Novel Optimal Control Algorithms with Application to the Parallel Hydraulic Hybrid Vehicle Power Train

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Robert Gregory Ertel

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science

October, 2010

Acknowledgements

Many people have earned my gratitude for their contribution to my time in graduate school. I would like to thank my adviser Professor Kim Stelson and Mike Gust for their continued support of my studies at the University of Minnesota. I would also like to thank my undergraduate adviser Professor Michael Hennessey for sparking my interest in optimal control. This research was done in association with the Center for Compact and Efficient Fluid Power (CCEFP), a National Science Foundation Engineering Research Center.

Dedication

To my family, and Al.

Abstract

The parallel Hydraulic Hybrid Vehicle (HHV) power train is quickly becoming a viable option among large (class 7-10) vehicles. This is due to its potentially vast improvements in fuel economy over non-hybrid power trains. Optimal control of the parallel HHV power train is critical to overall vehicle performance and is largely responsible for gains in efficiency. The research presented in this thesis aims to answer the question of how to best operate the power train to achieve maximum efficiency during driving intervals when vehicle speed is unspecified, except at boundary points. A state-space model of the parallel HHV power train is derived in the energy domain using a classical Lagrangian approach. Two optimal control algorithms are developed and applied to the vehicle. The first algorithm is gradient descent based and is derived using the calculus of variations. The second algorithm discretizes the optimal control problem in time and converts it to a non-linear program. Several optimal control problems are solved and the results offer valuable insight into efficient operation of the parallel HHV power train.

Contents

Acknowledgements **i**

Dedication **i**

Abstract **ii**

List of Tables **v**

List of Figures **vi**

1 Background and Overview **1**

1.1 A Brief History of Optimal Control 1

1.2 Optimal Control of the Hybrid Vehicle 2

1.3 Original Contribution of This Research 3

1.4 Introduction and Outline 4

2 Vehicle Modeling **5**

3 A Gradient Descent Based Approach to Optimal Vehicle Control **10**

3.1 Problem Statement 10

3.2 Derivation of Optimality Conditions 12

3.3 Optimality Conditions 16

3.4 Gradient Descent Optimal Control Algorithm 17

3.5 Results and Discussion 18

4 A Non-Linear Programming Approach to Optimal Vehicle Control 22
4.1 Problem Statement Reformulation 22
4.2 Optimal Control Reformulation 23
4.2.1 General Form of the Optimal Control Problem 24
4.2.2 Newton's Method for Solution to Optimality Conditions 28
4.3 Application to Vehicle Model 28
4.4 Results and Discussion 31

5 Conclusions and Future Work 35
5.1 Conclusions 35
5.2 Future Work 36

References 37

Appendix A. Matlab™ Code Listing for Gradient Descent Algorithm 39
A.1 OCAlgorithm1.m 39
A.2 euler_system.m 40
A.3 euler_adjoint.m 40
A.4 adjoint.m 41
A.5 system_test2.m 41

Appendix B. Matlab™ Code Listing for NLP Algorithm 42
B.1 OCAlgorithm2.m 42
B.2 vehicle.m 44
B.3 indices.m 51
B.4 ff1.m 52
B.5 dF_calc1.m 53

Appendix C. List of Acronyms and Notations 55
C.1 Acronyms 55
C.2 Notations 57

List of Tables

C.1 Acronyms 55
C.2 Notations 57

List of Figures

2.1 Pre-transmission parallel HHV power train architecture. 5
3.1 Optimal state trajectories. 19
3.2 Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$. 19
4.1 Optimal state trajectories. 32
4.2 Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$. 32
4.3 Optimal state trajectories. 33
4.4 Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$. 33

Chapter 1

Background and Overview

This chapter introduces optimal control and its role in the Hydraulic Hybrid Vehicle (HHV). It is broken into three sections. Section 1.1 presents a brief history of relevant aspects of optimal control theory. It covers works ranging from earlier analytical developments, to more recent numerical approaches, each having important implications in this research. Section 1.2 discusses optimal control research in the context of the HHV. Specifically, it addresses typical problems for such systems and different approaches to their solution. Section 1.3 is an overview of the research goals for this thesis. It explains the differences between existing research that which is presented here. Section 1.4 is an outline of thesis content.

1.1 A Brief History of Optimal Control

The purpose of optimal control is to find state and input trajectories of a dynamic system that minimize a specified cost functional. The so-called cost functional is a function of functions in that its input is a function and its output is a scalar (cost). Some examples of optimal control problems are minimum-time problems, minimum-input problems, shortest path etc.

Much of optimal control theory has foundations rooted in a well-developed branch of mathematics called the Calculus of Variations (COV). COV is concerned with finding extremal trajectories that represent optimal curves, shapes, parameters etc. Optimal Control is just one of its many applications. The work of Bryson [1] presents a COV

based approach to the solution of optimal control problems. Specifically, COV is used to derive certain optimality conditions that, when solved, guarantee the associated cost functional has been minimized.

Hasdorff [2] takes a similar approach to Bryson and proposes a simple numerical technique for solving optimal control problems. The numerical technique treats optimality conditions as gradient information of a given problem's cost functional. This is used in a gradient descent algorithm that effectively hones in on an optimal solution with each iteration. The way system constraints are enforced between Bryson and Hasdorff are fundamentally different, however. Bryson uses a Lagrange multiplier formulation, whereas Hasdorff takes a penalty-based approach to enforce simple equality constraints on the system. Equality constraints typically take the form of end conditions on state variables. The penalty on a given constraint is a weighting factor that, if large enough, should guide the system reasonably close to the desired final state. In contrast, the Lagrange multiplier formulation ensures constraints are enforced exactly.

Elements from these formulations are utilized in both optimal control algorithms presented here. Specifically, the Lagrange multiplier formulation from Bryson is used in combination with the gradient descent algorithm from Hasdorff. This enures exact constraint enforcement and gives a technique for solving complex optimal control problems numerically.

1.2 Optimal Control of the Hybrid Vehicle

This section is a brief overview of optimal control research as applied to the HHV.

The HHV has the potential for highly economical operation. It has been shown that the HHV is most promising for larger (class 7-10) vehicles subject to frequent start-stop duty cycles. This is due to the high power density of the hydraulic portion of the drive line, which is capable of efficiently recovering and restoring kinetic energy during large-mass vehicle operation [3]. The question of how to best operate the HHV power train so as to achieve maximum efficiency is one of great importance and falls within the realm of optimal control.

Hybrid vehicles, in general, have garnered much research attention in the area of optimal control. Typically, optimal control trajectories are determined for the hybrid

vehicle over representative driving schedules (drive cycles). Control laws are then extracted from the results and implemented in real time for efficient vehicle operation. In [4], work has been done on fuel consumption minimization for the hybrid electric vehicle (HEV) over multiple randomly generated drive cycles via stochastic dynamic programming. The result is a set of full state feedback control laws for real time efficient vehicle operation. A different approach to optimization over a drive cycle is by means of a robust approach (H-infinity control) and can be found in [5]. Similarly, the result is a state feedback controller for use in real time. Efficient real time control can be achieved through sub-optimal means as well, such as the Equivalent Consumption Minimization Strategy (ECMS). The core of ECMS is what is called the equivalence factor, which is essentially a tuneable parameter that dictates power flow [6], [7] and can achieve up to 50% of potential fuel savings over a drive cycle.

1.3 Original Contribution of This Research

The methods discussed in section 1.2 for improving vehicle efficiency work well when *a priori* knowledge of the vehicle's driving schedule is available. The goal of this research, however, is to maximize vehicle efficiency over driving events where vehicle trajectories (state and input) are not specified *a priori*. The resulting optimal vehicle trajectories offer valuable practical insight for efficient operation of the HHV power train.

There are inherent challenges with optimal control of complex systems such as the HHV. Inequality constraints are acting on both state and input variables which makes the problem significantly more difficult to solve. Under most conditions, this solution is discontinuous in nature causing the resulting optimal trajectories to be piece-wise continuous.

The research presented here offers two algorithms capable of quickly and precisely solving inequality constrained optimal control problems numerically. The first algorithm is based on a rigorous derivation of optimality conditions using COV. The second algorithm does away with functional analysis completely by converting the optimal control problem in to a Non Linear Program (NLP). The reason for the development of a second optimal control algorithm is that in some instances the first algorithm converges slowly, or not at all. Also, if any changes are made to the state-space model of the vehicle, a

complete re-derivation of optimality conditions needs to be made, which is unfavorable.

1.4 Introduction and Outline

This section is an outline of how the thesis is arranged. In chapter 2 a state-space model for the parallel HHV power train is derived. It will be used throughout for the purposes of optimal control. Chapter 3 is a development of a gradient descent based algorithm for optimal control of the parallel HHV power train. A problem statement is made, optimality conditions are derived, and the algorithm is enumerated. Several optimal control problems are solved. The results lead to rules for efficient practical operation of the vehicle as well as issues with both the problem statement and the algorithm. Chapter 4 is a development of a non-linear programming based algorithm for optimal control. A reformulation of the original problem statement is made. The new algorithm is developed and applied to the vehicle. Several optimal control problems are solved and the results are discussed. Chapter 5 presents final conclusions and a discussion of the research in this thesis.

Chapter 2

Vehicle Modeling

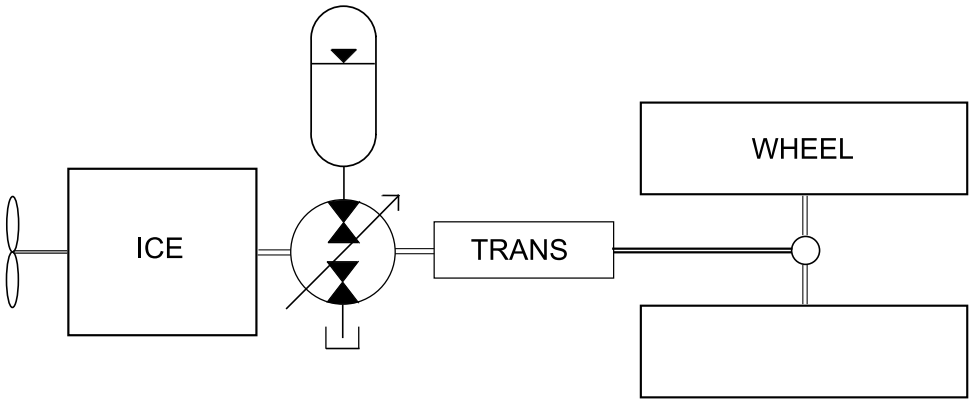

Figure 2.1: Pre-transmission parallel HHV power train architecture.

To begin, a state-space model for the pre-transmission parallel HHV power train is developed. Figure 2.1 is a schematic representation of the system and shows an internal combustion engine (ICE) coupled directly to a variable displacement hydraulic pump-motor unit. The pump-motor unit is in circuit with an accumulator and is located before the transmission, hence the term "pre-transmission." The transmission connects to the wheels via the drive shaft. Not shown in figure 2.1 are system losses which take the form of aerodynamic drag on the vehicle and volumetric loss over the pump-motor unit. The kinetic energy of the system is

$$T = KE_{ICE} + KE_{p/m} + KE_{wheel} + KE_{veh} \tag{2.1}$$

Applying the respective speeds and inertias to 2.1 gives

$$T = \frac{1}{2}J_{ICE}\omega_{in}\left(t\right)^2 + \frac{1}{2}J_{p/m}\omega_{in}\left(t\right)^2 + 4\left(\frac{1}{2}J_{wheel}\omega_{out}\left(t\right)^2\right) + \frac{1}{2}m_{veh}v_{veh}\left(t\right)^2 \quad (2.2)$$

where J_{ICE} is engine inertia $\left(kgm^2\right)$, $J_{p/m}$ is pump-motor inertia $\left(kgm^2\right)$, J_{wheel} is wheel inertia $\left(kgm^2\right)$, and m_{veh} is vehicle mass (kg). System speeds are related as follows

$$v_{veh}\left(t\right) = r_{wheel}\omega_{out}\left(t\right) \quad (2.3)$$

$$\omega_{in}\left(t\right) = N\omega_{out}\left(t\right) \quad (2.4)$$

where $v_{veh}\left(t\right)$ is vehicle speed (m/s), r_{wheel} is wheel radius (m), $\omega_{out}\left(t\right)$ is shaft speed after the transmission $(1/s)$, $\omega_{in}\left(t\right)$ is the shaft speed before the transmission $(1/s)$, and $N = Transmission\,Ratio$. Rewriting the kinetic energy in terms of $\omega_{in}\left(t\right)$ gives

$$T = \frac{1}{2}J_{ICE}\omega_{in}\left(t\right)^2 + \frac{1}{2}J_{p/m}\omega_{in}\left(t\right)^2 + 2J_{wheel}\frac{1}{N^2}\omega_{in}\left(t\right)^2 + \frac{1}{2}m_{veh}\frac{r_{wheel}^2}{N^2}\omega_{in}\left(t\right)^2 \quad (2.5)$$

The accumulator is now considered. Assuming an isothermal process, the ideal gas law gives

$$P_{gas}V_{gas} = mRT = k_{gas} \quad (2.6)$$

$$V_{gas} = V_a - V_{oil} \quad (2.7)$$

$$P_{gas} = \frac{k_{gas}}{V_a - V_{oil}} \quad (2.8)$$

where P_{gas} is accumulator pressure (Pa), V_{gas} is accumulator gas volume $\left(m^3\right)$, k_{gas} is the isothermal accumulator constant (kgm), V_a is the total accumulator volume $\left(m^3\right)$, and V_{oil} is the accumulator oil volume $\left(m^3\right)$. Ideal accumulator oil flow is a function of pump-motor speed and displacement such that

$$\dot{V}_{oil}\left(t\right) = \frac{D}{2\pi}\omega_{in}\left(t\right)u\left(t\right) \quad (2.9)$$

where D is total pump-motor displacement $\left(m^3\right)$, and $u\left(t\right) \in \{-1, 1\}$ is the pump-motor displacement command. It follows that $-1 \leq u\left(t\right) < 0$ is motor mode (accumulator oil volume decreasing), and $0 < u\left(t\right) \leq 1$ is pump mode (accumulator oil volume increasing). From [8] append a volumetric loss term to 2.9 giving

$$\dot{V}_{oil}\left(t\right) = \frac{D}{2\pi}\omega_{in}\left(t\right)u\left(t\right) - \frac{k_{gas}k_{loss}}{V_a - V_{oil}\left(t\right)} \quad (2.10)$$

where k_{loss} is based on pump-motor and working fluid characteristics such as oil viscosity, piston diameter, gap size etc. It is important to note that the volumetric loss term in 2.10 is always acting, even when the displacement command, $u(t)$, is equal to zero. Treating the contributions from the internal combustion engine and the hydraulic pump-motor unit as applied torques on the system gives

$$\tau_{hydraulics}(t) = \frac{D}{2\pi} \triangle P(t)\, u(t) \tag{2.11}$$

where

$$\triangle P(t) = P_{gas}(t) - P_{atm} = \frac{k_{gas}}{V_a - V_{oil}(t)} - P_{atm} \tag{2.12}$$

$$\tau_{ICE}(t) = input \tag{2.13}$$

P_{atm} is atmospheric pressure (Pa), $\tau_{hydraulics}(t)$ is applied hydraulic torque (Nm), and $\tau_{ICE}(t) = input$ is applied engine torque (Nm). Normally engine torque is a function of engine speed, fuel flow, throttle command, etc. For simplicity's sake it will be treated as an input. Aerodynamic loss can also be viewed as an applied torque which takes the form

$$\tau_{aero}(t) = -k_{aero} v_{veh}(t)^2 = -k_{aero} \frac{r_{wheel}^2}{N^2} \omega_{in}(t)^2 \tag{2.14}$$

Using a classical approach, the dynamics come from the generalized Euler-Lagrange differential equation

$$Q = \frac{d}{dt}\left(\frac{\partial T}{\partial \omega_{in}(t)}\right) - \frac{\partial T}{\partial \theta_{in}(t)} \tag{2.15}$$

where Q is the sum of the generalized torques i.e.

$$Q = \sum_{i=1}^{n} \tau_{applied_i}(t) \frac{\partial r_i}{\partial q} \tag{2.16}$$

In equation 2.16, q is a generalized coordinate, and r_i is the coordinate transform for each respective applied torque, $\tau_{applied_i}(t)$. Choose $q = \omega_{in}(t)$ as the generalized coordinate. Since all applied torques, $\tau_{ICE}(t)$, $\tau_{hydraulics}(t)$, and $\tau_{aero}(t)$ are either acting directly on the generalized coordinate, or have been converted to do so, there is only one coordinate transform; $r_1 = \omega_{in}(t)$, giving

$$Q = \left(\tau_{ICE}(t) + \tau_{hydraulics}(t) + \tau_{aero}(t)\right) \frac{\partial r_1}{\partial \omega_{in}(t)} \tag{2.17}$$

where $\frac{\partial r_1}{\partial \omega_{in}(t)} = \frac{\partial}{\partial \omega_{in}(t)}\omega_{in}(t) = 1$. For use in equation 2.15 compute

$$\frac{\partial T}{\partial \omega_{in}(t)} = J_{ICE}\omega_{in}(t) + J_{p/m}\omega_{in}(t) + \frac{4}{N^2}J_{wheel}\omega_{in}(t) + \frac{r_{wheel}^2}{N^2}m_{veh}\omega_{in}(t) \tag{2.18}$$

Putting together equations 2.15, 2.18, and 2.17, as well as 2.10 gives

$$\begin{aligned} J_{eq}\dot{\omega}_{in}(t) &= \tau_{ICE}(t) - \tfrac{D}{2\pi}\left(\tfrac{k_{gas}}{V_a - V_{oil}(t)} - P_{atm}\right)u(t) - k_{aero}\tfrac{r_{wheel}^2}{N^2}\omega_{in}(t)^2 \\ \dot{V}_{oil}(t) &= \tfrac{D}{2\pi}\omega_{in}(t)\,u(t) - \tfrac{k_{gas}k_{loss}}{V_a - V_{oil}(t)} \end{aligned} \tag{2.19}$$

which represent the second order, nonlinear state-space model for a pre-transmission parallel HHV power train. J_{eq} is the equivalent vehicle inertia and follows

$$J_{eq} = J_{ICE} + J_{p/m} + \frac{4}{N^2}J_{wheel} + \frac{r_{wheel}^2}{N^2}m_{veh}$$

Assuming a fixed transmission ratio, N, and the insignificance of P_{atm}, 2.19 can be written in a more compact form:

$$\begin{aligned} \dot{x}_1(t) &= K_e\tau_{ICE}(t) - \tfrac{K_a}{V_a - x_2(t)}u(t) - Bx_1(t)^2 \\ \dot{x}_2(t) &= K_v x_1(t)\,u(t) - \tfrac{K_L}{V_a - x_2(t)} \end{aligned} \tag{2.20}$$

with states

$$\begin{aligned} x_1(t) &= \omega_{in}(t) \\ x_2(t) &= V_{oil}(t) \end{aligned}$$

inputs

$$\begin{aligned} u(t) &= displacement\,command \\ \tau_{ICE}(t) &= engine\,torque \end{aligned}$$

and constants

$$K_e = \frac{1}{J_{ICE} + J_{p/m} + \frac{4}{N^2}J_{wheel} + \frac{r_{wheel}^2}{N^2}m_{veh}}$$

$$K_a = \frac{\frac{Dk_{gas}}{2\pi}}{J_{ICE} + J_{p/m} + \frac{4}{N^2}J_{wheel} + \frac{r_{wheel}^2}{N^2}m_{veh}}$$

$$B = \frac{k_{aero}\frac{r_{wheel}^2}{N^2}}{J_{ICE} + J_{p/m} + \frac{4}{N^2}J_{wheel} + \frac{r_{wheel}^2}{N^2}m_{veh}}$$

$$K_v = \frac{D}{2\pi}$$

$$K_L = k_{gas} k_{loss}$$

Typical constant values for a class 10 truck (4500 kg, N = 10) are $K_e = 5.9e-5 \frac{1}{kgm^2}$, $K_a = 1.7e6 \frac{1}{s^2}$, $B = 1e-4 \frac{1}{kg}$, and $K_L = 0.01Nm$.

Chapter 3

A Gradient Descent Based Approach to Optimal Vehicle Control

The parallel HHV power train represents a unique opportunity to apply well developed optimal control theory to a challenging problem. A problem statement is made first and is then cast in to a cost functional. From there, techniques rooted in the calculus of variations are applied to arrive at a set of optimal control laws. The model is subject to inequality constraints on both state and input trajectories, and terminal constraints on some, but not all states. Dealing with such complications relies on more modern contributions from [1], [9], and [10]. Such a formulation rules out the presence of a closed form analytical solution, but lends itself nicely to numerical solution using techniques developed by [2].

3.1 Problem Statement

The problem statement is as follows: Starting from rest, propel the vehicle to a certain speed at a specified final time using the least amount of effort from the engine, while obeying the system equations and their associated inequality constraints. The control

objective is

$$minimize \int_{t_0}^{t_f} \tau_{ICE}(t)^2 \, dt \tag{3.1}$$

such that

$$x_1(t_f) = k_1 \tag{3.2}$$

$$\begin{aligned} \dot{x}_1(t) &= K_e \tau_{ICE}(t) - \tfrac{K_a}{V_a - x_2(t)} u(t) - B x_1(t)^2 \\ \dot{x}_2(t) &= K_v x_1(t) u(t) - \tfrac{K_L}{V_a - x_2(t)} \end{aligned}$$

$$V_{min} \leq x_2(t) \leq V_{max} \tag{3.3}$$

$$u_{min} \leq u(t) \leq u_{max} \tag{3.4}$$

where V_{min} and V_{max} are the minimum and maximum accumulator oil volumes, and u_{min} and u_{max} are the minimum and maximum displacement commands. The control objective can be cast in to the following cost functional:

$$\begin{aligned} J = \int_{t_0}^{t_f} & \tau_{ICE}(t)^2 + \lambda_1(t) \left(\dot{x}_1(t) - K_e \tau_{ICE}(t) + \tfrac{K_a}{V_a - x_2(t)} u(t) + B x_1(t)^2 \right) \\ & + \lambda_2(t) \left(\dot{x}_2(t) - K_v x_1(t) u(t) + \tfrac{K_L}{V_a - x_2(t)} \right) \\ & + \mu_1(t) (u_{min} - u(t)) + \mu_2(t) (u(t) - u_{max}) \\ & + \mu_3(t) (V_{min} - x_2(t)) + \mu_4(t) (x_2(t) - V_{max}) \, dt \\ & + \nu_1 (x_1(t_f) - k1) \end{aligned} \tag{3.5}$$

which is a two-point boundary value problem where $x_1(t_0)$, $x_2(t_0)$, $x_1(t_f)$ are specified and $x_2(t_f)$ is unspecified.

- $\lambda_1(t)$ and $\lambda_2(t)$ are Lagrange multiplier functions that preserve the system state equations 2.20 and together constitute the so-called adjoint variables [1].
- $\mu_1(t)$ and $\mu_2(t)$ are Lagrange multiplier functions that preserve the inequality constraints on the the displacement command $u(t)$.
- $\mu_3(t)$ and $\mu_4(t)$ are Lagrange multiplier functions that preserve the inequality constraints on the state $x_2(t)$.
- ν_1 is a Lagrange multiplier constant that preserves the boundary condition on $x_2(t_f)$.

At this point, the cost functional contains all information regarding control objectives, the system, and its associated inequality constraints. The goal is to find input trajectories $\tau_{ICE}(t)^*$, $u(t)^*$ which minimize J. The calculus of variations is utilized to find these trajectories.

3.2 Derivation of Optimality Conditions

Taking the first variation on J 3.5 yields a set of optimality conditions the optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$ must satisfy in order for J to be minimized. The first variation is defined as

$$\delta J = \left[\begin{array}{c} \frac{d}{d\varepsilon} J\left(x_1(t) + \varepsilon h_{x_1}(t)\right) \\ \frac{d}{d\varepsilon} J\left(x_2(t) + \varepsilon h_{x_2}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\tau_{ICE}(t) + \varepsilon h_{T_e}(t)\right) \\ \frac{d}{d\varepsilon} J\left(u(t) + \varepsilon h_u(t)\right) \\ \frac{d}{d\varepsilon} J\left(\lambda_1(t) + \varepsilon h_{\lambda_1}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\lambda_2(t) + \varepsilon h_{\lambda_2}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\mu_1(t) + \varepsilon h_{\mu_1}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\mu_2(t) + \varepsilon h_{\mu_2}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\mu_3(t) + \varepsilon h_{\mu_3}(t)\right) \\ \frac{d}{d\varepsilon} J\left(\mu_4(t) + \varepsilon h_{\mu_4}(t)\right) \end{array} \right] \Bigg|_{\varepsilon=0} \tag{3.6}$$

where ε is a scalar and $h(t)$ is a smooth function. Adding $\varepsilon h(t)$ to any of the state or input functions represents a variation of that function. Setting $\delta J = 0$ gives an extremal (minimum) value of the cost functional, J. Also, 3.6 is a vector due to the presence of multiple functions in the cost functional. Before proceeding with analysis two useful tools need to be set forth – namely, the fundamental lemma of the calculus of variations, and differentiation under the integral sign. First, the fundamental lemma of the calculus of variations states that if an integral of the form

$$\int_{t_0}^{t_f} M(t)\, h(t)\, dt = 0$$

and both $M(t)$, and $h(t)$ are continuous functions with $h(t_0), h(t_f) = 0$, then $M(t)$ is also equal to zero. For a proof of the fundamental lemma of the calculus of variations

see [11]. The next tool, differentiation under the integral, states that

$$\frac{d}{d\varepsilon}\int_{t_0}^{t_f} f\left(t,\varepsilon\right)dt = \int_{t_0}^{t_f} \frac{\partial}{\partial\varepsilon} f\left(t,\varepsilon\right)dt \tag{3.7}$$

For a proof of 3.7 see [12]. With these tools in place, each term in 3.6 can be computed. For convenience all sub-notation on $h\left(t\right)$ will be dropped without any loss of generality. To start, take the first variation on J with respect to $\tau_{ICE}\left(t\right)$.

$$J\left(\tau_{ICE}\left(t\right)+\varepsilon h\left(t\right)\right) = \int_{t_0}^{t_f} \left(\tau_{ICE}\left(t\right)+\varepsilon h\left(t\right)\right)^2 - K_e\lambda_1\left(t\right)\left(\tau_{ICE}\left(t\right)+\varepsilon h\left(t\right)\right)dt \tag{3.8}$$

Using 3.7 gives

$$\frac{d}{d\varepsilon}J\left(\tau_{ICE}\left(t\right)+\varepsilon h\left(t\right)\right)\Big|_{\varepsilon=0} = \int_{t_0}^{t_f} 2\tau_{ICE}\left(t\right)h\left(t\right) - K_e\lambda_1\left(t\right)h\left(t\right)dt = 0 \tag{3.9}$$

Applying the fundamental lemma of the calculus of variations yields

$$\tau_{ICE}\left(t\right) = \frac{K_e}{2}\lambda_1\left(t\right) \tag{3.10}$$

Equation 3.10 is the control law for engine torque, $\tau_{ICE}\left(t\right)$. Next, the first variation on J with respect to state $x_1\left(t\right)$ is

$$\begin{array}{c} J\left(x_1\left(t\right)+\varepsilon h\left(t\right)\right) = \int_{t_0}^{t_f} \lambda_1\left(t\right)\left(\dot{x}_1\left(t\right)+\varepsilon\dot{h}\left(t\right)\right) + B\lambda_1\left(t\right)\left(x_1\left(t\right)+\varepsilon h\left(t\right)\right)^2 \\ -K_v\lambda_2\left(t\right)u\left(t\right)\left(x_1\left(t\right)+\varepsilon h\left(t\right)\right)dt + \nu_1\left(k_1 - x_1\left(t_f\right) - \varepsilon h\left(t_f\right)\right) \end{array} \tag{3.11}$$

$$\begin{array}{c} \frac{d}{d\varepsilon}J\left(x_1\left(t\right)+\varepsilon h\left(t\right)\right)\big|_{\varepsilon=0} = \int_{t_0}^{t_f} \lambda_1\left(t\right)\dot{h}\left(t\right) + 2B\lambda_1\left(t\right)x_1\left(t\right)h\left(t\right) \\ -K_v\lambda_2\left(t\right)u\left(t\right)h\left(t\right)dt - \nu_1 h\left(t_f\right) = 0 \end{array} \tag{3.12}$$

In order to deal with the first term under the integral in 3.12 employ integration by parts using

$$\begin{array}{cc} u = \lambda_1\left(t\right) & du = \dot{\lambda}_1\left(t\right)dt \\ v = h\left(t\right) & dv = \dot{h}\left(t\right)dt \end{array}$$

which yields

$$\begin{array}{c} \lambda_1\left(t_f\right)h\left(t_f\right) - \lambda_1\left(t_0\right)h\left(t_0\right) + \int_{t_0}^{t_f} -\dot{\lambda}_1\left(t\right)h\left(t\right) + 2B\lambda_1\left(t\right)x_1\left(t\right)h\left(t\right) \\ -K_v\lambda_2\left(t\right)u\left(t\right)h\left(t\right)dt - \nu_1 h\left(t_f\right) = 0 \end{array} \tag{3.13}$$

Since $x_1(t_0)$ is specified, the second term on the left vanishes because the state variation $h(t_0) = 0$. Also, choose $\lambda_1(t_f) = \nu_1$ so as to eliminate any state variation at the terminal time, $h(t_f)$. Applying the fundamental lemma of the calculus of variations yields

$$\begin{gathered} \dot{\lambda}_1(t) = 2B\lambda_1(t)\, x_1(t) - K_v\lambda_2(t)\, u(t) \\ \lambda_1(t_f) = \nu_1 \end{gathered} \tag{3.14}$$

The first variation on J with respect to state $x_2(t)$ is

$$\begin{gathered} J(x_2(t) + \varepsilon h(t)) = \int_{t_0}^{t_f} \frac{K_a\lambda_1(t)u(t)}{V_a - x_2(t) - \varepsilon h(t)} + \lambda_2(t)\left(\dot{x}_2(t) + \varepsilon\dot{h}(t)\right) \\ + \frac{K_L}{V_a - x_2(t) - \varepsilon h(t)} + (\mu_3(t) - \mu_4(t))(x_2(t) + \varepsilon h(t))\, dt \end{gathered} \tag{3.15}$$

$$\begin{gathered} \frac{d}{d\varepsilon} J(x_2(t) + \varepsilon h(t))\big|_{\varepsilon=0} = \int_{t_0}^{t_f} \frac{K_a\lambda_1(t)u(t) - K_L\lambda_2(t)}{(V_a - x_2(t))^2} h(t) + \lambda_2(t)\,\dot{h}(t) \\ + (\mu_3(t) - \mu_4(t))\, h(t)\, dt = 0 \end{gathered} \tag{3.16}$$

To deal with the second term under the integral sign, again use integration by parts

$$\begin{aligned} u &= \lambda_2(t) \quad & du &= \dot{\lambda}_2(t)\, dt \\ v &= h(t) & dv &= \dot{h}(t)\, dt \end{aligned}$$

which yields

$$\begin{gathered} \lambda_2(t_f)\, h(t_f) - \lambda_2(t_0)\, h(t_0) + \int_{t_0}^{t_f} -\dot{\lambda}_2(t)\, h(t) + \frac{K_a\lambda_1(t)u(t) - K_L\lambda_2(t)}{(V_a - x_2(t))^2} h(t) \\ + (\mu_3(t) - \mu_4(t))\, h(t)\, dt = 0 \end{gathered} \tag{3.17}$$

Since $x_2(t_0)$ is specified, the second term on the left vanishes because the state variation $h(t_0) = 0$. Also, choose $\lambda_2(t_f) = 0$ so as to eliminate any state variation at the terminal time, $h(t_f)$. Applying the fundamental lemma of the calculus of variations yields

$$\begin{gathered} \dot{\lambda}_2(t) = \frac{K_a\lambda_1(t)u(t) - K_L\lambda_2(t)}{(V_a - x_2(t))^2} + \mu_3(t) - \mu_4(t) \\ \lambda_2(t_f) = 0 \end{gathered} \tag{3.18}$$

Together, equations 3.14 and 3.18 are called the adjoint, or influence equations [1]. Since $\lambda_1(t_f)$ and $\lambda_2(t_f)$ are known, the adjoint system must be solved backwards in time. The first variation on J with respect to $\lambda_1(t)$ and $\lambda_2(t)$ simply amounts to the system equations 2.20, which is expected. The first variation on J with respect to $\mu_1(t)$, $\mu_2(t)$, $\mu_3(t)$, and $\mu_4(t)$ leaves us with the original inequality constraints. In order to impose the inequality constraints on both state and input variables, the Karush Kuhn Tucker

(KKT) Necessary Conditions for Optimality [10] are utilized which, for this system, state that

$$\begin{aligned} u_{min} - u(t) &\leq 0 \\ u(t) - u_{max} &\leq 0 \\ V_{min} - x_2(t) &\leq 0 \\ x_2(t) - V_{max} &\leq 0 \end{aligned} \tag{3.19}$$

3.19 are the so-called Primal Feasibility conditions.

$$\mu_i \geq 0, \ for \ i = 1, 2, 3, 4 \tag{3.20}$$

3.20 are the so-called Dual Feasibility conditions.

$$\begin{aligned} \mu_1(t)(u_{min} - u(t)) &= 0 \\ \mu_2(t)(u(t) - u_{max}) &= 0 \\ \mu_3(t)(V_{min} - x_2(t)) &= 0 \\ \mu_4(t)(x_2(t) - V_{max}) &= 0 \end{aligned} \tag{3.21}$$

3.21 are the so-called Complementary Slackness Conditions. The first variation on J with respect to input $u(t)$ is

$$\begin{aligned} J(u(t) + \varepsilon h(t)) = \int_{t_0}^{t_f} \left(\frac{K_a \lambda_1(t)}{V_a - x_2(t)} - K_v x_1(t) \lambda_2(t) \right) (u(t) + \varepsilon h(t)) \\ + (\mu_1(t) - \mu_2(t))(u(t) + \varepsilon h(t)) \, dt \end{aligned} \tag{3.22}$$

$$\begin{aligned} \frac{d}{d\varepsilon} J(u(t) + \varepsilon h(t))\big|_{\varepsilon=0} = \int_{t_0}^{t_f} \left(\frac{K_a \lambda_1(t)}{V_a - x_2(t)} - K_v x_1(t) \lambda_2(t) \right) h(t) \\ + (\mu_1(t) - \mu_2(t)) h(t) \, dt = 0 \end{aligned} \tag{3.23}$$

Applying the fundamental lemma of the calculus of variations yields

$$\frac{K_a \lambda_1(t)}{V_a - x_2(t)} - K_v x_1(t) \lambda_2(t) + \mu_1(t) - \mu_2(t) = 0 \tag{3.24}$$

Equation 3.24 is interesting because it is not an explicit control law for $u(t)$. Instead it is an equality constraint acting on both state and adjoint variables. Usually imposing such a constraint amounts to differentiating it with respect to time and appending the result to whatever system it is acting on. This however cannot be done due to the presence of $\mu_1(t)$ and $\mu_2(t)$, which are discontinuous in nature. Section 3.4 shows how to enforce this constraint numerically. Finally, the first variation on J with respect to ν_1 gives the original end condition 3.2.

3.3 Optimality Conditions

The optimality conditions derived in 3.2 are listed below as 3.25. They preserve the original system, give the time histories of the adjoint variables, and define control laws for the input functions. They also preserve the boundary condition on the final state, $x_1(t_f)$, and all inequality constraints on the system states and inputs.

$$\begin{array}{rl}
\dot{x}_1(t) = K_e\tau_{ICE}(t) - \frac{K_a}{V_a - x_2(t)}u(t) - Bx_1(t)^2 & x_1(t_0) = specified \\
\dot{x}_2(t) = K_v x_1(t)\, u(t) - \frac{K_L}{V_a - x_2(t)} & x_2(t_0) = specified \\
\\
\dot{\lambda}_1(t) = 2B\lambda_1(t)\, x_1(t) - K_v\lambda_2(t)\, u(t) & \lambda_1(t_f) = \nu_1 \\
\dot{\lambda}_2(t) = \frac{K_a\lambda_1(t)u(t) - K_L\lambda_2(t)}{(V_a - x_2(t))^2} + \mu_3(t) - \mu_4(t) & \lambda_2(t_f) = 0
\end{array} \tag{3.25}$$

$$\tau_{ICE}(t) = \frac{K_e}{2}\lambda_1(t)$$

$$\frac{\partial J}{\partial u(t)} = \frac{K_a\lambda_1(t)}{V_a - x_2(t)} - K_v x_1(t)\,\lambda_2(t) + \mu_1(t) - \mu_2(t) = 0$$

$$\frac{\partial J}{\partial \nu_1} = k_1 - x_1(t_f) = 0$$

$$\begin{array}{c}
\frac{\partial J}{\partial \mu_1(t)} = u_{min} - u(t) \le 0 \\
\frac{\partial J}{\partial \mu_2(t)} = u(t) - u_{max} \le 0 \\
\frac{\partial J}{\partial \mu_3(t)} = V_{min} - x_2(t) \le 0 \\
\frac{\partial J}{\partial \mu_4(t)} = x_2(t) - V_{max} \le 0
\end{array}$$

$$\mu_i(t) \ge 0,\ for\ i = 1,2,3,4$$

$$\begin{array}{c}
\mu_1(t)\,(u_{min} - u(t)) = 0 \\
\mu_2(t)\,(u(t) - u_{max}) = 0 \\
\mu_3(t)\,(V_{min} - x_2(t)) = 0 \\
\mu_4(t)\,(x_2(t) - V_{max}) = 0
\end{array}$$

The optimal input sequences, $\tau_{ICE}(t)^*$, $u(t)^*$, are those which satisfy the optimality conditions. Since they cannot be found directly, their solution relies on an iterative numerical technique. A gradient descent based algorithm for solving the optimality conditions is developed next.

3.4 Gradient Descent Optimal Control Algorithm

An algorithm for finding optimal input trajectories, $\tau_e(t)^*$, $u(t)^*$, is now developed. It is a gradient descent based algorithm and its goal is to solve the optimality conditions derived in 3.2 and listed in section 3.3. The algorithm is as follows.

- Initialize the control sequences $\tau_{ICE}(t)^{k=0}$ and $u(t)^{k=0}$. Typically zeros suffice i.e. $\tau_{ICE}(t)^{k=0} = u(t)^{k=0} = 0 \forall t \in \{t_0, t_f\}$. Also, initialize $\nu_1^{k=0}$. Again, $\nu_1^{k=0} = 0$ works fine.

- Solve the system equations 2.20 forward in time using $x_1(t_0) = specified$, $x_2(t_0) = specified$, $t_f = specified$ and the control sequences $\tau_{ICE}(t)^k$ and $u(t)^k$.

- Solve the adjoint equations 3.25 backwards in time using $\lambda_1(t_f) = \nu_1^k$ and $\lambda_2(t_f) = 0$.

- Compute gradient terms

$$\frac{\partial J}{\partial u(t)}^k = \frac{K_a \lambda_1(t)^k}{V_a - x_2(t)^k} - K_v x_1(t)^k \lambda_2(t)^k + \mu_1(t)^k - \mu_2(t)^k$$
$$\frac{\partial J}{\partial \nu_1}^k = k_1 - x_1(t_f)^k$$
$$\frac{\partial J}{\partial \mu_1(t)}^k = u_{min} - u(t)^k$$
$$\frac{\partial J}{\partial \mu_2(t)}^k = u(t)^k - u_{max}$$
$$\frac{\partial J}{\partial \mu_3(t)}^k = V_{min} - x_2(t)^k$$
$$\frac{\partial J}{\partial \mu_4(t)}^k = x_2(t)^k - V_{max}$$

- Update

$$\tau_{ICE}(t)^{k+1} = \frac{K_e}{2} \lambda_1(t)^k$$
$$u(t)^{k+1} = u(t)^k - \alpha_u \frac{\partial J}{\partial u(t)}^k$$
$$\nu_1^{k+1} = \nu_1^k - \alpha_{\nu_1} \frac{\partial J}{\partial \nu_1}^k$$
$$\mu_1(t)^{k+1} = max\left(0,\ \mu_1(t)^k + \alpha_{\mu_1} \frac{\partial J}{\partial \mu_1(t)}^k\right)$$
$$\mu_2(t)^{k+1} = max\left(0,\ \mu_2(t)^k + \alpha_{\mu_2} \frac{\partial J}{\partial \mu_2(t)}^k\right)$$
$$\mu_3(t)^{k+1} = max\left(0,\ \mu_3(t)^k + \alpha_{\mu_3} \frac{\partial J}{\partial \mu_3(t)}^k\right)$$

$$\mu_4(t)^{k+1} = max\left(0,\ \mu_4(t)^k + \alpha_{\mu_4}\frac{\partial J}{\partial \mu_4(t)}^k\right)$$

- Evaluate Termination Criteria

$$\begin{aligned}\Gamma^k = & \left(\frac{\partial J}{\partial \nu_1}^k\right)^2 + \left(\frac{\partial J}{\partial u(t)}^k\right)^T \frac{\partial J}{\partial u(t)}^k + \\ & \left(\mu_1(t)^k\right)^T \frac{\partial J}{\partial \mu_1(t)}^k + \left(\mu_2(t)^k\right)^T \frac{\partial J}{\partial \mu_2(t)}^k + \\ & \left(\mu_3(t)^k\right)^T \frac{\partial J}{\partial \mu_3(t)}^k + \left(\mu_4(t)^k\right)^T \frac{\partial J}{\partial \mu_4(t)}^k\end{aligned}$$

The update formulas achieve convergence through gradient descent where α_{xx} is the step size (typically small) and $\frac{\partial J}{\partial_{xx}}$ is the search direction. Much research has been done in the realm of gradient descent in terms of step size selection and convergence analysis [10] [2]. For our purposes, we elect to use a constant step size, α, in each of the update formulas. The $\mu_i(t)$ update formulas preserve the first two Karush Kuhn Tucker (KKT) necessary conditions [10]. Qualitatively, they keep each $\mu_i(t)$ positive when its respective inequality constraint is active, and zero when it is inactive. When the termination criteria, Γ^k, is sufficiently close to zero, convergence has occurred causing the algorithm to stop. Γ is also responsible for preserving the third KKT condition on the inequality constraints. For an implementation of this algorithm in Matlab$^{\text{TM}}$ see A.

3.5 Results and Discussion

The optimal control algorithm developed in the last chapter is now applied to a series of problems. The problems are such that starting from rest ($x_1(t_0) = 0$) and with a nearly empty accumulator ($x_2(t_0) = 0.3 * V_a$), the vehicle speed must achieve a specified final value over different aerodynamic and volumetric loss constants; B and K_L.

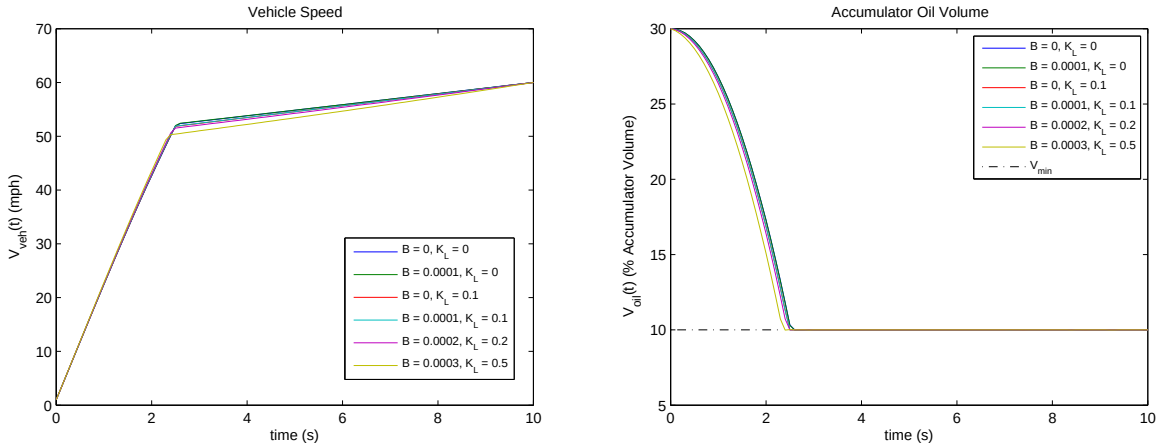

Figure 3.1: Optimal state trajectories.

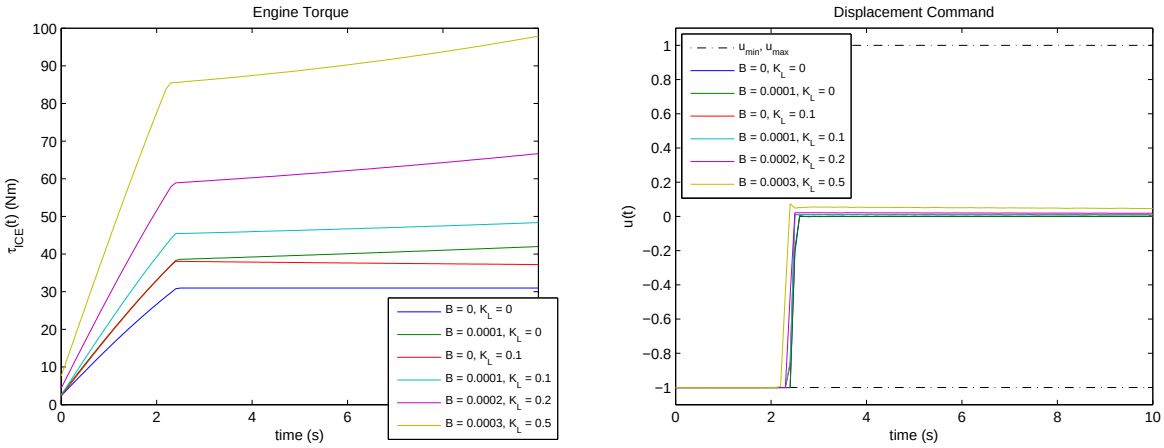

Figure 3.2: Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$.

Figures 3.1 and 3.2 were generated in Matlab^{TM}. The optimal input and state trajectories in each case are subject to active inequality constraints at some point over the time interval. One trait common among all cases is that the pump-motor unit should be set to full negative displacement (motoring) until all energy in the accumulator is exhausted. Because of this, the engine does not need to produce as much torque during the period in which the pump-motor unit is active. Once the energy in the accumulator

is depleted, however, the engine must make up the necessary torque. Another set of traits common in each case are the speed profiles. They are largely linear both before and after the accumulator is empty. This optimal speed characteristic will help explain the various input trajectories.

An interesting and reassuring result is that when leakage is high ($K_L = 0.5$) and the state inequality constraint is active, the pump-motor displacement must be slightly greater than zero because leakage is always occurring, independent of displacement. In the loss-less case ($B = K_L = 0$), once the accumulator is empty, the resulting engine torque is constant. In the presence of zero leakage and aerodynamic drag ($B = 0.0001$, $K_L = 0$), after the accumulator is empty, the resulting engine torque increases, making up for aerodynamic losses. Contrasting this, in the presence of leakage and zero aerodynamic drag ($B = 0$, $K_L = 0.1$), the engine torque spikes up to a point while the accumulator is releasing its energy. Once the accumulator is depleted, the engine torque slowly decreases. This is because the engine must put a certain amount of energy back in to the accumulator so as to make up for any leakage. It does so in a manner that preserves the nearly linear speed characteristic common in each case.

These results lead to a few rules for efficient practical operation of a pre-transmission parallel HHV power train: If a speed increase is desired and the accumulator has energy in it, set the pump-motor unit to full displacement (motor mode) until the accumulator is depleted. While the accumulator is dumping energy into the system, adjust the engine torque such that the vehicle maintains constant acceleration. Once the accumulator is depleted, the engine torque should continue to maintain constant acceleration, even in the presence of aerodynamic and volumetric losses. A lookup table coupled with a feedback controller could be employed to obey these rules in a real time application.

The results also illicit an important discrepancy in the original problem formulation 3.1. Looking at the plot of vehicle speed in figure 3.1, the vehicle achieves 90% of its final velocity in the first 2.5 seconds, which is not neccesarily desirable. In minimizing effort from the engine, the vehicle is reckless with accumulator energy. The root cause goes back to the original problem statement where the goal was to minimize effort from the engine only. No specification was made to minimize effort from the hydraulic branch, or control its behavior at all. Due to this discrepancy a new formulation is required – one which values the energy in the hydraulic branch.

In terms of algorithm performance, 3.4 is fully capable of solving optimal control problems of the type simulated in this section. For the most part, convergence occurs within a few minutes. Unfortunately the convergence rate is very sensitive to step size selection. If the step sizes are too large, the algorithm diverges; if too small, convergence is slow. There is room for improvement.

Chapter 4

A Non-Linear Programming Approach to Optimal Vehicle Control

In chapter 3 the original optimal control problem statement was made. Optimality conditions were derived and the problem was solved numerically using the Gradient Descent Optimal Control Algorithm. Due to discrepancies in the original problem statement, and the algorithm, a new formulation is required for both.

4.1 Problem Statement Reformulation

The issue with the original problem statement is that it does not place value on effort from the hydraulic branch. In reformulation, the new problem statement becomes

$$minimize \int_{t_0}^{t_f} \tau_{ICE}(t)^2 + \tau_{hydraulics}(t)^2 \, dt \tag{4.1}$$

subject to the same constraints as in 3.1 and one additional constraint on the engine:

$$\tau_{ICE}(t) \geq 0$$

The additional constraint on the engine prevents it from doing any braking in the case of a deceleration event. This causes all braking to be done by the hydraulics, thus storing

the vehicle's kinetic energy. The difference between 4.1 and 3.1 is the addition of the hydraulic effort term, $\tau_{hydraulics}(t)^2$. The problem statement now values effort from the engine and hydraulics equally. For simplicity's sake, we will use the ideal (loss-less) vehicle model. Setting $B = K_L = 0$ in 2.20 gives

$$\begin{aligned} \dot{x}_1(t) &= K_e \tau_{ICE}(t) - \frac{K_a}{V_a - x_2(t)} u(t) \\ \dot{x}_2(t) &= K_v x_1(t)\, u(t) \end{aligned} \tag{4.2}$$

In 4.2 the applied hydraulic torque is

$$\tau_{hydraulics}(t) = \frac{-D k_{gas} u(t)}{2\pi (V_a - x_2(t))} \tag{4.3}$$

Squaring 4.3 gives

$$\tau_{hydraulics}(t)^2 = \frac{K_4 u(t)^2}{(V_a - x_2(t))^2}$$

where $K_4 = \left(\frac{-D k_{gas}}{2\pi}\right)^2$. At this point a cost functional can be formulated from which a new set of optimality conditions can be found, and the Gradient Descent Optimal Control Algorithm 3.4 can be used to solve them. Unfortunately finding the optimality conditions requires taking the first variation of the cost functional and is a tedious endeavor (see 3.2). In the next section a new approach to optimal control is developed that side-steps COV entirely, and results in a far more robust optimal control algorithm.

4.2 Optimal Control Reformulation

In this section a new approach to the solution of optimal control problems is developed. The fundamental idea is to discretize the problem in time and convert it to a non-linear program (NLP). The NLP contains mixed inequality/equality constraints which are enforced using a Lagrange multiplier formulation. The NLP is solved using an implementation of Newton's Method.

4.2.1 General Form of the Optimal Control Problem

The general form of the continuous-time optimal control problem with some initial and final conditions specified and fixed initial and final times is

$$\begin{aligned} &minimize \int_{t_0}^{t_f} Q\left(x\left(t\right), u\left(t\right), t\right) dt \; s.t. \\ &\dot{x}\left(t\right) = f\left(x\left(t\right), u\left(t\right), t\right) \\ &g\left(x\left(t\right), u\left(t\right), t\right) \leq 0 \\ &h_{t_0}\left(x\left(t_0\right)\right) = 0 \\ &h_{t_f}\left(x\left(t_f\right)\right) = 0 \end{aligned} \tag{4.4}$$

- $x\left(t\right) \in n \times 1$ is the system state vector.
- $u\left(t\right) \in m \times 1$ is a vector of system inputs.
- t_0, t_f are fixed initial and final times.
- $Q\left(x\left(t\right), u\left(t\right), t\right)$ is the unconstrained cost, i.e. that which is being minimized.
- $g\left(x\left(t\right), u\left(t\right), t\right) \leq 0 \in p \times 1$ is the set of inequality constraints acting on state and/or input variables.
- $h_{t_0}\left(x\left(t_0\right)\right) = 0$, $h_{t_f}\left(x\left(t_f\right)\right) = 0$ are initial and terminal constraints on the states. They allow for certain initial and final conditions to be specified.

By setting $\dot{x}_{n+1}\left(t\right) = Q\left(x\left(t\right), u\left(t\right), t\right)$ and appending it to the states, 4.4 becomes

$$\begin{aligned} &minimize \; x_{n+1}\left(t_f\right) \; s.t. \\ &\dot{x}\left(t\right) = f\left(x\left(t\right), u\left(t\right), t\right) \\ &g\left(x\left(t\right), u\left(t\right), t\right) \leq 0 \\ &h_{t_0}\left(x\left(t_0\right)\right) = 0 \\ &h_{t_f}\left(x\left(t_f\right)\right) = 0 \end{aligned} \tag{4.5}$$

where $x_{n+1}\left(t_f\right) = x_{n+1}\left(t_0\right) + \int_{t_0}^{t_f} \dot{x}_{n+1}\left(t\right) dt$, and $x_{n+1}\left(t_0\right) = 0$. The problem can be discretized in time using Euler's method, which states that an ordinary differential equation (ODE) of the form $\dot{x}\left(t\right) = f\left(x\left(t\right), u\left(t\right), t\right)$ can be integrated approximately

by $x(t+h) = x(t) + hf(x(t), u(t), t)$. Where h is a small increment in time. In discretization, the optimal control problem takes the form

$$\begin{gathered}
minimize\ x_{n+1}(t_f)\ \ s.t. \\
x(t_1) = x(t_0) + hf(x(t_0), u(t_0), t_0) \\
x(t_2) = x(t_1) + hf(x(t_1), u(t_1), t_1) \\
\vdots \\
x(t_f) = x(t_f - h) + hf(x(t_f - h), u(t_f - h), t_f - h) \\
g(x(t_0), u(t_0), t_0) \leq 0 \\
g(x(t_1), u(t_1), t_1) \leq 0 \\
\vdots \\
g(x(t_f), u(t_f), t_f) \leq 0 \\
h_{t_0}(x(t_0)) = 0 \\
h_{t_f}(x(t_f)) = 0
\end{gathered} \tag{4.6}$$

In its current form, the optimal control problem is an NLP. For simplicity, re-write the discretized state vector $x(t) \in \{t_0, t_f\}$ as

$$[x(t_0), x(t_0 + h), \cdots, x(t_f)] \equiv [x(1), x(2), \cdots, x(T)]$$

and the discretized input vector $u(t) \in \{t_0, t_f - h\}$ as

$$[u(t_0), u(t_0 + h), \cdots, u(t_f - h)] \equiv [u(1), u(2), \cdots, u(T-1)]$$

where $T = \frac{t_f - t_0}{h} + 1$. The Lagrangian of 4.6 is

$$\begin{gathered}
L = x_{n+1}(T) + \\
\lambda_1(x(2) - x(1) - hf(x(1), u(1))) + \\
\lambda_2(x(3) - x(2) - hf(x(2), u(2))) + \\
\vdots \\
\lambda_{T-1}(x(T) - x(T-1) - hf(x(T-1), u(T-1))) + \\
\mu_1\left(g(x(1), u(1)) + z_1^2\right) + \\
\mu_2\left(g(x(2), u(2)) + z_2^2\right) + \\
\vdots \\
\mu_T\left(g(x(T)) + z_T^2\right) + \\
\nu_0 h_{t_0}(x(1)) + \\
\nu_T h_{t_f}(x(T))
\end{gathered} \tag{4.7}$$

- $\lambda_i \in n+1 \times 1$ are Lagrange multiplier vectors enforcing equality constraints at each time step, i.
- $\mu_i \in p \times 1$ are Lagrange multiplier vectors enforcing inequality constraints at each time step, i.
- $z_i \in p \times 1$ are slack variable vectors enforcing inequality constraints at each time step, i. They convert the inequality constrained problem to an equality constrained problem, and are squared because they must be positive [10].
- ν_0, ν_T are Lagrange multiplier vectors enforcing initial and final conditions on specified states.

The goal is to minimize the Lagrangian 4.7. Taking the partials of each variable and setting equal to zero gives the following optimality conditions:

- $\frac{\partial L}{\partial x} = \begin{bmatrix} \frac{\partial L}{\partial x(1)} \\ \frac{\partial L}{\partial x(2)} \\ \vdots \\ \frac{\partial L}{\partial x(T)} \end{bmatrix} = 0$ is a column vector where $\frac{\partial L}{\partial x(i)} \in n+1 \times 1$
- $\frac{\partial L}{\partial u} = \begin{bmatrix} \frac{\partial L}{\partial u(1)} \\ \frac{\partial L}{\partial u(2)} \\ \vdots \\ \frac{\partial L}{\partial u(T-1)} \end{bmatrix} = 0$ is a column vector where $\frac{\partial L}{\partial u(i)} \in m \times 1$
- $\frac{\partial L}{\partial \lambda} = \begin{bmatrix} \frac{\partial L}{\partial \lambda_1} \\ \frac{\partial L}{\partial \lambda_2} \\ \vdots \\ \frac{\partial L}{\partial \lambda_{T-1}} \end{bmatrix} = 0$ is a column vector where $\frac{\partial L}{\partial \lambda_i} \in n+1 \times 1$
- $\frac{\partial L}{\partial \mu} = \begin{bmatrix} \frac{\partial L}{\partial \mu_1} \\ \frac{\partial L}{\partial \mu_2} \\ \vdots \\ \frac{\partial L}{\partial \mu_T} \end{bmatrix} = 0$ is a column vector where $\frac{\partial L}{\partial \mu_i} \in p \times 1$

- $\frac{\partial L}{\partial z} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \\ \vdots \\ \frac{\partial L}{\partial z_T} \end{bmatrix} = 0$ is a column vector where $\frac{\partial L}{\partial z_i} \in p \times 1$
- $\frac{\partial L}{\partial \nu_0} = 0$ is a column vector where number of rows is dependent on number of specified initial conditions.
- $\frac{\partial L}{\partial \nu_T} = 0$ is a column vector where number of rows is dependent on number of specified final conditions.

Defining the vector of all system variables as

$$X \equiv \begin{bmatrix} x \\ u \\ \lambda \\ \mu \\ z \\ \nu_0 \\ \nu_T \end{bmatrix}$$

allows the optimality conditions to be written succinctly as

$$\frac{\partial L}{\partial X} = \begin{bmatrix} \frac{\partial L}{\partial x} \\ \frac{\partial L}{\partial u} \\ \frac{\partial L}{\partial \lambda} \\ \frac{\partial L}{\partial \mu} \\ \frac{\partial L}{\partial z} \\ \frac{\partial L}{\partial \nu_0} \\ \frac{\partial L}{\partial \nu_T} \end{bmatrix} = 0 \tag{4.8}$$

By discretizing the optimal control problem in time, optimality conditions can be derived easily, without the use of COV. The slack variables z_i and lagrange multipliers μ_i convert the inequality constrained problem to an equality constrained problem. This is important because it allows the problem to be solved with an aggressive solver, specifically Newton's Method.

4.2.2 Newton's Method for Solution to Optimality Conditions

Newton's Method is used to solve systems of non-linear equations. It is an iterative process and works by solving a linear approximation to a non-linear system. The solution is plugged back in to the non-linear system and the process is repeated until convergence. The optimality conditions 4.8 are a system of non-linear equations and can be solved using Newton's Method. A derivation of Newton's Method begins with a linear approximation to the non-linear system $\frac{\partial L}{\partial X}$

$$\frac{\partial L}{\partial X}^{k+1} - \frac{\partial L}{\partial X}^{k} = \frac{\partial^2 L}{\partial X^2}^{k} \left(X^{k+1} - X^k \right) \tag{4.9}$$

- X^k is the current vector of system variables.
- X^{k+1} is the next vector of system variables.
- $\frac{\partial L}{\partial X}^{k}$ is the non-linear system evaluated at X^k.
- $\frac{\partial L}{\partial X}^{k+1}$ is the non-linear system evaluated at X^{k+1}.
- $\frac{\partial^2 L}{\partial X^2}^{k}$ is the slope of the non-linear system evaluated at X^k.

Setting $\frac{\partial L}{\partial X}^{k+1} = 0$ and rearranging gives

$$X^{k+1} = X^k - \left(\frac{\partial^2 L}{\partial X^2}^{k} \right)^{-1} \frac{\partial L}{\partial X}^{k} \tag{4.10}$$

4.10 is Newton's Method for the solution to the optimality conditions 4.8. It iterates until $\frac{\partial L}{\partial X}$ is sufficiently close to zero. Notice the presence of second derivative information; it is leveraged to effectively determine step size for the algorithm. Newton's Method achieves quadratic convergence rates [10], and is thus capable of precisely solving optimal control problems very quickly.

4.3 Application to Vehicle Model

The algorithm developed in the last section is now applied to 4.1. Setting

$$\dot{x}_3(t) = \tau_{ICE}(t)^2 + \frac{K_4 u(t)^2}{(V_a - x_2(t))^2}$$

and appending it to 4.2 gives

$$\begin{gathered}\dot{x}_1(t) = K_e \tau_{ICE}(t) - \frac{K_a}{V_a - x_2(t)} u(t) \\ \dot{x}_2(t) = K_v x_1(t)\, u(t) \\ \dot{x}_3(t) = \tau_{ICE}(t)^2 + \frac{K_4 u(t)^2}{(V_a - x_2(t))^2}\end{gathered}$$

The optimal control problem 4.1 is now

$$\begin{gathered}minimize\ x_3(t_f)\ \ s.t. \\ \dot{x}_1(t) = K_e \tau_{ICE}(t) - \frac{K_a}{V_a - x_2(t)} u(t) \\ \dot{x}_2(t) = K_v x_1(t)\, u(t) \\ \dot{x}_3(t) = \tau_{ICE}(t)^2 + \frac{K_4 u(t)^2}{(V_a - x_2(t))^2} \\ x_1(t_0) = specified \\ x_2(t_0) = specified \\ x_3(t_0) = 0 \\ x_1(t_f) = specified \\ -1 \le u(t) \le 1 \\ x_{2_{min}} \le x_2(t) \le x_{2_{max}} \\ 0 \le \tau_{ICE}(t)\end{gathered} \tag{4.11}$$

After discretization, the Lagrangian of 4.11 is

$$
\begin{gathered}
L = x_3(T) + \\
\lambda_1(1)\left(x_1(2) - x_1(1) - h\left(K_e\tau_{ICE}(1) - \frac{K_a}{V_a - x_2(1)}u(1)\right)\right) + \\
\lambda_1(2)\left(x_2(2) - x_2(1) - h\left(K_v x_1(1)\, u(1)\right)\right) + \\
\lambda_1(3)\left(x_3(2) - x_3(1) - h\left(\tau_{ICE}(1)^2 + \frac{K_4 u(1)^2}{(V_a - x_2(1))^2}\right)\right) + \\
\vdots \\
\lambda_{T-1}(1)\left(x_1(T) - x_1(T-1) - h\left(K_e\tau_{ICE}(T-1) - \frac{K_a}{V_a - x_2(T-1)}u(T-1)\right)\right) + \\
\lambda_{T-1}(2)\left(x_2(T) - x_2(T-1) - h\left(K_v x_1(T-1)\, u(T-1)\right)\right) + \\
\lambda_{T-1}(3)\left(x_3(T) - x_3(T-1) - h\left(\tau_{ICE}(T-1)^2 + \frac{K_4 u(T-1)^2}{(V_a - x_2(T-1))^2}\right)\right) + \\
\mu_1(1)\left(-1 - u(1) + z_1(1)^2\right) + \\
\mu_1(2)\left(u(1) - 1 + z_1(2)^2\right) + \\
\mu_1(3)\left(x_{2_{min}} - x_2(1) + z_1(3)^2\right) + \\
\mu_1(4)\left(x_2(1) - x_{2_{max}} + z_1(4)^2\right) + \\
\mu_1(5)\left(0 - \tau_{ICE}(1) + z_1(5)^2\right) + \\
\vdots \\
\mu_T(1)\left(x_{2_{min}} - x_2(T) + z_T(1)^2\right) + \\
\mu_T(2)\left(x_2(T) - x_{2_{max}} + z_T(2)^2\right) + \\
\nu_0(1)\left(x_1(1) - x_1 t_0\right) + \\
\nu_0(2)\left(x_2(1) - x_2 t_0\right) + \\
\nu_0(3)\left(x_3(1) - 0\right) + \\
\nu_T(1)\left(x_1(T) - k_1\right)
\end{gathered}
$$

The components of $\frac{\partial L}{\partial X}$ take the form

$$\frac{\partial L}{\partial x} = \begin{bmatrix} \frac{\partial L}{\partial x_1(1)} \\ \vdots \\ \frac{\partial L}{\partial x_1(T)} \\ \frac{\partial L}{\partial x_2(1)} \\ \vdots \\ \frac{\partial L}{\partial x_2(T)} \\ \frac{\partial L}{\partial x_3(1)} \\ \vdots \\ \frac{\partial L}{\partial x_3(T)} \end{bmatrix} \quad \frac{\partial L}{\partial u} = \begin{bmatrix} \frac{\partial L}{\partial \tau_{ICE}(1)} \\ \vdots \\ \frac{\partial L}{\partial \tau_{ICE}(T-1)} \\ \frac{\partial L}{\partial u(1)} \\ \vdots \\ \frac{\partial L}{\partial u(T-1)} \end{bmatrix} \quad \frac{\partial L}{\partial \lambda} = \begin{bmatrix} \frac{\partial L}{\partial \lambda_1(1)} \\ \frac{\partial L}{\partial \lambda_1(2)} \\ \frac{\partial L}{\partial \lambda_1(3)} \\ \vdots \\ \frac{\partial L}{\partial \lambda_{T-1}(1)} \\ \frac{\partial L}{\partial \lambda_{T-1}(2)} \\ \frac{\partial L}{\partial \lambda_{T-1}(3)} \end{bmatrix} \quad \frac{\partial L}{\partial \mu} = \begin{bmatrix} \frac{\partial L}{\partial \mu_1(1)} \\ \frac{\partial L}{\partial \mu_1(2)} \\ \frac{\partial L}{\partial \mu_1(3)} \\ \frac{\partial L}{\partial \mu_1(4)} \\ \frac{\partial L}{\partial \mu_1(5)} \\ \vdots \\ \frac{\partial L}{\partial \mu_T(1)} \\ \frac{\partial L}{\partial \mu_T(2)} \end{bmatrix}$$

$$\frac{\partial L}{\partial z} = \begin{bmatrix} \frac{\partial L}{\partial z_1(1)} \\ \frac{\partial L}{\partial z_1(2)} \\ \frac{\partial L}{\partial z_1(3)} \\ \frac{\partial L}{\partial z_1(4)} \\ \frac{\partial L}{\partial z_1(5)} \\ \vdots \\ \frac{\partial L}{\partial z_T(1)} \\ \frac{\partial L}{\partial z_T(2)} \end{bmatrix} \quad \frac{\partial L}{\partial \nu_0} = \begin{bmatrix} \frac{\partial L}{\partial \nu_0(1)} \\ \frac{\partial L}{\partial \nu_0(2)} \\ \frac{\partial L}{\partial \nu_0(3)} \end{bmatrix} \quad \frac{\partial L}{\partial \nu_T} = \begin{bmatrix} \frac{\partial L}{\partial \nu_T(1)} \end{bmatrix}$$

Term-by-term computation of $\frac{\partial L}{\partial X}$ and $\frac{\partial^2 L}{\partial X^2}$ is straightforward and will be omitted for brevity's sake. For Matlab$^{\text{TM}}$ implementation of this algorithm, see appendix B.

4.4 Results and Discussion

The algorithm developed in this chapter is now used to solve two representative problems for the vehicle system:

1. With little energy in the accumulator, the vehicle at 45 mph needs to go to zero.
2. With plenty of energy in the accumulator, the vehicle at zero needs to go to 60 mph.

The first problem demonstrates the most efficient way for the vehicle to decelerate. The figures below are optimal vehicle trajectories.

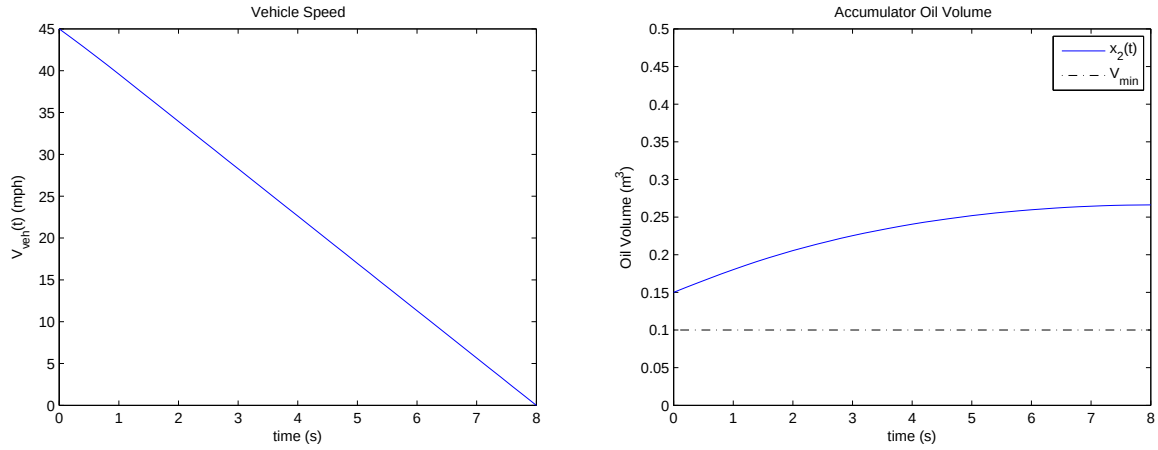

Figure 4.1: Optimal state trajectories.

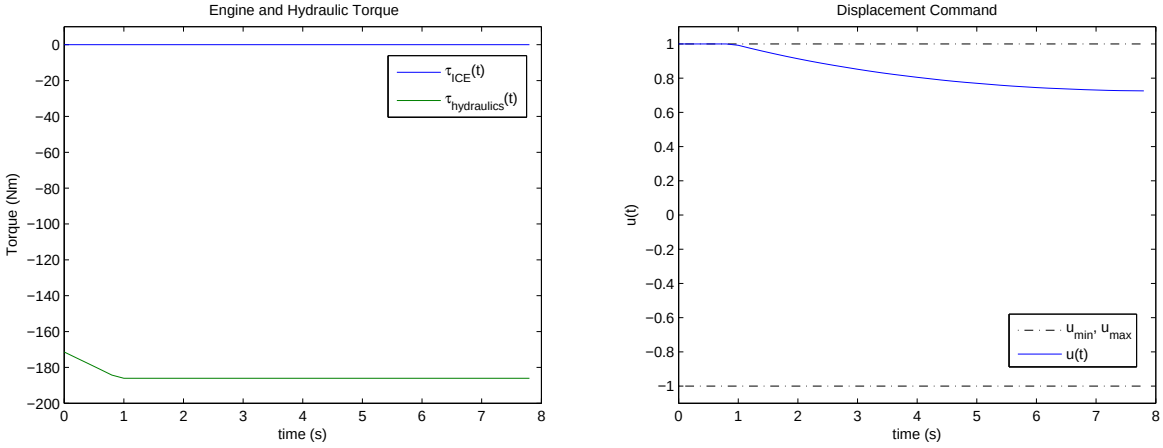

Figure 4.2: Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$.

The plot of vehicle speed in figure 4.1 shows a nearly constant vehicle deceleration. The resulting increase in oil level due to hydraulic braking is shown in the adjacent plot. Figure 4.2 shows all braking torque is supplied by the hydraulics, while engine torque

is a constant zero, which is expected. The associated displacement command clips its upper limit for the first second or so.

The second problem demonstrates the most efficient way for the vehicle to accelerate with a low accumulator oil levels. The figures below are optimal vehicle trajectories.

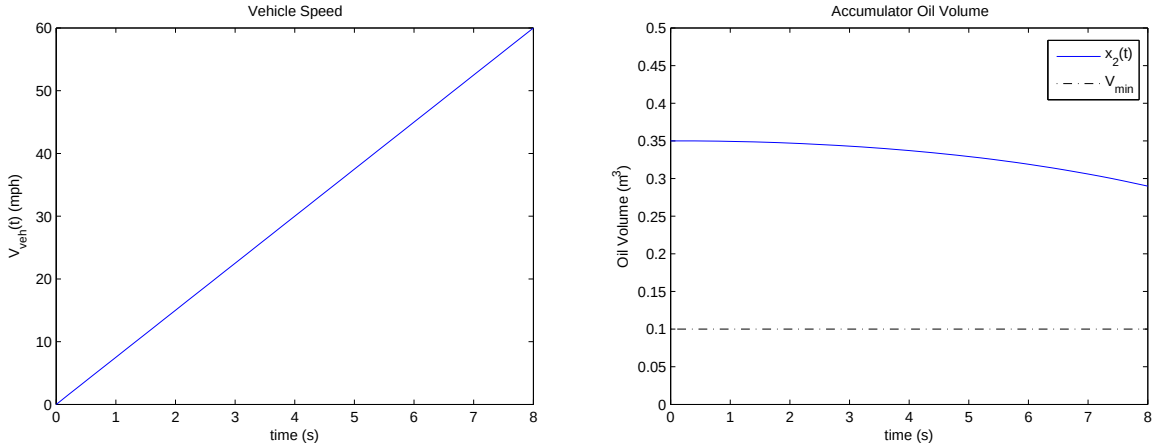

Figure 4.3: Optimal state trajectories.

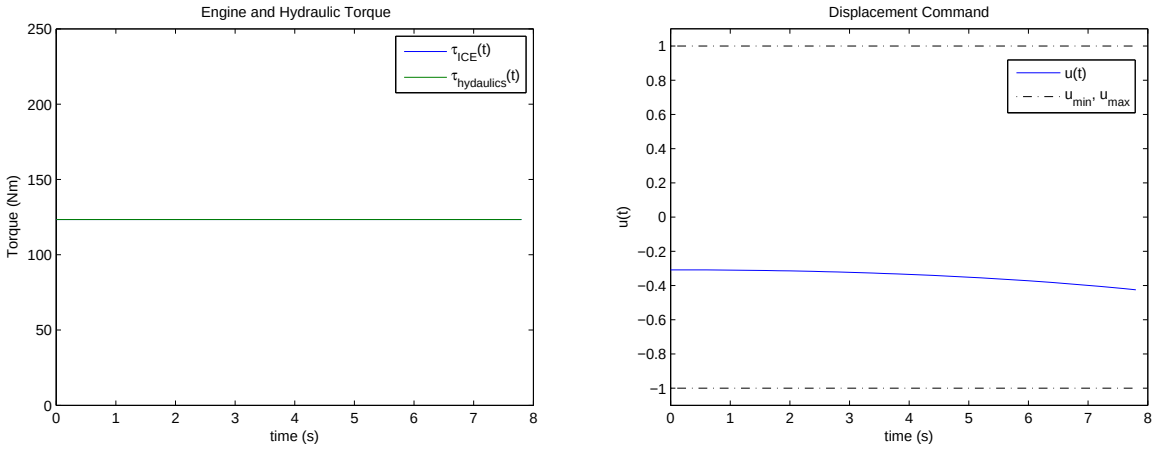

Figure 4.4: Optimal input trajectories $\tau_{ICE}(t)^*$ and $u(t)^*$.

Figure 4.3 shows a perfectly linear increase in speed. The adjacent plot shows the oil level in the accumulator decreasing due to the hydraulics supplying torque. Figure 4.4 shows engine and hydraulic torque are equal and constant. The corresponding displacement command for constant hydraulic torque is shown in the associated plot.

There are several takeaways from the above problems for practical, efficient operation of the vehicle.

- Both increases and decreases in speed should be linear. This is seen in each case as torque from the engine and hydraulics are both constant, or nearly constant.
- When the accumulator has enough energy, hydraulic torque should match engine torque. This result is expected due to the problem formulation valuing hydraulic and engine torque equally.
- All braking should be done through the hydraulic branch. This result is also expected due to the problem formulation which places an additional constraint on the engine causing it to only supply positive torque.
- Placing value on hydraulic effort results in less overall effort required from the power train. This is the most important takeaway. Valuing hydraulic effort prevents reckless usage of accumulator energy, which resolves the major discrepancy in the results from the original problem formulation.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Two algorithms have been developed, each capable of successfully solving pre-transmission parallel HHV power train optimal control problems. The goal of the original optimal control problem 3.1 was to minimize effort from the engine such that the vehicle achieves specified boundary values. The first optimal control algorithm 3.4 was developed to solve this problem. It is based on a COV framework, and utilizes a simple form of gradient descent. The resulting optimal vehicle trajectories pointed to areas where the problem formulation itself was lacking. Specifically, energy in the accumulator was being used recklessly. A new optimal control problem was formulated which resolved this issue by placing value on effort from the hydraulics. A second optimal control algorithm 4.3 was developed to solve the new problem. It works by discretizing the problem in time and converting it to an NLP. It is faster and more robust than the first algorithm, and eliminates the need for COV altogether. The solution minimized effort from the engine and hydraulics resulting in less overall effort from the power train.

The results provide practical insight for efficient power train operation: Increases or decreases in speed should be linear. The engine and the hydraulics should supply equal and constant torque to the power train when a speed increase is required. During deceleration, the hydraulics should supply all braking torque. This allows the vehicle to store kinetic energy and prevents the engine from doing any braking.

5.2 Future Work

The software developed for the NLP algorithm 4.3 is specific to the ideal vehicle model. It could easily be reworked to solve problems involving more sophisticated vehicle models such as 2.20. Completely different vehicle architectures could be investigated as well. These might include post-transmission parallel, series, power-split etc. With accurate engine BSFC data and hydraulic pump-motor efficiency data, a performance comparison between each architecture could be made. The NLP algorithm could be generalized to solve a myriad of optimal control problems; minimum time, multiple boundary values etc. The framework is not limited to vehicle models either, it could be applied to many different optimal control problems involving dynamic systems subject to inequality/equality constraints. There are limits to the algorithm, however. Since it relies on gradient information, it is not capable of solving singular optimal control problems such as bang-bang, on-off, and discrete-input type systems. Further investigation is necessary before modifications to the algorithm can be made so it is capable of solving such problems.

References

[1] A.E. Bryson and Yu-Chi Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, 1975.

[2] Lawrence Hasdorff. *Gradient Optimization and Nonlinear Control*. Wiley, 1976.

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

[4] C Lin, H Peng, and Grizzle J.W. A stochastic control strategy for hybrid electric vehicles. In *Proceedings of the 2004 American Control Conference Boston*, Massachusetts, June 30 - July 2 2004.

[5] P. Pisu and G. Rizzoni. H-infinity control for hybrid electric vehicles. In *43rd IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, December 14-17 2004.

[6] T.M. Guerra, S. Delprat, J. Santin, M. Delhom, and E. Combes. Simulation and assessment of power control strategies for a parallel hybrid car. *Journal of Automobile Engineering*, 214(D7):705–17, 2000.

[7] A. Sciarretta, M. Back, and L. Guzzella. Optimal control of parallel hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 12(3), 2004.

[8] W. E. Wilson. *Positive-Displacement Pumps and Fluid Motors*. Sir Issac Pittman and Sons, London, 1950.

[9] D. H. Jacobson, M. M. Lele, and J. L. Speyer. New necessary conditions of optimality for control problems with state-variable inequality constraints. Technical

report, Harvard University-Cambridge, Massachusetts. Division of Engineering and Applied Physics, 1969.

[10] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

[11] D. Widder. *Advanced Calculus.* Prentice-Hall, 1961.

[12] J. Troutman. *Variational Calculus with Elementary Convexity.* Springer-Verlag, 1980.

Appendix A

Matlab^{TM} Code Listing for Gradient Descent Algorithm

Matlab^{TM} implementation of the optimal control algorithm developed in 3.4. Execute OCAlgorithm1.m with euler_system.m, euler_adjoint.m, adjoint.m, and system_test2.m in the same directory. The output of OCAlgorithm1.m are optimal state and input vectors, and a gradient report in the command window.

A.1 OCAlgorithm1.m

```
% Optimization of a parallel hydraulic hybrid vehicle
clear all

% Define Constants
Ka = 150; Va = 100; Ke = 1; Kv = 0.03; B = 0.0001; Kloss = 0.1;
t0 = 0; tf = 10; h = 0.1; Tlist = t0:h:tf; last = length(Tlist);
X = [1 0.3*Va]'; k1 = 60;
Vmax = 0.9*Va; Vmin = 0.1*Va;
umin = -1; umax = 1;

% initialize
u = zeros(length(Tlist),1); Te = ones(length(Tlist),1); V1 = 1;
mu1 = zeros(length(Tlist),1); mu2 = mu1; mu3 = mu1; mu4 = mu1;
grad = inf;
while grad >= 1e-4
    % integrate the system forward in time
    Xlist = euler_system(X,Te,u);
```

```
    X1 = Xlist(:,1); X2 = Xlist(:,2); x1tf = X1(last); x2tf = X2(last);

    % integrate the adjoint system backwards in time
    A2tf = 0;
    Alist = euler_adjoint([V1 A2tf]',u,X1,X2,mu3,mu4);
    A1 = Alist(:,1); A2 = Alist(:,2);

    % gradient
    dJdV1 = x1tf - k1;
    dJdu = (Ka*A1)./(Va-X2) - Kv*A2.*X1 + mu1 - mu2;
    dJdmu1 = u - umax;
    dJdmu2 = umin - u;
    dJdmu3 = X2 - Vmax;
    dJdmu4 = Vmin - X2;

    % update
    V1 = V1 - 0.01*dJdV1;
    u = u - 0.001*dJdu;
    mu1 = max(0,mu1+0.1*dJdmu1);
    mu2 = max(0,mu2+0.1*dJdmu2);
    mu3 = max(0,mu3+0.1*dJdmu3);
    mu4 = max(0,mu4+0.1*dJdmu4);
    Te = (Ke/2)*A1;

    % termination criteria
    grad = dJdV1^2 + (mu1'*dJdmu1)^2 + (mu2'*dJdmu2)^2 + ...
           (mu3'*dJdmu3)^2 + (mu4'*dJdmu4)^2 + dJdu'*dJdu
end
```

A.2 euler_system.m

```
function Xlist = euler_system(X,Te,u)
Xlist = X';
h = evalin('base','h');
for i = 1:length(u)-1
    dX = system_test2(X,Te(i),u(i));
    X = X + h*dX;
    Xlist = vertcat(Xlist,X');
end
```

A.3 euler_adjoint.m

```
function Alist = euler_adjoint(A,u,X1,X2,mu3,mu4)
Alist = A';
h = evalin('base','h');
for i = length(u):-1:2
    dA = adjoint(A,u(i),X1(i),X2(i),mu3(i),mu4(i));
    A = A - h*dA;
    Alist = vertcat(A',Alist);
end
```

A.4 adjoint.m

```
function dA = adjoint(A,u_t,X1_t,X2_t,mu3_t,mu4_t)

Ka = evalin('base','Ka');
Va = evalin('base','Va');
Kv = evalin('base','Kv');
B = evalin('base','B');
Kloss = evalin('base','Kloss');

dA(1,1) = 2*B*A(1)*X1_t - Kv*A(2)*u_t;
dA(2,1) = (Ka*A(1)*u_t + Ka*Kloss*A(2))/(Va-X2_t)^2 + mu3_t - mu4_t;
```

A.5 system_test2.m

```
function dX = system_test2(X,Te_t,u_t)

Ka = evalin('base','Ka');
Va = evalin('base','Va');
Ke = evalin('base','Ke');
Kv = evalin('base','Kv');
B = evalin('base','B');
Kloss = evalin('base','Kloss');

dX(1,1) = Ke*Te_t - (Ka/(Va-X(2)))*u_t - B*X(1)^2;
dX(2,1) = Kv*X(1)*u_t - (Ka*Kloss)/(Va-X(2));
```

Appendix B

Matlab^{TM} Code Listing for NLP Algorithm

Matlab^{TM} implementation of the optimal control algorithm developed in 4.3. Execute OCAlgorithm2.m with vehicle.m, indices.m, ff1.m, and dF_calc1.m in the same directory. The output of OCAlgorithm2.m are optimal state and input plots, and a gradient report in the command window.

B.1 OCAlgorithm2.m

```
1
2 % Optimization of a pre-transmission parallel hydraulic hybrid vehicle
3 clear all; clc
4
5 % Define constants
6 N = 2;
7 mveh = 50; %kg
8 Va = 0.5; %m^3
9 D = 6e-5; %m^3
10 rwh = 0.381; %m
11 mwh = 6.8; %kg
12 P0 = 2e6; %Pa = 300psi
13 Kg = P0*Va; %Nm? = 1e6
14 Ke = N^2/(rwh^2*(mveh + 4*mwh));
15
16 % time
17 t0 = 0; tf = 8; h = 0.2; Tlist = t0:h:tf;
```

```
Tlength = length(t0:h:tf);

% ICs
Vveh_mpht0 = 0;
x1t0 = Vveh_mpht0*2.23693629*N/rwh;

Vveh_mphtf = 60;
k1 =  Vveh_mphtf*2.23693629*N/rwh;

x2t0 = 0.7*Va;

% Inequality constraint params
x2min = 0.2*Va; x2max = 0.9*Va;
umin = -1; umax = 1;
Temin = 0;

% length of variable vector
xlength = 18*(Tlength - 1) + 11;
x = 10*ones(xlength,1);

% assign indices
[x1,x1ii,x2,x2ii,Te,Teii,u,uii,J,Jii,A,Aii, ...
 mu1,mu1ii,mu2,mu2ii,mu3,mu3ii,mu4,mu4ii,mu5,mu5ii ...
 z1,z1ii,z2,z2ii,z3,z3ii,z4,z4ii,z5,z5ii] = indices(Tlength,x);

% solve
grad = inf;
x(x1ii) = x1t0; x(x2ii) = x2t0;
x(uii) = 0.1;

while grad >= 1e-8
    [F,dF] = vehicle(x);
    x = x - dF\F;
    grad = F'*F;
    disp(grad);
end

% plot
x1 = x(x1ii);
x2 = x(x2ii);
u =  x(uii);
Te = x(Teii);
Vveh = (rwh/N)*x1; Vveh_mph = Vveh/2.23693629;
for i = 1:length(u)
    Thyd(i,1) = -Kg*D*u(i)/(Va-x2(i)); %#ok<AGROW>
end
```

```
64
65 Tlist2 = Tlist(1:length(Tlist)-1);
66
67 figure(1); plot(Tlist,Vveh_mph);
68 title 'Vehicle Speed'; xlabel 'Time (s)'; ylabel 'Velocity (mph)';
69
70 figure(2); plot(Tlist,x2); hold on;
71 plot(Tlist,x2min*ones(length(Tlist),1),'-.','Color','black');
72 ylim([0 0.5]); title 'Accumulator Oil Volume'; xlabel 'Time (s)';
73 ylabel 'Oil Volume (m^3)';
74
75 figure(3); plot(Tlist2,u); ylim([-1.1 1.1]);
76 title 'Displacement Command'; xlabel 'Time (s)'; ylabel 'u(t)'
77
78 figure(4); plot(Tlist2,Te,Tlist2,Thyd);
79 title 'Engine and Hydraulic Torque'; xlabel 'Time (s)';
80 ylabel 'Torque (Nm)'; ylim([0 250]);
```

B.2 vehicle.m

```
1  function [F,dF] = vehicle(x)
2  % help
3  %
4  % This function file calculates F := dL/dX and  dF := d^2L/dX^2
5  %
6  % x1(i+1) = x1(i) + h*Ke*(Te(i) - Kg*D*u(i)/(Va-x2(i)));
7  % x2(i+1) = x2(i) + h*D*x1(i)*u(i);
8  % J(i+1)  = J(i)  + h*(Te(i)^2 + (-Kg*D*u(i))^2/(Va-x2(i))^2);
9  %
10 % x1(2) = x1(1) + h*Ke*Te(1) - h*Ke*Kg*D*u(1)/(Va-x2(1))
11 % x2(2) = x2(1) + h*D*x1(1)*u(1)
12 % J(2)  = J(1)  + h*Te(1)^2 + h*(-Kg*D)^2*u(1)^2/(Va-x2(i))^2
13 %
14 % x1(2) - x1(1) - h*Ke*Te(1) + h*Ke*Kg*D*u(1)/(Va-x2(1))
15 % x2(2) - x2(1) - h*D*x1(1)*u(1)
16 % J(2)  - J(1)  - h*Te(1)^2 - h*(-Kg*D)^2*u(1)^2/(Va-x2(1))^2
17 %
18 % x1(2) - x1(1) - K1*Te(1) + K2*u(1)/(Va-x2(1))
19 % x2(2) - x2(1) - K3*x1(1)*u(1);
20 % J(2)  - J(1)  - h*Te(1)^2 - K4*u(1)^2/(Va-x2(1))^2
21 %
22 % x1(2) - x1(1) - h*(Ke*Te(1) - Ke*Kg*D*u(1)/(Va-x2(1))) = 0;
23 % x1(2) - x1(1) - h*Ke*(Te(1) - Kg*D*u(1)/(Va-x2(1))) = 0;
```

```
% x1(2) - x1(1) - h*Ke*(Te(1) + Thyd(1)) = 0;
%
%   Thyd(1) = -Kg*D*u(1)/(Va-x2(1))
%   Thyd(1)^2 = (-Kg*D*u(1))^2/(Va-x2(1))^2
%
% min | Te^2 + Thyd^2
% ------------------------------------------------------------
% x1(2) - x1(1) - K1*Te(1) + K2*u(1)/(Va-x2(1))
% x2(2) - x2(1) - K3*x1(1)*u(1);
% J(2)  - J(1)  - h*Te(1)^2 - K4*u(1)^2/(Va-x2(1))^2

Tlength = evalin('base','Tlength');
x1t0 = evalin('base','x1t0'); k1 = evalin('base','k1');
x2t0 = evalin('base','x2t0');
h = evalin('base','h');
Ke = evalin('base','Ke'); Kg = evalin('base','Kg');
D = evalin('base','D'); Va = evalin('base','Va');
umin = evalin('base','umin'); umax = evalin('base','umax');
x2min = evalin('base','x2min'); x2max = evalin('base','x2max');
Temin = evalin('base','Temin');

K1 = h*Ke; K2 = h*Ke*Kg*D; K3 = h*D;
K4 = h*(-Kg*D)^2;

[x1,x1ii,x2,x2ii,Te,Teii,u,uii,J,Jii,A,Aii, ...
 mu1,mu1ii,mu2,mu2ii,mu3,mu3ii,mu4,mu4ii,mu5,mu5ii ...
 z1,z1ii,z2,z2ii,z3,z3ii,z4,z4ii,z5,z5ii] = indices(Tlength,x);

number_of_steps = Tlength-1; kend = 3*number_of_steps;

F = []; dF = zeros(length(x));
j = 1; Fii = 0;
for k = 0:3:kend

    if k == 0 %t0
%  L = A(1)[x1(1) - x1t0] +
%      A(2)[x2(1) - x2t0] +
%      A(3)[J(1) - 0] +
%      A(4)[x1(2) - x1(1) - K1*Te(1) + K2*u(1)/(Va-x2(1))] +
%      A(5)[x2(2) - x2(1) - K3*x1(1)*u(1)] +
%      A(6)[J(2) - J(1) - h*Te(1)^2 - K4*u(1)^2/(Va-x2(1))^2] +
%      mu1(1)[umin - u(1) + z1(1)^2] +
%      mu2(1)[u(1) - umax + z2(1)^2] +
%      mu3(1)[x2min - x2(1) + z3(1)^2] +
%      mu4(1)[x2(1) - x2max + z4(1)^2] +
%      mu5(1)[Temin - Te(1) + z5(1)^2] +
```

```
70
71 ff= [A(k+1) - A(k+4) - A(k+5)*K3*u(j)                                  %dx1(1)
72
73      A(k+2) + A(k+4)*K2*u(j)/(Va-x2(j))^2 - A(k+5) ...                 %dx2(1)
74      - A(k+6)*2*K4*u(j)^2/(Va-x2(j))^3 - mu3(j) + mu4(j)
75
76     -A(k+4)*K1 - A(k+6)*h*2*Te(j) - mu5(j)                             %dTe(1)
77
78      A(k+4)*K2/(Va-x2(j)) - A(k+5)*K3*x1(j) ...                         %du(1)
79      - A(k+6)*K4*2*u(j)/(Va-x2(j))^2 - mu1(j) + mu2(j)
80
81      A(k+3) - A(k+6)                                                    %dJ(1)
82
83      umin - u(j) + z1(j)^2                                             %dmu1(1)
84      u(j) - umax + z2(j)^2                                             %dmu2(1)
85      x2min - x2(j) + z3(j)^2                                           %dmu3(1)
86      x2(j) - x2max + z4(j)^2                                           %dmu4(1)
87      Temin - Te(j) + z5(j)^2                                           %dmu5(1)
88      2*mu1(j)*z1(j)                                                     %dz1(1)
89      2*mu2(j)*z2(j)                                                     %dz2(1)
90      2*mu3(j)*z3(j)                                                     %dz3(1)
91      2*mu4(j)*z4(j)                                                     %dz4(1)
92      2*mu5(j)*z5(j)                                                     %dz5(1)
93
94      x1(j) - x1t0                                                       %dA(1)
95      x2(j) - x2t0                                                       %dA(2)
96      J(j) - 0];                                                         %dA(3)
97
98       dF(Fii+1,Aii(k+1)) = 1;
99       dF(Fii+1,Aii(k+4)) = -1;
100      dF(Fii+1,Aii(k+5)) = -K3*u(j);
101      dF(Fii+1,uii(j)) = -A(k+5)*K3;
102
103      dF(Fii+2,Aii(k+2)) = 1;
104      dF(Fii+2,Aii(k+4)) = K2*u(j)/(Va-x2(j))^2;
105      dF(Fii+2,Aii(k+5)) = -1;
106      dF(Fii+2,Aii(k+6)) = -2*K4*u(j)^2/(Va-x2(j))^3;
107      dF(Fii+2,uii(j))   = A(k+4)*K2/(Va-x2(j))^2 ...
108                           - A(k+6)*2*K4*2*u(j)/(Va-x2(j))^3;
109      dF(Fii+2,x2ii(j))  = dF_calc1(2,'xx2',x2(j));
110      dF(Fii+2,mu3ii(j)) = -1;
111      dF(Fii+2,mu4ii(j)) = 1;
112
113      dF(Fii+3,Aii(k+4)) = -K1;
114      dF(Fii+3,Aii(k+6)) = -h*2*Te(j);
115      dF(Fii+3,Teii(j))  = -A(k+6)*h*2;
```

```
116       dF(Fii+3,mu5ii(j)) = -1;
117
118       dF(Fii+4,Aii(k+4)) = K2/(Va-x2(j));
119       dF(Fii+4,x2ii(j))  = dF_calc1(4,'xx2',x2(j));
120       dF(Fii+4,Aii(k+5)) = -K3*x1(j);
121       dF(Fii+4,x1ii(j))  = -A(k+5)*K3;
122       dF(Fii+4,Aii(k+6)) = -K4*2*u(j)/(Va-x2(j))^2;
123       dF(Fii+4,uii(j))   = -A(k+6)*K4*2/(Va-x2(j))^2;
124       dF(Fii+4,mu1ii(j)) = -1;
125       dF(Fii+4,mu2ii(j)) = 1;
126
127       dF(Fii+5,Aii(k+3)) = 1;
128       dF(Fii+5,Aii(k+6)) = -1;
129
130       dF(Fii+6,uii(j)) = -1; dF(Fii+6,z1ii(j)) = 2*z1(j);
131       dF(Fii+7,uii(j)) = 1;  dF(Fii+7,z2ii(j)) = 2*z2(j);
132       dF(Fii+8,x2ii(j)) = -1; dF(Fii+8,z3ii(j)) = 2*z3(j);
133       dF(Fii+9,x2ii(j)) = 1; dF(Fii+9,z4ii(j)) = 2*z4(j);
134       dF(Fii+10,Teii(j)) = -1; dF(Fii+10,z5ii(j)) = 2*z5(j);
135
136       dF(Fii+11,mu1ii(j)) = 2*z1(j); dF(Fii+11,z1ii(j)) = 2*mu1(j);
137       dF(Fii+12,mu2ii(j)) = 2*z2(j); dF(Fii+12,z2ii(j)) = 2*mu2(j);
138       dF(Fii+13,mu3ii(j)) = 2*z3(j); dF(Fii+13,z3ii(j)) = 2*mu3(j);
139       dF(Fii+14,mu4ii(j)) = 2*z4(j); dF(Fii+14,z4ii(j)) = 2*mu4(j);
140       dF(Fii+15,mu5ii(j)) = 2*z5(j); dF(Fii+15,z5ii(j)) = 2*mu5(j);
141
142       dF(Fii+16,x1ii(j)) = 1;
143       dF(Fii+17,x2ii(j)) = 1;
144       dF(Fii+18,Jii(j)) = 1;
145     end
146
147     if (3 <= k) && (k < kend) %t1
148 %      A(4)[x1(2) - x1(1) - K1*Te(1) + K2*u(1)/(Va-x2(1))] +
149 %      A(5)[x2(2) - x2(1) - K3*x1(1)*u(1)] +
150 %      A(6)[J(2) - J(1) - h*Te(1)^2 - K4*u(1)^2/(Va-x2(1))^2] +
151
152 %      A(7)[x1(3) - x1(2) - K1*Te(2) + K2*u(2)/(Va-x2(2))] +
153 %      A(8)[x2(3) - x2(2) - K3*x1(2)*u(2)] +
154 %      A(9)[J(3) - J(2) - h*Te(2)^2 - K4*u(2)^2/(Va-x2(2))^2] +
155 %      mu1(2)[umin - u(2) + z1(2)^2] +
156 %      mu2(2)[u(2) - umax + z2(2)^2] +
157 %      mu3(2)[x2min - x2(2) + z3(2)^2] +
158 %      mu4(2)[x2(2) - x2max + z4(2)^2] +
159 %      mu5(2)[Temin - Te(2) + z5(2)^2] +
160
161 ff= [A(k+1) - A(k+4) - A(k+5)*K3*u(j)                              %dx1(2)
```

```
162
163      A(k+2) + A(k+4)*K2*u(j)/(Va-x2(j))^2 - A(k+5) ...              %dx2(2)
164      - A(k+6)*2*K4*u(j)^2/(Va-x2(j))^3 - mu3(j) + mu4(j)
165
166     -A(k+4)*K1 - A(k+6)*h*2*Te(j) - mu5(j)                          %dTe(2)
167
168      A(k+4)*K2/(Va-x2(j)) - A(k+5)*K3*x1(j) ...                      %du(2)
169      - A(k+6)*K4*2*u(j)/(Va-x2(j))^2 - mu1(j) + mu2(j)
170
171      A(k+3) - A(k+6)                                                 %dJ(2)
172
173      umin - u(j) + z1(j)^2                                          %dmu1(2)
174      u(j) - umax + z2(j)^2                                          %dmu2(2)
175      x2min - x2(j) + z3(j)^2                                        %dmu3(2)
176      x2(j) - x2max + z4(j)^2                                        %dmu4(2)
177      Temin - Te(j) + z5(j)^2                                        %dmu5(2)
178      2*mu1(j)*z1(j)                                                  %dz1(2)
179      2*mu2(j)*z2(j)                                                  %dz2(2)
180      2*mu3(j)*z3(j)                                                  %dz3(2)
181      2*mu4(j)*z4(j)                                                  %dz4(2)
182      2*mu5(j)*z5(j)                                                  %dz5(2)
183
184      x1(j) - x1(j-1) - K1*Te(j-1) + K2*u(j-1)/(Va-x2(j-1))           %dA(4)
185      x2(j) - x2(j-1) - K3*x1(j-1)*u(j-1)                             %dA(5)
186      J(j) - J(j-1) - h*Te(j-1)^2 - K4*u(j-1)^2/(Va-x2(j-1))^2];      %dA(6)
187
188       dF(Fii+1,Aii(k+1)) = 1;
189       dF(Fii+1,Aii(k+4)) = -1;
190       dF(Fii+1,Aii(k+5)) = -K3*u(j);
191       dF(Fii+1,uii(j)) = -A(k+5)*K3;
192
193       dF(Fii+2,Aii(k+2)) = 1;
194       dF(Fii+2,Aii(k+4)) = K2*u(j)/(Va-x2(j))^2;
195       dF(Fii+2,Aii(k+5)) = -1;
196       dF(Fii+2,Aii(k+6)) = -2*K4*u(j)^2/(Va-x2(j))^3;
197       dF(Fii+2,uii(j))   = A(k+4)*K2/(Va-x2(j))^2 ...
198                            -A(k+6)*2*K4*2*u(j)/(Va-x2(j))^3;
199                            %dF_calc2(2,'xx2',x2(j));
200       dF(Fii+2,x2ii(j))  = 2*A(k+4)*K2*u(j)/(Va-x2(j))^2 ...
201                            - 3*A(k+6)*2*K4*u(j)^2/(Va-x2(j))^4;
202       dF(Fii+2,mu3ii(j)) = -1;
203       dF(Fii+2,mu4ii(j)) = 1;
204
205       dF(Fii+3,Aii(k+4)) = -K1;
206       dF(Fii+3,Aii(k+6)) = -h*2*Te(j);
207       dF(Fii+3,Teii(j))  = -A(k+6)*h*2;
```

```
208       dF(Fii+3,mu5ii(j)) = -1;
209
210       dF(Fii+4,Aii(k+4)) = K2/(Va-x2(j));
211                            %dF_calc2(4,'xx2',x2(j));
212       dF(Fii+4,x2ii(j))  = A(k+4)*K2/(Va-x2(j))^2 ...
213                            - 4*A(k+6)*K4*u(j)/(Va-x2(j))^3;
214       dF(Fii+4,Aii(k+5)) = -K3*x1(j);
215       dF(Fii+4,x1ii(j))  = -A(k+5)*K3;
216       dF(Fii+4,Aii(k+6)) = -K4*2*u(j)/(Va-x2(j))^2;
217       dF(Fii+4,uii(j))   = -A(k+6)*K4*2/(Va-x2(j))^2;
218       dF(Fii+4,mu1ii(j)) = -1;
219       dF(Fii+4,mu2ii(j)) = 1;
220
221       dF(Fii+5,Aii(k+3)) = 1;
222       dF(Fii+5,Aii(k+6)) = -1;
223
224       dF(Fii+6,uii(j)) = -1;   dF(Fii+6,z1ii(j)) = 2*z1(j);
225       dF(Fii+7,uii(j)) = 1;    dF(Fii+7,z2ii(j)) = 2*z2(j);
226       dF(Fii+8,x2ii(j)) = -1;  dF(Fii+8,z3ii(j)) = 2*z3(j);
227       dF(Fii+9,x2ii(j)) = 1;   dF(Fii+9,z4ii(j)) = 2*z4(j);
228       dF(Fii+10,Teii(j)) = -1; dF(Fii+10,z5ii(j)) = 2*z5(j);
229
230       dF(Fii+11,mu1ii(j)) = 2*z1(j); dF(Fii+11,z1ii(j)) = 2*mu1(j);
231       dF(Fii+12,mu2ii(j)) = 2*z2(j); dF(Fii+12,z2ii(j)) = 2*mu2(j);
232       dF(Fii+13,mu3ii(j)) = 2*z3(j); dF(Fii+13,z3ii(j)) = 2*mu3(j);
233       dF(Fii+14,mu4ii(j)) = 2*z4(j); dF(Fii+14,z4ii(j)) = 2*mu4(j);
234       dF(Fii+15,mu5ii(j)) = 2*z5(j); dF(Fii+15,z5ii(j)) = 2*mu5(j);
235
236       dF(Fii+16,x1ii(j)) = 1;
237       dF(Fii+16,x1ii(j-1)) = -1;
238       dF(Fii+16,Teii(j-1)) = -K1;
239       dF(Fii+16,uii(j-1)) = K2/(Va-x2(j-1));
240       dF(Fii+16,x2ii(j-1)) = K2*u(j-1)/(Va-x2(j-1))^2;
241
242       dF(Fii+17,x2ii(j)) = 1;
243       dF(Fii+17,x2ii(j-1)) = -1;
244       dF(Fii+17,x1ii(j-1)) = -K3*u(j-1);
245       dF(Fii+17,uii(j-1)) = -K3*x1(j-1);
246
247       dF(Fii+18,Jii(j)) = 1;
248       dF(Fii+18,Jii(j-1)) = -1;
249       dF(Fii+18,Teii(j-1)) = -2*h*Te(j-1);
250       dF(Fii+18,uii(j-1)) = -K4*2*u(j-1)/(Va-x2(j-1))^2;
251       dF(Fii+18,x2ii(j-1)) = -2*K4*u(j-1)^2/(Va-x2(j-1))^3;
252     end
253
```

```
    if k == kend %t2
%      A(7)[x1(3) - x1(2) - K1*Te(2) + K2*u(2)/(Va-x2(2))] +
%      A(8)[x2(3) - x2(2) - K3*x1(2)*u(2)] +
%      A(9)[J(3) - J(2) - h*Te(2)^2 - K4*u(2)^2/(Va-x2(2))^2] +
%      A(10)[x1(3) - k1] +
%      J(3)
%      mu3(3)[x2min - x2(3) + z3(3)^2] +
%      mu4(3)[x2(3) - x2max + z4(3)^2] +

ff= [A(k+1) + A(k+4)                                                %dx1(3)
     A(k+2) - mu3(j) + mu4(j)                                       %dx2(3)
     1 + A(k+3)                                                      %dJ(3)
     x2min - x2(j) + z3(j)^2                                       %dmu3(3)
     x2(j) - x2max + z4(j)^2                                       %dmu4(3)
     2*mu3(j)*z3(j)                                                 %dz3(2)
     2*mu4(j)*z4(j)                                                 %dz4(2)
     x1(j) - x1(j-1) - K1*Te(j-1) + K2*u(j-1)/(Va-x2(j-1))           %dA(7)
     x2(j) - x2(j-1) - K3*x1(j-1)*u(j-1)                             %dA(8)
     J(j) - J(j-1) - h*Te(j-1)^2 - K4*u(j-1)^2/(Va-x2(j-1))^2        %dA(9)
     x1(j) - k1];                                                   %dA(10)

      dF(Fii+1,Aii(k+1)) = 1;
      dF(Fii+1,Aii(k+4)) = 1;

      dF(Fii+2,Aii(k+2)) = 1;
      dF(Fii+2,mu3ii(j)) = -1;
      dF(Fii+2,mu4ii(j)) = 1;

      dF(Fii+3,Aii(k+3)) = 1;

      dF(Fii+4,x2ii(j)) = -1; dF(Fii+4,z3ii(j)) = 2*z3(j);
      dF(Fii+5,x2ii(j)) = 1;  dF(Fii+5,z4ii(j)) = 2*z4(j);
      dF(Fii+6,mu3ii(j)) = 2*z3(j); dF(Fii+6,z3ii(j)) = 2*mu3(j);
      dF(Fii+7,mu4ii(j)) = 2*z4(j); dF(Fii+7,z4ii(j)) = 2*mu4(j);

      dF(Fii+8,x1ii(j)) = 1;
      dF(Fii+8,x1ii(j-1)) = -1;
      dF(Fii+8,Teii(j-1)) = -K1;
      dF(Fii+8,uii(j-1)) = K2/(Va-x2(j-1));
      dF(Fii+8,x2ii(j-1)) = K2*u(j-1)/(Va-x2(j-1))^2;

      dF(Fii+9,x2ii(j)) = 1;
      dF(Fii+9,x2ii(j-1)) = -1;
      dF(Fii+9,x1ii(j-1)) = -K3*u(j-1);
      dF(Fii+9,uii(j-1)) = -K3*x1(j-1);

```

```
      dF(Fii+10,Jii(j)) = 1;
      dF(Fii+10,Jii(j-1)) = -1;
      dF(Fii+10,Teii(j-1)) = -2*h*Te(j-1);
      dF(Fii+10,uii(j-1)) = -K4*2*u(j-1)/(Va-x2(j-1))^2;
      dF(Fii+10,x2ii(j-1)) = -2*K4*u(j-1)^2/(Va-x2(j-1))^3;

      dF(Fii+11,x1ii(j)) = 1;
    end

    F = vertcat(F,ff);
    j = j + 1;
    Fii = Fii + 18;
end
```

B.3 indices.m

```
function [x1,x1ii,x2,x2ii,Te,Teii,u,uii,J,Jii,A,Aii, ...
          mu1,mu1ii,mu2,mu2ii,mu3,mu3ii,mu4,mu4ii,mu5,mu5ii ...
          z1,z1ii,z2,z2ii,z3,z3ii,z4,z4ii,z5,z5ii] = indices(Tlength,x)

% ========================================================================
x1start = 1; x1end = Tlength;
x1 = x(x1start:x1end); x1ii = x1start:x1end;

x2start = x1end + 1; x2end = x2start + Tlength - 1;
x2 = x(x2start:x2end); x2ii = x2start:x2end;

Testart = x2end + 1; Teend = Testart + Tlength - 2;
Te = x(Testart:Teend); Teii = Testart:Teend;

ustart = Teend + 1; uend = ustart + Tlength - 2;
u = x(ustart:uend); uii = ustart:uend;

Jstart = uend + 1; Jend = Jstart + Tlength - 1;
J = x(Jstart:Jend); Jii = Jstart:Jend;
% ========================================================================

% ========================================================================
mu1start = Jend + 1; mu1end = mu1start + Tlength - 2;
mu1 = x(mu1start:mu1end); mu1ii = mu1start:mu1end;

z1start = mu1end + 1; z1end = z1start + Tlength - 2;
z1 = x(z1start:z1end); z1ii = z1start:z1end;
```

```
28
29 mu2start = z1end + 1; mu2end = mu2start + Tlength - 2;
30 mu2 = x(mu2start:mu2end); mu2ii = mu2start:mu2end;
31
32 z2start = mu2end + 1; z2end = z2start + Tlength - 2;
33 z2 = x(z2start:z2end); z2ii = z2start:z2end;
34
35
36 mu3start = z2end + 1; mu3end = mu3start + Tlength - 1;
37 mu3 = x(mu3start:mu3end); mu3ii = mu3start:mu3end;
38
39 z3start = mu3end + 1; z3end = z3start + Tlength - 1;
40 z3 = x(z3start:z3end); z3ii = z3start:z3end;
41
42 mu4start = z3end + 1; mu4end = mu4start + Tlength - 1;
43 mu4 = x(mu4start:mu4end); mu4ii = mu4start:mu4end;
44
45 z4start = mu4end + 1; z4end = z4start + Tlength - 1;
46 z4 = x(z4start:z4end); z4ii = z4start:z4end;
47
48
49 mu5start = z4end + 1; mu5end = mu5start + Tlength - 2;
50 mu5 = x(mu5start:mu5end); mu5ii = mu5start:mu5end;
51
52 z5start = mu5end + 1; z5end = z5start + Tlength - 2;
53 z5 = x(z5start:z5end); z5ii = z5start:z5end;
54
55 Astart = z5end + 1; Aend = length(x);
56 A = x(Astart:Aend); Aii = Astart:Aend;
```

B.4 ff1.m

```
1 function ffrow = ff1(rownumber)
2
3 % This function file assists with numerical gradient calculation
4
5 h = evalin('base','h');
6 Ke = evalin('base','Ke'); Kg = evalin('base','Kg');
7 D = evalin('base','D'); Va = evalin('base','Va');
8
9 K1 = h*Ke; K2 = h*Ke*Kg*D; K3 = h*D;
10 K4 = h*(-Ke*Kg*D)^2;
11
```

```
if rownumber == 2 %dx2
    A2 = evalin('caller','A2'); A4 = evalin('caller','A4');
    A5 = evalin('caller','A5'); A6 = evalin('caller','A6');
    mmu3 = evalin('caller','mmu3'); mmu4 = evalin('caller','mmu4');
    uu = evalin('caller','uu'); xx2 = evalin('caller','xx2');
    ffrow =  A2 + A4*K2*uu/(Va-xx2)^2 - A5 ...
             - A6*2*K4*uu^2/(Va-xx2)^3 - mmu3 + mmu4;
end
if rownumber == 4 %du
    A4 = evalin('caller','A4'); A5 = evalin('caller','A5');
    A6 = evalin('caller','A6'); xx1 = evalin('caller','xx1');
    uu = evalin('caller','uu'); xx2 = evalin('caller','xx2');
    mmu1 = evalin('caller','mmu1'); mmu2 = evalin('caller','mmu2');
    ffrow = A4*K2/(Va-xx2) - A5*K3*xx1 - A6*K4*2*uu/(Va-xx2)^2 ...
            - mmu1 + mmu2;
end
```

B.5 dF_calc1.m

```
function dFout = dF_calc1(rownumber,varname,varvalue)          %#ok<INUSD>

% This function file assists with numerical gradient calculation

% dF(Fii+2,Aii(k+5)) = dF_calc1(2,'A5',A(k+5));
del = 1e-10;
A1 = evalin('caller','A(k+1)'); A2 = evalin('caller','A(k+2)'); %#ok<NASGU>
A3 = evalin('caller','A(k+3)'); A4 = evalin('caller','A(k+4)'); %#ok<NASGU>
A5 = evalin('caller','A(k+5)'); A6 = evalin('caller','A(k+6)'); %#ok<NASGU>
xx1 = evalin('caller','x1(j)'); xx2 = evalin('caller','x2(j)'); %#ok<NASGU>
TTe = evalin('caller','Te(j)'); uu = evalin('caller','u(j)');   %#ok<NASGU>
JJ = evalin('caller','J(j)');                                   %#ok<NASGU>
mmu1= evalin('caller','mu1(j)'); zz1= evalin('caller','z1(j)'); %#ok<NASGU>
mmu2= evalin('caller','mu2(j)'); zz2= evalin('caller','z2(j)'); %#ok<NASGU>
mmu3= evalin('caller','mu3(j)'); zz3= evalin('caller','z3(j)'); %#ok<NASGU>
mmu4= evalin('caller','mu4(j)'); zz4= evalin('caller','z4(j)'); %#ok<NASGU>

% A5 = A(k+5)+del; fdel = ff1(2);
% A5 = A(k+5);     f = ff1(2);
% dF(Fii+2,Aii(k+5)) = (fdel-f)/del;

qq = genvarname(varname);                                       %#ok<NASGU>
eval([qq '= varvalue+del;']); fdel = ff1(rownumber);            %#ok<NASGU>
eval([qq '= varvalue;']);     f = ff1(rownumber);               %#ok<NASGU>
```

```
25 dFout = (fdel-f)/del;
```

Appendix C

List of Acronyms and Notations

C.1 Acronyms

Acronym	Meaning
HHV	Hydraulic Hybrid Vehicle
HEV	Hybrid Electric Vehicle
COV	Calculus of Variations
NLP	Non Linear Program
KKT	Karush Kuhn Tucker
ECMS	Equivalent Consumption Minimization Strategy
ICE	Internal Combustion Engine

Table C.1: Acronyms

C.2 Notations

Notation	Meaning	Units
D	Pump-motor total displacement	m^3
J_{ICE}	Engine inertia	kgm^2
$J_{p/m}$	Pump-motor unit inertia	kgm^2
J_{wheel}	Wheel inertia	kgm^2
m_{veh}	Vehicle mass	kg
$\omega_{in}(t)$	Shaft speed (pre-transmission)	$\frac{1}{s}$
$\omega_{out}(t)$	Shaft speed (post-transmission)	$\frac{1}{s}$
$v_{veh}(t)$	Vehicle speed	$\frac{m}{s}$
r_{wheel}	Wheel radius	m
V_{gas}	Accumulator gas volume	m^3
$V_{accumulator}$	Total accumulator volume	m^3
$V_{oil}(t)$	Accumulator oil volume	m^3
P_{gas}	Accumulator gas pressure	Pa
k_{gas}	Isothermal accumulator gas constant	kgm
$\tau_{hydraulics}(t)$	Applied hydraulic torque	Nm
$\tau_{ICE}(t)$	Applied engine torque	Nm
$\tau_{aero}(t)$	Applied aerodynamic torque	Nm
$P_{atmosphere}$	Atmospheric pressure	Pa
K_e	Engine torque constant	$\frac{1}{kgm^2}$
K_a	Accumulator constant	$\frac{1}{s^2}$
K_v	Oil volume constant	m^3
K_L	Volumetric loss constant	Nm
B	Aerodynamic drag constant	$\frac{1}{kg}$
$u(t)$	Pump-motor displacement command	-
V_{min}	Minimum accumulator oil volume	m^3
V_{max}	Maximum accumulator oil volume	m^3
u_{min}	Minimum displacement command	-
u_{max}	Maximum displacement command	-

Table C.2: Notations