694157487,564.251804364420,375.436025472046,187.718012736023;];
belta = 100;
% updating parameters for the penalty function
gama = 0.2; beta = 2;
MFEval = 1e3; % maximum function evaluation number for the inner stage optimization
% input
load Pwp_US6
t = Data_Pwp(:,1);
Pwcp = Data_Pwp(:,2:n+1);
% engine OOP Line
Peo1 = [-1e50 0
636.074561681784,1908.22368504535,2532.93977241139,3805.08889577496,4441.1634574567
4,5077.23801913853,5713.31258082031,7067.00135167805,8079.15499795419,9687.28779157
751,11074.7254292459,12462.1630669143,14076.6424080615,14938.9635800201,17078.55938
68772,19218.1551937343,20080.4763656929,22596.1512571444,25598.7355463260,25967.005
6798283,30098.2531655800,34588.2580625944,39087.7756818483,43587.2933011022,48086.8
109203560,57076.3334366244,1e50];
efupro1 = [-1e20 0
0.544464471404623,0.297323633726156,0.267625786993999,0.236336134222391,0.227372851
657720,0.220655402755786,0.227468737552148,0.221746101637281,0.215755236833728,0.21
8996281068938,0.213339826354583,0.217609092854815,0.213136052833294,0.220593616307
321,0.214732397324911,0.217125939401324,0.218002796361895,0.219494016638424,0.23062
2328564479,0.223539058433262,0.237290847435876,0.245393103770643,0.254804982536398,
0.265585658646697,0.265267747140208,0.267803467385868,1e20];
Pe_o = Peo1;
fc_o_gWh = efupro1;
fc_o = fc_o_gWh/3600.*Pe_o; % unit: g/W/h --> g/s
% initial conditions
% ------ (1)
result_fit = [1.29E-05    1.71E-05          2.57E-05          5.14E-05
27.57925543    28.15637466    28.7816803     29.39857406
0.005044       0.005371945    0.006027631    0.007999528
4.17E-07       4.74E-06       7.75E-06       5.65E-05
953.2477222    953.2889945    435.0499285    225.5133684
1.591087265    1.33780788     0.918188024    0.498861964
0.049846337    1.816498924    4.448105781    6.502090769
0.006171547    0.006169865    0.006169202    0.006248172
0.057452116    0.057452161    0.057452117    0.057411594
1.230432321    1.230433849    1.23043456     1.230410262
0.051245869    0.051245971    0.051245517    0.051205999];
ij = 2;
k1 = result_fit(5,ij);
k2 = result_fit(6,ij);
```

```
k3 = result_fit(7,ij);
n1 = result_fit(8,ij);
ke = result_fit(9,ij);
n2 = result_fit(10,ij);
kb = result_fit(11,ij);
x00 = [k1 k2];
y00 = [k3, n1, ke, n2, kb];
m1 = length(y00); % number of gains of the base controller (# of y)
m2 = length(x00); % number of gains of the actuator controller (# of xi)
xvar = [y00 x00];
% variable bounds
k1l = 100; k2l = 0; k1u = 1000; k2u = 5;
k3l = k3*0.9; n1l = n1*0.9; kel = ke*0.9; n2l = n2*0.9; kbl = kb*0.9;
k3u = k3*1.1; n1u = n1*1.1; keu = ke*1.1; n2u = n2*1.1; kbu = kb*1.1;
lb = [k3l n1l kel n2l kbl k1l k2l];
ub = [k3u n1u keu n2u kbu k1u k2u];
% scale to [-1, 1]
d_scale = 2./(ub-lb);
b_scale = -(ub+lb)./(ub-lb);
% new scale
lb = -ones(1,m1+m2);
ub = ones(1,m1+m2);
xvar_scale = d_scale.*xvar+b_scale;
y0 = xvar_scale(1:m1); x0 = xvar_scale(m1+1:m1+m2);
y00 = xvar(1:m1); x00 = xvar(m1+1:m1+m2);
n_head = 4; %(Bs, MPG, cost_bat, cost_driv)
Data1 = zeros(n_head+m1+m2,n); % each column includes (Bs, fuel, cost_bat, cost_driv, y, x)
for j = 1:n
   Data1(:,j) = [zeros(1,n_head), y0, x0]';
   testa(:,j) = [zeros(1,n_head), y00, x00]'; % to see the initial actual value
end
tests = Data1; % to see the initial scaled value
Datay(:,1) = y00';
ebslon = 10^(-5);      %stop tolerance
%  ------------   penalty function ----------------
% initial penalty function weights
% V = zeros(n*m1,1); % linear penalty weights
% W = zeros(n*m1,iterom); % quadratic penalty weights
% the first columns of V and W are initial values of v and w,
%termed as v0,w0
V(1:n*m1,1) = zeros(m1*n,1); % v0
W(1:n*m1,1) = 2*ones(m1*n,1); % w0
% initial discrepancy of consistency constraints
cc = 0*y0';
```

```
% Discrepancy of consistency constraints  (n*m1,1);
for i = 1:n
   CC((m1*(i-1)+1):(m1*i),1) = cc;
end
pf = zeros(n,1); % calculated in objfuni.m
options = optimset('TolX',1e-6,'TolFun',1e-
6,'MaxFunEvals',MFEval,'Display','iter','LargeScale','off');
clr = {'r--','b:','g-','k-.','m-x'};
for itero = 1:200
   yt = y0;
   for iteri = 1:n
      x0i = [yt, Data1(n_head+m1+1:n_head+m1+m2,iteri)'];  % scaled value
      Bs = Bs0(iteri);
      Pwc = Pwcp(:,iteri);
      soc_d = soc_d0(iteri); soc_i = soc_i0(iteri);
      socfmin = socfmin0(iteri); socfmax = socfmax0(iteri);
      % penalty function
      % the previous step value of v and w.
      % note itero+1 is the current v and w.
      v = V((m1*(iteri-1)+1):(m1*iteri),itero)';
      w = W((m1*(iteri-1)+1):(m1*iteri),itero)';
      cc0 = CC((m1*(iteri-1)+1):(m1*iteri),itero);
      v = v+2*w.*w.*cc0';    % update of v
      for j = 1:m1           % update of w
         if cc(j)> gama*cc0(j)
            w(j) = beta*w(j);
         end
      end
      V((m1*(iteri-1)+1):(m1*iteri),itero+1) = v';
      W((m1*(iteri-1)+1):(m1*iteri),itero+1) = w';
      [xi, h, exitflag] = fmincon(@objfuni_2,x0i,[],[],[],[],lb,ub,@confuni_2,options);
      x_show = (xi-b_scale)./d_scale; % actual data
      Data(:,iteri+(itero-1)*n) = [Bs, MPG, cost_bat, cost_driv, x_show]';
      % update the solution only when inner loop converges
      if h < 1e3 && h > 0
         Data1(:,iteri) = [Bs, MPG, cost_bat, cost_driv, xi]';
      end
      con_max_all(iteri) = con_max;
      % penalty function
      % discrepancy of equality constraints
      for j = 1:m1
          cc(j) = Data1(j+n_head,iteri)-yt(j);
      end
      CC((m1*(iteri-1)+1):(m1*iteri),itero+1) = cc;
```

```
        figure(10*itero);
        plot(t,soca,clr{iteri},'LineWidth',2.5);
        hold on;
        ylabel('soc','FontSize',12.5);
        xlabel('time (sec)','FontSize',12.5);
        legend('Bs = 5e-6','Bs = 10e-6', 'Bs = 15e-6', 'Bs = 20e-6');
        axis tight
        figure(1+10*iteri);
        plot(t,Pwc,'r:','LineWidth',2.5);
        hold on;
        plot(t,Pe,'b-','LineWidth',2.5);
        hold on;
        plot(t,Pb,'g--','LineWidth',2.5);
        hold on;
        legend('P_{w,cmd}','P_{e}','P_{b}');
        xlabel('time (sec)','FontSize',12.5);
        axis tight
        figure(2+10*iteri);
        plot(t,Pwc-Pwa,'r:','LineWidth',2.5)
        xlabel('time (sec)','FontSize',12.5);
        axis tight
    end
    % analytical solution
    ys1 = zeros(1,m1);
    ys2 = ys1;
    ys3 = ys1;
    for ki = 1:n
        v = V((m1*(ki-1)+1):(m1*ki),itero+1)';
        w = W((m1*(ki-1)+1):(m1*ki),itero+1)';
        ys1 = ys1+w.*w.*Data1(n_head+1:n_head+m1,ki)';
        ys2 = ys2+v;
        ys3 = ys3+w.*w;
    end
    y0 = (ys1-1/2*ys2)./ys3;
    y0_show = (y0-b_scale(1:m1))./d_scale(1:m1);
    Datay(:,itero+1) = y0_show';
    % stop criteria
    error = max(abs((CC(:,itero+1))))
    cma = max(con_max_all)  % check the constraints feasibility
    if error < ebslon && cma < 1e-5
        break;
    end
end
```

```
% Distributed controller for CS mode for the PHEV in distribution Case 1
% objective function
function h = objfuni_2(xi)
global v w cc cc0 V W CC Data1 n m1 m2 iteri itero pf gama beta con_max
global d_scale b_scale yt k1 k2 k3 k4 k5 k6 n1 n2 ke kb soca Pe Pb Pwa
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl max_Ped fuel_e_g E_factor
belta
global Bs b_scale d_scale soc_d soc_i t Pwc Pe_o fc_o soc_f cost_driv cost_bat
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb iteri
global eff_e eff_b eff_m Pe_c MPG
% scale of the variables
x = (xi-b_scale)./d_scale;
k3 = x(1); n1 = x(2); ke = x(3); n2 = x(4);  kb = x(5);
k1 = x(6); k2 = x(7);
k4 = -k1/eff_b*eff_e;
k5 = -k2/eff_b*eff_e;
k6 = 0;
K = [k1 k2 k3;k4 k5 k6];
K1 = K(1,:); K2 = K(2,:);
% closed loop system
Adiv = (1-kb*ke)+((1-kb*ke)==0)*1e-8;
Acl = [Bs*(K2+kb*K1)/Adiv; 1 0 0; -eff_m*eff_e*(K1+ke*K2)/Adiv-
eff_m*eff_b*(K2+kb*K1)/Adiv];
Bcl = [Bs*(n2+kb*n1)/Adiv; 0; 1-eff_m*eff_e*(n1+ke*n2)/Adiv-
eff_m*eff_b*(n2+kb*n1)/Adiv];
Ccl = [1 0 0; eff_e*(K1+ke*K2)/Adiv; eff_b*(K2+kb*K1)/Adiv];
Dcl = [0; eff_e*(n1+ke*n2)/Adiv; eff_b*(n2+kb*n1)/Adiv];
sys1 = ss(Acl,Bcl,Ccl,Dcl);
% simulation
y = lsim(sys1,Pwc,t);
delt_soc = y(:,1);
soca = soc_d*ones(size(y(:,1)))-y(:,1);
Pe = y(:,2);
Pb = y(:,3);
Pwa = eff_m*(Pb+Pe);
figure(1);
plot(t,soca,'LineWidth',2.5)
ylabel('soc','FontSize',12.5)
xlabel('time (sec)','FontSize',12.5)
figure(2);
plot(t,Pb,'b--',t,Pe,'g-',t,Pwa,'r-o',t,Pwc,'c-.','LineWidth',2.5)
legend('Pb','Pe','Pw_a','Pw_{cmd}')
```

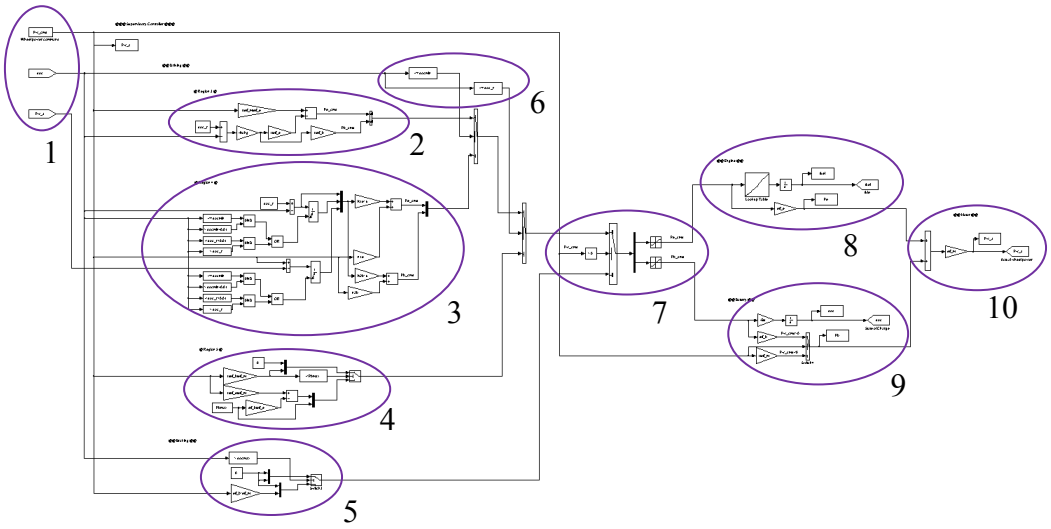
```
xlabel('time (sec)','FontSize',12.5)
% calculate constraints related variables
poles_cl = eig(Acl);
soc_min = min(soca); soc_max = max(soca);
soc_f = soca(length(soca));
Pe_min = min(Pe); Pe_max = max(Pe);
Pb_min = min(Pb); Pb_max = max(Pb);
% Pe_dot
for i = 1:length(t)-1
   Ped(i) = Pe(i+1)-Pe(i);
end
max_Ped = max(Ped);
% objective function
% % engine OOP Line
Pe_c = Pe/eff_e*1e3;
fuel0 = interp1(Pe_o,fc_o,Pe_c,'linear');
fuel = sum(fuel0);
cost_bat = (soc_d - soc_f);
cost_driv = sum(abs(Pwc-Pwa));
alpha = E_factor(iteri);
% MPG
% equivalent fuel consumption gram -> gallon
FUEL = (fuel+alpha*cost_bat)/(1000*3.8*0.75);
distance_US6 = 8.008; % US06
MPG = distance_US6./FUEL; % mile per gallon
f = fuel+alpha*cost_bat+belta*cost_driv;
% penalty function
for j = 1:m1
   cc(j,1) = xi(j)-yt(j);
end
% quadratic penalty function
QuaPen = (w.*cc')*(w.*cc')';
% Linear penalty function
LinPen = v*cc;
% objective function of the outer stage
pf(iteri) = LinPen+QuaPen;
h = f + pf(iteri);
```

```
% Distributed controller for CS mode for the PHEV in distribution Case 1
% constraint function
function [c,ceq] = confuni_2(xi)
global v w cc cc0 V W CC Data1 n m1 m2 iteri itero pf gama beta con_max
```

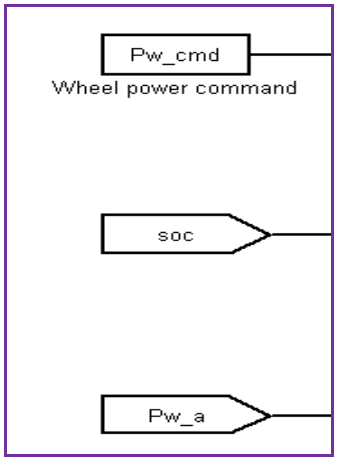
```
global Pb_max Pb_min Pe_max Pe_min soc_max soc_min poles_cl max_Ped fuel fuel_eq
global Bs b_scale d_scale soc_d soc_i t Pwc Pe_o fc_o soc_f socfmin socfmax
global Pemax Pemin Pbmax Pbmin socmax socmin c Pe Pb iteri Pe_chg_max
% inequalities c <= 0
c(1) = Pe_max - Pemax;
c(2) = Pemin - Pe_min;
c(3) = Pb_max - Pbmax;
c(4) = Pbmin - Pb_min;
c(5) = soc_max - socmax;
c(6) = socmin - soc_min;
c(7) = max_Ped - Pe_chg_max;
np = length(poles_cl);
for i = 1:np
   c(7+i) = real(poles_cl(i));
end
c(8+np) = socfmin - soc_f;
c(9+np) = soc_f - socfmax;
con_max = max(c);
% equality constraints
ceq = [];
```

Appendix B: SIMULINK Models Used in Chapter IV and V

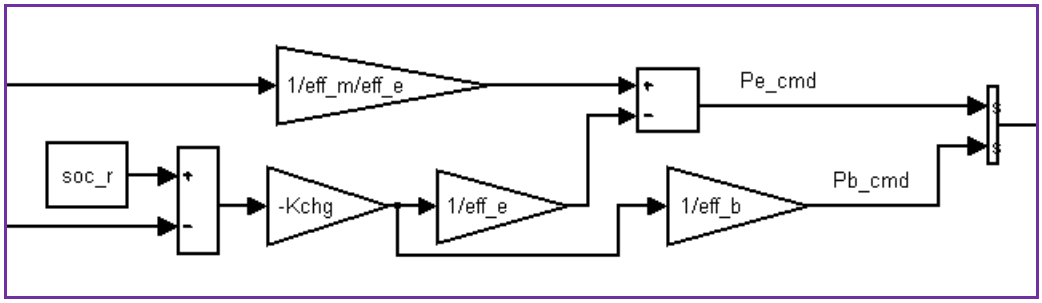
a. Vehicle model for the centralized control case



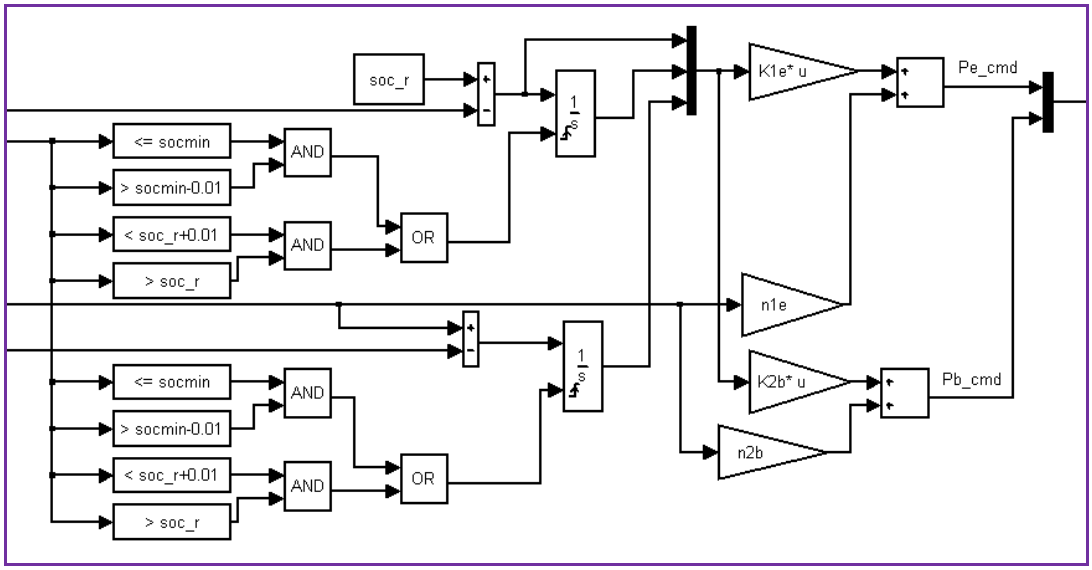
 # 1



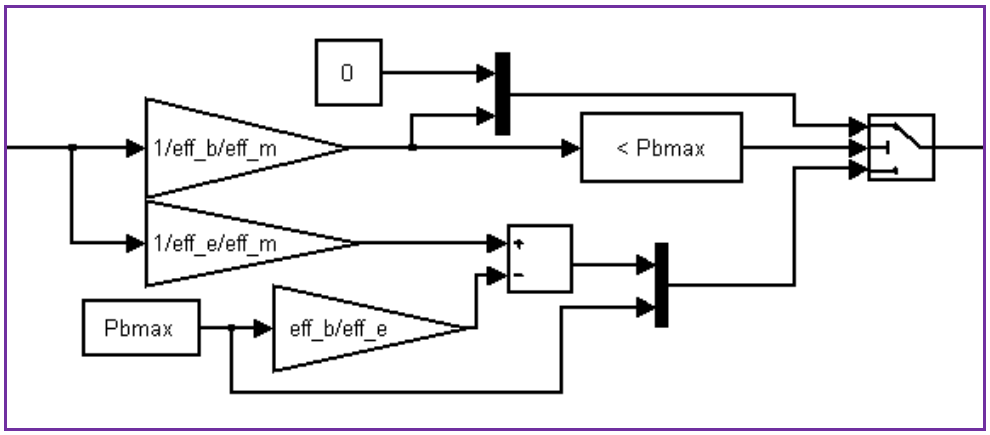
 # 2 Controller when $soc \in (0, soc_{min}]$ in the VSC



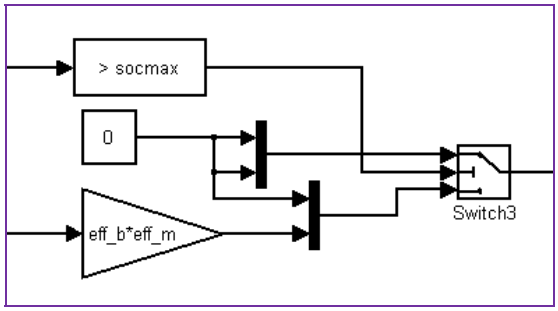
 # 3 CS mode controller in the VSC



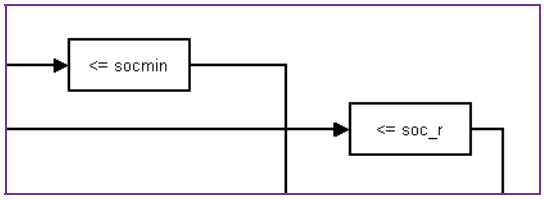
 # 4 CD mode controller in the VSC



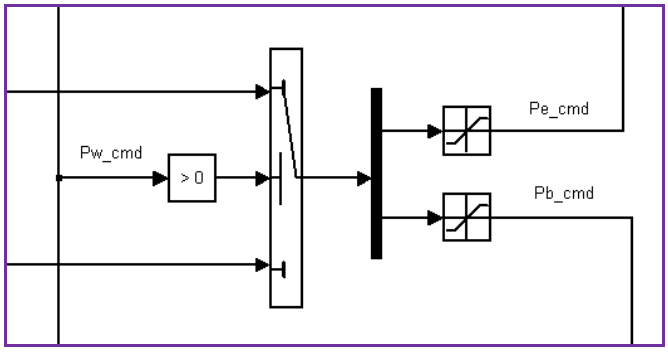
 # 5 Regenerative braking controller in the VSC



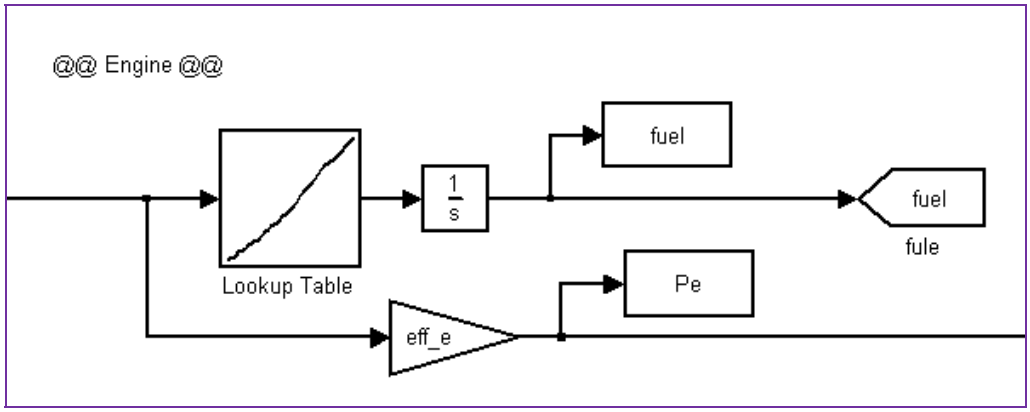
 # 6



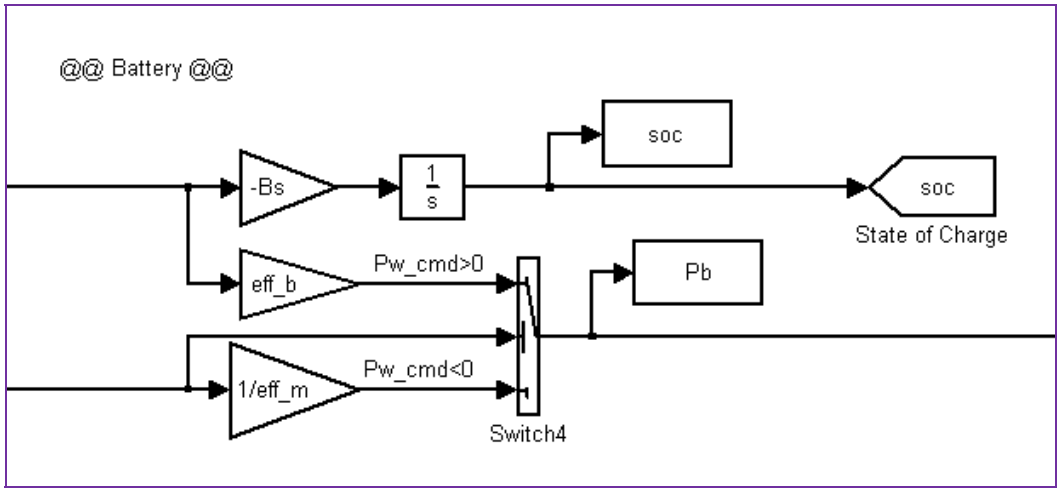
 # 7



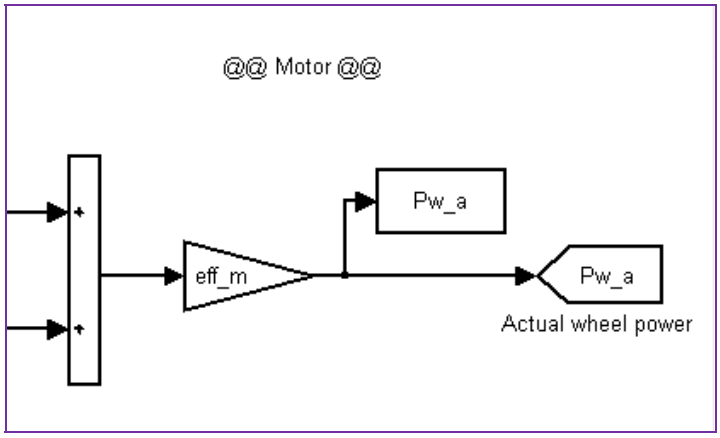
 # 8



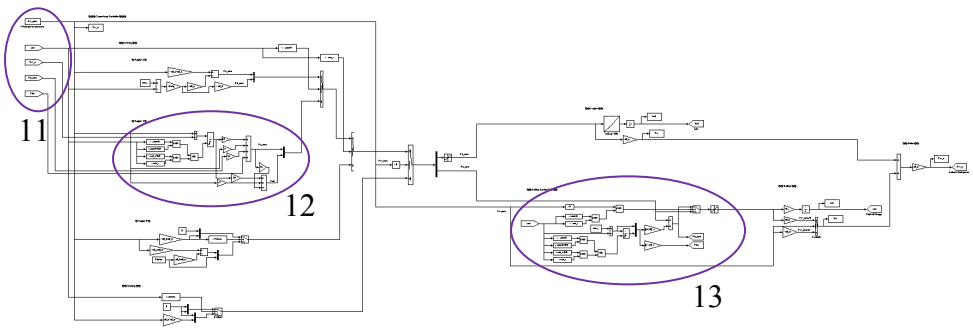
 # 9



 # 10

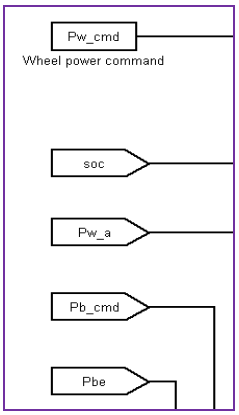


b. Vehicle model for the distributed control Case 1

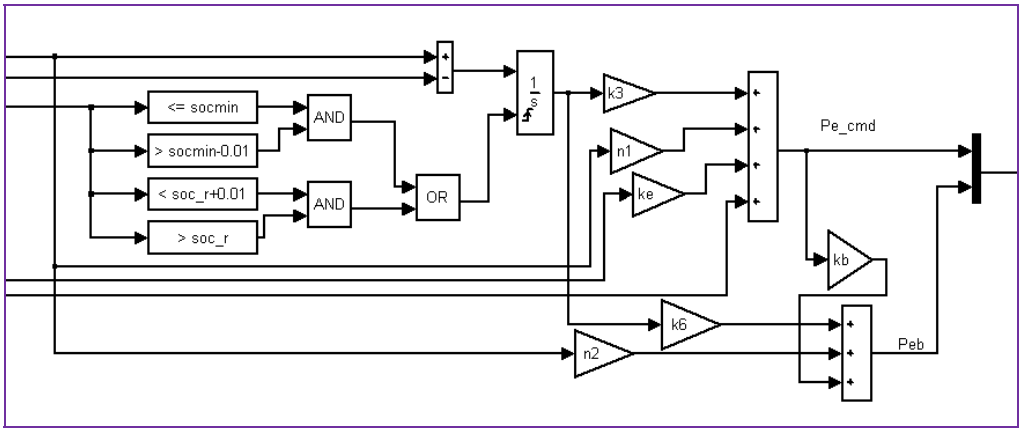


Compared to the centralized control case, the only difference is the supervisory controller for CS mode. It is distributed into two parts, the VSC and the BSC.

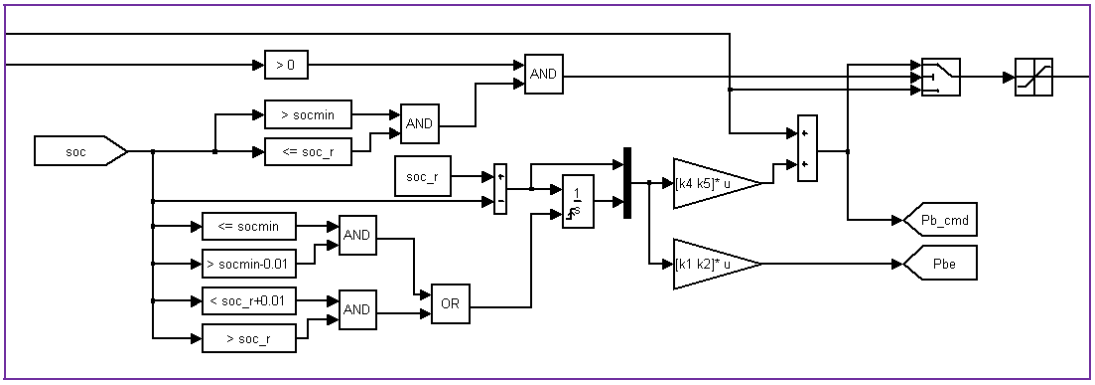
 # 11



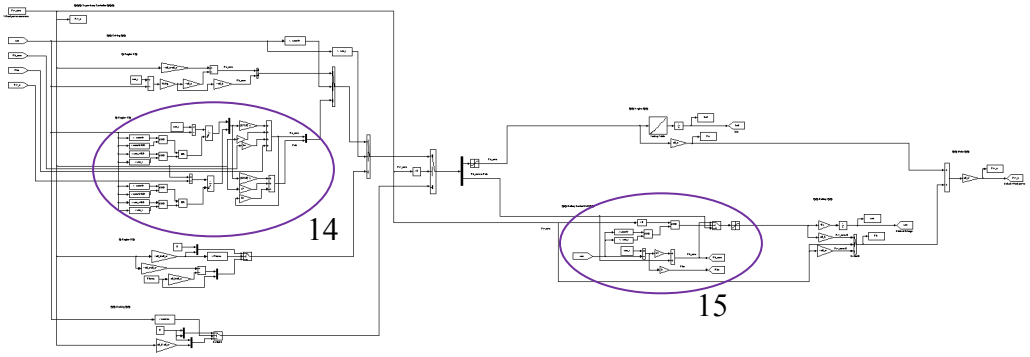
 # 12 CS mode controller in the VSC in Case 1



 # 13 CS mode controller in the BSC in Case 1

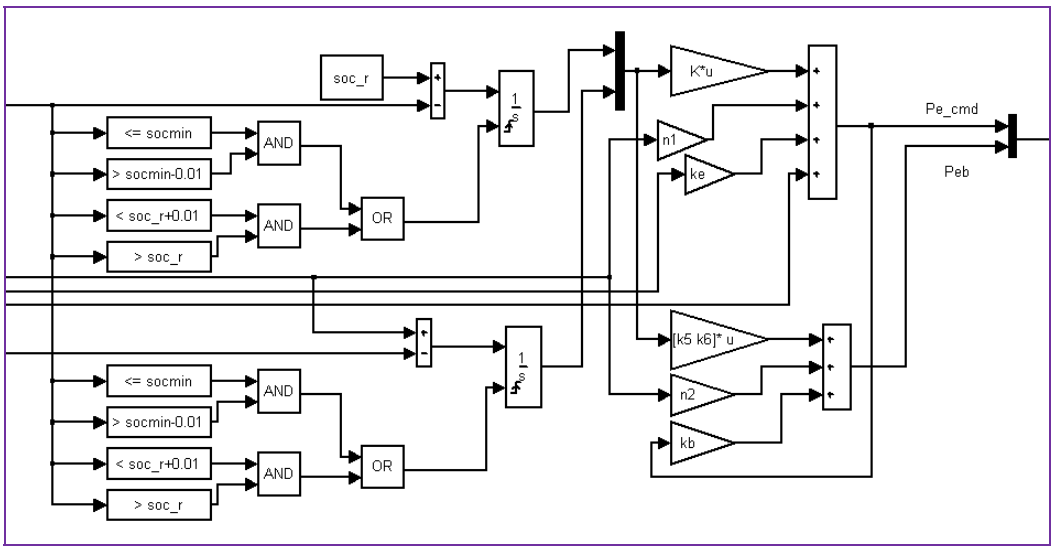


c. Vehicle model for the distributed control Case 2

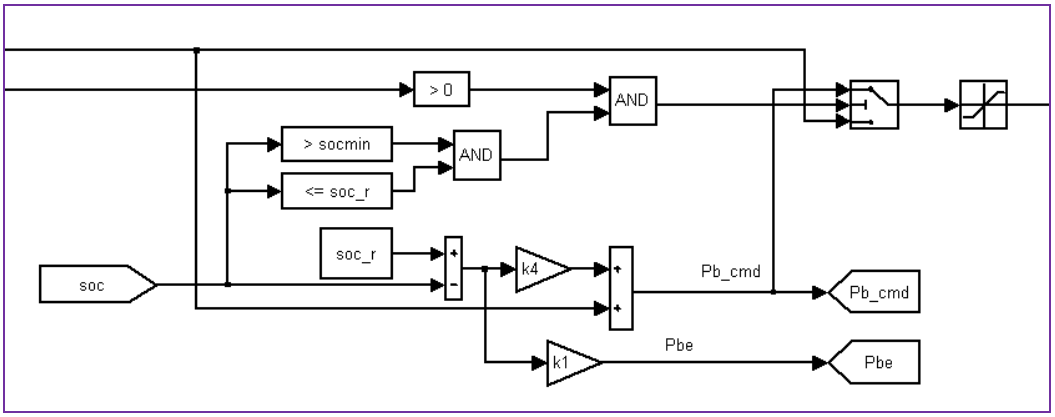


Compared to the distributed control case 1, the only difference is the CS mode controller in the VSC and the BSC.

 # 14 CS mode controller in the VSC in Case 2



 # 15 CS mode controller in the BSC in Case 2



Appendix C: Ideas on Distributed Controller Design Using LMI and BMI for CSM

a. Linear matrix inequalities

A linear matrix inequality (LMI) has the form [67]:

$$F(x) \triangleq F_0 + \sum_{i=1}^{m} x_i F_i > 0 \tag{C.1}$$

where $x \in R^m$ is the variable and the symmetric matrices $F_i = F_i^T \in R^{n \times n}$, $i = 0, \ldots, m$, are given. The inequality symbol in (C.1) means that $F(x)$ is positive-definite, i.e., $u^T F(x) u > 0$ for all nonzero $u \in R^n$.

b. Bilinear matrix inequalities

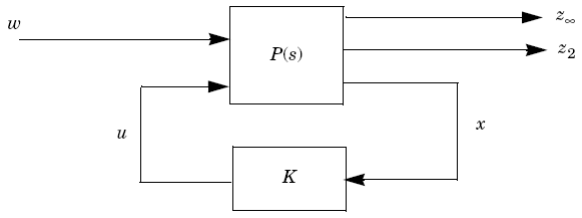
A bilinear matrix inequality (BMI) is of the form [67]:

$$F(x, y) \triangleq F_0 + \sum_{i=1}^{m} x_i F_i + \sum_{j=1}^{n} y_j G_j + \sum_{i=1}^{m} \sum_{j=1}^{n} x_i y_j H_{ij} > 0 \tag{C.2}$$

where G_j and H_{ij} are symmetric matrices of the same dimension as F_i, and $y \in R^n$

c. Controller design using LMI formulation

For instance, for state feedback control for a single LTI model, we assume full measurement of its state vector x. The control structure is as follows,



Consider a regulation problem with disturbance d, and let e denote the regulation error. Setting $\omega = d,\ z_\infty = e,\ z_2 = \begin{bmatrix} x \\ u \end{bmatrix}$

Given a state-space realization of the plant $P(s)$,

$$\dot{x} = Ax + B_1\omega + B_2u$$
$$z_\infty = C_1x + D_{11}\omega + D_{12}u$$
$$z_2 = C_2x + D_{22}u$$

The closed loop system is given in state-space form by

$$\dot{x} = (A + B_2K)x + B_1\omega$$
$$z_\infty = (C_1 + D_{12}K)x + D_{11}\omega$$
$$z_2 = (C_2 + D_{22}K)x$$

The H_2, H_∞ performance and pole placement can be formulated as LMIs [68]. For instance, for H_2 performance, the H_2 norm of the closed-loop transfer function from ω to z_2 does not exceed v if and only if there exist two symmetric matrices X_2 and Q such that,

$$\begin{bmatrix} (A + B_2K)X_2 + X_2(A + B_2K)^T & B_1 \\ B_1^T & -I \end{bmatrix} < 0$$

$$\begin{bmatrix} Q & (C_2 + D_{22}K)X_2 \\ X_2(C_2 + D_{22}K)^T & X_2 \end{bmatrix} > 0 \qquad \text{(C.3)}$$

$$Trace(Q) < v^2$$

With the change of variable Y: = KX_2, the above inequalities lead to LMI. Then the controller gains in K can be calculated from $Y = KX_2$ [68].

d. Distributed controller structure for LMI formulation

If the structure and the parameter dependency of the distributed controllers are assumed *a priori*, the overall distributed controller (equivalent centralized controller) as calculated by equation (3.13) results in a fixed structure controller. The design of a fixed structure controller reduces, under appropriate assumptions, to a bi-linear matrix inequality (BMI) problem as given in Appendix C. Unfortunately, BMI solvers are not

capable of efficiently handling large number of design variables. Hence a nonlinear optimization formulation is proposed in this dissertation.

For the distributed control system in Figure 3.2, assume that the distributed controllers in state-space have the following form,

$$\begin{aligned} \dot{x}_{b1} &= A_{b1}x_{b1} + B_{b1}y_{ac} \\ u_{ca1} &= C_{b1}x_{b1} \end{aligned} \tag{C.4}$$

$$\begin{aligned} \dot{x}_{b2} &= A_{b2}x_{b2} + B_{b2}e \\ u_{ca2} &= C_{b2}x_{b2} \end{aligned} \tag{C.5}$$

$$\begin{aligned} \dot{x}_{a1} &= A_{a1}x_{a1} + B_{a1}u_{ca} \\ y_{ac} &= C_{a1}x_{a1} \end{aligned} \tag{C.6}$$

$$\begin{aligned} \dot{x}_{a2} &= A_{a2}x_{a2} + B_{a2}u_{ca} \\ u &= C_{a2}x_{a2} \end{aligned} \tag{C.7}$$

Equations (C.4) to (C.7) represent the model for C_{BC11}, C_{BC12}, C_{A11} and C_{A21}, respectively.

Combine the sub-controllers, we get the overall distributed controller as,

$$\begin{bmatrix} \dot{x}_{b1} \\ \dot{x}_{b2} \\ \dot{x}_{a1} \\ \dot{x}_{a2} \end{bmatrix} = \begin{bmatrix} A_{b1} & 0 & B_{b1}C_{a1} & 0 \\ 0 & A_{b2} & 0 & 0 \\ B_{a1}C_{b1} & B_{a1}C_{b2} & A_{a1} & 0 \\ B_{a2}C_{b1} & B_{a2}C_{b2} & 0 & A_{a2} \end{bmatrix} \begin{bmatrix} x_{b1} \\ x_{b2} \\ x_{a1} \\ x_{a2} \end{bmatrix} + \begin{bmatrix} 0 \\ B_{b2} \\ 0 \\ 0 \end{bmatrix} e$$

$$u = \begin{bmatrix} 0 & 0 & 0 & C_{a2} \end{bmatrix} \begin{bmatrix} x_{b1} \\ x_{b2} \\ x_{a1} \\ x_{a2} \end{bmatrix}$$

Assume the controlled plant model as,

$$\dot{x}_p = A_p x + B_p u + E_p d$$
$$y = C_p x_p$$

The closed loop system with the states

$$x = \begin{bmatrix} x_{b1} & x_{b2} & x_{a1} & x_{a2} & x_p \end{bmatrix}^T$$

in state-space form is:

$$\dot{x} = A_{cl} x + B_{cl} d$$
$$z_2 = C_{cl2} x + D_{cl2} u = C_{cl} x$$

where:

$$A_{cl} = \begin{bmatrix} A_{b1} & 0 & B_{b1}C_{a1} & 0 & 0 \\ 0 & A_{b2} & 0 & 0 & -B_{b2}C_p \\ B_{a1}C_{b1} & B_{a1}C_{b2} & A_{a1} & 0 & 0 \\ B_{a2}C_{b1} & B_{a2}C_{b2} & 0 & A_{a2} & 0 \\ 0 & 0 & 0 & B_p C_{a2} & A_p \end{bmatrix}, \; B_{cl} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ E \end{bmatrix}$$

$$C_{cl} = C_{cl2} + D_{cl2} \begin{bmatrix} 0 & 0 & 0 & C_{a2} & 0 \end{bmatrix}$$

If all the sub-controllers are assumed to be in the controllable canonical form, the A_{cl} and C_{cl} will be linear with respect to the unknown controller gains. For example, for a fourth order C_{BC11}, we can assume

$$A_{b1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_{b14} & -a_{b13} & -a_{b12} & -a_{b11} \end{bmatrix}, \; B_{b1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \; C_{b1} = \begin{bmatrix} c_{b14} & c_{b13} & c_{b12} & c_{b11} \end{bmatrix}.$$

The unknown controller gains are a_{b14}, a_{b13}, a_{b12}, a_{b11}, c_{b14}, c_{b13}, c_{b12} and c_{b11}.

For the H_2 norm performance, the H_2 norm of the closed-loop transfer function from ω to z_2 does not exceed v if and only if there exist two symmetric matrices X_2 and Q such that,

$$\begin{bmatrix} A_{cl}X_2 + X_2A_{cl}^T & B_{cl} \\ B_{cl}^T & -I \end{bmatrix} < 0$$

$$\begin{bmatrix} Q & C_{cl}X_2 \\ X_2C_{cl}^T & X_2 \end{bmatrix} > 0 \qquad \text{(C.8)}$$

$$Trace(Q) < v^2$$

Compared to the inequalities in (C.3), if we use the change of variables ($Y_1 := A_{cl}X_2$ and $Y_2 := C_{cl}X_2$) to make the above inequalities into LMI, the major computation hurdle is to solve for the unknown controller gains and the entries of X_2 and Q from $Y_1 = A_{cl}X_2$ and $Y_2 = C_{cl}X_2$.

The inequalities in (C.8) are actually BMIs with the variables as the controller gains in A_{cl} and C_{cl}, and the entries of X_2 and Q. Solution algorithms for BMIs can be applied. The global algorithms, which are applicable to modest size problems with a few variables, include Branch and Bound algorithm [69], lagrangian dual global optimization algorithm [70], generalized benders decomposition [71]. The local algorithms, which are computationally fast, but depend on initial condition and may not converge to the global optima, include coordinate descent method [72], rank-minimization method [73], XY-centering algorithm [74], and path-following method [75].

Meanwhile, in order to achieve CSM for the actuator, the actuator controller (C_{BC11} and C_{BC12}), can change with the actuator module when the actuator changes, but the base controller (C_{A11} and C_{A21}) remains the same. Thus, the controller gains of the base controller and the actuator controller have different design freedom. This needs to be considered in the solution algorithm as well. Two approaches, an all in one optimization and a bi-level optimization, have been introduced in Chapter II, Section 2.5, to handle the different design freedom of the controller gains.

BIBLIOGRAPHY

[1] C. Y. Baldwin and K. B. Clark, "Managing in an age of modularity," *Harvard Business Review,* vol. 75, pp. 84-84, 1997.

[2] K. Ulrich and K. Tung, "Fundamentals of Product Modularity," in *Issues in Design/Manufacture Integration, ASME*, New York, 1991, pp. 73-79.

[3] J. K. Gershenson, G. J. Prasad, and Y. Zhang, "Product modularity: definitions and benefits," *Journal of Engineering Design,* vol. 14, pp. 295-313, 2003.

[4] J. K. Gershenson, G. J. Prasad, and S. Allamneni, "Modular Product Design: A Life-cycle View," *Journal of Integrated Design and Process Science,* vol. 3, pp. 3-26, 1999.

[5] H. Chun-Che and A. Kusiak, "Modularity in design of products and systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 28, pp. 66-77, 1998.

[6] S. Gupta and G. E. Okudan, "Modular design: A review of research and industrial applications," *Proceedings of IIE Annual Conference and Expo 2007 - Industrial Engineering's Critical Role in a Flat World* pp. 1563-1568, 2007.

[7] Y. Zhang, J. K. Gershenson, and G. J. Prasad, "Product modularity: measures and design methods," *Journal of Engineering Design,* vol. 15, pp. 33-51, 2004.

[8] A. G. Michelsen and J. Stoustrup, "High level model predictive control for plug-and-play process control with stability guaranty," *Proceedings of the 49th IEEE Conference on Decision and Control,* 2010.

[9] J. Bendtsen, K. Trangbaek, and J. Stoustrup, "Hierarchical Model Predictive Control for Resource Distribution," *49th IEEE Conference on Decision and Control,* pp. 2468-2473, 2010.

[10] J. M. Giron-Sierra, C. Insaurralde, M. Seminario, J. F. Jimenez, and P. Klose, "CANbus-based distributed fuel system with smart components," *Aerospace and Electronic Systems, IEEE Transactions on,* vol. 44, pp. 897-912, 2008.

[11] V. X. W. Yong Zhang ; Yikang Gu ; Vlatkovic, " Progress of smart sensor and smart sensor networks " in *Intelligent Control and Automation, Fifth World Congress on*. vol. 4, 2004, pp. 3600 - 3606

[12] W. Wang and O. A. Jianu, "A Smart Sensing Unit for Vibration Measurement and Monitoring," *IEEE/ASME Transactions on Mechatronics, ,* vol. 15, pp. 70-78, Feb. 2010.

[13] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson, and K. V. Prasad, "Control, computing and communications: technologies for the twenty-first century model T," *Proceedings of the IEEE,* vol. 95, pp. 334-55, 2007.

[14] B. A. Warneke and K. S. J. Pister, "Exploring the Limits of System Integration with Smart Dust," in *ASME International Mechanical Engineering Congress and Exposition*, New Orleans, 2002, pp. 621-625.

[15] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE,* vol. 95, pp. 138-172, 2007.

[16] S. Li, M. Cakmakci, I. V. Kolmanovsky, and A. G. Ulsoy, "Throttle actuator swapping modularity design for idle speed control " *Proceedings of American Control Conference,* pp. 2702-2707, 2009.

[17] L. Feng-Li, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control Systems Technology,* vol. 10, pp. 297-307, 2002.

[18] R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE control systems magazine,* vol. 23, pp. 20-33, 2003.

[19] W. Zhang, "Stability of networked control systems," *IEEE control systems magazine,* vol. 21, p. 84, 2001.

[20] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Transactions on Control Systems Technology,* vol. 10, pp. 438-446, 2002.

[21] J. K. Yook, D. M. Tilbury, and N. R. Soparkar, "Trading computation for bandwidth: reducing communication in distributed control systems using state estimators," *IEEE Transactions on Control Systems Technology,* vol. 10, pp. 503-18, 2002.

[22] G. C. Walsh and Y. Hong, "Scheduling of networked control systems," *IEEE control systems magazine,* vol. 21, pp. 57-65, 2001.

[23] D. Barcelli, D. Bernardini, and A. Bemporad, "Synthesis of Networked Switching Linear Decentralized Controllers," *49th IEEE Conference on Decision and Control,* pp. 2480-2485, 2010.

[24] Melih Cakmakci and A. Galip Ulsoy, "Improving Component-Swapping Modularity Using Bidirectional Communication in Networked Control System," *IEEE/ASME Transactions on Mechatronics,* vol. 14, pp. 307 - 316 2009.

[25] M. Cakmakci and A. G. Ulsoy, "Swappable distrituted MIMO controller for a VCT engine," *IEEE Transactions on Control System Technology (in press)*.

[26] N. M. Alexandrov and R. M. Lewis, "Analytical and computational aspects of collaborative optimization for multidisciplinary design," *AIAA Journal,* vol. 40, pp. 301-309, 2002.

[27] P. Zadeh, V. Toropov, and A. Wood, "Metamodel-based collaborative optimization framework," *Structural and Multidisciplinary Optimization,* vol. 38, pp. 103-15, 2009.

[28] S. Tosserams, L. F. P. Etman, and J. E. Rooda, "An augmented Lagrangian decomposition method for quasi-separable problems in MDO," *Structural and Multidisciplinary Optimization,* vol. 34, pp. 211-227, 2007.

[29] I. P. Sobieski and I. M. Kroo, "Collaborative optimization using response surface estimation," *AIAA Journal,* vol. 38, pp. 1931-1938, 2000.

[30] J. G. Lin, "Analysis and enhancement of collaborative optimization for multidisciplinary design," *AIAA Journal,* vol. 42, pp. 348-360, 2004.

[31] B. D. Roth and I. M. Kroo, "Enhanced collaborative optimization: A decomposition-based method for multidisciplinary design " in *Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2008, pp. 927-936.

[32] V. DeMiguel and W. Murray, "A local convergence analysis of bilevel decomposition algorithms," *Optimization and Engineering,* vol. 7, pp. 99-133, 2006.

[33] A.A. Pesaran, T. Markel, H.S. Tataria, and D. Howell, "Battery Requirements for Plug-In Hybrid Electric Vehicles – Analysis and Rationale," NREL/CP-540-42240, 2009.

[34] T. Markel and A. Simpson, "Plug-In Hybrid Electric Vehicle Energy Storage System Design," NREL/CP-540-39614, 2006.

[35] S. G. Wirasingha and A. Emadi, "Classification and review of control strategies for plug-In hybrid electric vehicles," *Vehicular Technology, IEEE Transactions on,* vol. 60, pp. 111-122, 2011.

[36] N. Jalil, Kheir, N.A., Salman, M, "A rule-based energy management strategy for a series hybrid vehicle," *American Control Conference, 1997. Proceedings of the 1997,* vol. Volume 1, 4-6 June 1997 Page(s):689 - 693 vol.1 1997.

[37] S. Barsali, C. Miulli, and A. Possenti, "A control strategy to minimize fuel consumption of series hybrid electric vehicles," *Energy Conversion, IEEE Transaction on,* vol. 19, pp. 187-195, 2004.

[38] W. Jong-Seob, R. Langari, and M. Ehsani, "An energy management and charge sustaining strategy for a parallel hybrid vehicle with CVT," *Control Systems Technology, IEEE Transactions on,* vol. 13, pp. 313-320, 2005.

[39] H. Banvait, S. Anwar, and Y. Chen, "A rule-based energy management strategy for plug-in hybrid electric vehicle (PHEV)," *Proceedings of American Control Conference,* pp. 3938-3943, 2009.

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

[41] A. Sciarretta, M. Back, and L. Guzzella, "Optimal control of parallel hybrid electric vehicles," *Control Systems Technology, IEEE Transactions on,* vol. 12, pp. 352-363, 2004.

[42] C. Musardo, G. Rizzoni, and B. Staccia, "A-ECMS: An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 1816-1823.

[43] L. Jinming and P. Huei, "Modeling and Control of a Power-Split Hybrid Vehicle," *Control Systems Technology, IEEE Transactions on,* vol. 16, pp. 1242-1251, 2008.

[44] L. Chan-Chiao, P. Huei, J. W. Grizzle, and K. Jun-Mo, "Power management strategy for a parallel hybrid electric truck," *Control Systems Technology, IEEE Transactions on,* vol. 11, pp. 839-849, 2003.

[45] Edward Dean Tate Jr, J. W. Grizzle, and H. Peng, "Shortest path stochastic control for hybrid electric vehicles," *Int. J. Robust Nonlinear Control,* vol. 18, pp. 1409–1429, 2008.

[46] P. Jungme, C. Zhihang, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey, "Intelligent Vehicle Power Control Based on Machine Learning of Optimal Control Parameters and Prediction of Road Type and Traffic Congestion," *Vehicular Technology, IEEE Transactions on,* vol. 58, pp. 4741-4756, 2009.

[47] A. Konev, L. Lezhnev, and I. Kolmanovsky, "Control strategy optimization for a series hybrid vehicle " *SAE paper 2006-01-0663,* 2006.

[48] S. Di Cairano, W. Liang, I. Kolmanovsky, M. L. Kuang, and A. M. Phillips, "Engine power smoothing energy management strategy for a series hybrid electric vehicle," *Proceedings of the American Control Conference,* 2011.

[49] S. Li, I. V. Kolmanovsky, and A. G. Ulsoy, "Direct Optimal Design for Component Swapping Modularity in Control Systems," *IEEE/ASME Transactions on Mechatronics (submitted)*.

[50] S. Li, I. V. Kolmanovsky, and A. G. Ulsoy, "Distributed Supervisory Controller Design for Battery Swapping Modularity in Plug-in Hybrid Electric Vehicles," *Journal of Dynamic Systems, Measurement and Control (submitted)*.

[51] S. Li, I. V. Kolmanovsky, and A. G. Ulsoy, "Battery Swapping Modularity Design for Plug-in HEVs Using the Augmented Lagrangian Decomposition Method," *Proceedings of American Control Conference,* 2011.

[52] S. Li, I. V. Kolmanovsky, and A. G. Ulsoy, "Direct optimal distributed controller design for component swapping modularity with application to ISC," *Proceedings of American Control Conference,* pp. 5368-5373, 2010.

[53] H. M. Min, N. F. Michelena, P. Y. Papalambros, and T. Jiang, "Target cascading in optimal system design," *Transactions of the ASME, Journal of Mechanical Design,* vol. 125, pp. 474-480, 2003.

[54] K. Tammer, "The application of parametric optimization and imbedding for the foundation and realization of a generalized primal decomposition approach.," *Parametric optimization and related topics, Mathematical Reseach,* vol. 35, 1987.

[55] R. D. Braun and I. M. Kroo, "Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment," *Technical Report, NASA Langley Technical Report Server,* 1995.

[56] D. P. Bertsekas, *Nonlinear programming*: Athena Scientific, 2003.

[57] D. Hrovat and J. Sun, "Models and control methodologies for IC engine idle speed control design," *Control Engineering Practice,* vol. 5, pp. 1093-1100, 1997.

[58] M. Abate and V. DiNunzio, "Idle speed control using optimal regulation," in *SAE paper*, 1990.

[59] C. Carnevale and A. Moschetti, "Idle speed contol with H-infinity technique," *SAE paper,* 1993.

[60] K. Butts, N. Sivashankar, and J. Sun, "Feedforward and feedback design for engine idle speed control using l_1 optimization," in *American Control Conference, 1995. Proceedings of the*, 1995, pp. 2587-2590 vol.4.

[61] A. Gibson, I. Kolmanovsky, and D. Hrovat, "Application of disturbance observers to automotive engine idle speed control for fuel economy improvement," in *American Control Conference, 2006*, 2006, p. 6 pp.

[62] S.F. Alyaqout, P. Y. Papalambros, and A. G. Ulsoy, "Combined Robust Design and Robust Control of an Electric DC Motor," *IEEE/ASME Trans. Mechatronics, DOI: 10.1109/TMECH.2010.2047652 (in press)*.

[63] T. Markel, A. Brooker, T. Hendricks, V. Johnson, K. Kelly, B. Kramer, M. O'Keefe, S. Sprik, and K. Wipke, "ADVISOR: a systems analysis tool for advanced vehicle modeling," *Journal of Power Sources,* vol. 110, pp. 255-266, 2002.

[64] J.-F. Magni, S. Bennani, J. Terlouw, L. Faleiro, J. de la Cruz, and S. Scala, "Eigenstructure assignment," *Robust Flight Control, Springer Berlin / Heidelberg,* pp. 22-32, 1997.

[65] "Code of Federal Regulations 474.3, "http://cfr.vlex.com/source/code-federal-regulations-energy-1059, 2005.

[66] E. D. Arnheiter and H. Harren, "A typology to unleash the potential of modularity," *Journal of Manufacturing Technology Management,* vol. 16, pp. 699-711, 2005.

[67] J. G. VanAntwerp and R. D. Braatz, "A tutorial on linear and bilinear matrix inequalities," *Journal of Process Control,* vol. 10, pp. 363-385, 2000.

[68] P. Gahinet, A. Nemirovski, A. J. Laub, and M. Chilali, *LMI Control Toolbox User's Guide*, 1995.

[69] M. Kawanishi, T. Sugie, and H. Kanki, "BMI global optimization based on branch and bound method taking account ofthe property of local minima," *Proceedings of the 36th Conference on Decision and Control,* pp. 781-786, 1997.

[70] D. H. Tuan, P. Apkarian, and Y. Nakashima, "A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities," *Proceedings of American Control Conference,* pp. 1851-1855 1999.

[71] E. Beran, L. Vandenberghe, and S. Boyd, "A global BMI algorithm based on the generalized benders decomposition," *Proceedings of the European Control Conference, paper no.934,* 1997.

[72] D. G. Luenberger, *Linear and Non Linear Programming*: Addison Wesley, 1994.

[73] S. Ibaraki and M. Tomizuka, "Rank minimization approach for solving BMI problems with random search," *Proceedings of the American Control Conference,* pp. 1870-1875 2001.

[74] T. Iwasaki and R. E. Skelton, "The XY-centering algorithm for the dual LMI problem: A new approach to fixed order control design," *International Journal of Control,* vol. 62(6), pp. 1257–1272, 1995.

[75] A. Hassibi, J. How, and S. Boyd, "A path-following method for solving BMI problems in control," *Proceedings of the American Control Conference,* pp. 1385-1389, 1999.