The Development of a Power Management Strategy for a Hydraulic Hybrid Passenger Vehicle

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

Jonathan James Meyer

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Kim A. Stelson

July, 2014

Acknowledgements

During my time in graduate school, many people have helped me along on the journey in their own unique way, and while it is impossible to express my gratitude for all they have done, this is an attempt to at least recognize some of the influential people I have met. First and foremost, thank you to my advisor, Dr. Kim Stelson, for his guidance over the years. He has taught me how to be a researcher and to never stop asking questions and exploring, even when an answer may be found, and to continually learn. He not only helped with my research skills, but also my leadership and communication skills throughout the years. I also would like to thank my committee members - Dr. Demoz Gebre-Egziabher, Dr. Perry Li, Dr. Rajesh Rajamani, and Dr. Yiyuan Zhao for their input to improve this research.

My family has always been there to support me, and without them I would not be who I am today. My parents have always believed in me and taught me to follow my dreams, and if I put my mind to it, I can accomplish anything. My sisters have always been there when I needed a joke, and they didn't let the highs get too high or the lows too low. They were always there for me and a phone call away whenever I needed someone to talk to, and I couldn't ask for better role models.

During my time with the Center for Compact and Efficient Fluid Power (CCEFP), I have met a number of graduate students I am lucky to call friends today, many who started at the same time I did. Venkat Durbha, David Hafvenstein, Kathy (Braun) Houle, Mike Rannow, Caleb Sancken, Haink Tu, and Meng (Rachel) Wang were people I could count on if I had a technical question I couldn't quite figure out or if I just needed a laugh. The numerous happy hours and socials we had will forever be great memories of my graduate school career. Also, I want to thank all the members past and present who worked on the hydraulic hybrid passenger vehicle - Kai Loon Cheong, Zhekang

Du, Robert Ertel, David Grandall, Andrew Harm, Henry Kohring, Justin Lapp, and Stephen Sedler. I will never forget taking the vehicle outside and bringing it back in with all the methods we rigged up. While it was sometimes frustrating, it also provided a lot of memories I can now look back on and laugh, even though at the time it wasn't funny.

On my project I also had the fortune to collaborate with Dr. Andrew Alleyne and Tim Deppen at the University of Illinois - Urbana/Champaign. They were another support system I could count on when I had a question on how to proceed. Tim and I shared many project updates, paper deadlines, and annual report summaries and has been a great person to work with throughout this project. Tim provided numerous suggestions on how to make this project better and is a person I am glad to call a friend today.

The staff of the CCEFP has always been there for me whenever I needed them. Alyssa Burger started with the Center and saw me through my whole graduate school journey. She was always there when I needed to let out frustration and would let me spend countless hours in her office as a listening ear. Even though we didn't always agree, she was a valuable asset and taught me to see things in a different light. I am glad we became friends over the years. Mike Gust was always there when I needed someone to relieve stress and was always open to socializing by either getting a drink after work or hitting the golf course for a round. Your friendship has meant a lot to me. Don Haney would let me come into his office almost everyday when I needed a break from my work just to see how his day was going. I have enjoyed getting to know you and becoming a friend. Lisa Wissbaum would always make sure I got reimbursed after a trip and worked tirelessly behind the scenes to make sure the Center kept operating smoothly. Brad Bohlmann was another person I could count on whenever I needed a drink and get my mind off research for awhile. Though not a member of the CCEFP, John Gardner (aka The Gradfather) was always there to help me when I had a question, take me to lunch, provide a joke when I needed a laugh, and make sure I kept making progress towards my degree. He was an invaluable resource during my entire journey, always answering numerous questions I had along the way with a smile. I am glad to have gotten to know you over the years and call a friend today. All of you helped enhance my graduate school career for the better.

Finally, I was fortunate to meet people outside of the mechanical engineering department who kept me sane and shaped me into the person I am today. Thank you to Tadd Wilson with the University of Minnesota Athletic Department for giving me a chance at a part-time job with the video production at Gopher athletic events with no prior experience in that field. Even though I was getting paid, it never felt like a job and learned so much. Also, thank you to Sarah Myers, for keeping me in good spirits, providing a good laugh when I needed it, and always asking how I was doing. You have both become friends and have provided many good memories, even when the Gophers didn't do the best. But I would have never had the opportunity without Deborah Diamond Edwards. She introduced me to numerous terrific people I would have otherwise never met. Whenever I was ready to give up, she was there to support and encourage me. Thank you for the countless meals, yummy homemade treats, and always letting me complain, without ever complaining back. You have been a wonderful person to get to know, and an even greater friend through the years. Thanks for it all, Deb!

This work was performed within the Engineering Research Center for Compact and Efficient Fluid Power, supported by the National Science Foundation under Grant No. EEC-0540834.

Abstract

The amount of energy being consumed is increasing each year, with the highest sector being the transportation industry. Within the transportation sector, the highest area of oil consumption is in the small and lightweight vehicle category. With increasing oil prices and decreasing supply, methods of reducing oil consumption have been studied. One is by developing a hybrid vehicle, which combines the internal combustion engine with an additional power source. For lightweight vehicles, electric hybrid vehicles have been thoroughly studied. While hydraulic hybrids have been studied for larger applications such as delivery trucks and buses, little research has been done in the area of small, lightweight vehicles. Hydraulics have a higher power density than electronics, so hydraulic hybrids can get better performance than electric hybrids while reducing fuel consumption.

In this research, a series and power-split architecture is studied for a passenger vehicle. Because of the additional hydraulic power source along with energy storage, the optimal way to control these vehicles is not known. Therefore, an energy management strategy must be developed to determine the optimal strategy for splitting the power between the engine and the hydraulics.

Three different methods are used to develop the energy management strategy - a rule-based strategy based on dynamic programming results, stochastic dynamic programming, and model predictive control. An experimental hardware-in-the-loop setup is used to replicate a series hybrid in which the different energy management strategies are tried. Through simulation and experimentation, it was found that not one strategy works best in all scenarios, and variables such as knowledge of duty cycle and energy storage must be taken into account when developing the strategy.

An input-coupled power-split hybrid was also studied, which combines the mechanical efficiency of the parallel hybrid with the engine management of a series hybrid. Through a series of simulations, a strategy that declutched the engine from the drivetrain while the vehicle is stopped gave a significant reduction in fuel consumption. Another advantage of the power-split architecture is the ability to operate the vehicle in different modes by declutching the engine and removing hydraulic units by the use of valves. By using this strategy, the fuel economy can be almost doubled over a baseline strategy which operates only in power-split mode. Finally, the size of the accumulator can have an effect on the fuel consumption, with a smaller accumulator leading to less fuel consumed; however, if the accumulator is too small, the performance starts to degrade with a downsized engine.

The results of this research can be used to develop a toolbox that can be used for developing energy management strategies by having the user enter a model, objective function, and duty cycle for a system. By using other information, such as knowledge of duty cycle, the toolbox can determine the best method of developing the control strategy, reducing the amount of time and resources for developing an optimal control strategy.

Contents

List of Tables

List of Figures

List of Abbreviations

AEVPS	Augmented Earthmoving Vehicle Powertrain Simulator
CCEFP	Center for Compact and Efficient Fluid Power
DP	Dynamic Programming
EMS	Energy Management Strategy
HHPV	Hydraulic Hybrid Passenger Vehicle
MPC	Model Predictive Control
SDP	Stochastic Dynamic Programming
UDDS	Urban Dynamometer Driving Schedule

Nomenclature

A_f	Frontal Area
C	Counts for Transition Probabilities
C_d	Drag Coefficient
C_v	Valve Flow Coefficient
D	Maximum Displacement
F	Force
I	Inertia
J	Objective Function
L	Fuel Consumption in One Time Step
N	Number of Gear Teeth
N_a	Number of Discretized Accelerations
K	Total Number of Steps
K_{Eng}	Engine Friction Loss Factor
K_P	Pump Flow Gain
P	Pressure
Pr	Probability
Q	Volumetric Flow Rate
R_{diff}	Gear Ratio of Differential
R_{pd}	Gear Ratio of Planetray Gearset
R_S	Gear Ratio Between Pump/Motor S and Planetary Gearset
R_T	Gear Ratio Between Engine and Pump/Motor T
S	Total Number of States
T	Torque
U	Space for Control Decisions
V	Volume
Va	Value Function
a	Acceleration
b	Damping Coefficient
c	Transition Cost
$\bar{c}$	Average Cost
d	Diameter

f_o	Rolling Resistance Coefficient
f_s	Rolling Resistance Coefficient
g	Gravitational Constant
k	Time Index
k_f	Fuel Consumption Coefficient
m	Mass
n	Index of State at Next Time Step
o	Index of State at Current Time Step
p	Power
r	Radius
u	Control Input
v	Vehicle Velocity
x	State Variable
y	Summation Index for MPC
Δt	Time Step
β	Bulk Modulus
γ	Specific Heat Ratio
ϵ	Engine Speed Weighting Factor
ζ	Weighting Factor for MPC
η	Efficiency
θ	Road Grade
λ	Discount Factor
μ	Control Policy
ρ	Air Density
τ	Engine Throttle Command
ϕ	Fraction of Maximum Displacement
ψ	Leakage Coefficient
ω	Rotational Speed

Subscripts

$Accel$	Acceleration
$Drag$	Drag Force
Eng	Engine
L	Load
$Measured$	Measured Value
$Mech$	Mechanical
PMS	Pump/Motor S
PMT	Pump/Motor T
$Roll$	Rolling Resistance
$Slope$	Slope Force
$Total$	Total Force

Vol	Volumetric
a	Accumulator
d	Downstream of Valve
des	Desired
i	Index for Velocity of Current Time Step
j	Index for Acceleration at Current Time Step
is	Planetary Gearset Input (Ring Gear)
k	Time Index
l	Iteration Index
m	Motor
max	Maximum
$measured$	Measured Value
oil	Oil in Accumulator
p	Pump
pc	Planetary Gearset Output
pr	Precharge
q	Index for Acceleration at Next Time Step
u	Upstream of Valve
v	Valve
w	Wheel
z	Summation Index for MPC

Chapter 1

Introduction

A 2012 report by the United States Energy Information Administration shows the United States consumed about 100 quadrillion British Thermal Units (Quads) of energy in 2011 [1]. Figure 1.1 shows the breakdown of the energy flow. The right side lists the four main areas of consumption: Residential, commercial, industrial, and transportation. Transportation used about 27 Quads of energy in 2011, which is about 28% of the total energy consumption in the country.

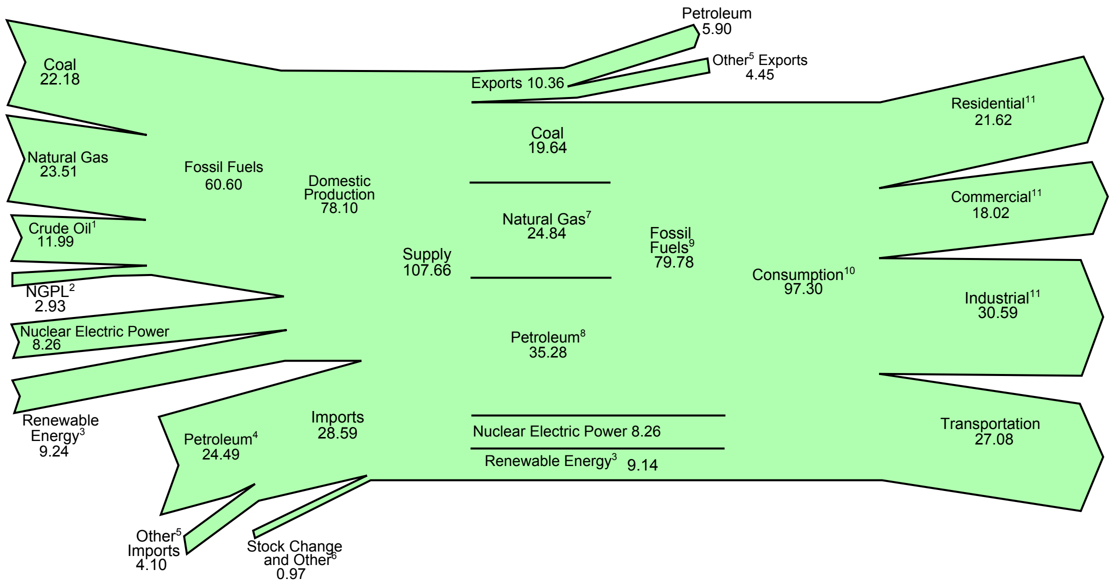


Figure 1.1: United States energy flow, 2011 (Adopted from [1])

In 2012, the average cost of a gallon of diesel fuel for on-highway applications was \$3.97 [3]. Diesel fuel has an energy density of approximately 138,000 Btu/gallon. Therefore, approximately \$775 billion was spent on energy for the transportation sector alone in the United States. Using these numbers, a 10% reduction in energy for transportation could lead to a savings of approximately \$76 billion a year. Furthermore, with the increasing cost of fuel each year, the savings will be even greater by making on-highway transportation more efficient. This significant savings of cost, along with lower emissions, is the motivation for developing a vehicle that dramatically improves fuel economy.

One way to accomplish this fuel economy improvement is by use of a hybrid vehicle. A hybrid vehicle is one that contains two sources of power, with one source most commonly being an internal combustion engine. The most common other types of power sources can be mechanical in the form of a flywheel, electric in the form of motor/generators and batteries, or hydraulic in the form of pumps/motors and accumulators. The hybrid also allows energy storage during braking events, allowing additional fuel consumption minimization.

To compliment the hybrid vehicle, an appropriate energy management strategy (EMS) must also be developed to distribute the power between the two power sources as efficiently as possible. At first glance, it may appear operating the engine most efficiently would be the best method for operating the system as a whole. However, by doing this, the other power source may be operating in a very low efficiency area, causing the entire system to be inefficient. Therefore, a systems approach must be taken to operate the system efficiently.

Currently, mass produced passenger hybrid vehicles have been electric hybrids. One reason is the technological advances that have been made in electronics over the past few decades. Also, electric batteries have high energy density, allowing large energy storage in compact batteries. However, a disadvantage of electric hybrids is the low power density of electric motors/generators and batteries.

To overcome this shortcoming, hydraulics have been proposed for use in passenger vehicles due to the large power density of hydraulic pumps/motors and accumulators. Also, hydraulic components are inexpensive when compared to their electrical counterparts, especially for state-of-the-art battery packs. Developments are being made in the area of digital hydraulic valves which could be used to turn a fixed displacement pump into a variable displacement pump to reduce weight and improve efficiency [4]. The efficiency of hydraulic pumps and motors is improving on innovations such as the floating cup principle [5] and digital displacement pumps such as the Artemis pump [6]. Higher energy density accumulators are also being studied to reduce the size and weight of accumulators [7]. These improvements make hydraulic technology more promising for passenger vehicles.

1.1 Hybrid Vehicle Architectures

Regardless of the secondary power source of the hybrid vehicle, three main types of architectures exist: parallel, series, and power-split. The power-split architecture can be either input-coupled or output-coupled depending on the location of the additional pump unit with respect to a planetary gear train. In this section, the overall operation of each will be described along with the similarities and differences. Figure 1.2 shows a schematic of each type of architecture.

1.1.1 Parallel Hybrid Vehicle

A parallel hybrid vehicle uses a conventional mechanical drive train. The engine shaft is directly connected to a transmission, which is connected to a differential to provide power to each wheel. In addition, the hydraulic pump/motors are connected to the drive shaft between the engine and the transmission to provide hydraulic energy stored in the accumulator for driving or store hydraulic energy during deceleration of the vehicle. Also, a clutch is placed between the engine and hydraulic pumps/motors so the engine can be decoupled completely from the road load and powered entirely by hydraulics if enough energy is contained in the accumulator. This allows the engine to be turned off when not needed, and turned back on when the accumulator becomes low. Optimal engine management cannot be obtained since the engine speed is related to the wheel speed by the transmission gear ratio.

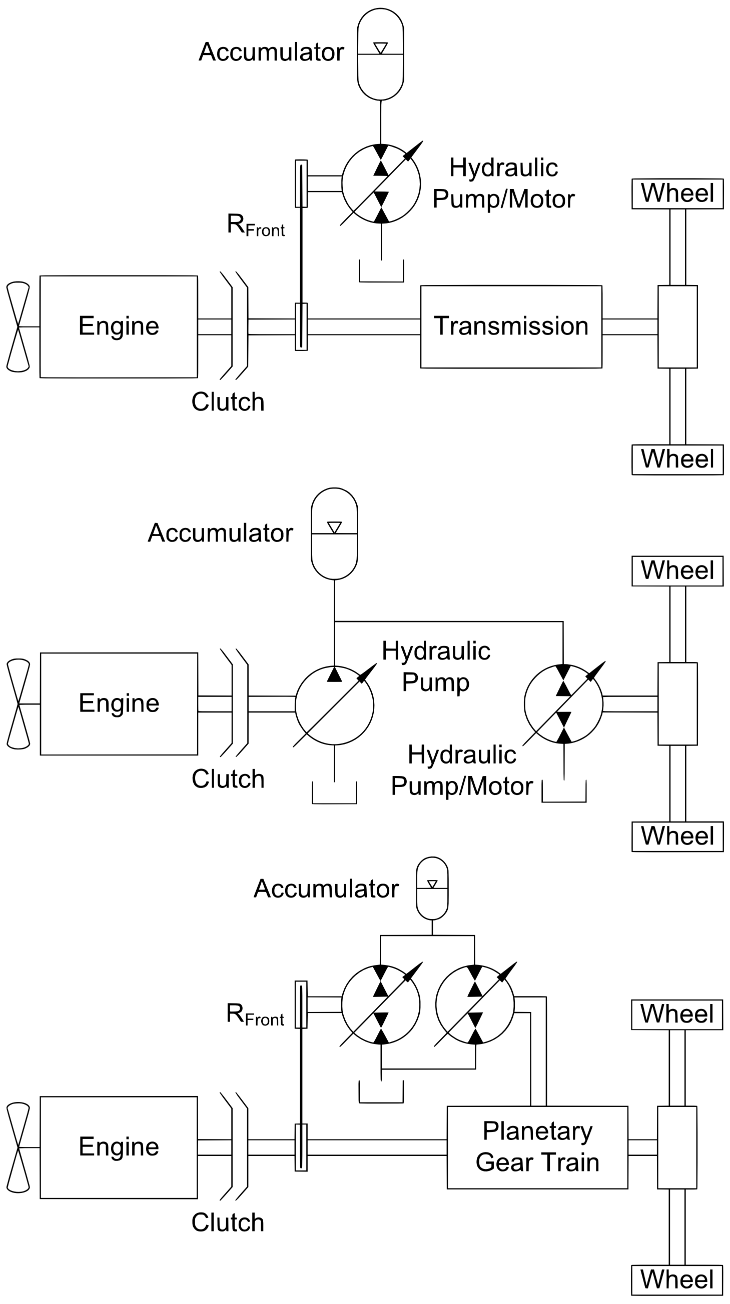


Figure 1.2: Schematic for the parallel hybrid (top), series hybrid (middle), and power-split hybrid (bottom) configurations

1.1.2 Series Hybrid Vehicle

In a series hybrid vehicle, the mechanical drive train is removed, and the vehicle is powered by a hydrostatic drive. The engine shaft is directly connected to a hydraulic pump/motor, which is connected to an accumulator to allow for energy storage. A hydraulic pump/motor is placed at each wheel to provide power and propel the vehicle. A clutch is again placed immediately downstream of the engine to allow for decoupling of the engine and on/off engine management. This architecture not only allows the engine output power to not match load demand, but also the speed does not need to match wheel speed due to the pump/motor at the wheels, allowing for optimal engine management.

1.1.3 Power-split Hybrid Vehicle

The power-split configuration combines the parallel and series architectures into one. The mechanical drive train is conventional as in the parallel hybrid design. However, hydraulic pump/motors are also connected to the drive wheel shafts as in the series hybrid design. This configuration allows for optimal engine management since, even though the engine is coupled mechanically to the drive wheels, the pump/motors at the wheels can be used to achieve desired wheel speed. The clutch immediately downstream of the engine allows the engine to be decoupled from the load, just as in the parallel and series configurations. The power-split combines the advantages of both the parallel and series configurations: the mechanical drive train enables highly efficient power transfer from engine to wheels of the parallel architecture while maintaining the optimal engine management of the series architecture.

Each of the three different types of architectures have their own advantages and disadvantages. While parallel is easy to implement of existing non-hybrid architectures, the fuel savings are not as great because the engine operation is not independent of the wheel load. The series architecture accomplishes this goal but with a penalty of transmission efficiency. The power-split is a combination of the parallel and series, which also means it is the most complex to control and the most efficient control strategy is not apparent.

1.2 Ways A Hybrid Vehicle Saves Energy

Figure 1.3 shows the estimated losses in a typical passenger vehicle. As can be seen in this figure, over 75% of the losses in a vehicle result from engine operation.

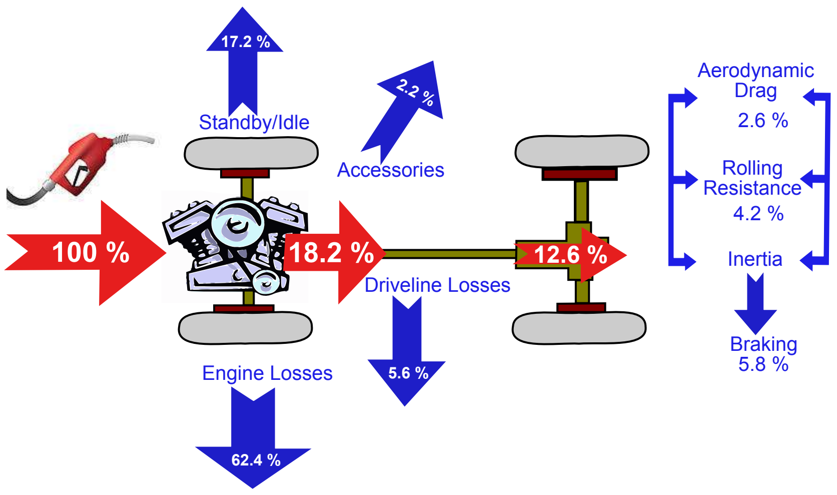


Figure 1.3: Estimated Losses In A Vehicle (adopted from [2])

A hybrid vehicle is able to reduce the losses in a vehicle in four ways.

1. **Operate the engine efficiently.** As shown in Figure 1.3, over 60% of the losses in a vehicle are from engine losses. This is because the engine has to be sized to handle high acceleration events, while the majority of the time this much power is not needed and is operating in a 10-15% efficient region. In a hybrid vehicle, the power output of the engine does not need to match the power demand of the vehicle since excess energy can be stored. Therefore, the engine can operate in a high power, high efficient region most of the time, reducing engine losses.

2. **Downsize the engine.** Similar to the previous point, because energy can be stored, the engine does not need to be sized for high power events. Instead, it can be sized for the typical power demand of a city drive cycle, and the stored energy can be used for short bursts of high power. By downsizing the engine, it is more appropriately sized for the majority of driving and therefore also helps in reducing engine losses.

3. **Turn the engine off.** As shown in Figure 1.3, the second most cause of losses in a vehicle are standby and idling losses. In a non-hybrid vehicle, the engine must be continually running and ready to provide power. Some advanced controllers allow the engine to turn off when the vehicle comes to a stop and then turn on again as soon as the driver steps on the accelerator pedal, but this results in a slight delay while the engine has to start. In a hybrid vehicle, the engine can turn off not only when stopped, but also when moving and enough energy is stored to power the vehicle. By doing this, the engine is rarely idling and the losses are almost completely eliminated from the vehicle.

4. **Regenerative braking.** In a non-hybrid vehicle, braking energy is lost to heat from the friction brakes. However, a hybrid vehicle is able to reduce this loss by running the motor as a pump (or generator) and recovering this energy. It can then use this stored energy to propel the vehicle and thus use the braking energy rather than losing it as heat.

1.3 Energy Management Strategy

As stated earlier, an appropriate energy management strategy must be developed to have the system operate as efficiently as possible. Figure 1.4 shows a block diagram for the EMS and how it fits into the operation of the hybrid vehicle.

The EMS consists of two closed-loop feedback system. The first is a controller which is responsible for tracking operator commands to ensure the vehicle accelerates and brakes when desired. The second is the EMS which is responsible for fulfilling the operator commands as efficiently as possible. In a non-hybrid vehicle, the operator's command is fed directly into the controller, which generally increases or decreases the throttle command to the engine. However, in a hybrid vehicle, the operator's command is fed into the EMS, which is interpreted as a desired torque command. This desired torque is then broken down into power commands for the engine and secondary power source. The EMS also inputs vehicle states, such as energy stored, to determine the optimal way to distribute the power.

Three EMS methods are studied in this project. The first is a rule-based strategy, which is a set of rules developed based on dynamic programming results. The second is

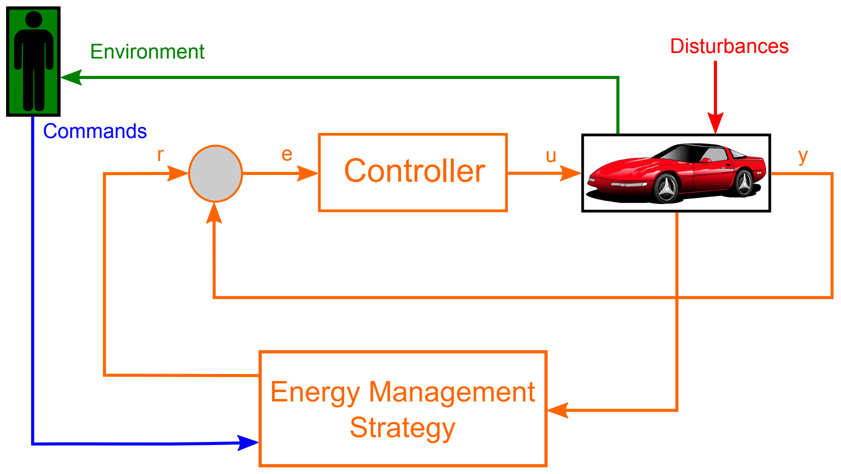


Figure 1.4: Block diagram of the energy management strategy for a hybrid vehicle

stochastic dynamic programming (SDP), which is a lookup table based on statistics of driving behavior. The final is model predictive control (MPC), which uses a dynamic model to predict a short horizon span in the future.

1.4 Outline

The remaining chapters are arranged as follows.

- Chapter 2 is a literature review describing the history of optimizing the control strategy for hybrid vehicles using various algorithms.
- Chapter 3 describes the model for the power-split architecture of the passenger vehicle.
- Chapter 4 introduces the dynamic programming algorithm, the development of a rule-based strategy from the dynamic programming results, and implementation on a dynamic model of the vehicle.
- Chapter 5 describes the experimental vehicle, presents results from experimental tests, and compares the experiment results to the simulation.
- Chapter 6 introduces the stochastic dynamic programming algorithm, the development of the Markov chain for transition probabilities, and results from simulation

of the vehicle.

- Chapter 7 presents a variety of different strategies that could be used on the vehicle, along with a study on varying the accumulator size.
- Chapter 8 describes the Augmented Earthmoving Vehicle Powertrain Simulator and presents results using a rule-based strategy, stochastic dynamic programming, and model predictive control.
- Chapter 9 presents the conclusions and future work.

Chapter 2

Literature Review

2.1 Overview of Hydraulic Hybrids

The idea of using hydraulics for hybrid vehicle applications is not new. Work began as early as the 1960's for using hydrostatic power-splitting transmissions for use in agricultural equipment [8]. Two very broad categories of these power-split transmissions exist: three-shaft and four-shaft systems. The three-shaft systems can further be broken down into three subcategories: Input coupled, output coupled, and hydraulic differential. All utilize a planetary gear train with two hydraulic units. The authors concluded that one configuration cannot be recommended for all applications since each has its own advantages and disadvantages, and the designer must look at the specifications of the application before one is chosen.

The University of Wisconsin-Madison began work on hydraulic hybrid vehicles in the 1980's. A passenger vehicle using one hydraulic pump/motor with an accumulator was studied [9]. Since a clutch is placed immediately downstream of the engine, three modes of operation are possible: Direct hydrostatic when enough pressure exists in the accumulator, hydrostatic power split where the pump/motor absorbs or provides additional power, and direct mechanical drive. Either a gear transmission or continuously-variable mechanical transmission (CVT) can be incorporated into this configuration. A basic control strategy is suggested where the engine runs at an established minimum allowable power output until the accumulator reaches an allowable pressure corresponding to vehicle speed, the engine shuts off until the accumulator reaches a lower setpoint, at which time the engine turns back on and the process is repeated. When a CVT is used, an additional choice exists: whether ratio control or torque control should be used. Ratio control means that the ratio of the output speed to input speed is proportional to an input signal, while torque control means the torque is proportional to the input signal. It was concluded that ratio control provided better stable operation under all conditions than torque control.

A series configuration was also proposed where a hydraulic pump is being driven by an engine, a hydraulic pump/motor is connected to the wheels, and an accumulator is connected between the two to allow for energy storage [10]. Several control strategies are described. The first, and simplest, is forward drive with accumulator where the engine pump is charging the accumulator, and the pump/motor acts as a motor to drive the vehicle. If the driver requires more torque than the hydraulic motor can provide at the accumulator pressure, the accumulator can be shut off and enter a direct hydrostatic drive. This is more difficult since flow continuity and power balance between the two hydraulic units must be maintained. The vehicle enters regenerative braking mode to stop the vehicle, where the pump/motor either goes over center to act as a pump, or valving is used to reverse the high and low pressure ports. The transitions between modes must be carefully designed for smooth transitions. Examples of these transitions include shutting off and reconnecting the accumulator to the high pressure line, switching between forward drive or direct hydrostatic drive to regenerative braking, and turning the engine on and off.

The University of Wisconsin-Madison also suggested that it would be possible to design an “All-Hydraulic Car” by using a dual pump/motor configuration with accumulator energy storage as described above, but replacing the internal combustion engine and hydraulic pump with a free piston engine/pump ([11], [12]). The free piston engine works as follows. When the engine fires, a piston assembly pumps oil to the drive motor or the accumulator. Some goes into a rebound accumulator, which is used to push the piston back for the next cycle. To stop the engine, the valve connecting the rebound accumulator is closed so the piston does not return to the starting position. All other systems work as described previously. One modification of adding pump/motors at all wheels could be included to provide traction control. When compared to a conventional automobile, little, if any, addition weight is added since components such as the transmission, differentials, and starter can be eliminated. It is expected that improvements in both fuel economy and performance could be achieved by making use of this configuration, but complexity is added to the system.

2.2 Modeling and Simulation

Modeling and simulation are valuable tools to determine how different power trains will perform before implementation on physical systems. This was used for all types of applications, including buses, commercial and military vehicles, and passenger vehicles.

2.2.1 General Hydraulic Hybrid Power Trains

Modeling and simulation of hydraulic hybrid vehicle power trains began in the early 1970's. In one study a family car, an urban car, a bus, and a delivery van each utilizing a series architecture with energy storage were simulated to examine fuel consumption and emissions over a given test cycle [13]. The simulation includes losses in the hydraulic pumps and motors, a basic engine model which splits fuel consumption into five regions, an accumulator model that includes heat transfer by a time constant, an emissions model based on a fourth-order least squares polynomial, a vehicle model that includes aerodynamic drag, rolling resistance, and inertial acceleration, and pressure losses in hydraulic lines. A basic control strategy where accumulator pressure is decreased as vehicle velocity is increased to allow for energy storage during braking and a torque-speed schedule to provide minimum fuel consumption while utilizing maximum engine capacity is used. Several conclusions were made from the simulations. First, as the top speed of the vehicle was lowered, the accumulator size and engine power decrease. Also, the engine power is about one-half of the power required for a similar non-hybrid vehicle. Even though the fuel consumption decreased in urban driving conditions, the emissions did not meet standards. The authors concluded that buses and delivery vans would be ideal applications of hydraulic hybrid power trains, but family and urban vehicles would not be suitable due to the size and weight of the accumulators needed for energy storage.

More recently, many different software packages have been used to model and simulate hydraulic hybrid vehicles, including ADVISOR, AMESim, and Matlab/Simulink

[14]. ADVISOR incorporates a backward-facing model, meaning the power required at the wheels is calculated and then the power is calculated backward through the drive train to determine engine speed and torque. AMESim incorporates a forward-facing model, which uses a driver model to specify throttle and brake commands to follow a desired vehicle velocity trajectory. AMESim was used to model the hydraulic subsystem, while Matlab/Simulink was used to model the vehicle. The AMESim and Matlab/Simulink combined model was successful for evaluating hydraulic hybrid vehicle performance over various drive cycles and can be used for preliminary control strategy development. More recently, researchers have used the Autonomie software [15], which is software for designing and analyzing different powertrains developed at Argonne National Laboratory.

2.2.2 Buses

In the late 1970's and 1980's, researchers began developing simulations and trying different control strategies for city buses to improve fuel economy ([16], [17], [18]). Three different control strategies were simulated using a parallel hydraulic hybrid power train: on-off control, "Best efficiency" control, and "Constant IC Engine Torque" control [17]. On-off control is the simplest of the three where, during accelerating from a standstill, only the hydraulic motor is used until the accumulator has dropped to a certain level, at which time the engine provides the power needed. The "Best efficiency" control strategy is an extension of the on-off control, where the efficiencies of the engine and hydraulic motor are compared at the desired power level, and the more efficient device is used (given the accumulator is not empty). The third control strategy operates the engine at constant torque during acceleration and constant speed periods, and the hydraulic motor is used for peak load conditions. Each control strategy was simulated using three different buses over a specified driving cycle. All three strategies resulted in an improvement of 30% in fuel savings when the correct component sizes are used.

Simulations of regenerative braking systems in buses continued into the 1980's. Different hydraulic pump/motor sizes and accumulator minimum pressures were tried [16]. The results showed that increasing the pump/motor size had a greater effect on braking and acceleration performance than the minimum accumulator pressure. To achieve desirable acceleration and deceleration rates, either large units must be used or an additional unit must be added. The latter is preferred since the additional unit can be disconnected when not needed with no loss in transmission efficiency. With the correct choice for unit sizes, at least 50% of the kinetic energy could be captured and stored. Also, different component sizes were simulated using different drive cycles to determine the effect of each on fuel efficiency [18]. One result was that fuel savings is very sensitive to the drive cycle used. Also, for a drive cycle that has 4 stops per kilometer, a smaller accumulator was found to give slightly better fuel savings than using an accumulator twice the size. Even though not all braking energy is able to be recovered, rolling resistance decreases due to the lower weight of the components. The results also show that acceleration performance can improve by approximately 10% over a conventional bus. Therefore, adding hydraulics not only improves fuel consumption but also acceleration performance.

2.2.3 Heavy Commercial and Military Vehicles

Research has also been done on incorporating a hydraulic hybrid power train for heavy commercial and military vehicles, with emphasis on modeling the Permo-Drive Regenerative Energy Management Strategy (PDREMS) ([19], [20], [21], [22]). The PDREMS is a parallel architecture with the hydraulic pump/motor placed downstream of the transmission. This system was modeled and simulated using the software package ADVISOR ([19], [21]). Since ADVISOR was developed for electric hybrid vehicles, models were developed for the hydraulic pump/motors and accumulator. The simulation was then verified using an experimental vehicle with simple acceleration and deceleration drive cycles. The fuel consumption prediction from the simulation was within 2% of the experimental results.

This simulation model was then used to develop a fuzzy logic controller for the PDREMS [20]. A two-stage fuzzy logic control is used—one for the hydraulic subsystem and one for the power request. The first stage takes pressure, pump/motor speed, efficiency, and state-of-charge to determine the overall hydraulic power rank. The second stage takes the hydraulic power rank and requested power rank to determine the overall rank. This is converted into a power command that is given to the engine. This was implemented in the ADVISOR simulation, which resulted in a 2% improvement in fuel economy for the Federal Urban Driving Schedule and a 5% improvement in acceleration performance over the baseline control strategy used in [19].

A study was also done to minimize fuel consumption by finding optimal values for pump/motor size, accumulator size, and accumulator pre-charge pressure [22]. The ADVISOR model was used along with the DIRECT algorithm available with the Matlab Optimization Toolbox. The results showed an increase in pump/motor size, accumulator size, and higher pre-charge pressure yielded a 2% improvement in fuel economy for the Federal Urban Driving Schedule. The authors hypothesized that a greater increase could be accomplished if the optimization was applied to the entire drive train.

2.2.4 Passenger Vehicles

In the late 1970's, work began to minimize the fuel consumption of a hydraulic hybrid drive train without any loss in driving performance ([23], [24]). A series configuration is studied where an engine drives an axial-piston variable-displacement pump to generate hydraulic power, which is converted to mechanical power by an axial-piston variable-displacement motor connected to the wheels to drive the vehicle. An accumulator is connected to the hydraulic circuit to provide energy storage and regenerative braking. A dynamic model was developed taking into account engine dynamics, pump and motor dynamics, pressure losses in valves and hoses, polytropic compression-expansion in the accumulator, and a vehicle load model. The controller gains for the engine throttle and pump displacement were optimized using the complex method to minimize fuel consumption over a modified LA-4 drive cycle. The results from the optimization show approximately a 10% improvement in fuel consumption over the baseline simulation where the controller gains were found by the Ziegler-Nichols method.

A computer simulation of the parallel configuration described in [9] was developed to determine the effects of using different gear ratios and accumulator pressure ranges on fuel economy [25]. Both a 3-speed and 4-speed transmission were simulated, and different gear ratios were tried until a combination that minimized the fuel consumption over the Federal Urban Drive Cycle (FUDC) were found. The control strategy for selecting the transmission gear was to select the ratio that would provide the lowest possible engine speed while still meeting the torque requirements. No attempt was made to minimize gear shifts, but a penalty was added every time the engine was started to minimize engine cycling. The results show very little difference in fuel economy using a 3-speed or 4-speed transmission. Also, when compared to a baseline simulation of a conventional vehicle, the fuel economy doubled for the hybrid vehicle over the FUDC, with approximately 70% of the improvement due to regenerative braking. Finally, it was suggested that the hydraulic subsystem be disconnected at highway speeds to avoid frequent engine cycling and has little advantage if proper transmission ratios for high speed driving are chosen.

The series configuration described in [10] was modeled and simulated using a computer [26]. The operating strategy for the accumulator is divided into two parts. When the vehicle is operating below a certain speed, the capacity kept in reserve for regenerative braking is equal to the vehicle kinetic energy. Once the vehicle goes above that speed, the capacity for braking is kept constant to allow for power decoupling. To reduce engine cycling and improve system efficiency, the engine is operated at a lower speed whenever the power required is less than that produced by the engine at that speed. If the power required is more than the engine power at that speed and if the accumulator is empty, the engine would be operated at a higher power level. Also, once the engine is on, it stays on until the accumulator reaches the level described above, and if the engine is off it stays off until the accumulator is depleted. Since a 2-speed final drive is used, the operating policy is to select the ratio that the pump/motor can supply enough torque, and if both are satisfactory, then the one that allows the pump/motor to operate at a higher efficiency is chosen. To avoid excessive shifting, the system must stay in one gear ratio for at least 10 seconds. Several parameters were varied to determine their effect on fuel consumption. First, when the minimum operating speed of the engine was increased, fuel consumption decreased since the engine is operating more efficiently. However, if set too high, losses involved in storing more energy and engine cycling become factors. When the maximum operating pressure of the accumulator increases, fuel economy decreases due to higher losses in the pump/motors. By using a 2-speed final drive, the losses in the pump/motors can be significantly reduced, resulting in higher fuel economy. Finally, the fuel economy is very sensitive to the efficiency of the hydraulic units, which is expected since this is a series hybrid where all the power goes through the hydraulics.

Another study compared the fuel economy improvement of a series, parallel, and power-split hydraulic hybrid architecture to a conventional drive train [27]. All the simulations included the same engine model, a 4-speed automatic transmission, and 30 liter accumulator. The series and power-split configurations used one 57 cc/rev variable displacement hydraulic pump connected to the engine and one 95 cc/rev variable displacement pump/motor connected upstream of the 4-speed transmission. The parallel configuration used one 95 cc/rev variable displacement pump/motor. For urban driving, the simulation results showed the parallel configuration had a 71% improvement over the conventional vehicle, while the series and power-split each had approximately 50% improvement over the conventional vehicle. Even more improvement could be obtained for the parallel configuration using a 57 cc/rev pump/motor unit and refined energy management strategy. Therefore, it was concluded that the parallel configuration is the most promising for passenger vehicle applications.

2.3 Experimental Prototypes

Many experimental hydraulic hybrid prototypes have been built by different researchers for comparing simulation and experimental results. These include buses, commercial vehicles, laboratory test stands, and passenger vehicles.

2.3.1 Prototype Buses

Experimental buses were used to test parallel hydraulic hybrid power trains that were previously simulated ([28], [29]). The most basic control strategy was used where the kinetic energy from braking is stored in the accumulator, the bus is accelerated using the stored energy to drive the hydraulic motor, and once cruising speed is reached the engine solely powers the wheels. A shut-off valve was placed between the accumulator and hydraulic pump/motor to minimize leakage when the bus is stopped. The engine, pump/motor displacement, and shut-off valve are all controlled electronically. First, an acceleration test was performed to see how fast the bus could accelerate from 0 to 40 km/hr. The hybrid version was able to accelerate to this speed nearly 1 second faster while using 40% less fuel than the conventional counterpart. A comparison was also made using the M15 standard driving schedule. The fuel consumption of the hybrid bus was reduced by approximately 30% over the conventional vehicle. This shows that a reduction in fuel consumption for buses is possible using a hydraulic hybrid power train, and this reduction would likely increase using a more complex control strategy.

2.3.2 Heavy Commercial Vehicles

An experimental prototype incorporating the PDREMS power train was built to obtain experimental data ([21], [30]). The power train was incorporated into a Freightliner semi-truck and tested at an airstrip for controlled results. The vehicle was tested both with and without the PDREMS system functional. The results showed a 37% improvement in fuel savings averaged over all runs when PDREMS was activated, along with an improvement in acceleration/deceleration performance.

2.3.3 Laboratory Experiments

A test rig was designed and built for experimental studies in the lab for both parallel and series configurations for a passenger vehicle ([31], [32]). The inertia of the vehicle was simulated using a flywheel, while the engine was simulated using a hydraulic power supply, a hydraulic motor, an orifice, and a servovalve. The servovalve acts as a throttle on an engine, while the orifice is used to give the same torque-speed relationship as an engine. This enables different control strategies to be tested before being implemented on a vehicle.

The test rig was used to determine the efficiency of the hydraulic pump/motors and accumulator, efficiency of regenerative braking, and the stability of the system to simulate hybrid vehicles [33]. The effect of foam in the gas side of the accumulator was also tested. Without foam, the temperature of the nitrogen varied from about 400°F to -90°F and dropped 1150 psig with a holding time of 100 seconds. With the foam, the temperature difference was only about 55°F with a pressure drop of 200 psig in the same holding time. The inclusion of foam also increases the energy storage by about 4%. The accumulator efficiency was determined using two methods. The first is calculated by taking the flywheel energy at a succeeding peak and dividing by the energy of the flywheel at the preceding peak. The other method takes the total energy transferred during an acceleration/deceleration cycle divided by the total energy obtained. Both took into account losses for the flywheel. The results show that larger displacements have a higher regenerative cycle efficiency, which is expected due to the higher efficiency of the units at higher displacements. On average the regenerative cycle efficiency is about 70%. Finally, a controllability test of the test rig was performed [34]. To do this, the flywheel started from rest, accelerated to a speed, maintained a constant speed, and then decelerated by regenerative braking. An exact constant flywheel speed was not maintained due to changes in accumulator pressure. However, this would not be a problem in a vehicle since the driver provides feedback to adjust for the speed. The results concluded that the system is controllable, and transitions between engine-driving to accumulator-driving were smooth due to the accumulators.

2.3.4 Passenger Vehicles

A study was done where a Lincoln Navigator Sport Utility Vehicle was modified into a parallel hydraulic hybrid power train [35]. The engine was downsized, a 150 cc/rev bent axis variable displacement pump/motor was added downstream of the transmission, and two, 14.4 gallon accumulators were added. The vehicle was tested on a chassis dynamometer, and the results showed a 23% improvement in fuel economy for the EPA City Cycle and a 35% improvement in fuel economy for a heavier acceleration cycle over the conventional power train. The results also showed a decrease in emissions and a slight increase in acceleration performance, even with the downsized engine. While the noise level is slightly more than the conventional vehicle, it is still within acceptable limits. It was concluded that a parallel hydraulic hybrid vehicle could be used for this application.

2.4 Energy Management Strategy

In the early 2000's, modeling and simulations were used to optimize the energy management strategy for hybrid vehicles. Initially electric hybrid architectures were studied and later hydraulic hybrid vehicles were examined for delivery trucks and military vehicles.

2.4.1 Electric Hybrid Vehicles

The deterministic dynamic programming method was used to optimize the control strategy for an electric parallel hybrid truck to minimize fuel consumption [36] along with emissions [37]. First, a baseline control strategy was developed using engineering intuition and efficiency maps. Dynamic programming was then used to determine the globally optimal results, which led to a 28% improvement in fuel economy and a 12% improvement in emissions. However, since these results cannot be directly implemented, a new gear shift strategy, power-split strategy, and charge sustaining strategy were formed from the optimized results. With these new strategies an improvement of about 24% in fuel economy and about 5% in emissions. The strategy was also tried in simulation on drive cycles not included in the optimization, which also showed improvements over the baseline strategy.

Rule-based strategies were also developed using fuzzy logic control for series electric hybrid vehicle [38] and parallel hybrid vehicles ([39], [40], [41]). For the series hybrid, the energy consumption was minimized in simulation and tried on a physical prototype. The results showed robustness to disturbances and smoothness in operation, while requiring minimal time for development. In [41], the drivetrain losses were minimized for a parallel hybrid vehicle using fuzzy logic. In this strategy the accelerator and brake pedal commands are converted to a power demand, which is used along with the state of charge and motor speed to determine the optimal power split between the engine and electric motor. The simulation results showed about a 10% reduction in losses versus the default strategy in the software. In both [39] and [40] the fuel usage and emissions were minimized for the parallel architecture and compared to the default strategy along with different fuzzy logic rules. The results showed a improvement in fuel economy and emissions for a variety of drive cycles over the default strategy. Also, different rule-based strategies lead to different results, but some drain the battery excessively, so a charge-sustaining strategy should be used.

Research has also been done using the equivalent consumption minimization strategy (ECMS), with one study being applied to a parallel electric hybrid powertrain [42]. Two energy flows exist in this architecture: battery discharge and battery recharge. During battery discharge, both the engine and electric motor are used to drive the wheel. However, sometimes in the future the engine will need to be used to provide power to the motor (now acting as a generator) to recharge the battery. This implies an extra fuel consumption in the future, so the penalty for using the electric motor is positive. During battery recharge, the engine is used to power the vehicle, and additional power is used to recharge the battery through the electrical generator. At a future time, this electrical power can be used to power the vehicle, and the engine will have to produce less mechanical power. Therefore, the penalty for this mode is negative. An equivalent consumption coefficient is then defined for the electric path, and the objective is to minimize the fuel consumption of the engine plus the equivalent fuel consumption of the electric path, as in [43]. The results show that the engine operating points are chosen in high efficiency areas with a 17.5% reduction of fuel used in simulation. This verifies that the equivalent fuel consumption method can be used to develop a control strategy that uses less fuel than a baseline strategy. These results were compared to dynamic optimization results [44]. The dynamic optimization results showed a 1.45% higher fuel economy over the equivalent fuel consumption strategy for the UDDS drive cycle, but the ECMS used 0.036% of the calculation time. These results show that the equivalent fuel consumption strategy can be used to obtain a fuel economy close to the global result, but depends heavily on the conversion factor for electric power consumption.

Model predictive control (MPC) has also been used to develop energy management strategies for a power-split electric hybrid vehicle ([45], [46], [47]). In [45], the problem is formulated as a nonlinear optimization problem and the powertrain model are linearized. The resulting strategy was tried on a variety of drive cycles, and for each the MPC strategy did better than the default strategy in the software program used to model the vehicle. This was further developed into a controller nonlinear model predictive control [46]. Using this method showed even further improvement in the fuel economy. MPC was also used to compare the rule-based energy management strategies in the simulation tool ADVISOR for the power-split architecture [47]. Three different drive cycle were used and the resulting fuel economy from both were compared. Using the MPC algorithm resulted in a significant improvement over the default rule-based strategy, with two of the drive cycles showing over 30% improvement.

Stochastic dynamic programming is another method that can be used for a variety of applications, including generating random drive cycles, optimizing a gear shift strategy for a vehicle, and optimizing engine operation to minimize emissions [48]. This method was applied to the parallel hybrid architecture ([49], [50]). In [49], a parallel electric hybrid is modeled and simulated. A Markov chain based on power demend is used for the stochastic dynamic programming. The objective function is a combination of fuel consumption and emissions. The resulting control strategy is applied to a variety of standard drive cycles. The results show that the SDP results are as good or better than a rule-based strategy based on dynamic programming. This was taken one step further by using knowledge of the drive cycle and traffic from a global positioning system (GPS) device [50]. Without the GPS the results were within 3% of the global solution, but by adding information from the GPS into the controller led to a fuel economy within 0.2% of the global solution.

2.4.2 Hydraulic Hybrid Vehicles

The deterministic dynamic programming technique was also applied to a parallel hydraulic hybrid configuration for a military vehicle [51] and a delivery truck [52]. The dynamic programming results for the delivery truck showed about a 33% improvement over the baseline strategy, where the modified rule-based strategy was about 11% more efficient. The optimization of the military vehicle was slightly different because both component sizes and control strategy were optimized in a two-stage process. The component sizes were optimized using Sequential Quadratic Programming (SQP), the control strategy was optimized using dynamic programming, rules were extracted from the optimized results, and these new rules were used to run the SQP algorithm again to optimize component sizes. By optimizing both the component sizes and control strategy, an improvement of 32% in fuel economy is obtained. This shows significant improvement in fuel economy can be made to hydraulic hybrid vehicles by optimizing the control strategy.

An instantaneous optimization based control has been developed for an output-coupled hydraulic hybrid powertrain ([53], [54]). This optimization operates the powertrain at the most efficient point at throughout operation. When an accumulator is not present, this is done by optimizing the engine operation and maximizing the efficiency of the power flow paths to the wheels. When an accumulator is present, an extra degree of freedom is added, and now the power split ratio and engine operating point are control decisions. This is done by maximizing the accumulator energy as much as possible without compromising performance. Using this strategy resulted in a better fuel economy than the Toyota Prius for both urban and highway driving.

An input-coupled power-split hydraulic hybrid architecture has also been studied by colleagues at the University of Minnesota. Through simulation it was concluded that this architecture could outperform the series architecture in fuel economy using a simple control strategy, and more complex control strategies could lead to even further improvement [55]. A comparison to both parallel and series architectures using a simple optimal control study was also done, where it was shown the power-split architecture led to higher fuel economy numbers than the other architectures [56]. Due to both the mechanical and hydraulic branch of the power-split architecture, controlling the system can be complex. An internal speed variable is calculated which combines the hydraulic and mechanical branches. Simulation and experimental results show that regulating this variable is crucial for the operation of this type of hybrid architecture [57]. A three-level control strategy was also developed for this architecture, where the high level determines the accumulator power level, the mid level determines the operating points of the componentes, and the low level determines the actuator commands. This type of control strategy was successfully implemented in experiments using a rule-based strategy [58]. Besides the control strategy, the size of the components also have a significant impact on the efficiency of the overall system. For this reason, a study was done to find the optimal sizes of the components [59]. The power-split architecture also has the ability to remove components from the drive train by locking up the pump/motor units using valves, leading to further improvements in efficiency [60].

The input and output-coupled architectures were compared to each other to determine which gives better system efficiency ([61], [62]). In this study, a basic optimal control strategy was developed for both. The results shows similar fuel economy results for both architectures with smaller component sizes used for the input-coupled architecture. However, the control strategy for each was different - the input-coupled architecture is better suited with wheel torque control while the input-coupled architecture benefits with wheel speed control. Hybrid electric vehicles were also compared, which showed similar fuel economy results to the hydraulic hybrid. The major difference was in component sizes where a large battery pack is needed to absorb and provide the power needed, which is the major advantage of using a hydraulic accumulator instead of an electric battery for energy storage.

Drive cycle statistics were also used to develop real-time implementable strategies. Stochastic dynamic programming was used on a parallel hydraulic hybrid architecture

([63],[64]). A baseline control study was first done to develop a thermostatic control strategy. Stochastic dynamic programming was then used and compared to the baseline strategies. Both strategies were tried over three different control strategies, and in each case the SDP strategy used less fuel than the baseline strategy. Stochastic model predictive control (SMPC) was also tried on a series hybrid architecture [65]. The SMPC optimizes the fuel consumption over a distribution of future driver's request represented by a Markov model much like the SDP strategy. The resulting strategy was simulated through an urban drive cycle and compared to three other strategies - one baseline and two from model predictive control. The results showed that the SMPC strategy did the best out of all the strategies, showing that including statistics of the driving behavior can be beneficial in developing the control strategy for a hybrid powertrain.

2.5 Areas Not Currently Addressed

As the literature review shows, much research has been done on the development of energy management strategies for electric hybrid vehicles. However, less research has been done for hydraulic hybrid vehicles, and the majority that has been done is for large applications such as buses, delivery trucks, and military vehicles. The studies that have been done for passenger vehicles have mainly been for parallel or series architectures. The power-split architecture that was studied was an output-coupled type of architecture. Therefore, this research aims to fill the current gaps in the hydraulic hybrid community.

- Develop an energy management strategy for an **input-coupled** hydraulic hybrid architecture for use in a **passenger sized vehicle.**

- Study the effect of **accumulator size** on fuel economy and performance to determine the optimal size for passenger vehicle applications.

- Use a variety of methods to develop an **array of energy management strategies** and compare them to different driving scenarios and compare each to the global optimal solution.

- Develop a system where energy management strategies can be **easily integrated into other types of architectures** without doing much rework.

- Form a **decision methodology** to help aid in which method should be used for developing an energy management strategy for a hydraulic hybrid application based on a variety of parameters to save time in the control strategy process.

Chapter 3

System Modeling

In this chapter, the system models for the hydraulic hybrid vehicle and the Augmented Earthmoving Vehicle Powertrain Simulator (AEVPS) are described. First the system architecture is described. Then the backwards-facing model is explained, taking into account engine efficiency and the efficiency of the hydraulic pump/motor units. The dynamic model used for the vehicle is also presented.

For the AEVPS system, the dynamic model is explained. From this dynamic model, the system is then discretized, which is used to develop the energy management strategies.

3.1 Hydraulic Hybrid Vehicle Model

In this section, the hydraulic hybrid vehicle model used in the development of the energy management strategy is described. The model is a backwards-facing model, meaning the amount of power needed at the wheels for a given duty cycle is calculated first, and then calculations are made backwards through the drivetrain to determine the amount of power the engine and hydraulic units must provide to meet the wheel load. Even though this is not how a vehicle operates, this type of model is good to use for optimization because the computation time is shorter than that of a forward-facing model, and the calculations for this type of model are simpler.

3.1.1 Vehicle Architecture

The architecture studied throughout this research was a power-split passenger vehicle. A power-split architecture can have two different forms: input-coupled and output-coupled. The form of power-split architecture is determined by where the power source is located in regards to a power-split device, which is usually a planetary gearset. An output-coupled architecture has the power source connected to the output of the planetary gearset which would then be connected to the wheels to power the vehicle. An input-coupled architecture, which is the type being studied in this research, has the power source connected between the output of the engine and the input to the planetary gearset. Figure 3.1 shows a stick diagram of the configuration studied in this research.

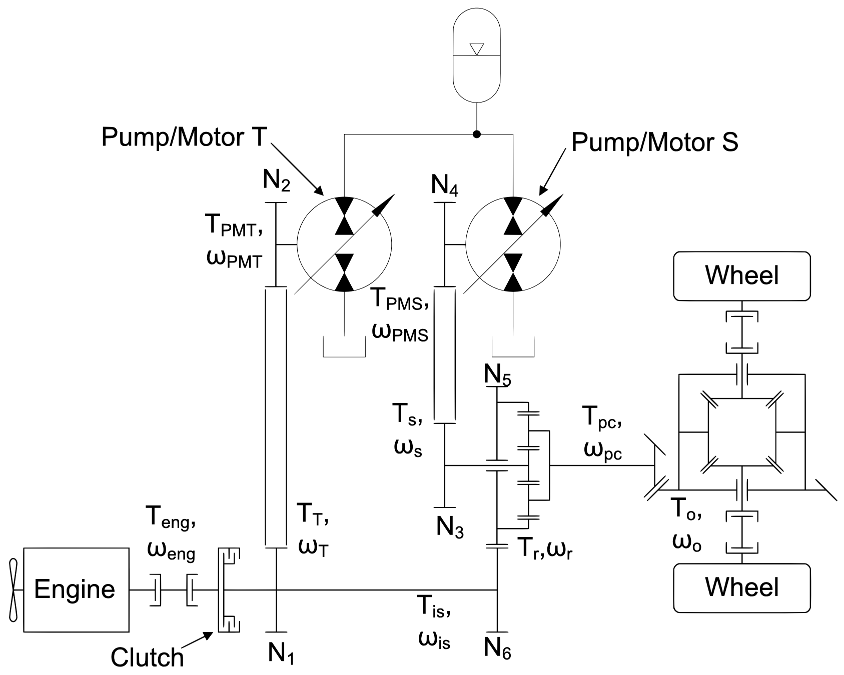


Figure 3.1: Stick diagram of the input-coupled, power-split architecture modeled for this research

As can be seen in Figure 3.1, hydraulic pump/motor T is connected to the output of the engine through a gear ratio. A clutch is placed between the engine and pump/motor

T to allow the engine to be decoupled completely from the system and run only on hydraulic power. This is connected to the ring gear of the planetary differential. Hydraulic pump/motor S is connected to the carrier of the planetary gearset through a gear ratio. Between the two hydraulic units is an accumulator to allow for energy storage. The sun gear of the planetary gearset is the output to a differential, which splits the power between the two rear wheels of the vehicle.

3.1.2 Backwards-Facing Model

The vehicle model used for the optimization of the energy management strategy is a backwards-facing model. The input to the model is a duty cycle, which gives the vehicle velocity as a function of time. From the velocity, the acceleration at each time step can be calculated using the forward difference method.

$$a_k = \frac{v_{k+1} - v_k}{\Delta t} \tag{3.1}$$

With the acceleration known, the force required at each time step is calculated. First, the force required to overcome the inertial acceleration of the vehicle is calculated.

$$F_{Accel,k} = m a_k \tag{3.2}$$

The vehicle must also overcome aerodynamic drag, given by Equation (3.3), and rolling resistance of the tires against the pavement, given by Equation (3.4).

$$F_{Drag,k} = \frac{1}{2} C_d A_f \rho v_k^2 \tag{3.3}$$

$$F_{Roll,k} = \left(f_o + 3.24 f_s \left(\frac{2.23693629 v_k}{100} \right)^{2.5} \right) mg \tag{3.4}$$

If the vehicle is on a hill, the slope of the hill must also be taken into account.

$$F_{Slope,k} = mg \tan(\theta) \tag{3.5}$$

The total force required to propel the vehicle is the sum of Equations (3.2) - (3.5).

$$F_{Total,k} = F_{Accel,k} + F_{Drag,k} + F_{Roll,k} + F_{Slope,k} \tag{3.6}$$

The power required at the wheels is equal to the total force multiplied by the vehicle velocity.

$$p_k = F_{Total,k} v_k \tag{3.7}$$

The torque needed at the wheels is the power required divided by the diameter of the wheel. The differential splits this torque between the two rear wheels.

$$T_{Wheel,k} = \frac{p_k}{d_{Wheel}} \tag{3.8}$$

With the output torque known, the torque output of the planetary gearset can be calculated using the gear ratio of the differential.

$$T_{pc,k} = \frac{T_{Wheel,k}}{R_{diff}} \tag{3.9}$$

The speed of the output shaft of the planetary gearset is calculated using the velocity of the wheels and the gear ratio of the differential.

$$\omega_{pc,k} = \frac{2R_{diff}}{d_{Wheel}} v_k \tag{3.10}$$

The torque on the ring gear of the planetary differential is calculated using the output torque and the ratio of the planetary differential.

$$T_{is,k} = -\frac{T_{pc,k}}{R_{pd} + 1} \tag{3.11}$$

The engine torque and engine speed are a vector of control variables and therefore inputs to the model. Knowing the torque of the engine and the torque on the input shaft to the planetary gearset, the torque on pump/motor T can be calculated.

$$T_{PMT,k} = \frac{T_{is,k} - T_{Eng}}{R_T} + 0.07\omega_{Eng} \tag{3.12}$$

In Equation (3.12), the last term accounts for mechanical losses in the system as a function of rotational engine speed, and R_T is the gear ratio between the engine and pump/motor T. Using the notation in Figure 3.1, this is expressed as the following.

$$R_T = \frac{N_2}{N_1} \tag{3.13}$$

The rotational speed of pump/motor T is calculated using the engine speed and R_T.

$$\omega_{PMT,k} = \omega_{Eng} R_T \tag{3.14}$$

The torque on pump/motor S is calculated using the output torque, the ratio of the planetary differential, and the gear ratio between pump/motor S and the carrier gear.

$$T_{PMS,k} = \frac{R_S R_{pd}}{R_{pd} + 1} T_{pc,k} \tag{3.15}$$

In Equation (3.15), the term R_S is the gear ratio between pump/motor S and the carrier shaft of the planetary gearset. Using the notation in Figure 3.1, this is expressed as the following.

$$R_S = \frac{N_4}{N_3} \tag{3.16}$$

The speed of pump/motor S is calculated using the speed of pump/motor T along with the speed of the output shaft of the planetary differential.

$$\omega_{PMS,k} = \frac{\omega_{PMT,k}}{R_S R_T R_{pd}} - \frac{R_{pd} + 1}{R_S R_{pd}} \omega_{pc,k} \tag{3.17}$$

3.1.3 Sign Convention

When power is being put into the planetary gearset, the sign is positive. Likewise, when power is being taken away, the sign convention is negative. For example, when the vehicle is braking, power is being put into the system from the wheels, so the sign on the power from the wheels is positive, even though the calculations from above would give a negative power since the acceleration is negative. Likewise, when the hydraulic unit is acting as a pump, it is taking power from the system, and the sign of this power is negative. Using this convention, the sum of the power on the carrier, ring gear, and sun gear should always be zero, assuming that the planetary gearset is 100% efficient. This provides a check in the model to ensure that the power flow in the system is correct.

3.1.4 Hydraulic Pump/Motor Units

The model of the hydraulic units is based on experimental data from the Rexroth 28 cc/rev variable displacement units installed on the experimental vehicle. The test data was obtained using a test stand assembled at the University of Minnesota [66]. A picture of the test stand is shown in Figure 3.2.

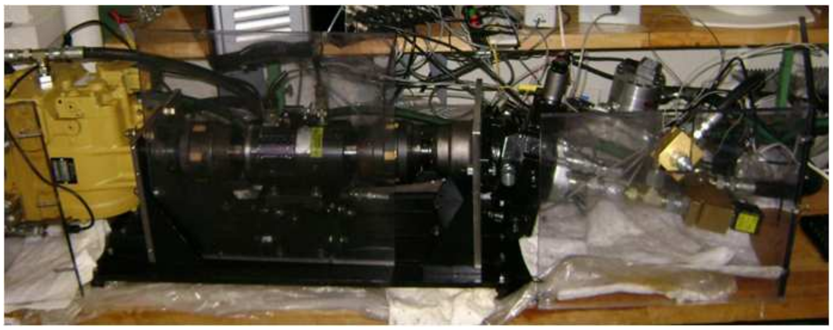


Figure 3.2: Pump test stand used to obtain experimental efficiency data

In Figure 3.2, the unit being tested is shown on the right. The driving unit (if the unit is being tested as a pump) or the driven unit (if the unit is being tested as a motor) is the unit on the left, which is larger than the unit being tested. In the middle is a torque and speed sensor. Pressure and flow sensors are located at the inlet, outlet, and case drain ports of the unit being tested. The xPC Target toolbox in Matlab is used for both the data acquisition and control. In a test, the pressure is set from a supply unit, the displacement of the test unit is set, and the speed is maintained constant by adjusting the displacement of the larger unit. By recording the pressure and flow rate at each port, along with the torque on the shaft, the mechanical and volumetric efficiency can be calculated. When the test unit is acting as a pump, the mechanical efficiency is given as

$$\eta_{Mech} = \frac{\phi D P}{T_{Measured}} \tag{3.18}$$

and the volumetric efficiency is given as

$$\eta_{Vol} = \frac{Q_{Measured}}{\phi D \omega} \tag{3.19}$$

When the test unit is acting as a motor, the inverse of Equations (3.18) and (3.19) would be used. The total efficiency is then calculated by multiplying the mechanical and volumetric efficiency.

$$\eta_{Total} = \eta_{Mech}\eta_{Vol} \tag{3.20}$$

By using gridpoints of different pressures, speeds, and displacements, a lookup table is generated for the mechanical and volumetric efficiencies, from which the total efficiency is calculated. A contour plot of the total efficiency for pumping and motoring at a pressure of 3000 psi is shown in Figure 3.3. At this pressure, the maximum total efficiency for motoring is approximately 84%, and the maximum total efficiency for pumping is slightly over 80%.

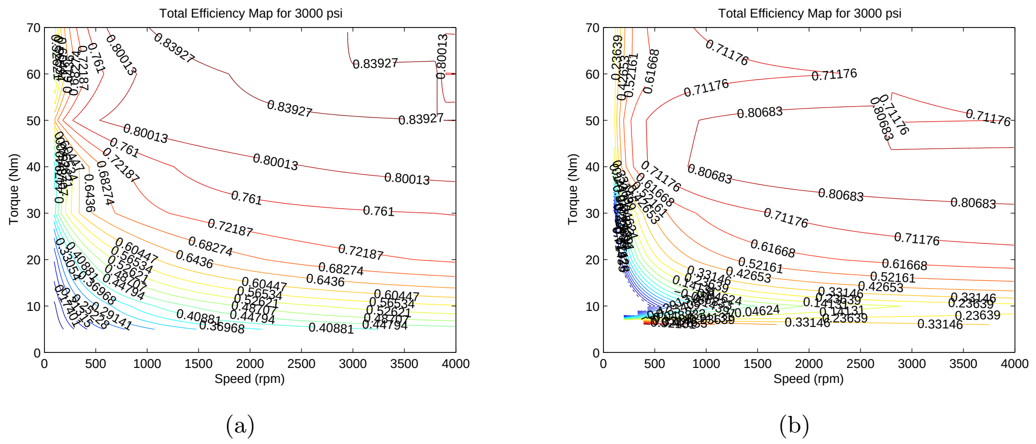


Figure 3.3: Experimental total efficiency maps for (a) motoring and (b) pumping modes of the hydraulic units

3.2 Engine

The engine model is based on experimental data obtained from the Perkins 404C-15 engine installed in the experimental vehicle. Since an engine dynamometer in the appropriate power range was not available, the experimental data was obtained with the engine installed on the vehicle. Pump/motor T was used to load the engine at various speeds and torques. Since a torque sensor was not available, the experimental pump map was used to estimate the torque knowing the pressure, displacement, and speed of the hydraulic unit. A turbine style fuel flow sensor was installed at the outlet of the gas tank to measure the flow consumption. Figure 3.4 shows how the fuel flow sensor was connected in the system to estimate the instantaneous fuel consumption.

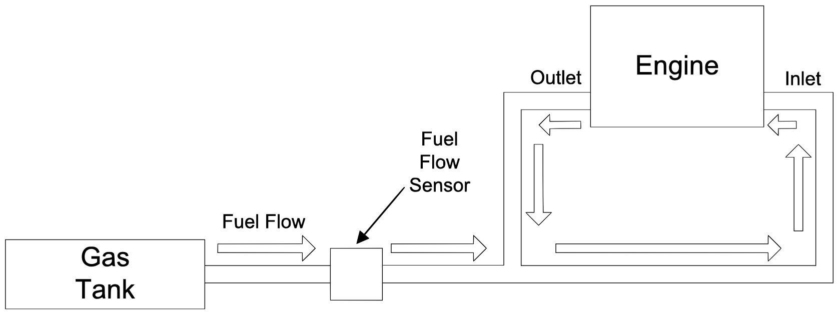


Figure 3.4: Diagram of how fuel sensor is connected in the system

Knowing the energy content in diesel fuel and the flow rate of fuel into the engine, the efficiency at each speed and torque point is calculated by taking the output power and dividing by the input power. This generates an efficiency contour plot, which is shown in Figure 3.5.

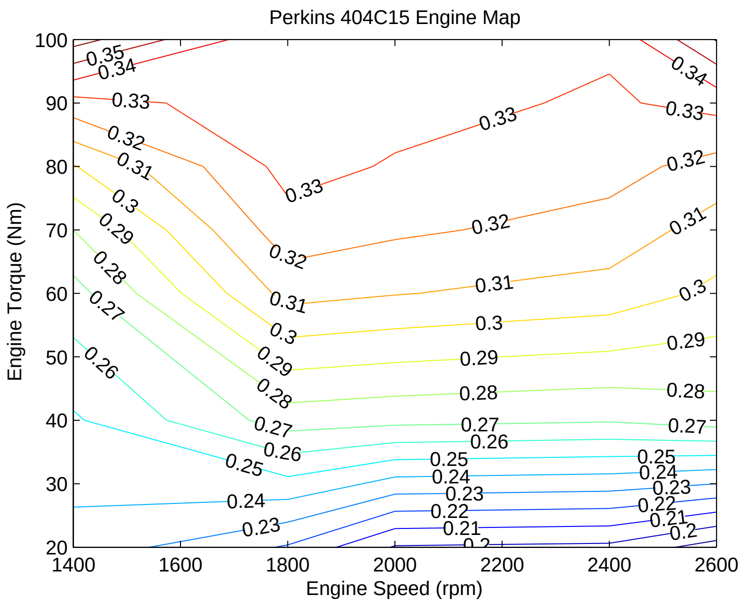


Figure 3.5: Engine efficiency map

3.2.1 Accumulator

The accumulator is a 38 liter bladder-type accumulator with a precharge pressure of 1600 psi. An isothermal model is used for this analysis, which has been shown to be accurate if the accumulator contains foam [67]. The flow rate from each of the hydraulic units is known from the volumetric efficiency maps. Keeping with the same sign convention, a negative flow rate indicates that the unit is pumping and taking power from the system, while a positive flow rate indicated the unit is motoring. Knowing the initial volume of oil in the accumulator, Euler's method is used to calculate the volume of oil in the accumulator at each time step. This is given by Equation (3.21).

$$V_{oil}\left(k+1\right) = V_{oil}\left(k\right) - \left(Q_{PMT} + Q_{PMS}\right)\Delta t \tag{3.21}$$

Once the volume of oil in the accumulator is known, the pressure is calculated from the known precharge pressure.

$$P\left(k\right) = \frac{P_{pr}V_a}{V_a - V_{oil}\left(k\right)} \tag{3.22}$$

This is repeated for each time step to find the pressure at each time step in the drive cycle.

3.2.2 Dynamic Model

A dynamic model was developed in Simulink that includes dynamics of the vehicle, engine, and hydraulic units. Figure 3.6 shows the top-most level of this model.

As shown in Figure 3.6, this model contains four main subsytems.

1. **Controller.** This contains the driver model, which is a feedforward plus PI controller to ensure that the vehicle velocity tracks the desired drive cycle. It also contains the high level controller which is the result of the optimization, the mid level controller which determines the engine power, and the low level controller which contains controllers for the hydraulic units as well as an engine controller for tracking the desired engine speed from the high level controller.

2. **Hybrid Powertrain.** This contains the models for the hydraulic units, including the same efficiency maps as those shown in Figure 3.3, the engine model for

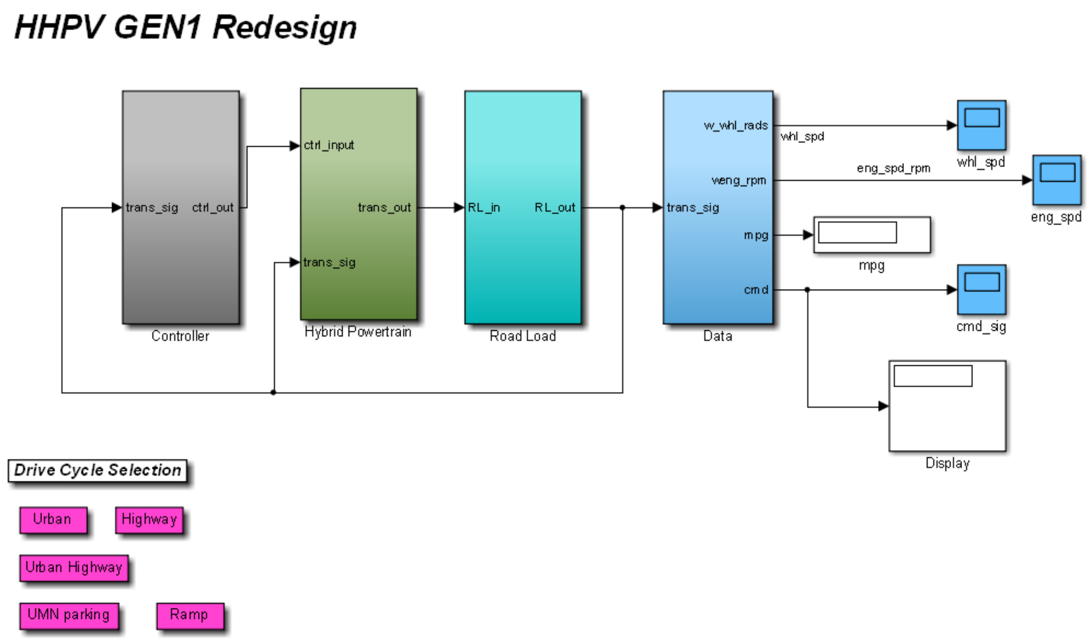


Figure 3.6: Top level of the dynamic model developed in Simulink

calculating the fuel consumption, the isothermal accumulator models, as well as valve models. This also includes the transmission dynamics and the kinematics described by Equations (3.9) - (3.17).

3. **Road Load.** This contains the model for the road load, which is the same as described by Equations (3.3) - (3.5).

4. **Data.** This saves all the data from the simulation so it can be plotted and analyzed.

This model is used to simulate the energy management strategies before implementation on the experimental vehicle.

3.3 Augmented Earthmoving Vehicle Powertrain Simulator Model

This section describes the dynamic model for the Augmented Earthmoving Vehice Powertrain Simulator (AEVPS) system by looking at each component within the system. For a more detailed description of the system model, see [68]. First, the engine is examined, taking into account frictional losses in the engine as well as the load from the hydraulic pump, as shown in Equation (3.23).

$$I_{Eng}\dot{\omega_{Eng}} = T_{Eng} - \frac{K_{Eng}}{2}\omega_{Eng}^2 - b_{Eng}\omega_{Eng} - K_p u_p P_u \tag{3.23}$$

The upstream and downstream pressures are based on the mass flow in those regions. For the upstream pressure, flow comes into the junction from the pump, and flow exits into the accumulator, valve, and leakage, as shown in Equation (3.24).

$$\dot{P_u} = \frac{\beta_u}{V_u}(Q_p - Q_a - Q_v - \psi_u P_u) \tag{3.24}$$

The flow from the pump is determined by the displacement and rotational speed as shown in Equation (3.25).

$$Q_p = K_p u_p \omega_{Eng} \tag{3.25}$$

The flow into the accumulator is a function of the precharge pressure, capacity, and specific heat ratio of the gas.

$$Q_a = \frac{V_a}{\gamma} P_{pr}^{\frac{1}{\gamma}} P_u^{\frac{-(\gamma+1)}{\gamma}} \dot{P_u} \tag{3.26}$$

Finally, the flow through the valve is modeled as the orifice equation as shown in Equation (3.27).

$$Q_v = C_v\sqrt{\Delta P} \tag{3.27}$$

The valve flow coefficient is found as a function of valve command based on experimental data.

$$C_v = 12u_v^3 - 40u_v^2 + 65u_v - 4.5 \tag{3.28}$$

$$\Delta P = P_u - P_d \tag{3.29}$$

For the downstream pressure, flow enters from the valve and exits to the hydraulic motor and leakage. The dynamics are shown in Equation (3.30).

$$\dot{P_d} = \frac{\beta_d}{V_d}(C_v\sqrt{\Delta P} - D_m\omega_m - \psi_d P_d) \tag{3.30}$$

Next, the dynamics of the hydraulic motor are examined. Torque is provided by the pressure and displacement, which is lost as inefficiencies and load torque. The dynamic equation for the hydraulic motor is given by Equation (3.31).

$$I_m\dot{\omega_m} = D_m P_d - b_m\omega_m - T_L \tag{3.31}$$

The load torque is calculated by taking into account aerodynamic drag and friction in the transmission. It is divided by four to by consistent with the 1/4 model that was used when modeling the engine.

$$T_L = \frac{1}{4}(b_w\omega_m + \frac{r_w}{2}\rho C_{drag}A_f(r_w\omega_m)^2) \tag{3.32}$$

Finally, the fuel consumption of the engine is estimated from the throttle command and speed of the engine, shown in Equation (3.33). When the engine is idling, a fuel rate of 0.43 g/s is used.

$$\text{fuel rate} = \max(k_f\omega_{Eng}\frac{\tau}{\tau_{max}}, 0.00043) \tag{3.33}$$

3.3.1 Discrete model

To use the discrete-time, discrete-space dynamic programming algorithm, the above continuous equations must be discretized. The Euler method was used to discretize the continuous differential equations into discrete difference equations. When Equation 3.32 is discretized, it becomes the following.

$$T_L(k) = \frac{1}{4}(b_w\omega_m(k) + \frac{r_w}{2}\rho C_{Drag}A_f(r_w\omega_m(k))^2) \tag{3.34}$$

When Equation (3.31) is discretized and rearranged, the downstream pressure can be calculated.

$$P_d(k) = \frac{\omega_m(k+1) - \omega_m(k)}{\Delta t}\frac{I_m}{D_m} + \frac{b_m}{D_m}\omega_m(k) + \frac{T_L(k)}{D_m} \tag{3.35}$$

The flow through the valve can be calculated using Equations (3.27) and (3.30).

$$Q_v(k) = \frac{V_d}{\beta_d}\frac{P_d(k+1) - P_d(k)}{\Delta t} + D_m\omega_m(k) + \psi_d P_d(k) \tag{3.36}$$

The upstream pressure is calculated from Equation (3.24).

$$P_u(k+1) = P_u(k) + \frac{Q_p(k) - Q_v(k) - \psi_u P_u(k)}{\frac{V_u}{\beta_u} + Q_a(k)}\Delta t \tag{3.37}$$

The valve command is calculated by setting Equation (3.36) equal to (3.27).

$$Q_v(k) = (12u_v^3(k) - 40u_v^2(k) + 65u_v(k) - 4.5)\sqrt{P_u(k) - P_d(k)} \tag{3.38}$$

The discrete model is used to develop the different energy management strategies for the AEVPS system, while the dynamic model is used to validate the results before being implemented on the physical system.

3.4 Conclusions

In this chapter, the models for the hydraulic hybrid vehicle and AEVPS system were presented. In both cases a backwards-facing model is used for the development of the energy management strategies, while a dynamic model is used to validate the results before implementation. In the next chapter, the methods for developing the energy management strategies for each system is presented.

Chapter 4

Theoretical Methods

In this chapter, the theory for using dynamic programming to find the global optimal solution is presented. However, since this method cannot be used for real-time implementation, a rule-based strategy is developed based on the results. To develop a strategy that can be implemented in real-time based on optimization, stochastic dynamic programming is used.

4.1 Dynamic Programming (DP)

Once the system configuration, components, and drive cycle are fixed, the fuel economy of the vehicle depends only on the strategy for splitting the power between the engine and hydraulics. The optimal control problem is formulated and solved by using the dynamic programming (DP) algorithm [69]. This is a powerful technique for solving optimal control problems for nonlinear, constrained dynamic problems since the true optimal solution is found. However, one drawback is the "curse of dimensionality," meaning computation time significantly increases as the dimension of the problem increases. However, since this problem is relatively small, computation times are kept reasonable.

The dynamic programming algorithm is based on Bellman's principle of optimality [70], which states that if a sequence of decisions is optimal, each subsequence must also be optimal. Using this principle, the algorithm can start at the end of the drive cycle, go one step back and find the optimal trajectory, go another step back and find the optimal trajectory, and continue this process until the beginning is reached. This significantly reduces the number of computations required than starting at the beginning and trying every decision at every time step to find the optimal trajectory.

The formulation of the problem for the hybrid vehicle is as follows. The objective is to find the optimal trajectory of control signals $u(k)$ to minimize the fuel consumption of the vehicle over an entire drive cycle. Mathematically, this is given by Equation 4.1.

$$\min_{u(k)} J = \sum_{k=0}^{k=K-1} L\left[x(k), u(k)\right] \tag{4.1}$$

In the above equation, L is the fuel consumption in one time segment, K is the number of time segments, x is the state vector, which includes vehicle speed and accumulator state of charge, and u is the control vector, which is engine power. The summation goes to $K-1$ and not K since a decision is not needed at the end of the drive cycle.

Now the dynamic programming algorithm can be implemented using the principle of optimality stated above. First the optimal cost at time step $K-1$ must be calculated.

$$J^*_{K-1}\left[x(K-1)\right] = \min_{u(K-1)} L\left[x(K-1), u(K-1)\right] \tag{4.2}$$

For all other time steps, the optimal control is found by minimizing the total cost.

$$J^*_k = \min_{u(k)} \left\{L\left[x(k), u(k)\right] + J^*_{k+1}\left[x(k+1)\right]\right\}, 0 < k < K-1 \tag{4.3}$$

Once the equation is solved backwards from step $K-1$ to 0, a lookup table is formed in which, given the state of charge of the accumulator at a time step, the optimal control is found to minimize fuel consumption. Then, given the initial state, the optimal control can be found from the lookup table. The model is executed to find the state at the next time step, and this can be propagated forward in time until the end of the drive cycle is reached. The resulting optimal control trajectory is then simulated to obtain the fuel economy result.

4.2 Rule-based Strategy

For the rule-based strategy, the results from dynamic programming are used to develop a control strategy that can be implemented in real-time by not using any future information. The rule-based strategy consists of two parts: A discrete part for engine idling, and a continuous part for how much power the engine should provide if the engine is not idling.

4.2.1 Engine idling

To determine if the engine should be idling or providing power, two state variables are used: Accumulator pressure and desired wheel torque. To keep the accumulator pressure from going to low, the engine provides power when the pressure falls below a lower threshold. Also, to avoid the accumulator pressure from getting too high, causing energy loss over the relief valve, the engine idles when the pressure is above a upper threshold.

When the pressure is between the lower and upper thresholds, the desired wheel torque is used to determine the engine idling state. Desired wheel torque can be divided into three regions: Braking, standing and small torques, and high torque events. Braking is when the wheel torque is negative as calculated by Equation (3.8). When the vehicle is braking, the engine idles as long as the pressure is greater than the lower pressure threshold. The other extreme is when a high torque demand is required. To ensure the system is able to provide the required power, the engine does not idle during these high torque demand events. For the standing/small torque demand, the previous engine idling mode is used.

4.2.2 Engine power

If the engine is not idling, the amount of power the engine provides must be determined. To determine the engine power, a curve fit is performed with wheel speed and wheel torque as the dependent variables. Several different curve fits are tried, and using linear regression, the one that results in the highest r^2 value is used.

4.3 Stochastic Dynamic Programming

One of the drawbacks of dynamic programming is that the end result is an acausal control law because it requires that the future is completely known. To obtain a causal control law, further analysis must be done, such as developing a rule-based strategy as described in the previous section. However, this leads to a suboptimal solution because the rule-based strategy does not use theory to develop the control law, but only approximates the DP solution.

To overcome this shortcoming, stochastic dynamic programming (SDP) is used to develop a causal control law directly. Rather than basing the solution on a single drive cycle, SDP uses the probabilities of a drive cycle to develop a control law based on what is likely to happen in the future. First, the transition probabilities must be calculated, and then the SDP method is applied to develop a lookup table based on the states of the system.

4.3.1 Transition Probabilities

The transition probabilities are calculated using the following steps.

1. Read in drive cycles and combine into one drive cycle. For this study, the Urban Dynamometer Driving Schedule (UDDS), West Virginia Interstate Driving Schedule, West Virginia Suburban Driving Schedule, and the West Virginia City Driving Schedule are used. The combined drive cycle is shown in Figure 4.1.

2. Calculate the acceleration at each time step using backwards difference.

3. Create discretized acceleration values. For this study, a vector of 20 uniformly spaced values from the minimum acceleration to the maximum acceleration of the drive cycle is used. The value closest to zero is set equal to zero.

4. Create discretized speed values. For this study, a vector of 20 uniformly spaced values from zero to the maximum speed of the drive cycle is used.

5. Find discretized speed values of the drive cycle by rounding the drive cycle speeds at each time step to the nearest discretized speed.

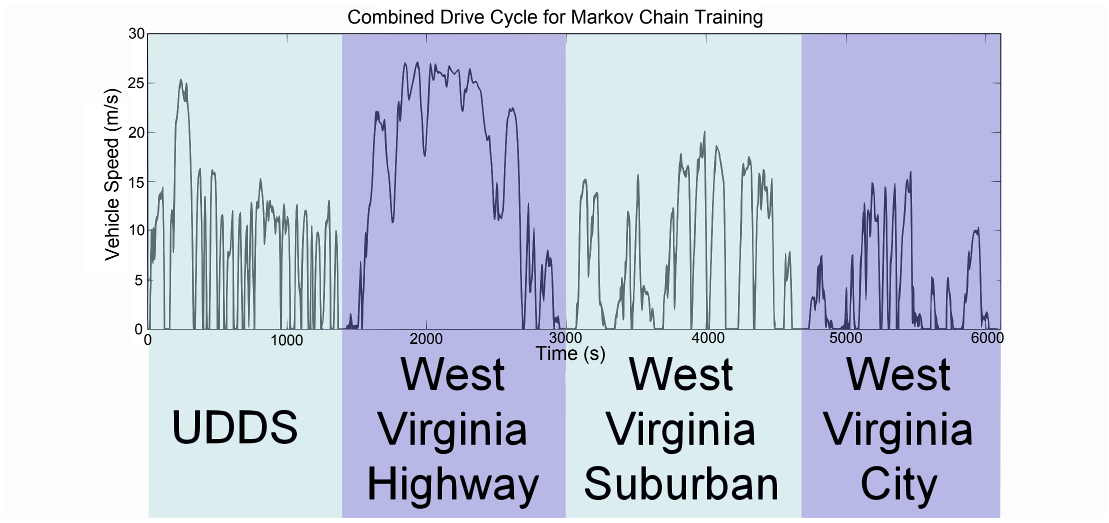


Figure 4.1: Combined drive cycle for Markov chain

6. Find discretized acceleration values of the drive cycle by rounding the drive cycle accelerations at each time step to the nearest discretized acceleration.

7. Calculate the transition probabilities from acceleration at the current time step to acceleration at the next time step for each discretized speed. The transition probabilities are calculated by the following.

   (a) Find all the points in the drive cycle that equal the discretized speed being investigated.

   (b) Record the accelerations at those time steps and the accelerations at the next time step.

   (c) Count the number of times each current acceleration and next acceleration occur in the drive cycle and store in a matrix. In the matrix, the rows correspond to current acceleration and columns correspond to next acceleration.

   (d) Sum the number of times each current acceleration occurs in the drive cycle.

   (e) Calculate the transition probabilities by taking the counts and dividing by the total number of times that acceleration occurs in the drive cycle. Mathematically, this is shown in Equation (4.4).

$$Pr_{i,j,q} = \frac{C_{i,j,q}}{\sum_{q=1}^{N_a} C_{i,j,q}} \tag{4.4}$$

(f) If the current acceleration never occurs in the drive cycle for the current velocity, set the probability to zero.

(g) Repeat for each discretized speed.

An example plot of transition probabilities at a given velocity is shown in Figure 4.2. As shown in this plot, the highest probabilities lie on the diagonal. According to this, if the vehicle is accelerating at a certain rate at the current time step, the probability of accelerating at the same rate during the next time step is high. Also, notice from this plot that there are some accelerations (e.g high deceleration rates) that never occur in the drive cycle and therefore have a probability of zero.

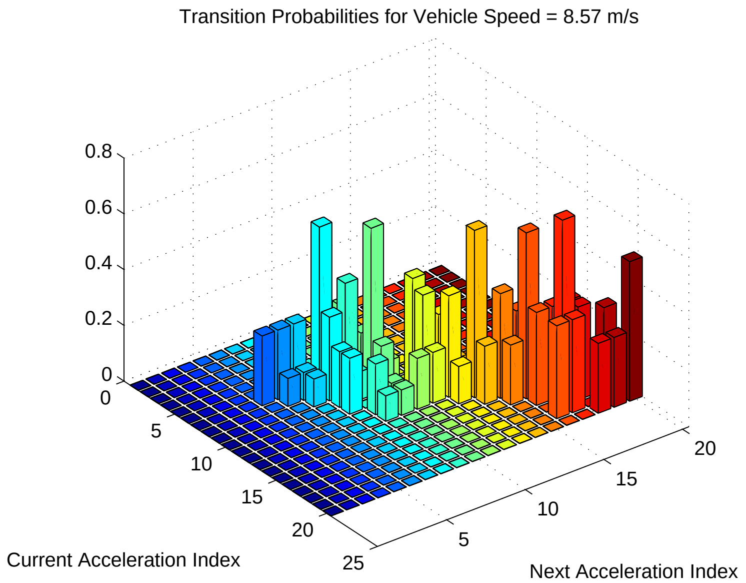


Figure 4.2: Transition probabilities for a given velocity

Once the transition probabilities are found, SDP is used to find the control law in the form of a lookup table. Two general versions of SDP exist - value iteration and policy iteration [71]. In value iteration, the value for each state is set to an initial value. By using the Bellman equation and the transition probability, new values are calculated until the difference between the values is within some tolerance. Even though this method does not require solving a set of linear equations, the values can get very high and is not guaranteed to converge for large problems. In policy iteration, an arbitrary control policy is used initially, and the value function is used to solve a set of linear equations to find a new policy until the policy does not change. Even though it requires more computational power than the value iteration since a set of linear equations need to be solved, this method is used since it will converge to a solution. For this study, the discounted policy iteration method is used.

4.3.2 Discounted Policy Iteration

The discounted policy iteration method starts with an initial policy and iterates until the solution converges. The first step is to calculate the average cost ($\bar{c}$) at each state (o), which is given by Equation (4.5).

$$\bar{c}(o, \mu_l(o)) = \sum_{n=1}^{S} Pr(o, \mu_l(o), n) c(o, \mu_l(o), n) \tag{4.5}$$

In the above equation, Pr is the probability of going from state o to state n using control policy $\mu_l(o)$, c is the cost to go from state o to state n using control policy $\mu_l(o)$, and S is the total number of states.

The cost function c is the fuel consumption to go from state o to state n using control policy μ_l at state o. The fuel consumption is a function of engine speed and torque.

$$c(o, \mu_l(o), n) = \text{fuel}(\omega_{Eng}, T_{Eng}) \tag{4.6}$$

Once the average cost at each state is calculated, the next step is policy evaluation, which evaluates the current policy and finds the value function. This is given by Equation (4.7).

$$Va_l(o) = \bar{c}(o, \mu_l(o)) + \lambda \sum_{n=1}^{S} Pr(o, \mu_l(o), n) Va_l(n) \tag{4.7}$$

The discount factor, λ, is less than 1. The meaning of the discount factor is that future costs do not matter as much as the same costs incurred at the present time. The lower the discount factor, the lower the importance of the future costs. $Va_l(o)$ is the value function for state o using control policy $\mu_l(o)$. The value function is solved at each state using the set of linear equations given by Equation (4.7). Once the value function at each state is known, the policy improvement step is performed according to Equation (4.8).

$$\mu_{l+1}(o) \in \arg\min_{u \in U(o)} \left[\bar{c}(o, \mu_l(o)) + \lambda \sum_{n=1}^{S} Pr(o, \mu_l(o), n) Va_l(n) \right] \tag{4.8}$$

The policy evaluation and policy improvement steps repeat until the policy for each state between iterations is the same, which is the optimal policy. The end result is a lookup table which inputs the current vehicle velocity, acceleration, and accumulator oil volume and outputs the control decision.

4.4 Random Drive Cycle Generator

The transition probabilities can also be used to develop random drive cycles that can be used for further studies. There are two components to creating the random drive cycles. The first is, if the vehicle is stopped, whether it should stay stopped or accelerate. The probability of the vehicle staying stopped is determined by taking the number of times the vehicle stays stopped and dividing it by the total number of time steps the vehicle is stopped. A uniform random number generator is then used to generate a number between 0 and 1. If the random number is less than that probability, the vehicle stays stopped. If it is above that probability, the vehicle accelerates.

To determine the acceleration, the transition probabilities are used. For a given speed and current acceleration, the probabilities for the next acceleration are known. A uniform random number between 0 and 1 is generated and compared to where it falls in the probabilities for acceleration. The acceleration that has the cumulative probability closest to the random number is chosen. This is repeated for the length of the drive cycle.

In order for the random drive cycle to be representative of typical urban driving, a few guidelines are made.

1. The drive cycle must have a length of at least 20 minutes. This is approximately the average time it takes to drive to work in the United States [72].
2. If after 20 minutes the drive cycle does not have zero speed, the algorithm continues until zero speed is reached.
3. When stopped, the vehicle must remain stopped for at least 5 seconds. This is to simulate stop-and-go traffic in the city. If the random drive cycle does not meet this criteria, it is not used.
4. The drive cycle cannot have a speed of 20 m/s for more than 100 seconds. The purpose of the drive cycle generator is to simulate city driving and not highway driving. If the drive cycle does not meet this criteria, it is not used.

An example of a random drive cycle created using the method described above is shown in Figure 4.3.

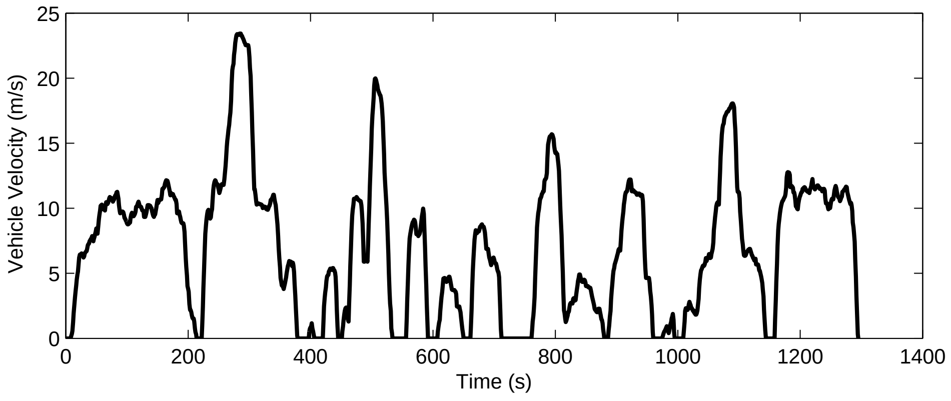


Figure 4.3: Example of a random drive cycle

4.5 Model Predictive Control

Model predictive control (MPC) is also used to develop an energy management strategy for the AEVPS system [73]. This is a different approach because, unlike the rule-based and SDP strategies, this is an online optimization method that requires no advanced knowledge of the duty cycle. Instead, a finite horizon is used for the prediction and no future information about the drive cycle is used.

The MPC strategy uses a discrete linear model for prediction and the objective function. The controller solves for the sequence of throttle commands, pump swashplate angles, and valve commands which minimizes the objective function over the specified horizon and then applies the first element of this sequence to the system. System operation is divided into two modes - engine on and engine idling - and a supervisory logic is used to determine when to switch between the modes. A dwell time constraint is included in the supervisory controller to reduce the frequency of the system switching between the two modes. The objective for engine on mode is composed of three parts - ensure motor tracking to the desired reference speed, operate the pump efficiently, and operate the throttling valve efficiently. The weighting factors for each are found by minimizing the fuel consumption over the UDDS, which are found to be $\zeta_1 = 1 \times 10^{-4}$ and $\zeta_2 = 9 \times 10^{-4}$. This objective function is shown in Equation (4.9).

$$J_1 = \sum_{z=1}^{K} \sum_{y=1}^{z} \left[\left(\frac{\omega_z(y) - \omega_{z,des}}{\omega_{z,max}} \right)^2 + \zeta_1 C_1 + \zeta_2 C_2 \right] \tag{4.9}$$

$$C_1 = \left(\frac{\omega_{Eng}(y) - \omega_{Eng,des1}}{\omega_{Eng,max}} \right)^2 + \left(\frac{u_{Eng}(y) - u_{Eng,des1}}{u_{Eng,max}} \right)^2 \tag{4.10}$$

$$C_2 = \left(\frac{P_u(y) - P_{u,des}}{P_{u,max}} \right)^2 \tag{4.11}$$

Engine idling mode is possible when the accumulator can supply the requested power or the motor speed is decelerating. The objective for this mode is given by Equation (4.12). To ensure the motor tracks the reference speed accurately, this is given a higher priority and ζ_3 is set to 1×10^{-3}.

$$J_2 = \sum_{z=1}^{K} \sum_{y=1}^{z} \left[\left(\frac{\omega_z(y) - \omega_{z,des}}{\omega_{z,max}} \right)^2 + \zeta_3 C_3 \right] \tag{4.12}$$

$$C_3 = \left(\frac{\omega_{Eng}(y) - \omega_{Eng,des2}}{\omega_{Eng,max}} \right)^2 + \left(\frac{u_{Eng}(y) - u_{Eng,des2}}{u_{Eng,max}} \right)^2 \tag{4.13}$$

4.6 Conclusions

In this chapter, the optimization methods of dynamic programming and stochastic dynamic programming were explained. Since the dynamic programming solution cannot be directly implemented on a real-time system, a method for developing a rule-based strategy from the dynamic programming results is also presented. In the next chapter, these methods will be used to develop energy management strategies for the hydraulic hybrid vehicle and AEVPS systems, and simulation results for each will be presented.

Chapter 5

Simulation Results

5.1 Urban Dynamometer Driving Schedule (UDDS)

For the development of the energy management strategies and the simulations, the Urban Dynamometer Driving Schedule (UDDS) is used as the duty cycle [74]. This is the duty cycle specified by the United States Environmental Protection Agency for fuel economy for light duty vehicles. While it mostly represents city driving, a highway speed event occurs at the beginning of the drive cycle. The UDDS is shown in Figure 5.1.

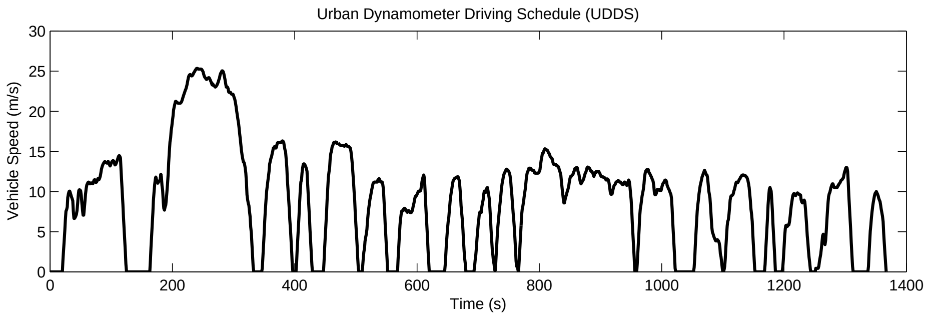


Figure 5.1: Urban Dynamometer Driving Schedule

5.2 Baseline Control Strategy

For the baseline simulation, control parameters were chosen using physical intuition about the system. One of the control parameters, engine idle, is a discrete state, while the other control parameter, engine power, is a continuous state. To determine the engine idle state, a constant lower and upper setpoint were set based on pressure. When the pressure is below the lower setpoint, the engine will turn on, and when the pressure is above the upper setpoint, the engine will turn off and the vehicle will be solely powered by the hydraulics. When the pressure is between the setpoints, the strategy uses the previous idle state. For this baseline study, the lower setpoint is set to 2500 psi, and the upper setpoint is set to 3500 psi.

To determine the engine power when the engine is not idling, the engine operates near its most efficient operating point. Looking at Figure 3.5, the most efficient point of approximately 33% is operating at a speed of 1800 rpm with a torque of 85 Nm.

When this baseline strategy was tried, no feasible solution was found to complete the urban drive cycle. During periods of high acceleration, the accumulator would empty, and the engine was not able to provide enough power to accelerate the vehicle to highway speed in the time specified. To account for this, two checks were made in the control strategy. The first is, if the solution was not feasible at the current time step, the engine idle state was changed and tried again. If this still did not yield a feasible solution, the engine operates at a higher power level, which was chosen to be an engine speed of 2500 rpm and torque of 85 Nm. Figure 5.2 shows the flowchart for the final baseline control strategy.

Figure 5.3 shows the results from this baseline strategy. As can be seen, most of the time the engine is operating at the most efficient point of 1800 rpm. When the vehicle is at highway speed and the accumulator pressure is low, the engine goes into the higher power mode. A few spikes in engine power also exist when the engine is switching from idle mode to power mode to provide power when the accumulator is near the lower constraint. The fuel economy using this strategy is 30.5 mpg.

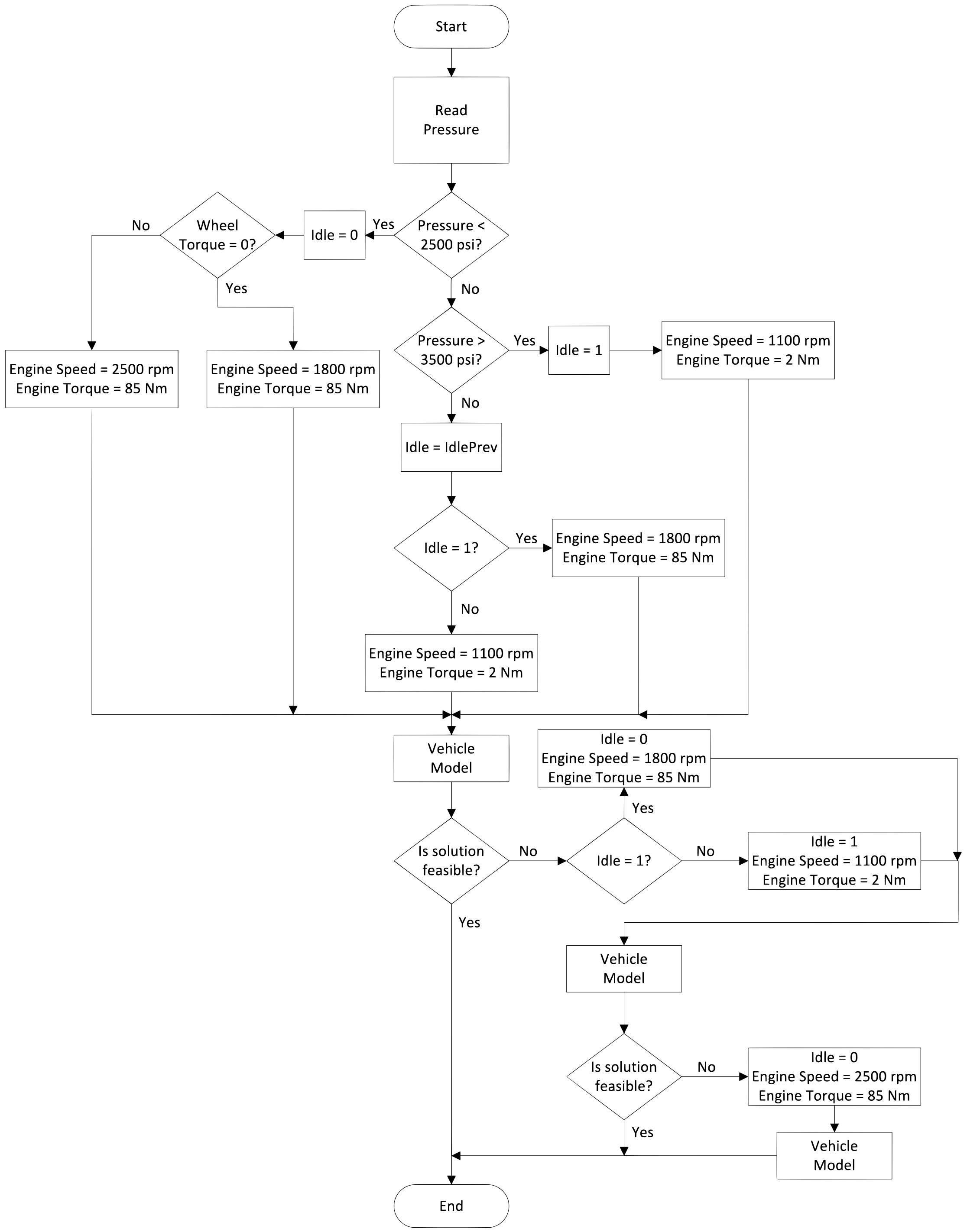


Figure 5.2: Flowchart for baseline control strategy

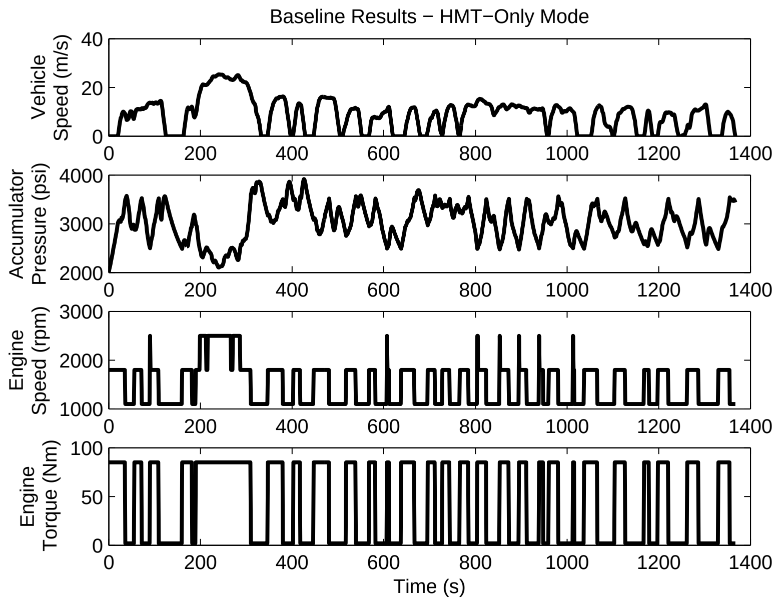


Figure 5.3: Results for baseline control strategy

5.3 Hydraulic Hybrid Vehicle Simulation Results

5.3.1 Dynamic Programming (DP) Results

For these simulations, the discrete time, discrete space DP algorithm was used. This requires discretizing time, state variables, and control variables. Time was discretized using 1 second time increments. The state variable is the oil volume in the accumulator, which is related to the pressure by Equation (3.22). This is discretized in 1×10^{-4} m^3 increments. The control variable is the engine speed, which is discretized in 10 rpm increments.

This method also makes it easy to add constraints to the state and control variables. The accumulator pressure must remain between 2000-4000 psi. By using this constraint and the grid spacing, accumulator oil volume is discretized into 152 points. Also, the engine must operate between 1400-2600 rpm, discretizing the engine speed into 121 points. The engine idling is also simulated at each time step for each state. At each time step, 18544 simulations are run, and linear interpolation is used to determine the next state and fuel used.

Two different mid-level controllers are used to determine engine power. The first is using a constant engine torque at all engine speeds. The second is to operate the engine along the best brake specific fuel consumption (BSFC) curve.

Constant Engine Torque

For these simulations, a constant engine torque of 85 Nm is used for all engine speeds. This strategy is a good starting point to see how much improvement can be made over the baseline strategy. One disadvantage is, even when the engine is running at idle, it still must be clutched to the drivetrain to keep the control strategy simple. This is included in all the simulations. Figure 5.4 shows the dynamic programming solution using this strategy.

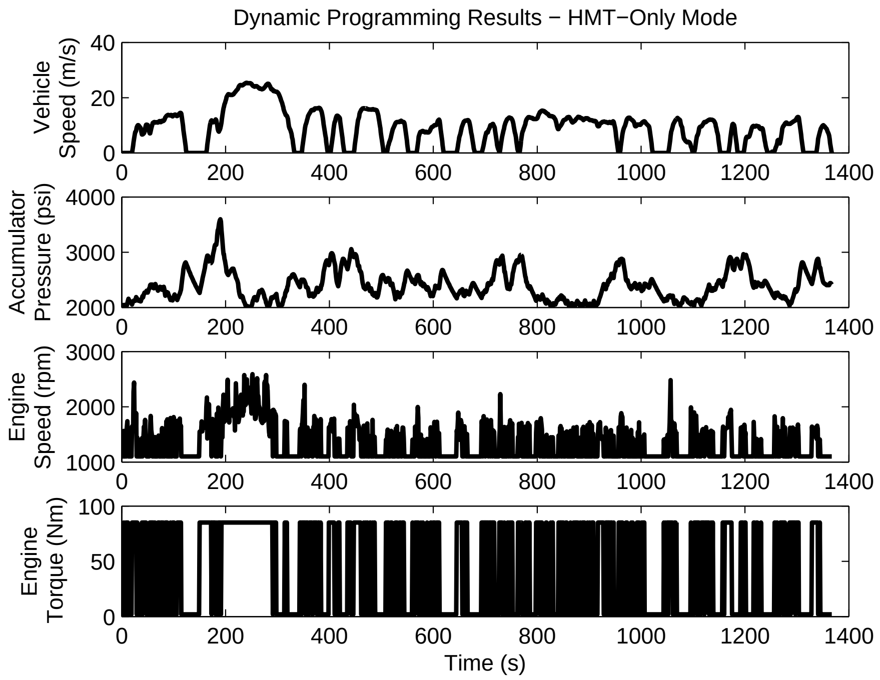


Figure 5.4: Dynamic programming solution with constant engine torque

Looking at the results in Figure 5.4, a few observations are made. First, the accumulator pressure is kept relatively low and never reaches the upper limit of 4000 psi. Secondly, when the vehicle is at highway speeds, the engine speed increases to provide more power, but otherwise the engine operates between 1600-2000 rpm the majority of the time. This is similar to the results from the baseline strategy. Finally, by looking at the engine torque plot, the engine is turning on and off very rapidly, which would not be feasible in an actual vehicle.

Figure 5.5 shows the engine operating points for this strategy when the engine is not idling.

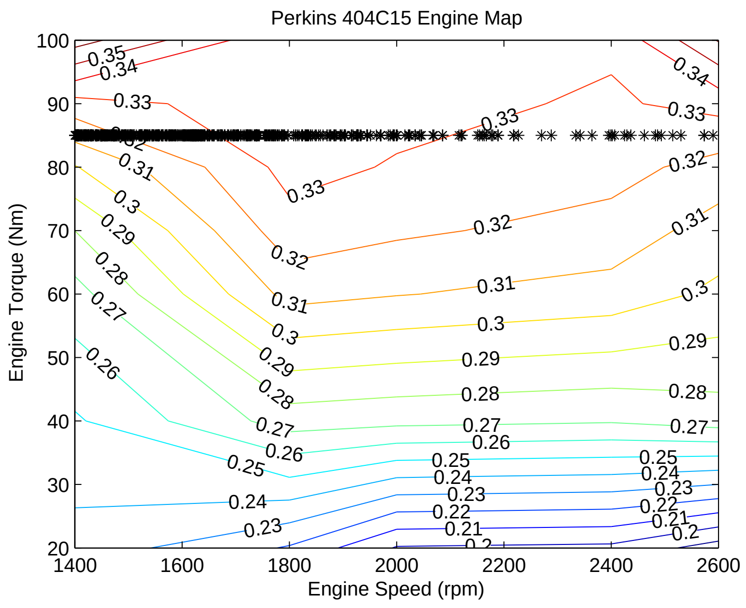


Figure 5.5: Engine operating points with constant engine torque

As seen in Figure 5.5, the engine very rarely operates above 2200 rpm. This only happens at high vehicle speed when the accumulator pressure is low and more power is needed. Also, even though numerous points lie within the most efficient region of the engine, there is also a cluster of points operating at engine speeds below this region. This is to keep the engine power low to put as much power through the hydraulics as possible to increase the efficiency of the overall system. Even though the engine is not always operating at its most efficient region, all operating points are at an engine efficiency of at least 31%. The fuel economy using this control strategy is 36.8 mpg, which is an improvement of 20% over the baseline strategy. However, as noted earlier, this would not be feasible due to the high frequency of engine going between idling and providing power.

Constant Engine Torque With Engine Penalty

To reduce the frequency of the engine idling and providing power, a penalty is added each time the idling mode at the current time step does not match the idling mode of the next time step. This is shown in Equation (5.1).

$$L(x,u) = \begin{cases} L(x,u) + Pe_{eng} & \text{Idle(k+1)} \neq \text{Idle(k)} \\ L(x,u) & \text{otherwise} \end{cases} \tag{5.1}$$

This becomes an optimization problem within an optimization problem. Different values for the engine penalty are tried until a reasonable result is obtained. If the engine penalty is too low, the engine will continue to frequently change between idling and providing power, where if the engine penalty is too large the engine will rarely switch to idling mode. An engine penalty of 10 was found to give a solution where the engine switched between idling and providing power at a reasonable frequency. Figure 5.6 shows the results from using this strategy.

As seen in the engine torque plot in Figure 5.6, the engine switches between idling and providing power approximately every 30 seconds. One difference between this strategy and the strategy without the engine penalty is the pressure in the accumulator now reaches the upper limit three times during the drive cycle. However, the majority of the time, the accumulator is well below the upper limit. The engine also operates similarly to that with no engine penalty.

Figure 5.7 shows the engine operating points for this strategy. As seen in this figure, the operating points are very similar to those shown in Figure 5.5. However, a few slight differences exist. First, more operating points are at a higher engine speed even though the engine only operates in this region at a high vehicle speed. Also, the operating points are spread more evenly at engine speeds below 2000 rpm than before. However, numerous points still lie at speeds below the most efficient region of the engine to keep the efficiency of the hydraulic units as high as possible. With this strategy, the engine is at idle for 738 seconds out of the 1368 seconds of the drive cycle, slightly over 54% of the drive cycle., while the strategy without the engine penalty is at idle for 793 seconds

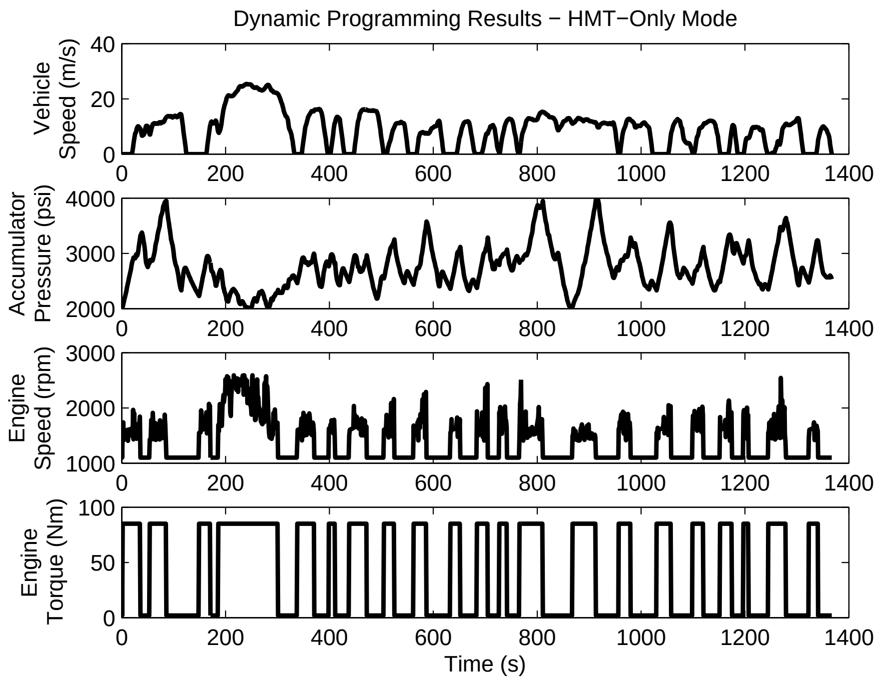


Figure 5.6: Dynamic programming solution with constant engine torque and engine penalty

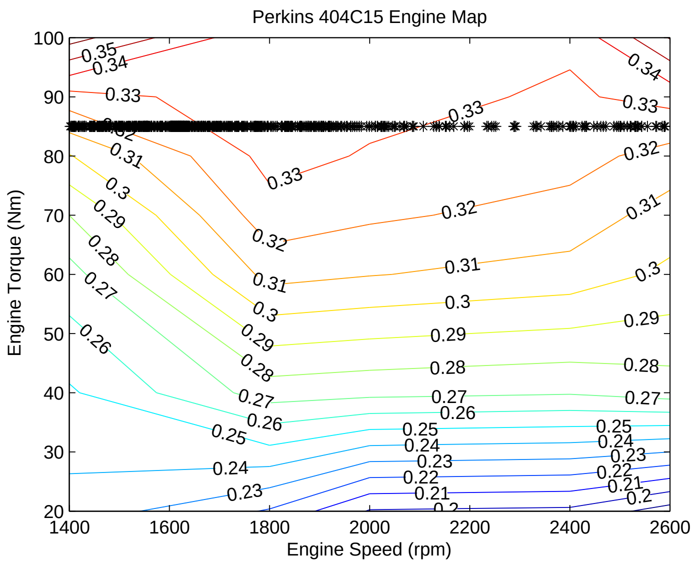


Figure 5.7: Engine operating points with constant engine torque and engine penalty

for 58% of the drive cycle. The fuel economy with this strategy is about 32.8 mpg for an improvement of 7.5% over the baseline strategy.

Best BSFC Curve

Rather than using a constant engine torque, these simulations operate along the best BSFC curve. To develop this curve, engine power is discretized from 2-24 kW in 1 kW increments. For each power level, torque is discretized from 5-85 Nm in 1 Nm increments. The corresponding engine speed is calculated by dividing the power by the torque. If the calculated engine speed does not fall in the feasible operating range of the engine (i.e. below 1000 rpm or above 2600 rpm), the data point is not included. For the remaining data points, the losses for each engine torque and speed are found, and the minimum loss point is recorded for that power level. The process is repeated for each power level. The best BSFC curve is shown is Figure 5.8.

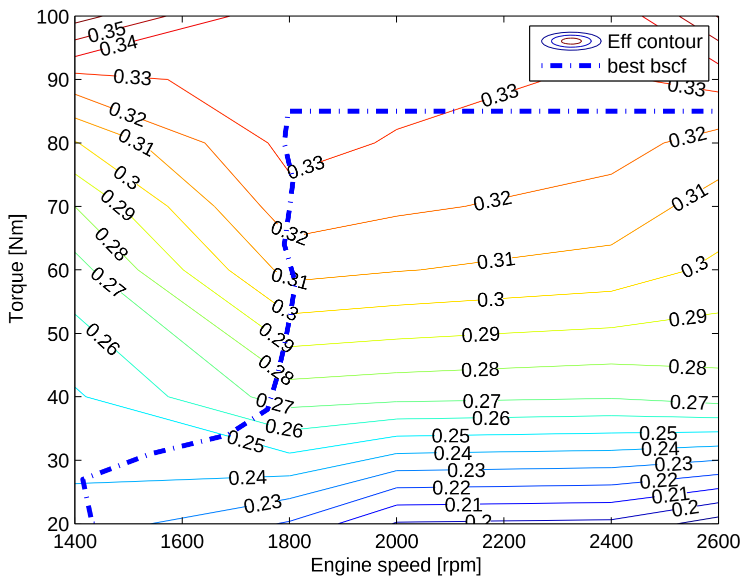


Figure 5.8: Best BSFC curve for 404C15 engine

With the best BSFC curve identified, this can be used with the DP algorithm by varying the engine power and using the corresponding engine speed and torque in the simulations. The results are shown in Figure 5.9.

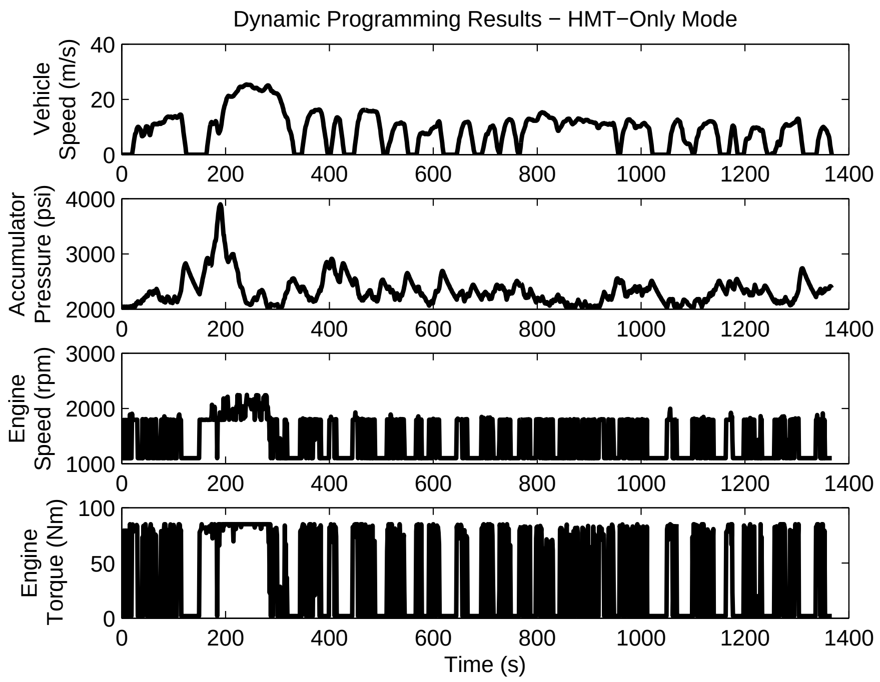


Figure 5.9: Dynamic programming solution with best BSFC curve

Comparing Figure 5.9 to the constant engine torque solution from Figure 5.4, many similarities exist. First, the pressure is maintained at a low level, even lower than the constant torque solution. Also, the frequency of the engine switching between idling and power mode is very rapid and not feasible on a physical system. Finally, except for high vehicle speeds, the engine is operated near 1800 rpm, which corresponds to the most efficient region of the engine.

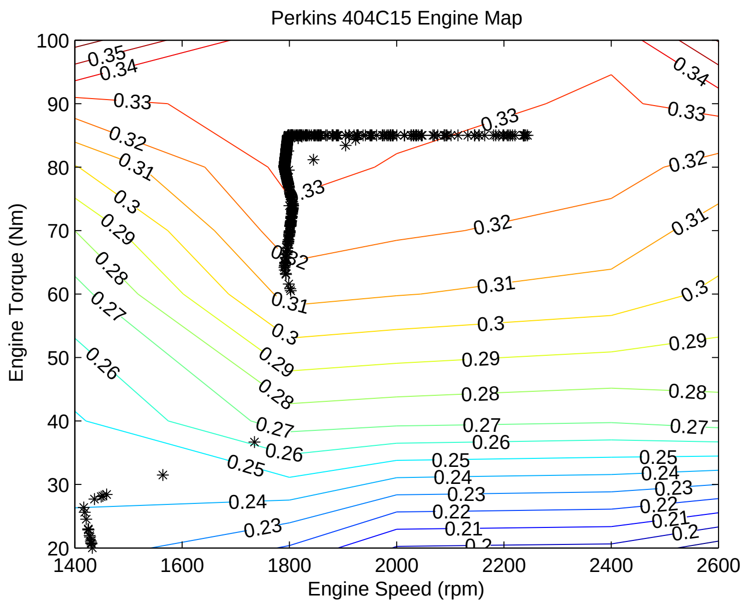


Figure 5.10: Engine operating points with best BSFC curve

Figure 5.10 shows the engine operating points. As shown in this figure, the engine never operates above 2300 rpm. Also, only a small grouping of points exist at low power levels. The majority of operating points are found in the most efficient region of the engine where the efficiency is above 31%. The fuel economy using this control strategy is 36.6 mpg, which is very close to the constant engine torque solution. However, a penalty must be added just like in the constant torque case to reduce the frequency of engine switching.

Best BSFC Curve with Engine Penalty

Just as in the constant torque case, an engine penalty must be applied to reduce the frequency of engine switching between idling and power mode. Equation (5.1) is again used to apply the engine penalty. However, when a penalty of 10 is used as in the constant torque case, the engine never idles. Therefore, a lower engine penalty of 1 is used to find a balance between too frequent of switching and no switching.

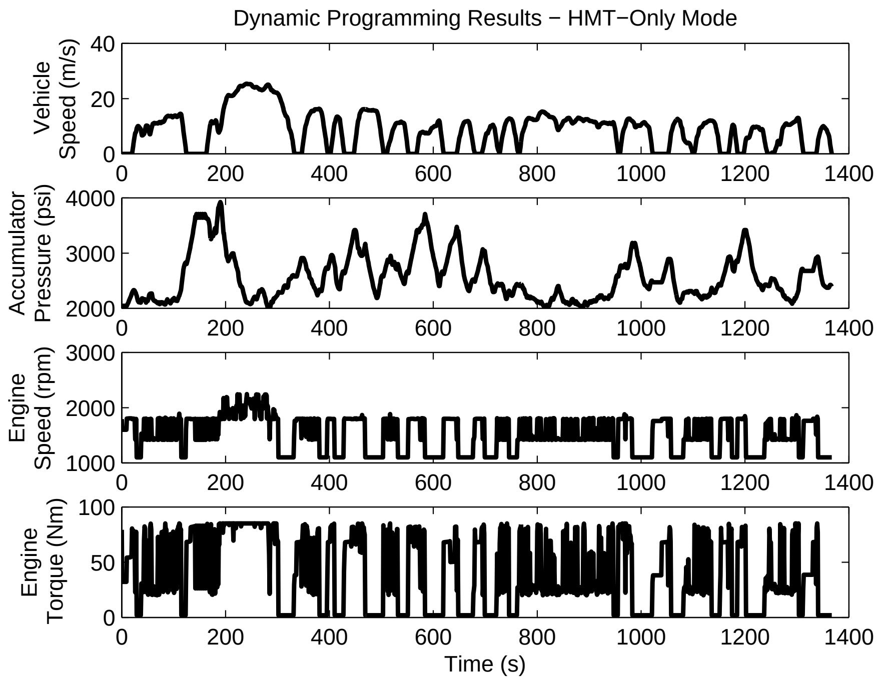


Figure 5.11: Dynamic programming solution with best BSFC curve and engine penalty

The results for this simulation are shown in Figure 5.11. As can be seen in this figure, the pressure in the accumulator is higher, but still does not reach the upper constraint. The engine idles for about 30 seconds at a time before it goes into power mode and operates near 1800 rpm. However, now the engine speed is fluctuating very rapidly, which would not be feasible physically.

Figure 5.12 shows the engine operating points. As shown in this figure, the engine still does not operate above 2300 rpm. However, more of the engine map is being used. More points are now at low power levels and fewer points are in the most efficient region.

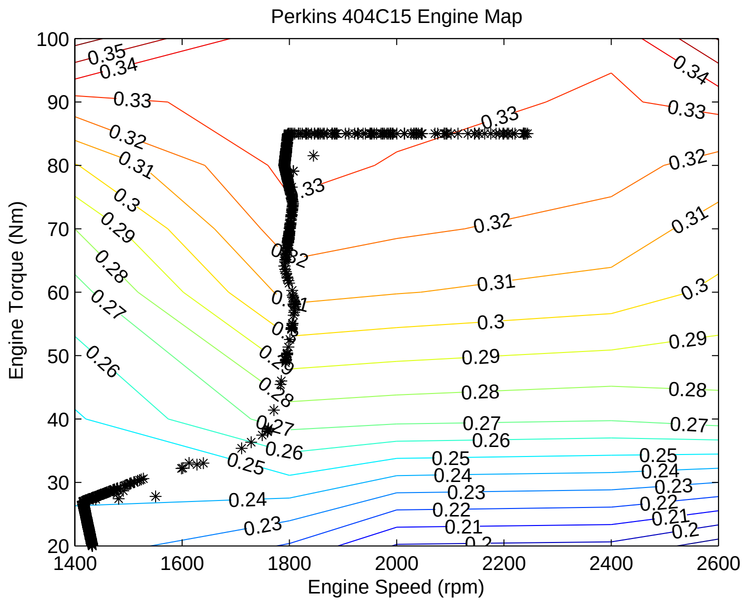


Figure 5.12: Engine operating points with best BSFC curve and engine penalty

Also, the engine is operating at mid-power levels, whereas without the engine penalty the engine never operated in this region. The fuel economy from this simulation is 32.6 mpg, which is again approximately the same as the constant torque case with engine penalty. However, the speed fluctuations must be minimized.

Best BSFC curve with engine penalty and engine speed weighting

To minimize the rapid fluctuations in the engine speed, a weighting factor is added to the difference in engine speed. This is shown in Equation (5.2)

$$L(x,u) = \begin{cases} L(x,u) + Pe_{eng} & \text{Idle(k+1)} \neq \text{Idle(k)} \\ L(x,u) + \epsilon\left(\omega_{Eng}(k+1) - \omega_{Eng}(k)\right) & \text{otherwise} \end{cases} \tag{5.2}$$

As with the engine speed penalty, if the weighting factor on engine speed is set too high, the engine speed will stay constant, and if set too low, the engine speed will fluctuate too much. A value of 0.0005 for the weighting factor is a good balance.

The results are shown in Figure 5.13. As shown in this figure, the accumulator

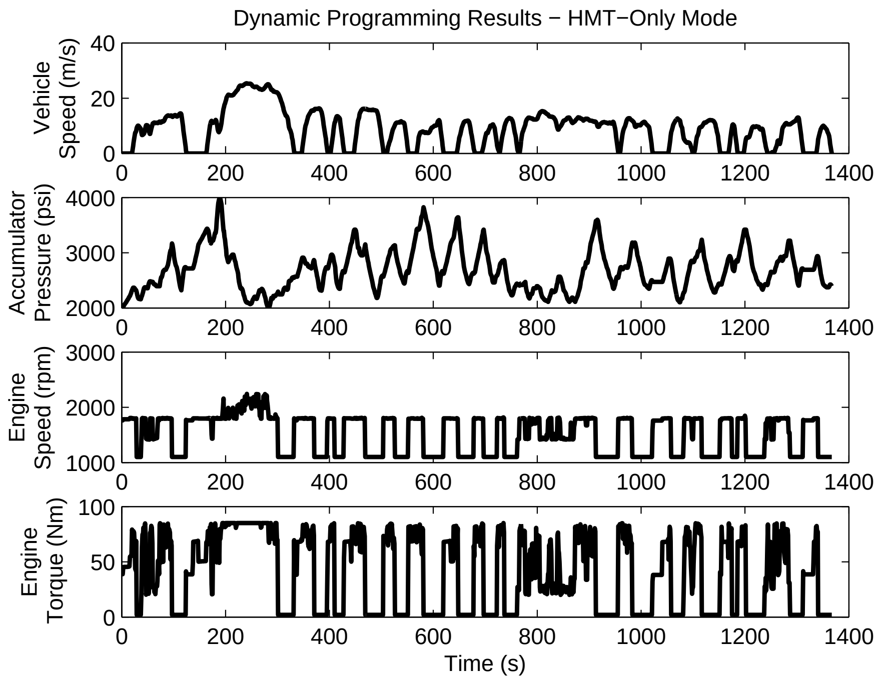


Figure 5.13: Dynamic programming solution with best BSFC curve, engine penalty, and engine speed weighting

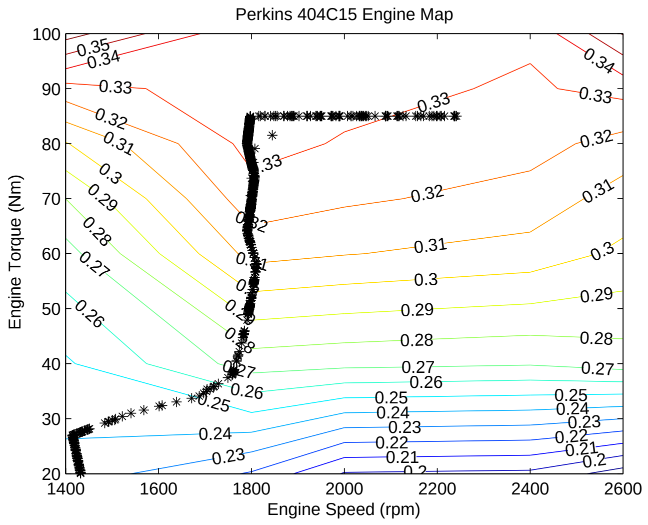


Figure 5.14: Engine operating points for HMT only with best BSFC curve, engine penalty, and engine speed weighting

pressure is similar to the engine penalty results shown in Figure 5.11, but at a slightly higher pressure. Also, the engine speed is similar, but the fluctuations in the engine speed have been reduced. Also, the engine is operating at 1800 rpm for the majority of the time except at high vehicle velocities, matching the results from the previous simulations. However, even though the engine speed is constant, the engine power is varying as shown in the engine torque plot.

Figure 5.14 shows the engine operating points. The engine operating points are very similar to the engine penalty operating points shown in Figure 5.12. The engine still does not operate above 2300 rpm, but now the engine operates at all power levels along the best BSFC curve. The fuel economy for this simulation is approximately 32 mpg, which is approximately a 5% increase over the baseline control strategy.

This control strategy is very similar to the baseline control strategy, so it is expected that the fuel economy improvement would not be that great. In both cases, the engine speed operates near 1800 rpm when the engine is not idling, and the speed increases when the vehicle speed is high and the accumulator pressure is low. However, this strategy obtains a slightly better fuel economy than the baseline strategy. The main difference is that the baseline strategy is constrained to operate at the engine's most efficient point, whereas the best BSFC curve strategy is allowed to move away from this point. This shows that operating the engine as efficiently as possible does not necessarily mean the system will operate as efficiently as possible. The reason for this is that the difference in operating the engine away from its most efficient point is less than operating the hydraulic units in their inefficient region.

5.3.2 Rule-Based Strategy Results

For the rule-based strategy, the constant engine torque mid-level control is used. This is a simpler strategy that achieved a slightly better fuel economy than the best BSFC curve strategy. As stated in Chapter 4, the rule-based strategy consists of two parts: A discrete part for engine idling and a continuous part for how much the power the engine should provide if the engine is not idling. Looking at the DP results in Figure 5.6, the lower threshold is set to 2500 psi - if the accumulator pressure drops below this pressure, the engine will supplement the hydraulics to provide power. The upper threshold is set to 3900 psi - above this pressure the engine will idle.

When the pressure is between 2500-3900 psi, the desired wheel torque is used to determine the engine idle state. In the UDDS, the wheel torque is negative for 320 time steps, which signifies braking events. Out of those times, the engine is providing power for only 9 time steps. Therefore, when the vehicle is braking, the engine is turned off as long as the pressure is greater than 2500 psi. The other extreme is when a high torque demand is required. For this study, high torque demand is defined as 300 Nm and greater. This occurs for 139 time steps in the drive cycle. From the DP results, the engine is providing power for all 139 time steps. Therefore, the engine does not idle during these high torque demand events. For the standing/small torque demand times, the previous engine idling mode is used.

If the engine is not idling, the amount of power the engine provides must be determined. Since the torque is fixed at a constant 85 Nm, the engine speed is the only variable that determines the engine power. To determine the engine speed, a curve fit is performed with wheel speed and wheel torque as the dependent variables. It was found that the engine speed depended more on the wheel speed than the wheel torque. The curve fit equation is given by Equation (5.3).

$$\begin{aligned}\omega_{Eng} = \frac{\pi}{30}(1622 + 9.977 * \omega_w + 0.3211 * T_w - 0.4402 * \omega_w^2 - \\ 0.02748 * T_w * \omega_w + 0.004245 * \omega_w^3 + 0.0008076 * \omega_w^2 * T_w)\end{aligned} \tag{5.3}$$

This gives an r^2 of 0.53. Even though this seems low, obtaining a higher r^2 value is very difficult due to the rapid fluctuations in the engine speed from the DP results.

Figure 5.15 shows the flowchart for the rule-based strategy.

Simulation

The rule-based strategy is simulated using the backwards-facing model to compare to the DP results. Comparing the rule-based strategy in Figure 5.16 to the DP solution in Figure 5.6, many similarities are seen. First, the accumulator pressure reaches the upper constraint a few times in the drive cycle. However, throughout the majority of the drive cycle, the pressure in both cases averages around 3000 psi. Also, the engine operation is about the same, with the DP results having slightly longer idling times. At the beginning of the drive cycle, some fluctuations are seen between engine idling and

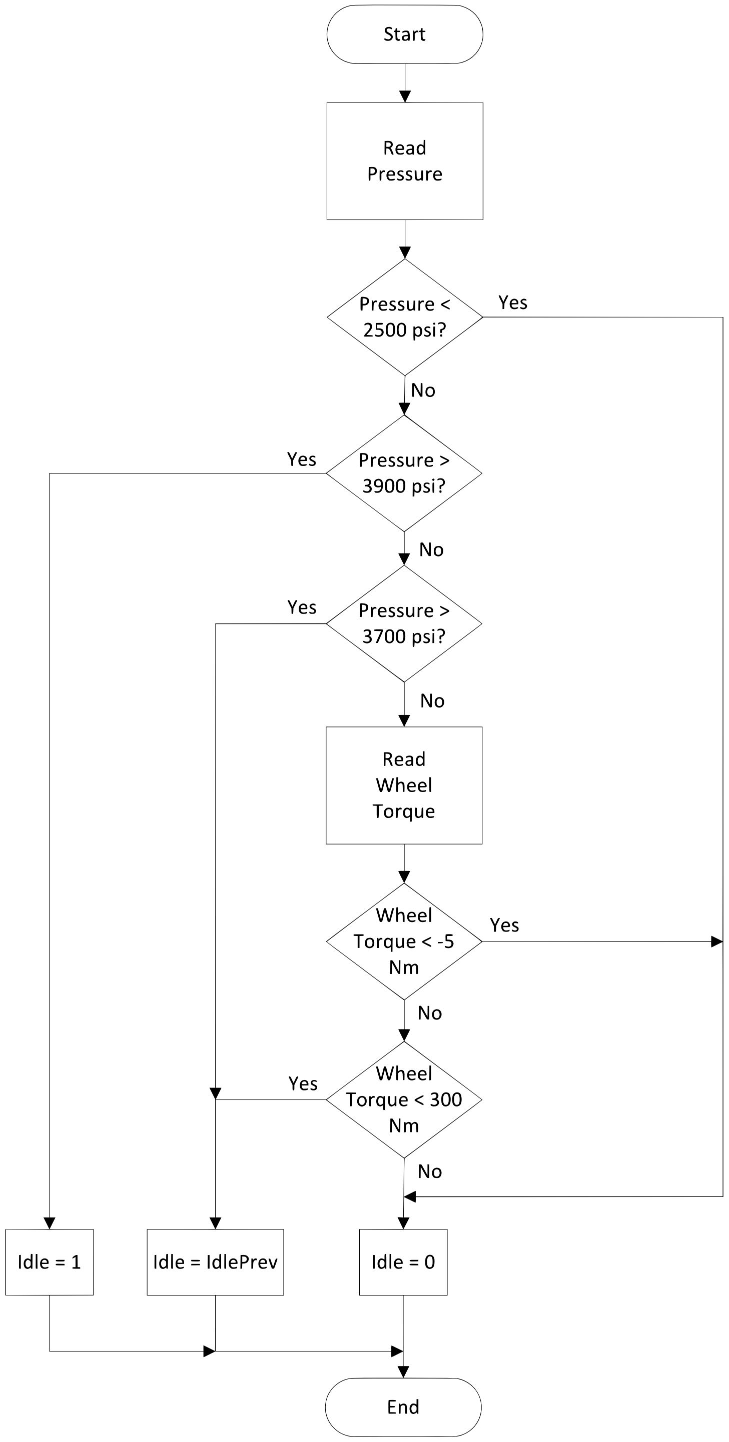


Figure 5.15: Flowchart for rule-based control strategy using constant engine torque

power modes. This occurs because the pressure is around the lower setpoint of 2500 psi and braking events are included, so it is fluctuating between keeping the pressure above 2500 psi and turning the engine off for braking.

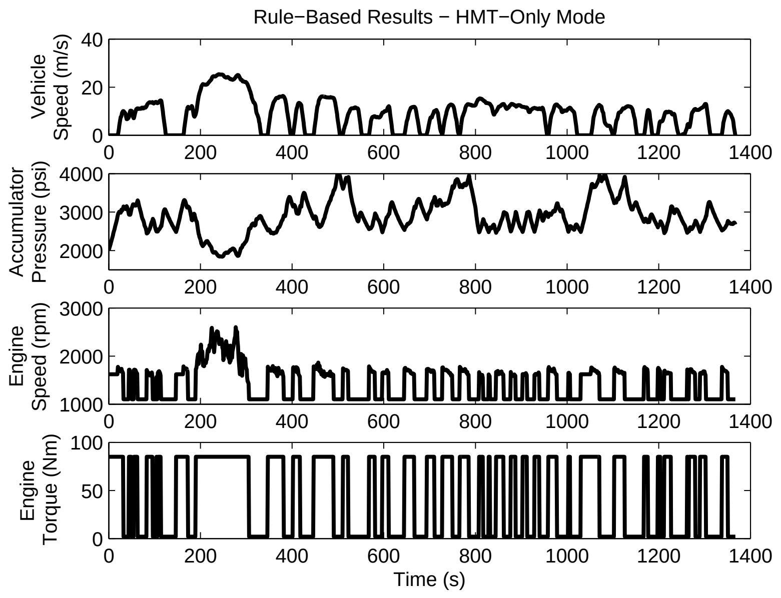


Figure 5.16: Results for rule-based strategy

Figure 5.17 shows the engine operating points for the rule-based strategy. Comparing this to the engine operating points for the DP solution in Figure 5.7, the operation above 2000 rpm is very similar. Also, the majority of the operating points are at a speed below the most efficient region similar to the DP results. However, unlike the DP results, the engine very rarely operates below 1550 rpm due to the curve fit that was used.

Figure 5.18 shows a comparison of non-idle engine operating points between the DP results and the rule-based results. The DP points are scattered and therefore makes it difficult to find a good fit. However, the fit at high vehicle speeds is very good, and the majority of the curve fit points lie where there are clusters of points from the DP solution, which makes this a good fit for the data.

The fuel economy obtained using the rule-based strategy is 32.4 mpg, which is a difference of about 1% from the DP solution. From these results, it can be concluded that this rule-based strategy is a good match for the UDDS duty cycle.

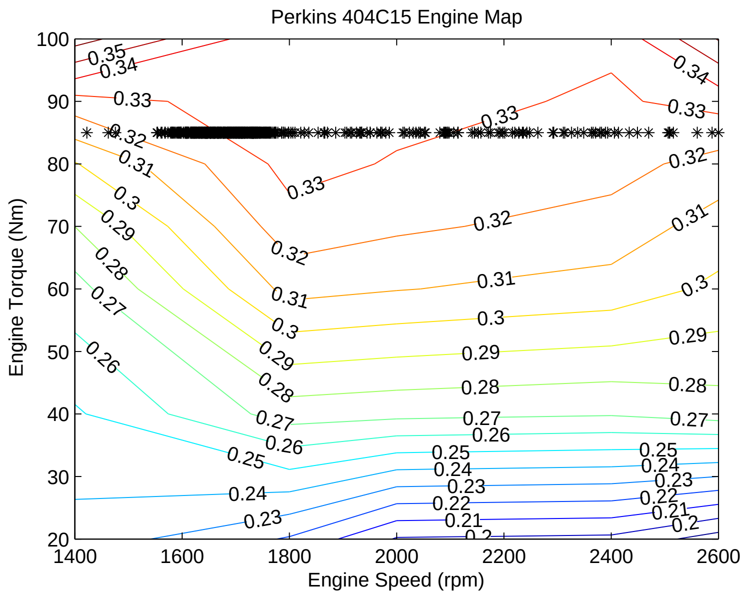


Figure 5.17: Engine operating points for rule-based strategy

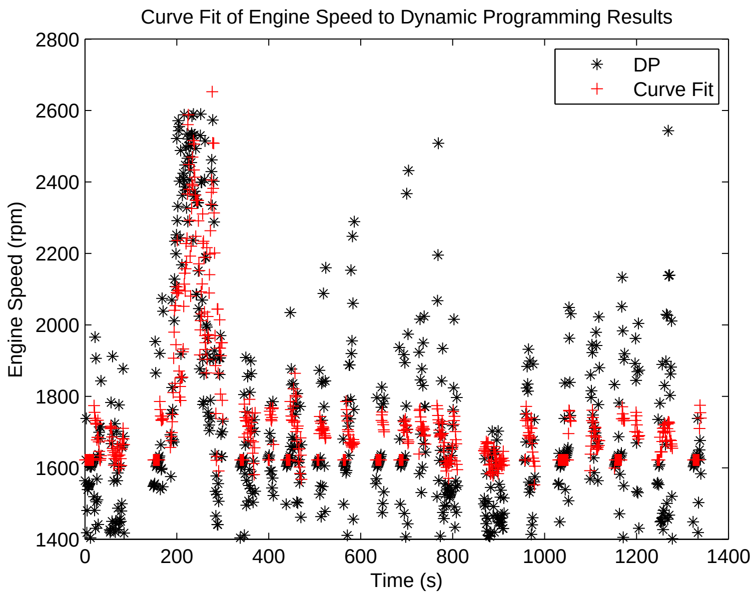


Figure 5.18: Comparison of engine speed for DP and rule-based results

Dynamic Model

The same rule-based strategy is applied to the dynamic model to compare the results to the backwards facing model. To implement in the dynamic model, the strategy given by the flowchart in Figure 5.15 is entered into the high-level subsystem. This outputs the desired engine speed, which is fed into the mid-level subsystem. The mid-level subsystem determines the desired engine torque based on the following condition.

$$T_{Eng,des} = \begin{cases} 2 & \text{if } \omega_{Eng,des} < 125.66\text{rad/s} \\ 85 & \text{otherwise} \end{cases} \tag{5.4}$$

The desired engine speed and torque are fed into the low-level subsystem, which determines the commands for displacement and engine throttle. A driver subsystem is also used to determine the desired wheel torque to follow the drive cycle.

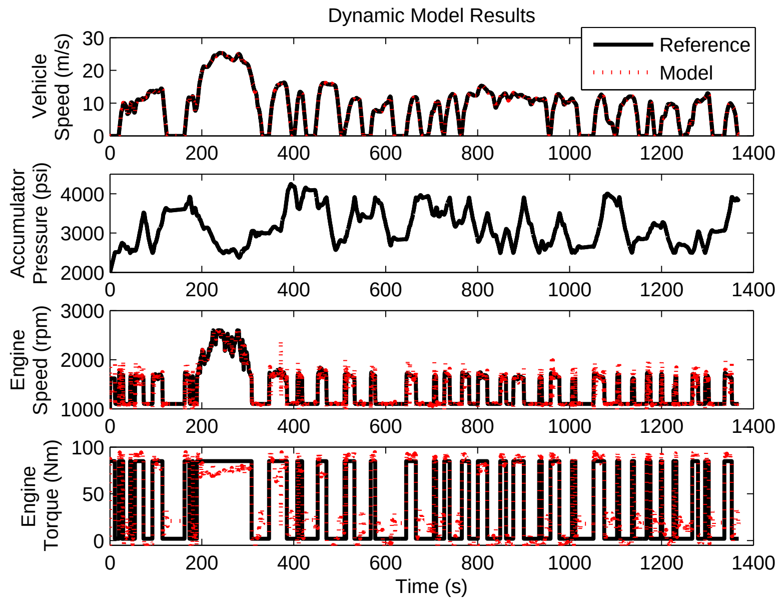


Figure 5.19: Dynamic model results using the rule-based strategy

The results using the rule-based strategy on the dynamic model are shown in Figure 5.19. Looking at these results in, the model is able to complete the drive cycle accurately with regard to the reference drive cycle. The engine speed is also able to track the desired engine speed from the rule-based strategy well. The fluctuations in the engine speed are caused by the dynamics of the engine and the drivetrain which are not captured in the backwards-facing model. The engine torque does not follow the reference trajectory as well as the engine speed, but the general pattern follows the reference trajectory. The reason for the error is because the desired engine torque is fed into an ideal pump model to determine the displacement command for pump/motor T. However, due to losses in the unit, the torque will not be equal to that value for the command. Using the actual efficiency map would help alleviate this problem, but then time and memory problems are encountered. The end result is a fuel economy of approximately 36.7 mpg.

The result from the dynamic model had an improvement of about 13% over the backwards-facing model. This can be attributed to several factors. First, the dynamic model uses the dynamics of the fuel solenoid to calculate the fuel used while the backwards-facing model uses a static map of the engine. Even though the static map is based on experimental results, modeling the dynamics of the fuel solenoid is a better approximation. Also, as stated above, the engine torque does not match the desired engine torque. Generally, the engine torque in the dynamic model is higher than the desired engine torque. A diesel engine is more efficient operating at high loads, so the dynamic model is operating slightly more efficiently. Finally, in the dynamic model, when the vehicle is stopped, the engine does not go to idle but rather provides torque to maintain the pressure in the accumulator. Therefore, even though the engine has to provide power, energy is not being lost from the accumulator.

Summary of Results

The rule-based strategy is within 2% of the DP solution when limiting the engine switching, showing the rule-based strategy is a good match for this drive cycle. The rule-based strategy is also simulated using the dynamic model. This strategy is able to complete the drivecycle entirely and gives a better result than the backwards-facing model, showing promise for implementation on the experimental vehicle. Table 5.1 shows a summary of all the results.

5.3.3 Stochastic Dynamic Programming Results

To apply the stochastic dynamic programming algorithm to the vehicle, the state and control variables are discretized. The vehicle speed is already discretized from the

Strategy	**Engine Penalty**	**Eng. Speed Weight**	**Fuel Economy**	**% Over Baseline**
Baseline	N/A	N/A	30.5 mpg	N/A
DP Constant Engine Torque	0	0	36.5 mpg	20.7
	10	0	32.8 mpg	7.5
DP Best BSFC Curve	0	0	36.6 mpg	20
	1	0	32.6 mpg	6.9
	1	0.0005	32.0 mpg	4.9
Rule-Based	N/A	N/A	32.4 mpg	6.2
Dynamic Model	N/A	N/A	36.7 mpg	N/A

Table 5.1: Summary of results from dynamic programming and rule-based strategy

transition probabilities, and the pressure is discretized from 2200 psi to 4000 psi in 100 psi increments. Even though the lower limit for accumulator pressure is 2000 psi, 2200 psi is used for the SDP algorithm to provide a buffer. Also, from the DP results, the lower limit of 2000 psi is rarely reached. The control variable is the engine speed, which is discretized from 1400 rpm to 2600 rpm in 100 rpm increments. The idle speed of 1100 rpm is also included. The constant engine torque mid-level control is used because the DP results show that strategy performs slightly better than the best BSFC curve strategy.

The discounted policy iteration algorithm is used with a discount factor of 0.9 and maximum number of iterations set to 15. One problem encountered was when a given acceleration had a zero probability because it never occurred in the drive cycle. Since no probability is given, the algorithm is not able to find a valid policy. To overcome this challenge, when the probability of a given acceleration is zero, the probability of the next acceleration is split evenly between all the next acceleration values so they each have an equal probability of occurring. For this study, since the acceleration is divided into 20 discrete values, the probability assigned is 1/20, or 0.05. Other methods can be used, but because no data is available using the chosen drive cycles, having an equal probability for all future accelerations was determined to be best. To get a more realistic probability map, more drive cycles that include accelerations in these ranges could be used.

The discounted policy iteration algorithm results in a lookup table to be used in real-time. In this case, the inputs to the lookup table are current vehicle speed, acceleration, and accumulator pressure, and the output is the engine speed. Linear interpolation is used between the discrete data points to determine the engine speed. The engine speed and torque are used in the backwards-facing model to solve for the accumulator pressure at the next time step, which is repeated for the entire drive cycle.

When the SDP results are implemented directly, a feasible solution is not found. The main reason is because not enough power is being provided to accelerate to highway speed. Another reason is, since linear interpolation is being used, engine speeds between 1100 rpm and 1400 rpm can exist. To develop a strategy that can make it through the drive cycle, the SDP strategy is modified as follows.

- If the vehicle is travelling above 20 m/s, the engine speed is set to 2500 rpm (i.e. engine provides all the power needed when at highway speeds)
- If the vehicle is decelerating, the engine speed is set to 1100 rpm (i.e. engine is idling and the hydraulics are used to decelerate the vehicle)
- If the pressure in the accumulator is less than 2200 psi, the engine speed is set to 1400 rpm (i.e. engine is used to provide some power to prevent accumulator pressure from dropping too low)
- If the lookup table gives an engine speed below 1150 rpm, the engine speed is set to 1100 rpm (i.e. engine is set to idle)
- If the lookup table gives an engine speed between 1150 and 1400 rpm, the engine speed is set to 1400 rpm (i.e. engine provides minimum power)
- For all other cases, the value given from the lookup table is used.

Using the modified control strategy, the simulation is able to successfully complete the drive cycle. The results are shown in Figure 5.20. Looking at the results, the accumulator pressure never reaches the maximum constraint, which is the same as for the DP result without the engine penalty. Also, except for where the modification was made to operate the engine at 2500 rpm, the engine operates at a low speed, often times at the minimum speed of 1400 rpm when it is not idling. This is to minimize the power of the engine to maximize the power through the hydraulic units so they operate as efficiently as possible. Also, even though the engine is providing almost maximum power at highway speeds, the accumulator pressure is still decreasing. This is due to the engine being downsized from the original engine and not designed to handle that large of a load. Finally, when the vehicle is stopped, the engine fluctuates rapidly between on and off to maintain the pressure in the accumulator at a constant level and to be ready for the next acceleration event. This shows that the optimal way to operate would be to maintain pressure in the accumulator, but the strategy of turning the engine on and off frequently to accomplish this would not be feasible. This strategy results in a fuel economy of 31.4 mpg, which is slightly worse than the rule-based strategy, but still better than the baseline strategy.

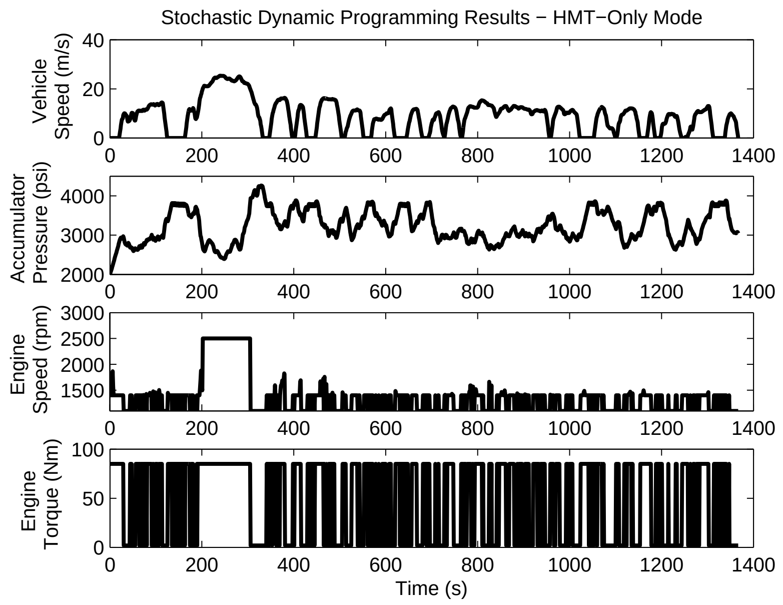


Figure 5.20: Stochastic dynamic programming results using constant engine torque for power-split architecture

Since the frequent on/off switching of the engine would not be feasible when the vehicle is stopped, a modification is made to the control strategy. If the vehicle is stopped and the pressure is greater than 2500 psi, the previous engine speed is used. The results from this modified strategy are shown in Figure 5.21. The accumulator pressure is slightly lower in this case than in the previous case. Also, while the engine behavior is similar, there is a point where the SDP algorithm must use more engine power because the accumulator pressure is reaching the bottom constraint. The engine still fluctuates rapidly when the vehicle is travelling at a near constant speed to try to maintain a constant accumulator pressure. This could be overcome by adding a time constraint that the engine must be on or off for a certain period of time before being allowed to switch. Using this strategy, a fuel economy of 33.1 mpg was obtained, which is better than the rule-based strategy.

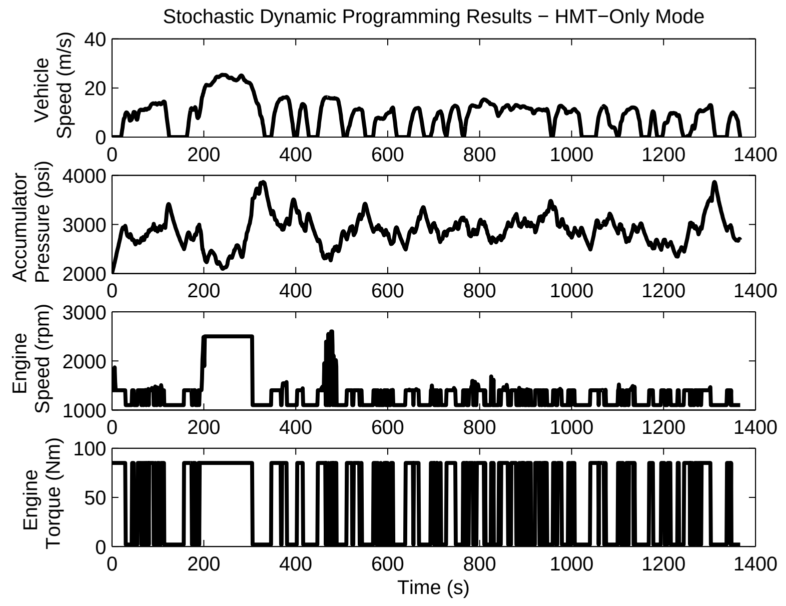


Figure 5.21: Stochastic dynamic programming results using constant engine torque for power-split architecture modified when vehicle is stopped

Summary of Results

When the lookup table from the SDP results was tried in simulation, a result could not be obtained since the hydraulics are not able to provide enough power at times. Therefore, the strategy has to be modified to obtain a control strategy that was able to make it through the drive cycle. When the vehicle is stopped, this strategy results in engine oscillations to maintain the accumulator pressure constant. To overcome this, the policy is modified to use the previous control decision whenever the vehicle is stopped.

This results in a higher fuel economy than the rule-based strategy. The results are summarized in Table 5.2.

Strategy	**Fuel Economy**	**% Over Baseline**
Baseline	30.5 mpg	N/A
SDP	31.4 mpg	2.8
SDP Modified When Stopped	33.1 mpg	8.7

Table 5.2: Summary of results from stochastic dynamic programming

5.4 Augmented Earthmoving Vehicle Powertrain Simulator

5.4.1 Dynamic Programming Results

With the equations for the AEVPS system discretized as given by Equations (3.34)-(3.38), the backwards-facing model of the system is used with the dynamic programming algorithm to determine the global optimal solution. The UDDS drive cycle shown in Figure 5.1 is again used for the optimization. Since the drive cycle is known, the load torque is calculated at each time step using Equation (3.34). The downstream pressure is also calculated at each time step using Equation (3.35). Once the downstream pressure is known, the flow through the valve is calculated using Equation (3.36). The upstream pressure is discretized into 0.01 MPa increments. For each value, the upstream pressure at the next time step is calculated using Equation (3.37). Finally, the valve command is solved for each upstream pressure by using Equation (3.38).

One advantage of the dynamic programming algorithm is that constraints on the state and control variables are easy to implement. For the AEVPS system, the constraints are given below.

$$5.2 \text{ MPa} \leq P_u \leq 19 \text{ MPa} \tag{5.5}$$

$$0 \text{ Nm} \leq u_{Eng} \leq 121 \text{ Nm} \tag{5.6}$$

$$75 \text{ rad/s} \leq \omega_{Eng} \leq 185 \text{ rad/s} \tag{5.7}$$

$$0 \leq \theta \leq 80 \tag{5.8}$$

$$0 \text{ rad} \leq u_p \leq 0.314 \text{ rad} \tag{5.9}$$

$$0 \text{ V} \leq u_v \leq 10 \text{ V} \tag{5.10}$$

A lower value of 6 MPa is used for the upstream pressure to ensure it never falls below the lower constraint of the system. Engine speed is discretized into 5 rad/s increments and the swashplate angle of the pump is discretized into 0.01 radian increments. Time is discretized into 1 second increments. The objective is to minimize fuel consumption over the length of the drive cycle, given by Equation (5.11).

$$J = \min \sum_{k=0}^{N} \text{fuel} = \min \sum_{k=0}^{N} \left[\max \left(K_f \omega_{Eng}(k) \frac{\theta(k)}{\theta_{max}}, 0.00043 \right) \Delta t \right] \tag{5.11}$$

By using Equations (4.2) and (4.3), the optimal control for each state is calculated starting at the end of the drive cycle to minimize the cost. Then, by using the initial conditions of the system, the optimal trajectory is found. Figure 5.22 shows the results for the dynamic programming, which used 0.99 kg of fuel.

5.4.2 Rule-Based Strategy Results

To develop the rule-based strategy, a curve fit of the DP data is performed for the engine throttle command and swashplate angle of the hydraulic pump. Looking at the DP data, the throttle command is dependent on the motor speed and engine speed, and the swashplate angle is dependent on the accumulator pressure and motor speed. These curve fits are given in Equations (5.12) and (5.13).

$$\theta(k) = -434.775 + 10.026\omega_{Eng}(k) + 0.017\omega_m(k) - 0.055\omega_{Eng}^2(k) + 0.002\omega_m^2(k) \tag{5.12}$$

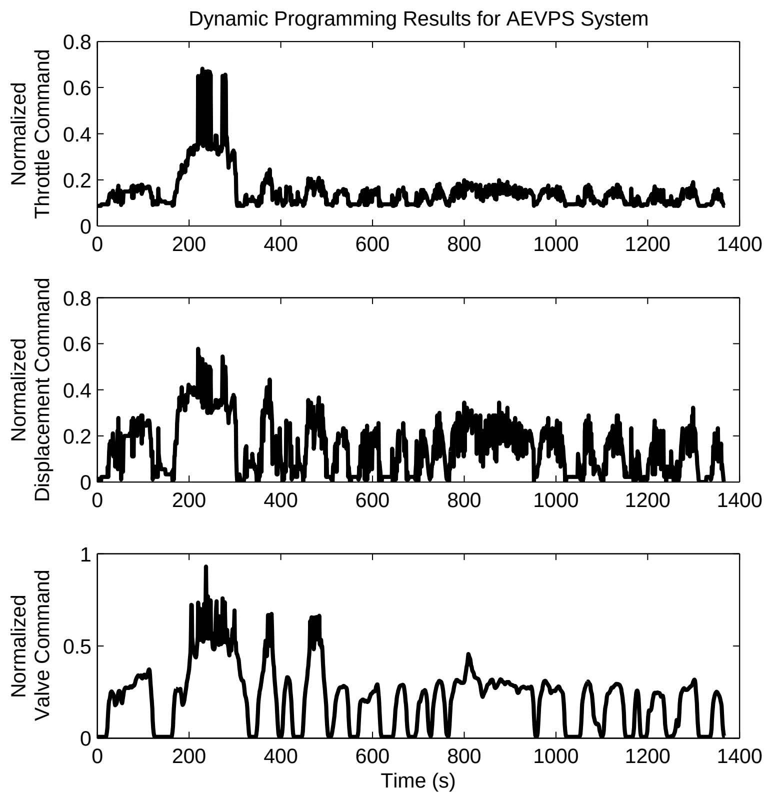


Figure 5.22: Dynamic programming results for AEVPS system

$$u_p(k) = -0.034 + 0.012P_u(k) + 0.001\omega_m(k) - 0.001P_u^2(k) + 4.219 \times 10^{-6}\omega_m^2(k) \quad (5.13)$$

The r^2 value for the throttle command is 0.8 and for the displacement command is 0.71. These are simulated using a PI controller for the valve command to track the motor speed to the reference drive cycle. Even though the r^2 values are relatively high, the equations do not capture the behavior of the DP results at high and low vehicle speeds. To account for this, a switching condition is added to these rules. When the motor is operating at low speeds, the engine throttle is set to idle. Also, when the motor is operating at high speed, the pressure drop across the throttling valve is low. Therefore, when the pressure drop is below a certain value, the throttle command is set to 55 degrees and the swashplate angle is set to 0.16 rad to more closely resemble the DP results. The set of rules are given below.

$$\Delta P_{Flag}(k) = \begin{cases} 0 & \Delta P \geq 1 \\ 1 & \Delta P \leq 0.5 \\ \Delta P_{Flag}(k-1) & 0.5 < \Delta P < 1 \end{cases} \tag{5.14}$$

$$\omega_{m,Flag}(k) = \begin{cases} 0 & \omega_m \geq 20 \\ 1 & \omega_m \leq 15 \\ \omega_{m,Flag}(k-1) & 15 < \omega_m < 20 \end{cases} \tag{5.15}$$

$$\theta(k) = \begin{cases} 55 & \text{if } \Delta P_{Flag}(k) = 1 \\ 7 & \text{else if } \omega_{m,Flag}(k) = 1 \\ \theta(k) & \text{otherwise} \end{cases} \tag{5.16}$$

$$u_p(k) = \begin{cases} 0.16 & \text{if } \Delta P_{Flag}(k) = 1 \\ u_p(k) & \text{otherwise} \end{cases} \tag{5.17}$$

The dynamic model is used to simulate the rule-based strategy with the switching criteria. A comparison between this rule-based strategy and the DP results for the UDDS drive cycle is shown in Figure 5.23.

This plot shows good comparison between the DP results and the rule-based strategy. The fuel consumption for the rule-based case in simulation is 1.10 kg, about an 11% increase from the DP results.

5.4.3 Stochastic Dynamic Programming Results

As stated earlier, the rule-based strategy is a sub-optimal solution since post-analysis of the DP results is needed. To overcome this, stochastic dynamic programming is used to develop a causal control law that is directly implementable from the results. The transition probabilities in chapter 4 are used with the discounted policy iteration algorithm. For this analysis, a discount factor of 0.95 is used.

The state and control variables are uniformly discretized between their minimum and maximum values. Upstream pressure values are discretized from 6 MPa to 19

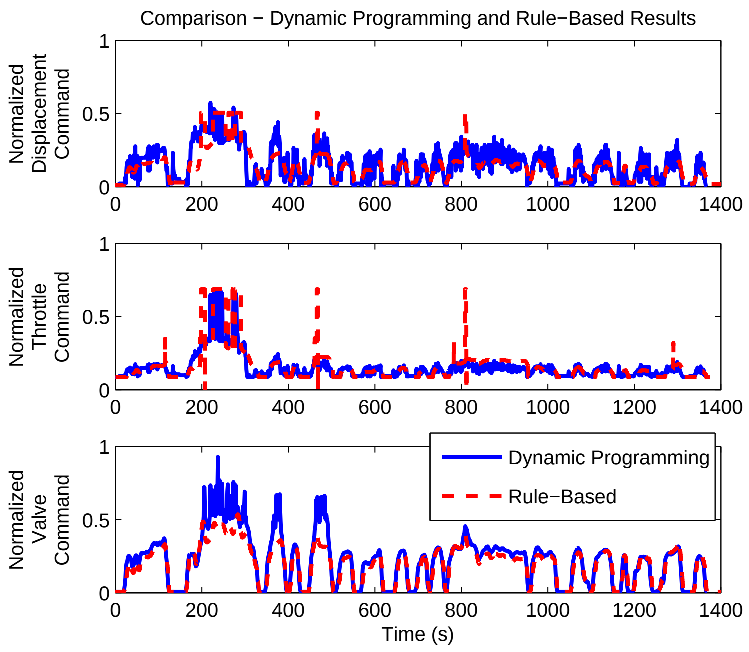


Figure 5.23: Comparison of DP and rule-based results for AEVPS system

MPa in 1 MPa increments. The values for swashplate angle are discretized between 0 radians and 0.30 radians in 0.01 radian increments. The engine speed is discretized between 75 rad/s and 185 rad/s in 5 rad/s increments. The throttle command is found using a lookup table based on engine speed and engine torque. The valve command is determined using a feedback PI controller on motor speed to improve speed tracking to the reference drive cycle.

Using this control strategy, a fuel consumption of 1.01 kg is obtained, which is only 2% more than that obtained from the DP results.

5.5 Concluding Remarks

In this chapter, simulation results were presented for both the hydraulic hybrid vehicle and the AEVPS system using the UDDS duty cycle. First a baseline control strategy developed using engineering intuition was presented. Then dynamic programming was used operating the engine at both a constant torque and along the best BSFC curve to calculate the global optimal solution. Then a rule-based strategy was developed using linear regression and the dynamic programming solution. Finally, stochastic dynamic programming was used using transition probabilities from four standard drive cycles. For both the hydraulic hybrid vehicle and the AEVPS system, the SDP solution was the closest to the DP solution. The rule-based strategy did not give as good as fuel economy as the SDP solution, but it was still better than the baseline strategy. For the AEVPS system, MPC was also used as a comparison, which gave the highest fuel usage of 1.30 kg over the UDDS drive cycle in comparison to 0.99 kg from DP, 1.10 from the rule-based, and 1.01 kg from SDP.

Chapter 6

Experimental Systems

This chapter describes the experimental systems for both the power-split hydraulic hybrid vehicle and the series Augmented Earthmoving Vehicle Powertrain Simulator. For both setups, the physical hardware and the electronics and controls used for each are described.

6.1 Hydraulic Hybrid Vehicle

The experimental setup for the hydraulic hybrid vehicle consists of two systems. The first is an experimental vehicle with the power-split transmission. The second is a hydrostatic dynamometer [75]. This was built to allow the vehicle power-train to be tested indoors in a controlled environment using any duty cycle. This also saves time by not having to transport the vehicle from the lab to outside as well as allow testing to be performed at any time regardless of weather.

6.1.1 Experimental Vehicle

This section describes the hardware and electronics used on the experimental vehicle.

Drive-train

The chassis of the vehicle is a Polaris Ranger XP all-terrain vehicle. The reason this platform was chosen over a standard passenger vehicle is because of its more open architecture over a commercial passenger vehicle, allowing for more room to work and make modifications. Also, the electronics are easier to integrate because no CAN signals are used in the Polaris Ranger. Besides the chassis, no other original drive-train components were used.

The original engine was a 683 cc, 30 kW, two cylinder gasoline engine. This was replaced by a Perkins 404C-15 engine, which is a 1500 cc, 26.5 kW, four cylinder diesel engine. The engine could be downsized from its original size because the hydraulics would provide the additional power necessary during acceleration. The original vehicle had a two-speed transmission - a low gear when high torque was needed and a high gear when traveling. However, it did not have the ability to switch gears while the vehicle was moving. This transmission was replaced with the planetary gearset used to split the mechanical power of the engine and the hydraulic power from the hydraulic units. The hydraulic units also act as an infinitely variable transmission. The original vehicle also contained a rear electronically-controlled locking differential, which was replaced with a standard automotive differential from a passenger car.

Hydraulics

The hydraulic circuit for the vehicle is shown in Figure 6.1. The two main components are the Rexroth A6 hydraulic units, which are bent axis, variable displacement units with a maximum displacement of 28 cc/rev. To allow for four quadrant operation (i.e. clockwise and counterclockwise direction as either a pump or a motor), a bi-directional valve is used for each one. Two 38 liter, bladder-type composite fiber accumulators are used on the high and low pressure lines. The high pressure accumulator has a maximum pressure of 5000 psi while the low pressure accumulator has a maximum pressure of 300 psi.

A charge circuit is used to provide flow to the low pressure side of the circuit. This is provided by a 1.2 cc/rev charge pump which is driven by an electric motor. Once the low pressure accumulator reaches a certain pressure, the charge pump does not run. By not having the charge pump connected to the engine, the auxiliary load on the engine is decreased and charge pressure is available at any engine speed, even when the engine is declutched. One challenge that was presented with this setup is that high pressure is needed for the pilot stage of the bi-directional valves. Therefore, when the vehicle is initially started and the high pressure accumulator is empty, the bi-directional valves cannot be actuated. To overcome this challenge, a start-up valve was added, which connects the output of the charge pump to the high pressure side rather than the low pressure side. This allows the charge pump to develop initial pressure on the high pressure side and the bi-directional valves to operate.

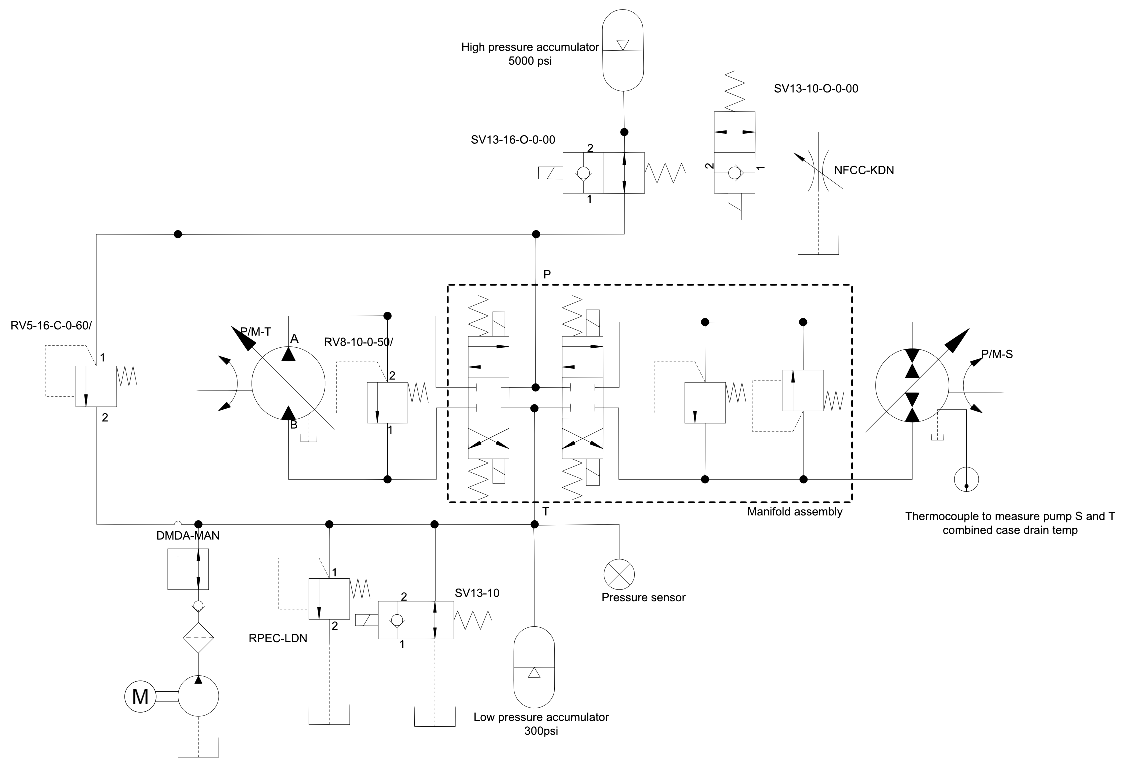


Figure 6.1: Hydraulic schematic of experimental hydraulic hybrid vehicle

Electronics and Controls

The main component of the control system for the vehicle is a 2nd generation MicroAutoBox by dSpace. This is a rugged control unit that is designed for mobile applications.

It contains analog inputs and outputs, digital inputs and outputs, timing channels for encoder and pulse-width modulated (PWM) signals, and CAN bus lines. It is powered by 12 volts DC, making it easy to connect to the vehicle battery without the need of an external power supply. The software is programmed using the Simulink interface from Matlab and the dSpace toolbox, which is then downloaded to the hardware for real-time execution.

Numerous sensors are located around the vehicle and connect to the MicroAutoBox unit to provide safe operation. Quadrature encoders are placed at the engine and the wheel to measure engine and vehicle speed and direction. Speed and direction sensors are also incorporated into the hydraulic units to measure the speed and direction of each. Pressure sensors measure the pressure in the high and low pressure accumulators. Finally, potentiometers are used to measure the accelerator and brake pedal positions, which creates the desired torque command from the operator. This makes the vehicle a drive-by-wire system since no mechanical connection exists between the pedal and the engine and hydraulic units. The mechanical brakes are still intact, along with an emergency stop switch, as safety features to allow the vehicle to stop.

The MicroAutoBox takes the inputs from the various sensors around the vehicle to command the actuators. The swashplate of the hydraulic units are controlled using analog voltage outputs and a voltage-to-current circuit since the swashplate angle is determined by the current going to the solenoid. Digital outputs are used to control the various valves in the hydraulic circuit, including the bi-directional valves, the shut-off valves for the high and low pressure accumulator to preserve pressure, and the start-up valve to connect the charge pump to the high pressure line. The engine output torque is controlled by a fuel solenoid valve.

6.1.2 Hydrostatic Dynamometer

To allow for testing indoors, a hydrostatic dynamometer was built. This allows the vehicle to run through any drive cycle as if the vehicle is moving while being stationary in the lab. This section briefly describes the hardware and electronic controls used on the dynamometer.

Hardware

To connect the dynamometer to the vehicle, the rear wheels and differential of the vehicle are removed, and the output shaft from the planetary gearset is connected to two 28 cc/rev, variable displacement, axial piston hydraulic units connected in tandem. The swashplate of these units are able to go over center, meaning the units can act as a pump or motor in both the clockwise and counterclockwise directions. A 16 cc/rev charge pump is connected in tandem to these units to provide charge flow for actuation of the displacement. A proportional bi-directional valve is connected to the hydraulic units. Normally this will be fully open as the displacement of the units can be changed to control the load. However, if a fast response time is needed for the load, the proportional valve will be used rather than changing the displacement since the response time of the valve is faster. A 38 liter, bladder-type, composite fiber accumulator is connected to the high pressure line. This is used to maintain a somewhat constant pressure, as well as providing power when needed. A continuously running hydraulic power unit that can provide approximately 19 liters/minute of flow at 200 bar is used to fill the accumulator as well as provide flow when the hydraulic units are motoring. Numerous valves are also included to prevent cavitation as well as provide safety to ensure the pressure in the lines does not get too high. A hydraulic schematic of the dynamometer is given in Figure 6.2.

Electronics and Controls

To control the dynamometer, a computer with data acquisition cards and xPC Target by the MathWorks is used. xPC Target is a toolbox for the Simulink environment to allow for real-time control. The data acquisition cards include analog inputs and outputs, digital inputs and outputs, and encoder inputs. Pressure sensors are placed at both ports of the hydraulic units to measure the high and low pressures. Between the output shaft of the planetary gearset of the vehicle and hydraulic unit of the dynamometer is a torque sensor. This requires an amplifier to obtain a reading that can be used by the data acquisition system. Incorporated into the torque sensor is a speed sensor to measure the output speed of the planetary gearset.

The actuators of the dynamometer include the displacement of the hydraulic units,

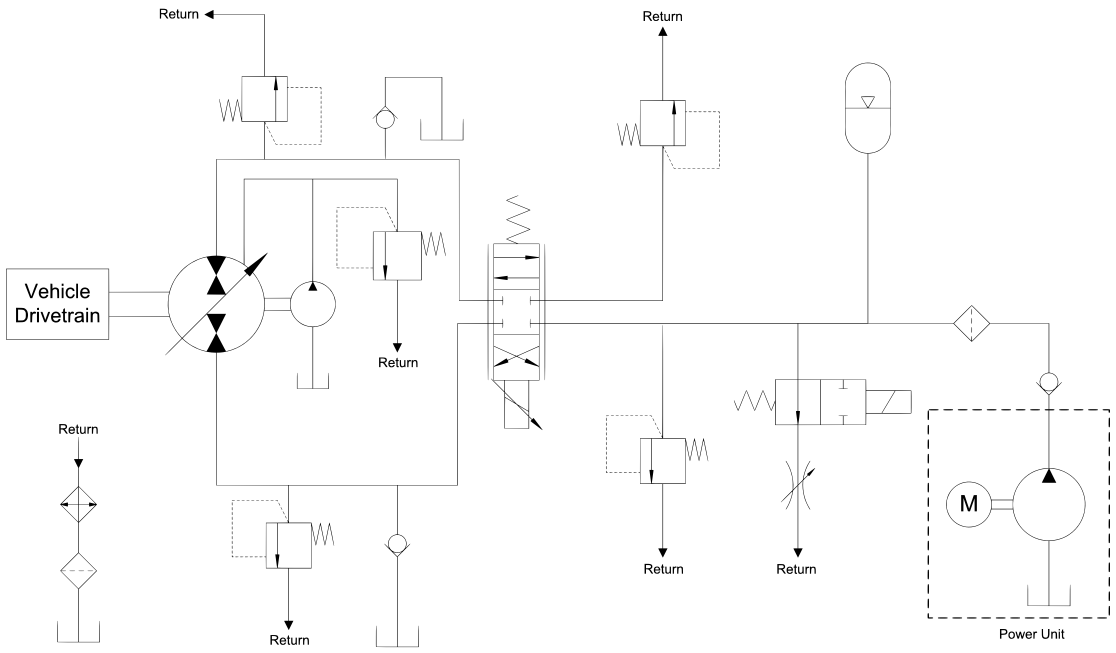


Figure 6.2: Hydraulic schematic of dynamometer for testing of experimental vehicle

the position of the bi-directional proportional valves, and signals for the other valves in the system for safety. Based on the pressure, torque, and speed readings, the displacement of the hydraulic unit is modified to match the reference load. If the displacement cannot respond fast enough, the proportional valve is then utilized to change the load. If the system needs to be shut down, the valve on the high pressure side opens as soon as power is turned off to drain the high pressure accumulator and remove any high pressure to put the system in a safe condition.

6.2 Augmented Earthmoving Vehicle Powertrain Simulator

To simulate a series hydraulic hybrid system, the AEVPS system at the University of Illinois-Urbana/Champaign is used [76],[77],[78]. In this system, a planetary gear train is not needed. A hydraulic pump is connected to the engine, which connects to a hydraulic motor to provide power to the load. An accumulator is placed between the two units to allow for energy storage. A photo of this system is shown in Figure 6.3.

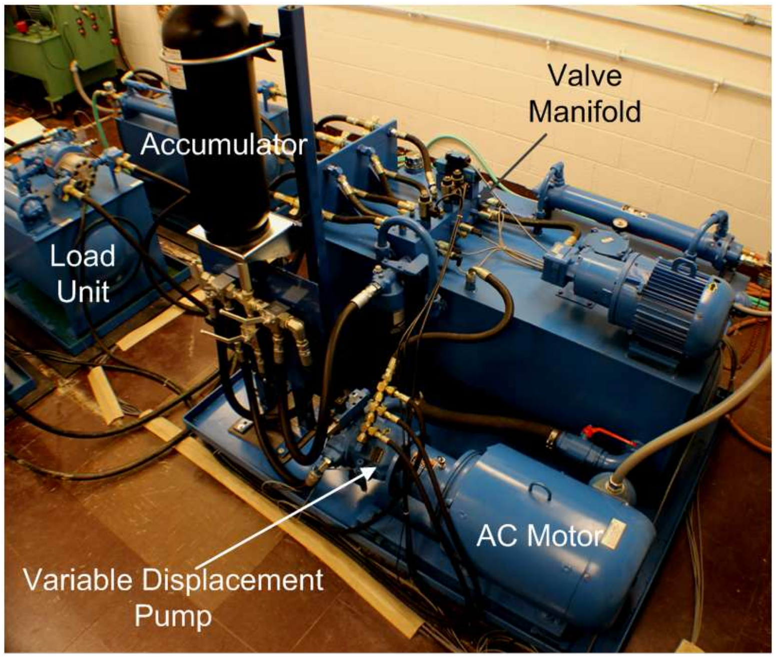


Figure 6.3: Photo of AEVPS system

Hardware

The diagram of the AEVPS system is shown in Figure 6.4. As shown, two units make up the complete system. The first is the powertrain unit, which is a series hybrid powertrain and the system being studied. The second is a load unit used to provide a load on the hydraulic motor. The powertrain unit consists of a prime mover, a variable displacement axial piston pump with a maximum flow rate of 128 liters/minute, a 18.9 liter gas charged accumulator with a precharge pressure of 5.19 MPa, and a fixed displacement 26.5 cc/rev hydraulic gear motor with a maximum flow rate of 79 liters/minute. Since a fixed displacement hydraulic motor is used in this system, a throttling valve is used to control the amount of flow going to the motor, which controls the output power of the motor. The maximum operating pressure is 20 MPa. The load unit consists of a 26 cc/rev hydraulic gear pump. Since the pump in the load unit is a fixed displacement pump, an electronically controlled proportional relief valve is used to control the load. The purpose of the load unit is to emulate the driving loads experienced by a passenger vehicle by regulating the pressure required to activate the pressure relief valve. The original system contained three load units, but only one was used for this study. The state variables for the series system are the engine speed, upstream pressure, downstream pressure, and hydraulic motor speed, while the control variables are the throttle command, swashplate angle, and valve command.

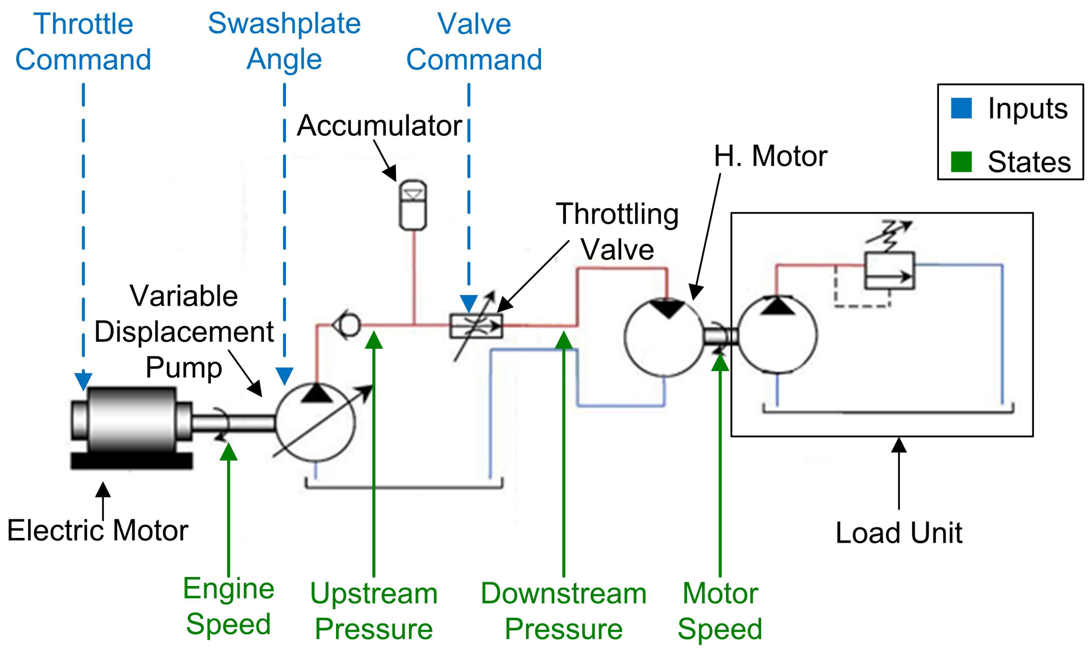


Figure 6.4: Inputs and states for the AEVPS system

The engine is emulated using an AC motor. An advantage of this is that a variety of different engines can be emulated by using computer control to make the AC motor behave according to engine dynamics [76], [77], [78]. In this study, since a 1/4 scale powertrain model is used, which scales the vehicle loads by a factor of 4, the maximum output power of the engine was chosen to be 1/4 that of a 2009 Toyota Prius engine, equal to 18 kW. The AC motor can provide up to 22.4 kW of power, so when it is used to emulate the scaled down engine, the maximum supplied power from the motor is limited to 18 kW. Figure 6.5 shows a normalized efficiency map for the emulated engine. This plot shows that the most efficient operating point of the engine is about 55% of maximum operating speed and 50% of maximum operating power. However, as shown in the previous studies with the power-split architecture, it may not be optimal to operate the engine in this region all the time.

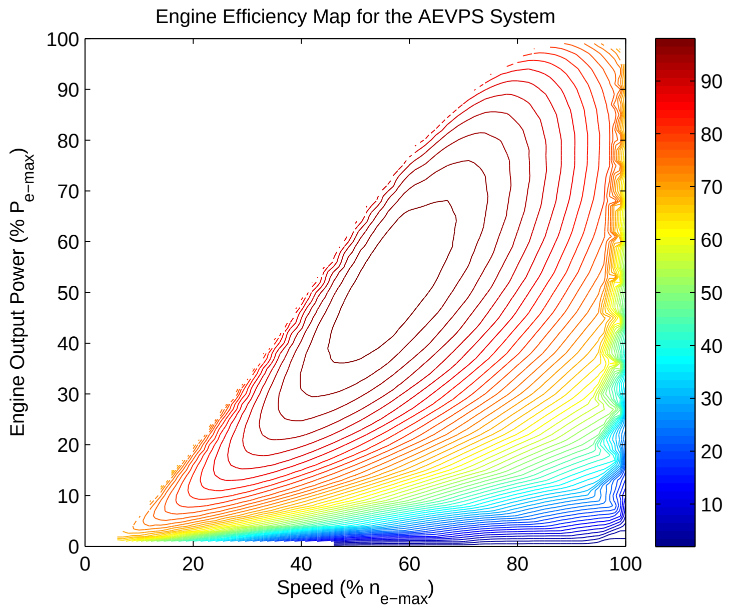


Figure 6.5: Engine efficiency map for AEVPS system

Electronics and Control

To control the AEVPS system, Matlab/Simulink models are used to generate code, which is then downloaded onto the dSpace hardware. The hardware takes inputs from the system and generates the outputs for controlling the system. Inputs to the controller include the torque and speed of the AC motor, pump displacement, upstream and downstream pressure, and the hydraulic motor speed. Outputs of the controller include the pump displacement command, throttling valve command to control the flow going to the motor, and the pressure relief valve command for the load unit. The engine throttle command is sent to the controller that emulates the engine, which predicts the engine speed based on a mapping of the throttle command and the engine load estimate.

6.2.1 Experimental Results

To validate each of the control strategies experiments on the physical hardware are performed using the UDDS drive cycle as the reference motor speed trajectory.

Rule-Based Strategy

For this strategy, a PI controller with a proportional gain of 0.01 and integral gain of 0.05 is used to regulate the throttle valve to track the motor speed to the reference speed. The rules given by Equations (5.12) and (5.13) are used to determine the engine throttle and pump displacement commands. Figure 6.6 shows a comparison between the simulation and experimental results for the control inputs. Both the displacement and valve commands are similar between the two. The engine throttle command is slightly less in the experiment than in the simulation since the engine idle speed is less in experiment than the speed in the simulation.

Figure 6.7 shows a comparison of the engine operating points between the experiment and simulation. Again, the two match very well, with a cluster of points at idle condition and a cluster of points in or slightly below the most efficient region. In the experiment, a few points are located in the high power region of the engine which do not appear in the simulation.

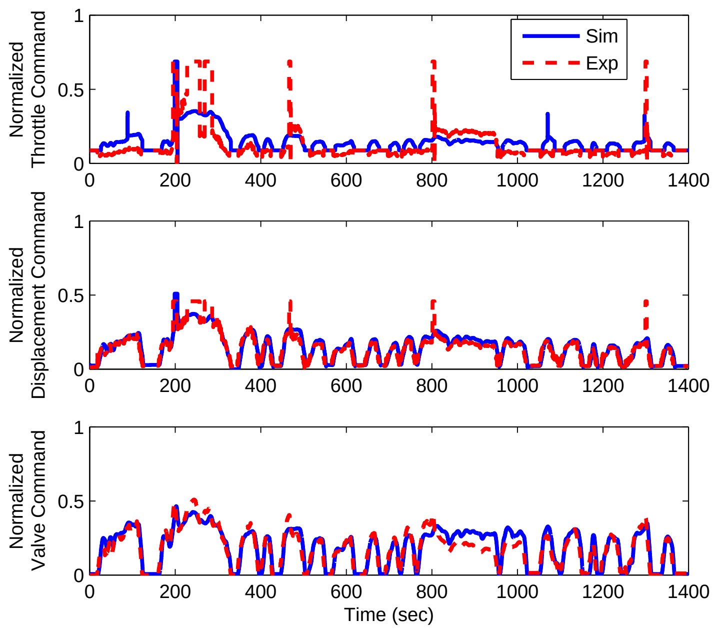


Figure 6.6: Comparison of simulation and experimental control inputs for rule-based strategy for AEVPS system

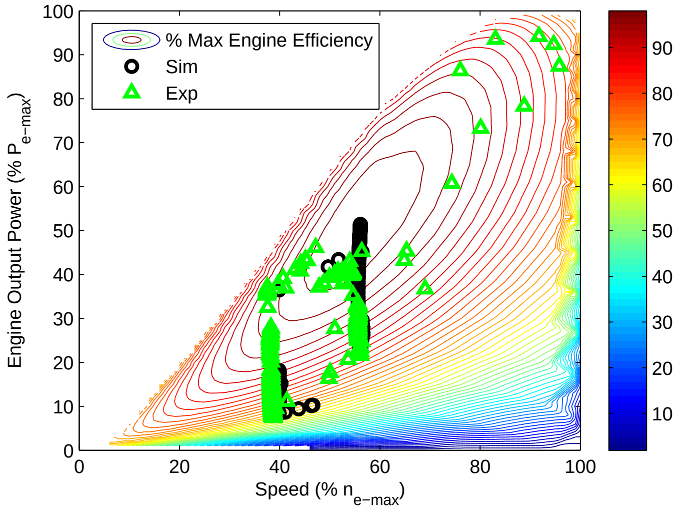


Figure 6.7: Comparison of simulation and experimental engine operating points for rule-based strategy for AEVPS system

Figure 6.8 compares the states from the experiment and simulation. The engine speed, downstream pressure, and motor speed all match well between the two. The accumulator in the experiment drops faster than in the simulation, which is why the high engine power points are needed in experiment and not simulation. Even though the experimental results are slightly off, the trajectory is still close. The total fuel consumption in the experimental run is 1.06 kg, which is within 4% of the simulation value of 1.10 kg.

Stochastic Dynamic Programming

For the stochastic dynamic programming implementation, the same PI controller as that in the rule-based strategy for the motor speed is used. The sampling time for the lookup table is 0.1 seconds to ensure enough time is given for the linear interpolation. The engine throttle and pump displacement commands are held constant during this time. Also, the output from the lookup table is passed through a low-pass filter to prevent high frequency switching. Figure 6.9 shows the comparison between the simulation and experiment for the control inputs. Even with the low-pass filter, some oscillations still exist in the throttle and pump displacement command when the motor is at high speed.

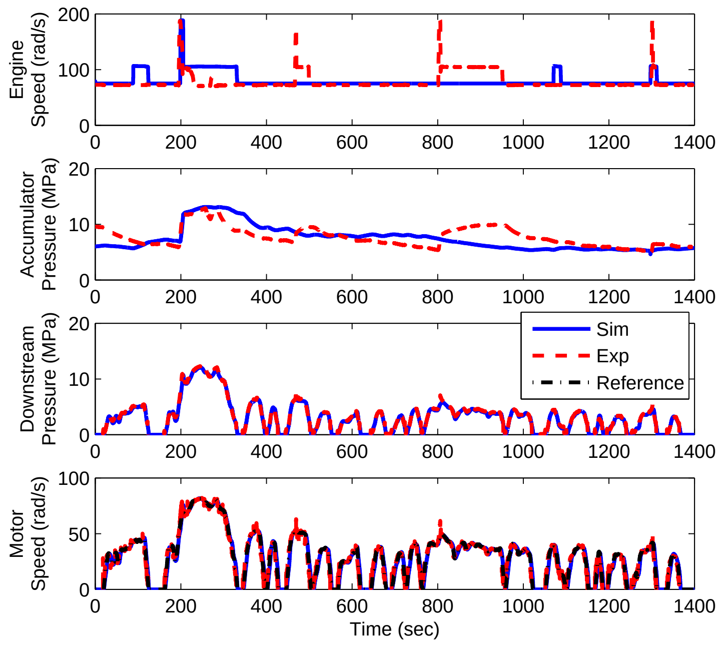


Figure 6.8: Comparison of simulation and experimental results for rule-based strategy for AEVPS system

Figure 6.10 shows a comparison of the engine operating points. Again, there is good correlation between the simulation and the experiment, with a cluster of points below the most efficient region of the engine. Just as in the power-split architecture, the engine operates slightly below the most efficient region to improve the efficiency of the overall system.

Figure 6.11 shows a comparison of the state variables between the simulation and experiment. Overall, all the state variables from the experiment match those from the simulation. When the motor is operating at a high speed, some oscillations in the speed tracking occur. This is due to the rapid switching of the control inputs to the system which the system cannot respond to fast enough. The total fuel consumption in the experimental run is 0.98 kg, which is within 3% of the simulation value of 1.01 kg. This is not surprising since the control inputs and states match very well between the two.

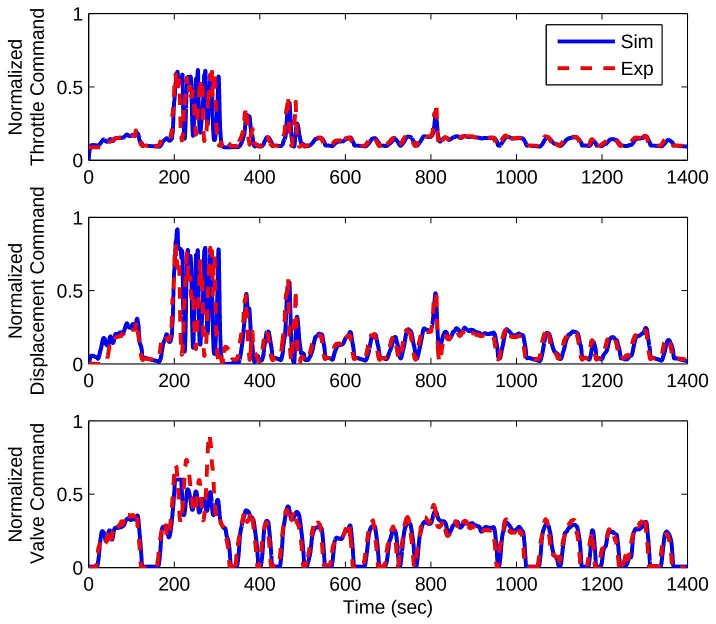


Figure 6.9: Comparison of simulation and experimental control inputs for SDP strategy for AEVPS system

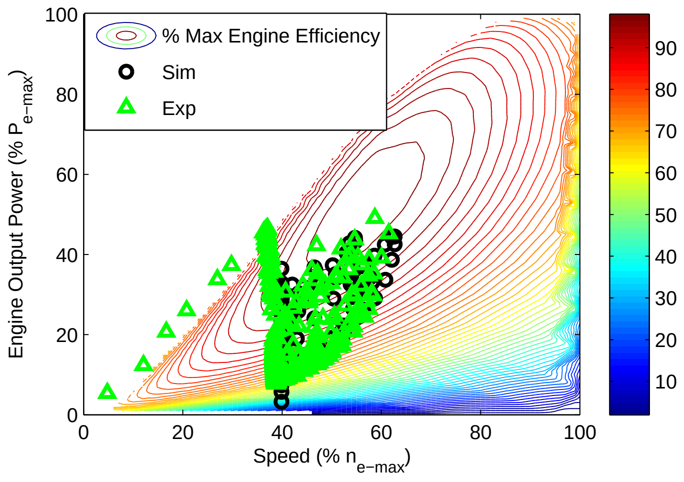


Figure 6.10: Comparison of simulation and experimental engine operating points for SDP strategy for AEVPS system

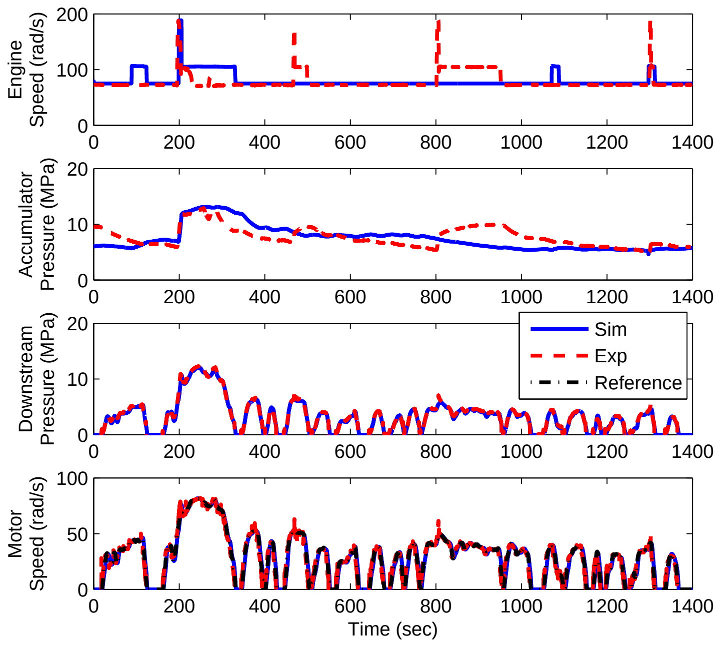


Figure 6.11: Comparison of simulation and experimental results for SDP strategy for AEVPS system

Model Predictive Control

For the model predictive control implementation, a sampling time of 1 second is used with a prediction horizon of 5 seconds. These were chosen to allow for real-time execution. The control signals are passed through a low-pass filter to smooth the input commands. A 10 second dwell time is used to prevent the engine from switching between on and idle modes frequently. Figure 6.12 shows a comparison between the simulation and experimental control inputs. As in the SDP case, oscillations in throttle and pump displacement occur at high motor speed, even with the low-pass filter. For the other times, the control inputs match with those from the simulation.

Figure 6.13 shows a comparison between the engine operating points for the simulation and experiment. This is a combination of the engine operating points from the SDP strategy and the rule-based strategy. As in the SDP strategy, the majority of points exists below the most efficient region. And like the rule-based strategy, some points are

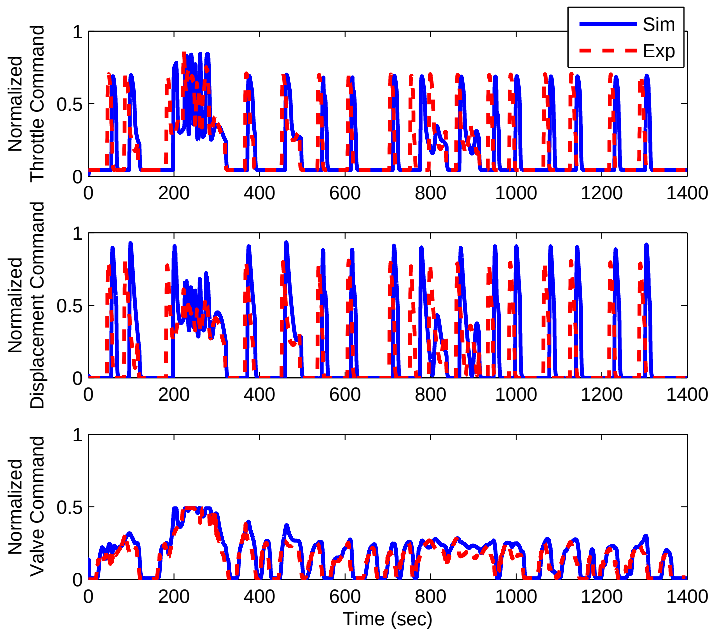


Figure 6.12: Comparison of simulation and experimental control inputs for MPC strategy for AEVPS system

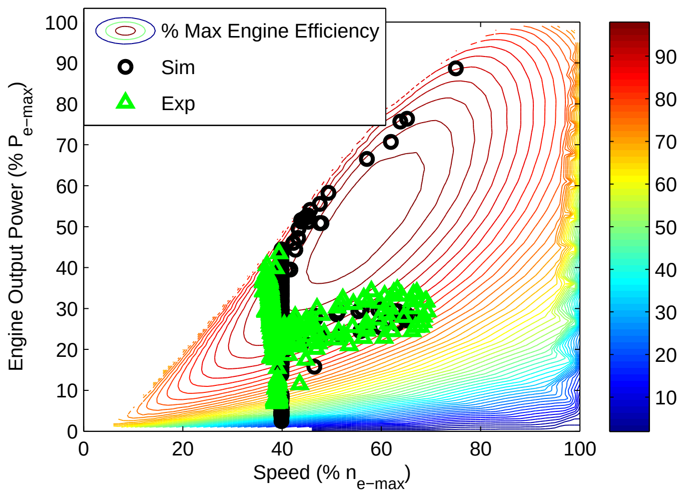


Figure 6.13: Comparison of simulation and experimental engine operating points for MPC strategy for AEVPS system

in the high power region to charge the accumulator.

Figure 6.14 shows the comparison of the state variables between the simulation and experiment. As in the rule-based and SDP cases, the experimental results match well with the simulation. The pressure in the accumulator drops faster in the experiment due to losses not modeled in the simulation. The fuel consumption for the experiment is 1.34 kg, which is within 3% of the simulation value.

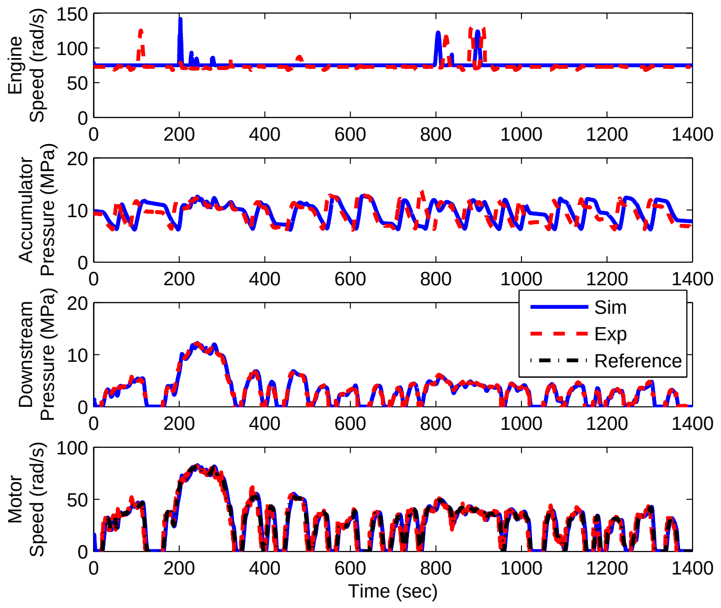


Figure 6.14: Comparison of simulation and experimental results for MPC strategy for AEVPS system

The experimental results show that for all three strategies, the experimental results match well with the simulation results, and the fuel consumption values are all within 5% of each other. This shows that the simulation is accurate for all three strategies, and simulations can be used for further studies on the system.

6.3 Concluding Remarks

In this chapter, the experimental setups for both the power-split hydraulic hybrid vehicle and the series hydraulic hybrid setup. Both of the setups contain two systems - one for the hybrid powertrain which is being studied and the other system to emulate the loads on the system. Even though the hardware and architecture are different for each one, they both use a similar procedure for implementing the control strategies. Both use a Matlab/Simulink model which is used to generate code to download onto dSpace hardware, which is used to control the systems. This was chosen so different control strategies could be interchanged easily between the two systems without much rework from simulation to implementation. The experimental results from the AEVPS system show a good match with the simulation results for the rule-based, SDP, and MPC energy management strategies, validating the simulation results.

Chapter 7

Additional Studies and Analysis

In this chapter, further simulation studies are presented for the power-split hydraulic hybrid vehicle and the series AEVPS system. The power-split system offers more freedom with the control strategy because of the planetary gearset, so modifications to the original control strategy were studied to see the effect on fuel consumption. Also, the size of the accumulator was varied to see which size offers the least fuel consumption. For the series hybrid, the different energy management strategies - rule-based, SDP, and MPC, were studied over different drive cycles to see how the effect of the drive cycle influenced each of the strategies.

7.1 Hydraulic Hybrid Vehicle

The other previous simulations required the engine to be clutched at all times to make the control strategy simple. In this chapter, more advanced control strategies will be used, such as declutching the engine, turning the engine off, and operating the vehicle in different modes by not using the engine or hydraulic units.

7.1.1 Declutch Engine When Stopped

The first strategy studied is declutching the engine when the vehicle is stopped. The engine will still always be on, but when the vehicle is stopped, the engine declutches from the drivetrain and runs at idle. When this happens, the bi-directional valves move to the center position so that pressure is maintained in the accumulator. For the following simulations, the valves are assumed not to leak and the pressure will not decrease while the engine is declutched. As in the previous simulations, two different mid-level controllers are used - one for constant engine torque and the other operating along the best BSFC curve.

Constant Engine Torque

For these simulations, a constant engine torque of 85 Nm is used just as in the previous studies. The dynamic programming algorithm is used to find the global optimal solution. However, when the vehicle is stopped, the engine declutches and the valves close. This means that no control decisions exist at these points making the optimization run faster than the previous studies. The dynamic programming results are shown in Figure 7.1.

The results for this strategy are similar to the results without declutching the engine. First, the accumulator pressure never reaches the maximum value, and for the majority of the drive cycle, the pressure is below 3000 psi. Also, the engine operates below 2000 rpm for the majority of the drive cycle and increases when the vehicle is operating at highway speed. Finally, the engine is frequently oscillating between idle and power modes, especially when the vehicle is moving at a relatively constant speed.

Figure 7.2 shows the engine operating points for this strategy. Once again it is very similar to the solution when the engine was always clutched. There are a cluster of points near low engine speeds, and as the engine speed increases, the number of operating points decreases. This resulted in a fuel economy of 40.6 mpg, an improvement of about 11% over not declutching the engine.

Just as before, this would not be physically possible due to the high frequency of switching the engine from idling to powering and back again. Therefore, an engine penalty of 10 is used to limit the engine switching. These results are shown in Figure 7.3.

Looking at the results, the accumulator pressure never reaches the maximum value, whereas in the results where the engine is always clutched in Figure 5.6 the accumulator pressure reaches the maximum value a few times throughout the drive cycle. However, the peaks and valleys in the pressure occur at the same points in the drive cycle. Also, when the vehicle is travelling at a relatively constant speed, such as around 800 seconds in the drive cycle, the pressure reaches the minimum value. Here the engine is used

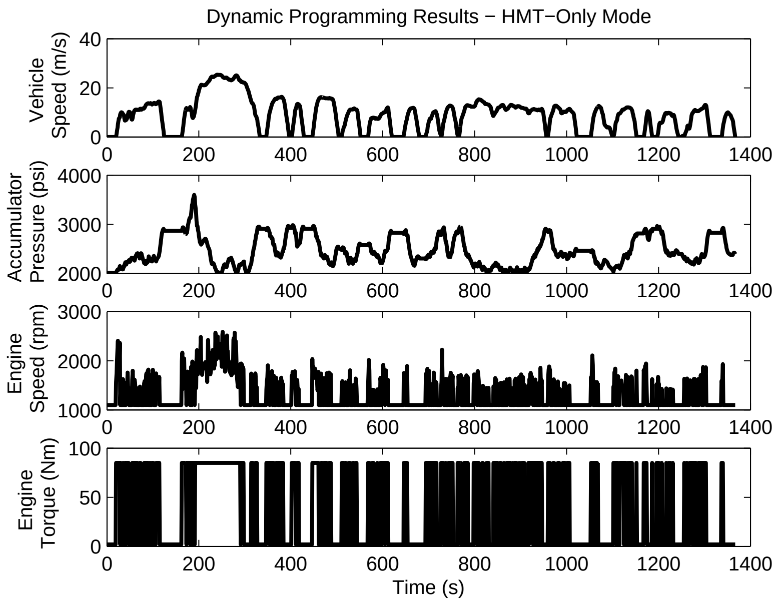


Figure 7.1: DP results for declutching the engine when vehicle is stopped using constant engine torque

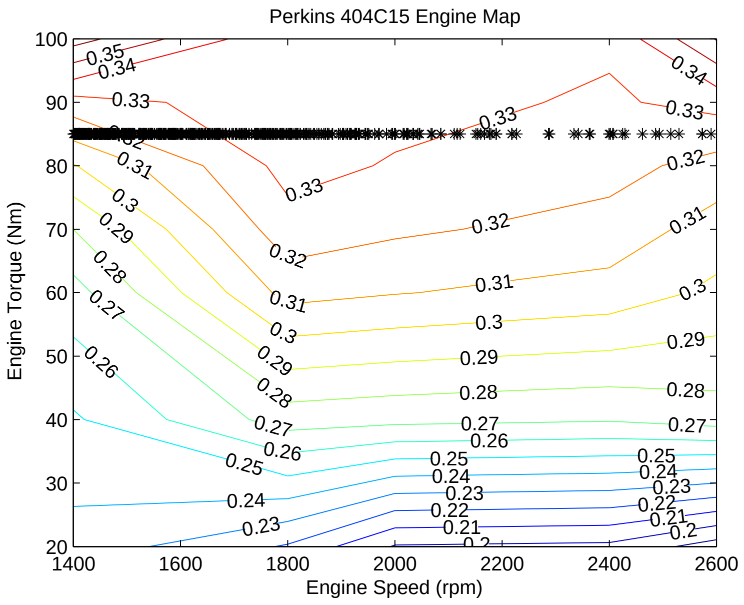


Figure 7.2: Engine operating points for declutching the engine when vehicle is stopped using constant engine torque

to refill the accumulator and then declutches again to power the vehicle solely by the hydraulics. This corresponds to the same point where the engine had high fluctuations between idling and powering without the penalty. Therefore, the engine is operating similarly between the penalty and non-penalty results, but the penalty makes it manageable on a physical system.

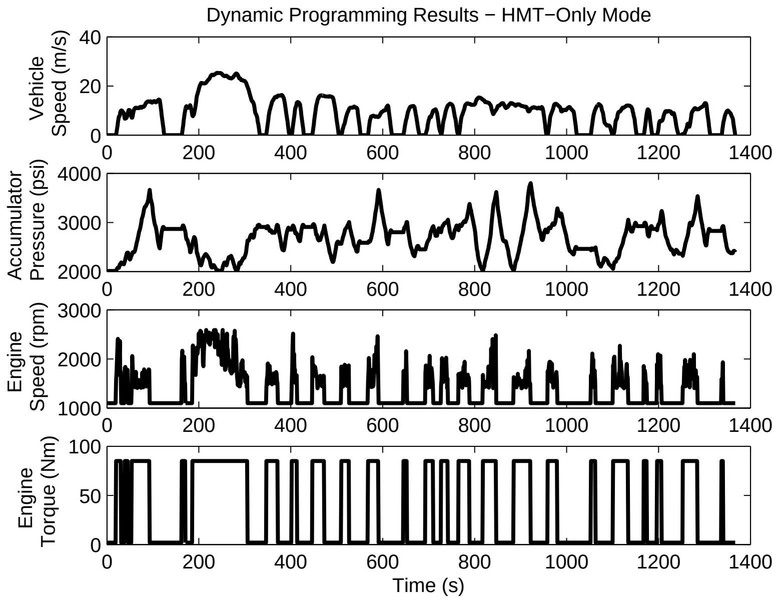


Figure 7.3: DP results for declutching the engine when vehicle is stopped with engine penalty and constant engine torque

Figure 7.4 shows the engine operating points using this strategy. Again a cluster of points exists below 2000 rpm, but more operating points exist at higher speeds than in the non-penalty case. This same behavior was observed when the engine was always clutched. The resulting fuel economy for this strategy is 37.2 mpg, which is about a 13% improvement over always having the engine clutched to the drivetrain.

Best BSFC Curve

For these simulations, the best BSFC curve is used to determine the engine power. Just as in the constant engine torque case, when the vehicle is stopped, the engine declutches and runs at idle, making for no control decisions at these points.

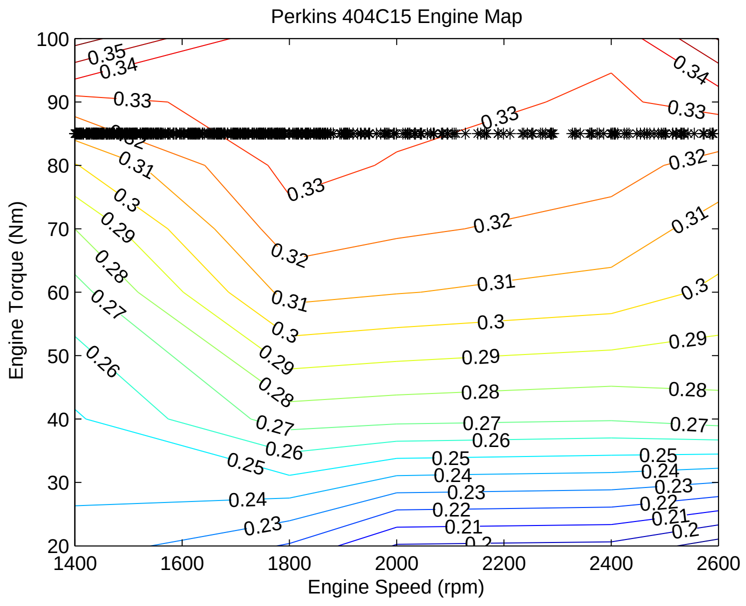


Figure 7.4: Engine operating points for declutching the engine when vehicle is stopped with engine penalty and constant engine torque

The results using this strategy are shown in Figure 7.5. Looking at these results and the results from not declutching the engine in Figure 5.11 show many similarities. First, the only time the accumulator pressure nearly reaches the maximum value is around 200 seconds before accelerating to highway speed. Other than that, the pressure stays below 3000 psi. The engine operation is also very similar, staying around 1800 rpm for the majority of the drivecycle and increasing slightly at highway speed.

Figure 7.6 shows the engine operating points for this strategy. Again, comparing this to the engine always clutched case in Figure 5.10 shows nearly identical operating points. The majority of operating points are near the most efficient operating point of the engine, with a few points at lower engine power. This strategy resulted in a fuel economy of 40.2 mpg, again less than using constant engine torque, but about a 10% improvement over not declutching the engine.

Again, due to the frequent switching of the engine, the above strategy is not implementable on a physical system. As in the previous studies, an engine penalty is used to limit the engine switching. The penalty used in this case is 1.

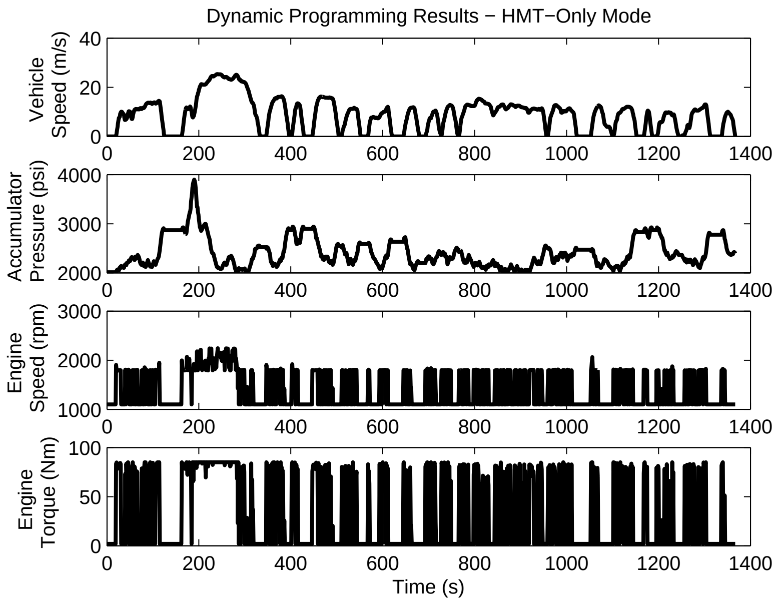


Figure 7.5: DP results for declutching the engine when vehicle is stopped using best BSFC curve

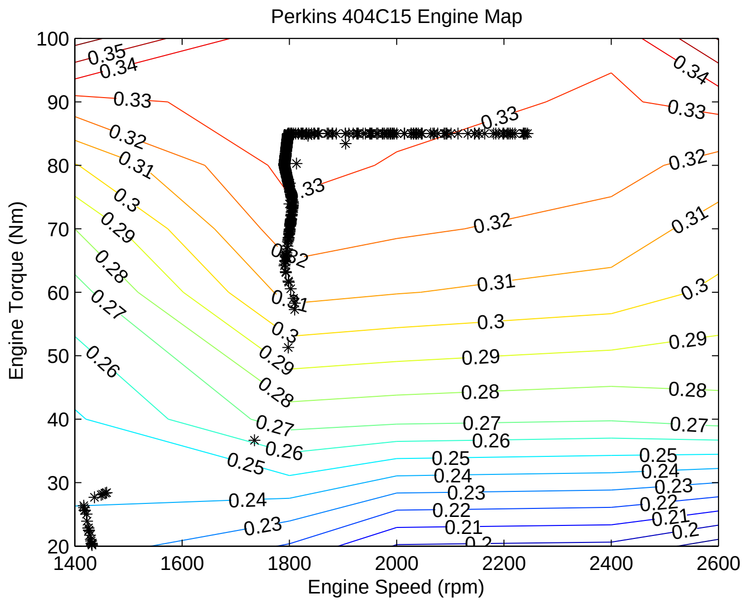


Figure 7.6: Engine operating points for declutching the engine when vehicle is stopped using best BSFC curve

Figure 7.7 shows the results for this case. Again the accumulator pressure only reaches near the maximum limit right before accelerating to highway speed, and the majority of the time it never goes about 3000 psi. Also, when the vehicle is near constant speed, the engine is switching between high and low power output frequently, although it is not going to idle. This is the same behavior observed when the engine was always clutched to the drivetrain.

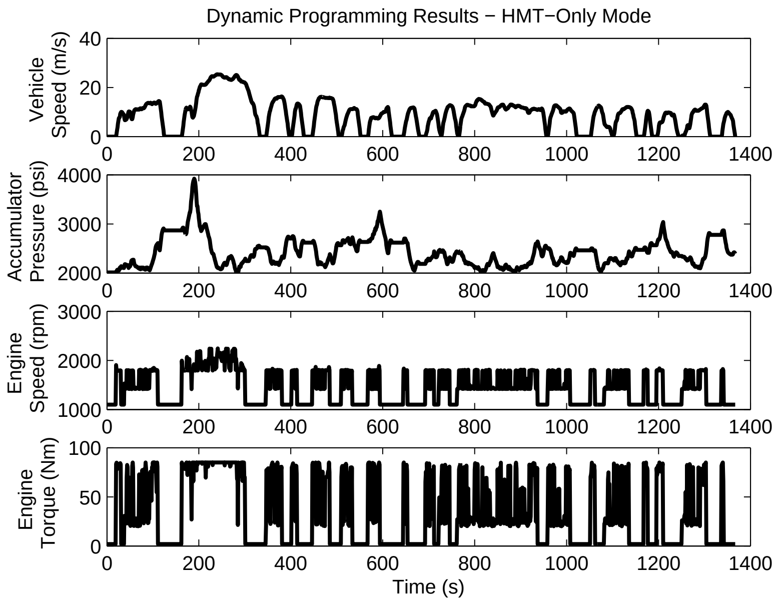


Figure 7.7: DP results for declutching the engine when vehicle is stopped using best BSFC curve and engine penalty

Figure 7.8 shows the engine operating points for this strategy. Comparing this to the engine always clutched case in Figure 5.12, the points are almost identical. The fuel economy using this strategy is 38.0 mpg, which is slightly better than the constant engine torque case, and a 16% improvement over not declutching the engine.

For all the cases above, the results for declutching the engine were all similar to the results when the engine was always clutched. From these results it can be concluded that a strategy that is developed for the case where the engine is always clutched will work just as well for the case where the engine is allowed to declutch from the drivetrain. However, over a 10% fuel economy increase is seen by declutching the engine. Since the

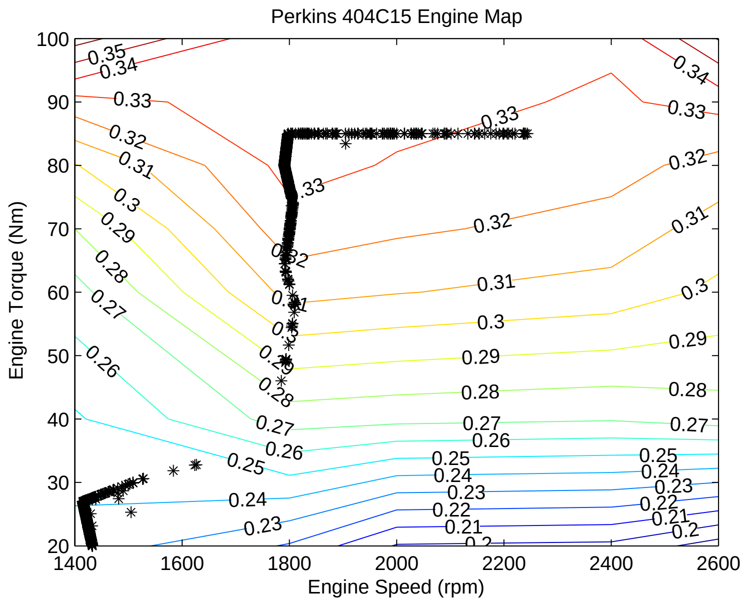


Figure 7.8: Engine operating points for declutching the engine when vehicle is stopped using best BSFC curve and engine penalty

engine operating points are nearly identical for each case between engine always clutched and engine declutched, this savings in fuel is a result of not allowing the pressure in the accumulator to be reduced while the vehicle is stopped.

7.1.2 Rule-based Strategy for Declutching the Engine

The rule-based strategy that was developed previously when the engine was always clutched to the drivetrain is also applied when the engine is allowed to declutch when the vehicle is stopped. Two minor modifications are made to the rule-based strategy to accommodate for the engine being allowed to declutch. First, if the vehicle is stopped and the accumulator pressure is greater than 2400 psi, the engine declutches and runs at idle. The valves for the hydraulic units are placed in the center position and no leakage flows through the valves, maintaining a constant pressure in the accumulator. Second, the lower setpoint for the accumulator pressure to turn the engine on is decreased from 2500 psi previously to 2400 psi, while the upper setpoint for the pressure to turn the engine off was decreased from 3900 psi to 3500 psi. These were lowered because the dynamic programming results show a lower pressure when the engine is allowed to declutch than when the engine is always clutched to the drivetrain. As with the previous rule-based strategy, a constant engine torque of 85 Nm is used.

The results using this rule-based strategy are shown in Figure 7.9. The accumulator pressure never reaches the maximum constraint of 4000 psi. The pressure does drop below the lower constraint of 2000 psi implemented in the dynamic programming algorithm. However, the pressure does not drop below 1600 psi, which is the precharge pressure of the accumulator. Also, even though the pressure setpoints were lowered for the engine operation, the pressure still ends at a pressure above 3000 psi.

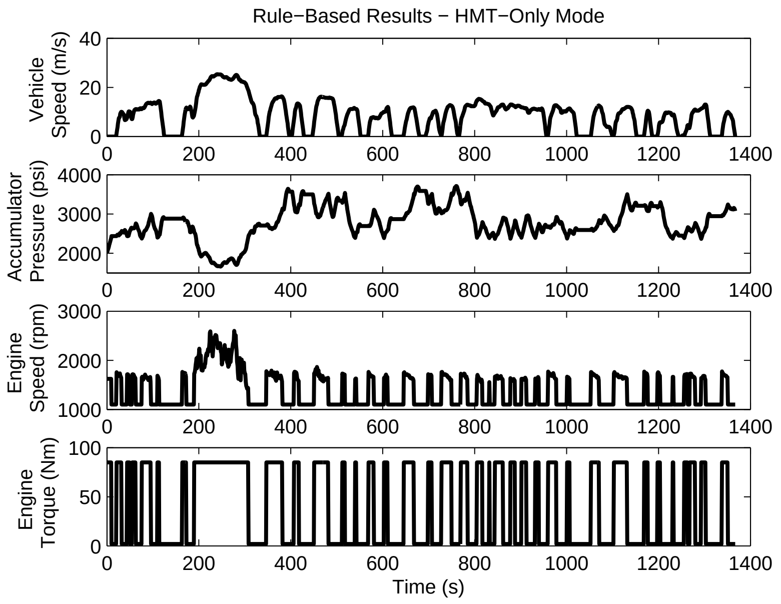


Figure 7.9: Results for rule-based strategy with declutching the engine when vehicle is stopped

The engine operating points for this rule-based strategy are shown in Figure 7.10. Few points are located below an operating speed of 1600 rpm, and the majority of operating points are located between 1600 and 1800 rpm, near the optimal operating point of the engine. The engine operates at high speed only at highway driving speeds when more power is needed and the accumulator is low on energy.

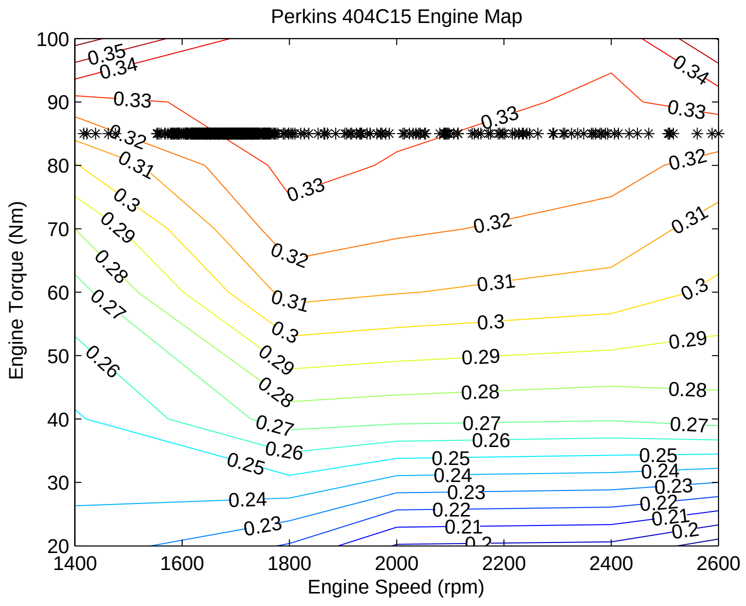


Figure 7.10: Engine operating points for rule-based strategy with declutching the engine when vehicle is stopped

This rule-based strategy results in a fuel-economy of 36.8 mpg, which is approximately a 13.5% improvement over a similar rule-based strategy when the engine did not declutch. This is a similar improvement that the dynamic programming results showed between the engine always clutched and declutching the engine when stopped. This shows that the rule-based strategy that was developed when the engine is always clutched can be used when the engine is allowed to declutch when the vehicle is stopped with minor modifications.

7.1.3 Turning Engine Off

In the previous section, the engine was declutched while the vehicle was stopped but the engine was still idling. In this section, the engine will be able to declutch during at any time in the drive cycle. Also, when the engine is declutched, the engine is turned off and the fuel consumption is set to zero.

When the engine is declutched, the speed of pump/motor T is no longer relative to the engine speed. Therefore, with the engine off, the speed of pump/motor T becomes the control variable rather than the engine speed. For these simulations, the speed of pump/motor T is allowed to vary between 500 rpm and 3600 rpm. As with the engine declutched case, the constant engine torque and best BSFC curve are examined.

Constant Engine Torque

In this section, a constant engine torque of 85 Nm is used when the engine is providing power. The range of feasible engine speed is the same as the previous studies - 1400 rpm to 2600 rpm.

Figure 7.11 shows the dynamic programming results for turning the engine off. Again, the accumulator pressure is kept below 3000 psi for the majority of the drive cycle and is near the lower limit when the vehicle is operating near a constant speed around 800 seconds. Also, the engine speed is around 1800 rpm for the majority of the drive cycle, with a slight increase when the vehicle is travelling at highway speed. Finally, the engine switches very frequently while the vehicle is moving, and the only time the engine is off for long periods of time is when the vehicle is stopped.

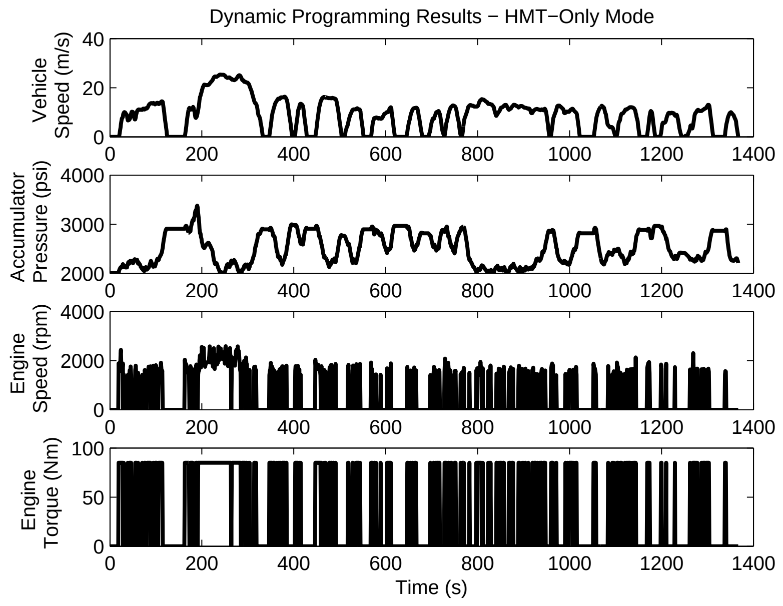


Figure 7.11: DP results for turning the engine off using constant engine torque

Figure 7.12 shows the engine operating points when the engine is on and providing power. The operating points are again clustered very similar to the case where the engine is always clutched to the drivetrain, with the majority of the points between 1400 rpm and 1800 rpm. Using this control strategy results in a fuel economy of 50.7 mpg, which is about a 25% improvement over the strategy to declutch the engine when the vehicle is stopped.

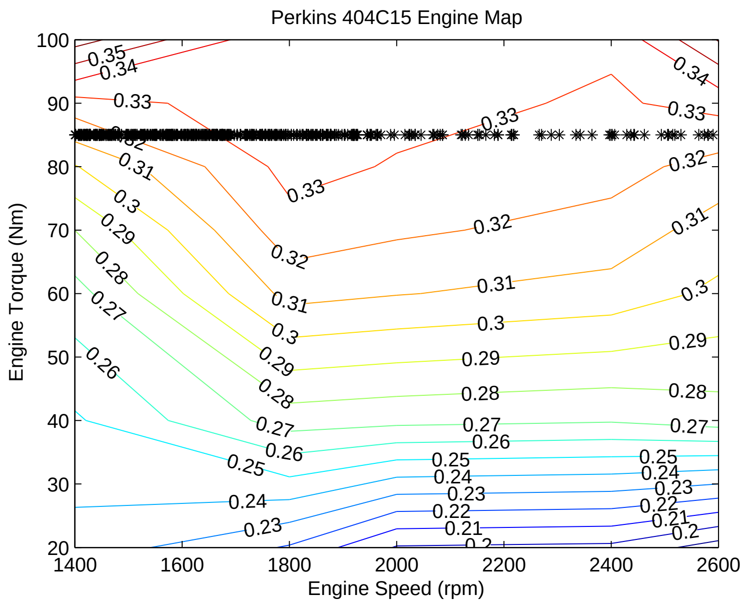


Figure 7.12: Engine operating points for turning the engine off using constant engine torque

Just as in the previous cases, to limit the engine switching rapidly between on and off, a penalty is added each time the engine switches between modes. For this case an engine penalty of 10 is used. The results from the dynamic programming are shown in Figure 7.13.

Comparing these results to the results without the penalty, one difference occurs right at the beginning of the drive cycle. While the strategy without the penalty keeps the engine off at the beginning of the drive cycle, the strategy with the penalty has the engine on to charge the accumulator. Also, the accumulator pressure reaches the maximum limit several times during the drive cycle, whereas the strategy without the penalty does not reach the upper limit once. This is even different than the previous

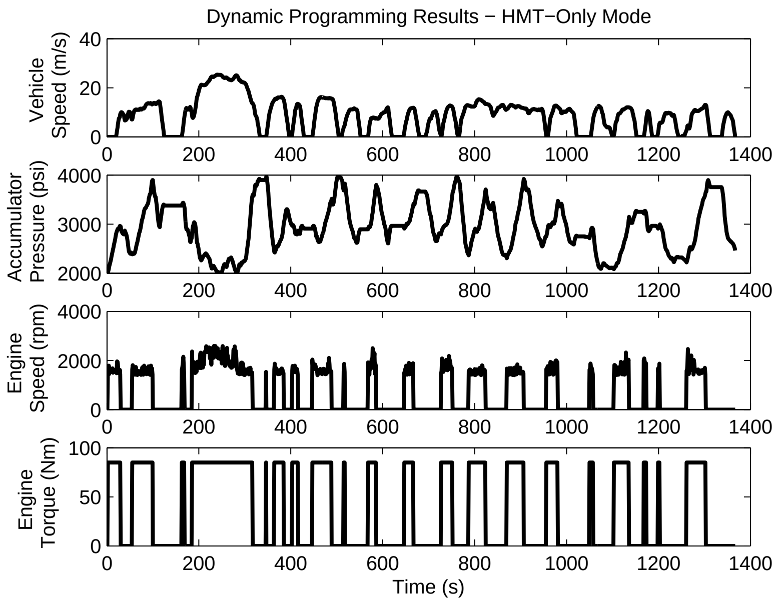


Figure 7.13: DP results for turning the engine off using constant engine torque and engine penalty

two cases where the engine stays clutched and where the engine is allowed to declutch. This is to allow the engine to stay on or off for as long as possible to minimize the penalty for engine switching.

Figure 7.14 shows the engine operating points for this strategy. Comparing these operating points to the previous cases, the operating points are very similar - as the engine speed increases, the number of operating points decreases. Using this strategy results in a fuel economy of 42 mpg, which is about a 13% improvement over the strategy of declutching the engine when the vehicle is stopped.

Best BSFC Curve

For these simulations, the best BSFC curve is used to determine the engine power. Just as in the constant engine torque case, the engine is allowed to declutch at anytime throughout the drive cycle, and when it declutches the engine turns off so no fuel is used.

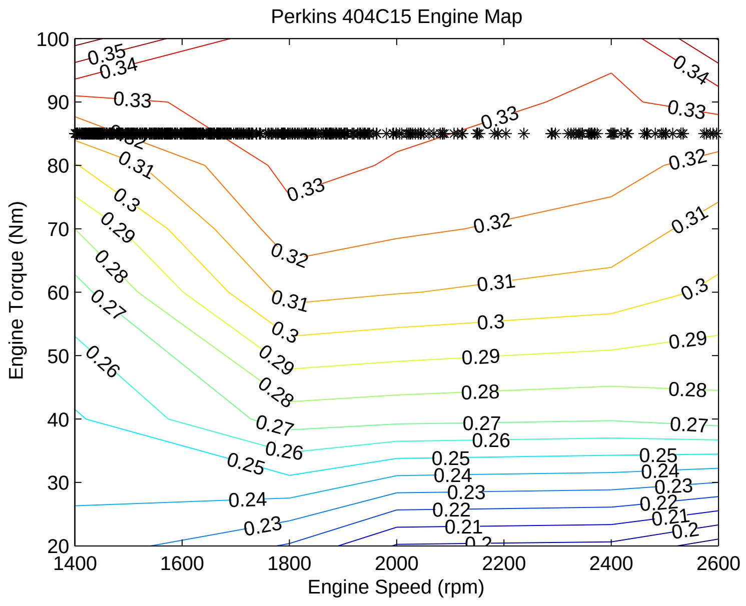


Figure 7.14: Engine operating points for turning the engine off using constant engine torque and engine penalty

Figure 7.15 shows the results from dynamic programming using this strategy. Comparing these results to the engine always on strategy in Figure 5.9, the accumulator pressure show slightly different characteristics. While they both maintain relatively low pressure, the engine off case maintains a slightly higher pressure than the engine always clutched case. Also, the pressure rises sooner in the drive cycle for the engine off case. For both cases the pressure is near the lower limit when the vehicle is operating near constant speed.

The engine operating points for this strategy are shown in Figure 7.16. Comparing these operating points to the operating points of declutching when stopped strategy in Figure 7.6, the declutched when stopped strategy has a cluster of operating points at low engine power, whereas the engine off strategy has only one point in the low engine power region. This shows, when the engine is on, it should be providing moderate to high power near the most efficient operating point. This strategy results in a fuel economy of 50.3 mpg, which is an improvement of about 25% over the declutch when stopped strategy.

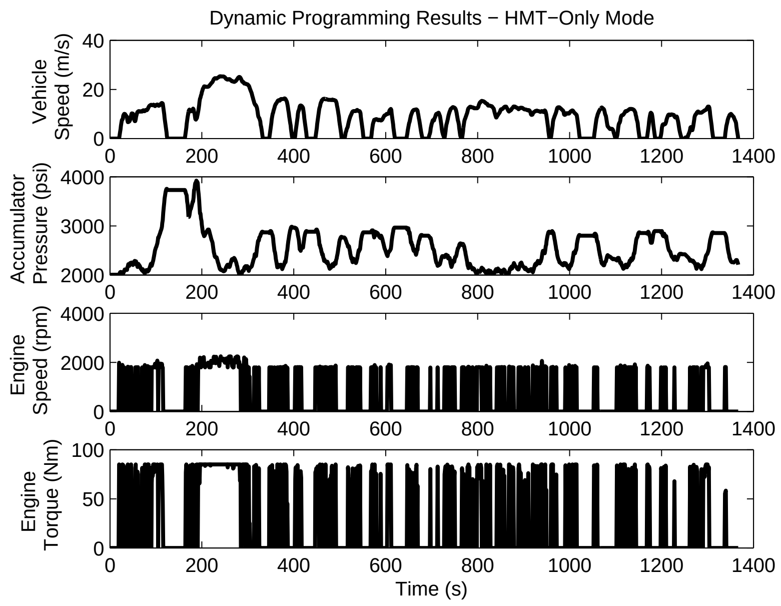


Figure 7.15: DP results for turning the engine off using best BSFC curve

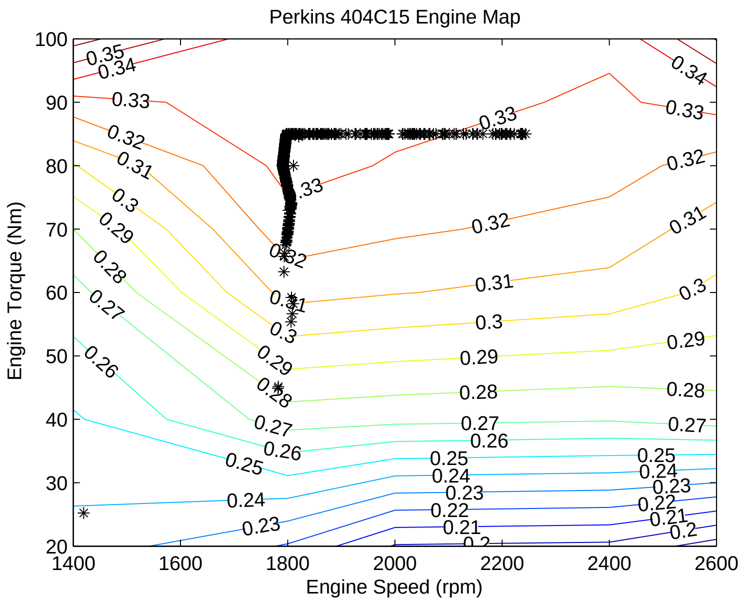


Figure 7.16: Engine operating points for turning the engine off using best BSFC curve

To reduce the rapid engine switching modes, an engine penalty of 1 is used each time the engine turns on or off. The dynamic programming results using this engine penalty with the engine off strategy are shown in Figure 7.17.

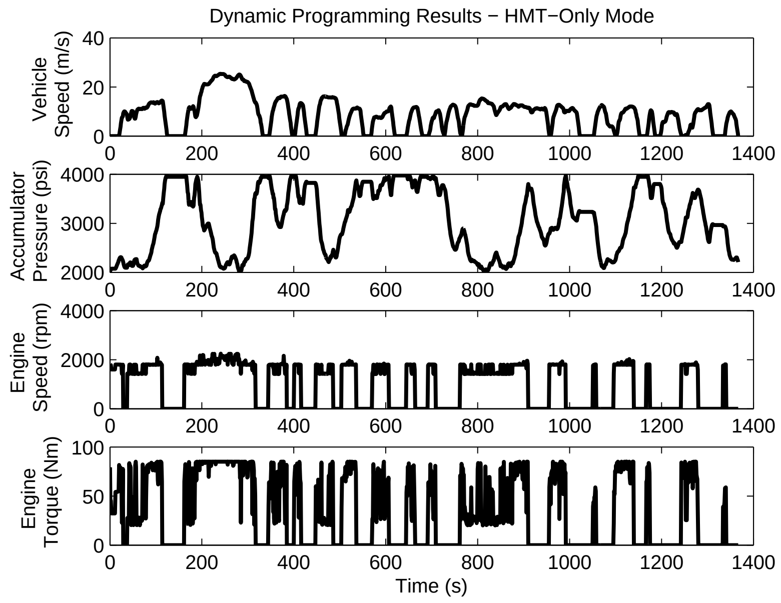


Figure 7.17: DP results for turning the engine off using best BSFC curve and engine penalty

The results show a completely different trajectory for the accumulator pressure compared to the previous cases. Whereas the engine always clutched and the engine declutched when stopped strategies try to maintain the pressure as low as possible, this strategy maintains a very high pressure, often times at or near the upper limit. This is so the engine is able to be on or off for longer periods of time, minimizing the penalties that are encountered during the drive cycle. A similarity between all the strategies is, when the vehicle is near a constant speed, the engine power fluctuates rapidly to maintain the accumulator pressure around a constant level.

Figure 7.18 shows the engine operating points for this strategy. Whereas without the penalty only one operating point was at a low engine power, now the engine operates on a wide spectrum of power. This is because, when the engine is not penalized, it prefers to turn off rather than operate at a low power. However, now with the penalty for

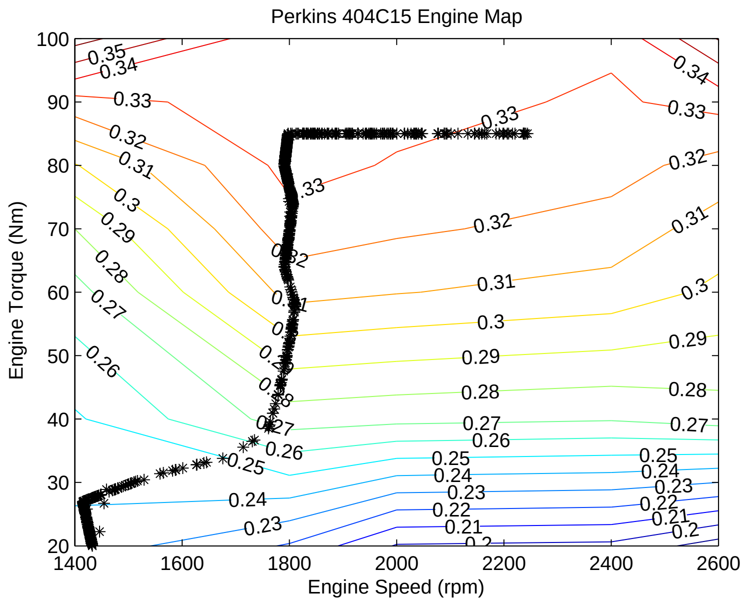


Figure 7.18: Engine operating points for turning the engine off using best BSFC curve and engine penalty

turning the engine off, the optimization would rather keep the engine on and operate at a lower power level. The fuel consumption for this strategy is 39.5 mpg, which is about a 4% improvement over the engine declutched when stopped strategy.

These results show that a significant improvement can be obtained by declutching the engine at any time and allowing the engine to shut off. Using the constant engine torque strategy, an improvement of over 10% can be achieved over just declutching the engine when stopped. Also, the results for turning the engine off were not as similar to the original strategy of always having the engine clutched to the drivetrain as the declutched when stopped strategy. Therefore, to obtain the best fuel economy possible using the engine off method, a new real-time strategy would need to be developed rather than using the existing one.

7.1.4 Four Mode Operation

One of the advantages of the power-split architecture is that it can operate in different modes since it is a combination of the parallel and series architectures. This is accomplished by turning certain components on and off. For this study, four distinct modes are used.

1. **HMT Mode.** This is the standard hydromechanical architecture that has been used throughout this research. The engine and both hydraulic units are used to power the vehicle.

2. **Parallel Mode.** This mode replicates a parallel architecture. Pump/motor S is locked and the valve is closed. In the simulation, the speed of pump/motor S is set to zero, and the flow through the unit is also set to zero.

3. **T-Only Mode.** This is similar to parallel mode except the engine is declutched from the drivetrain and turned off so the fuel consumption is zero. Just as in parallel mode, the speed and flow of pump/motor S are set to zero in the simulation.

4. **S-Only Mode**. This is similar to T-Only mode, except pump/motor T is locked and the flow is equal to zero. The engine is declutched from the drivetrain and turned off.

Figure 7.19 is a diagram that depicts each of these operating modes.

Dynamic programming is used to find the optimal control strategy for the four mode operation. A constant engine torque of 85 Nm is used when the engine is clutched to the drivetrain. Figure 7.20 shows the results from this optimization. The mode plot corresponds to the modes as defined above. Just as in the HMT mode, the accumulator pressure is kept relatively low and never reaches the upper limit. Also, when the vehicle is operating near constant speed, the accumulator pressure is kept near the lower limit. So even though the vehicle is allowed to operate in different modes, the results are similar to the previous results.

Figure 7.21 shows the engine operating points for the four mode operation. Again, just as in the previous studies, the engine operation is very similar by minimizing the number of operating points at high operating speeds. Rather, the optimization prefers to operate a low engine speeds between 1400 rpm and 2000 rpm.

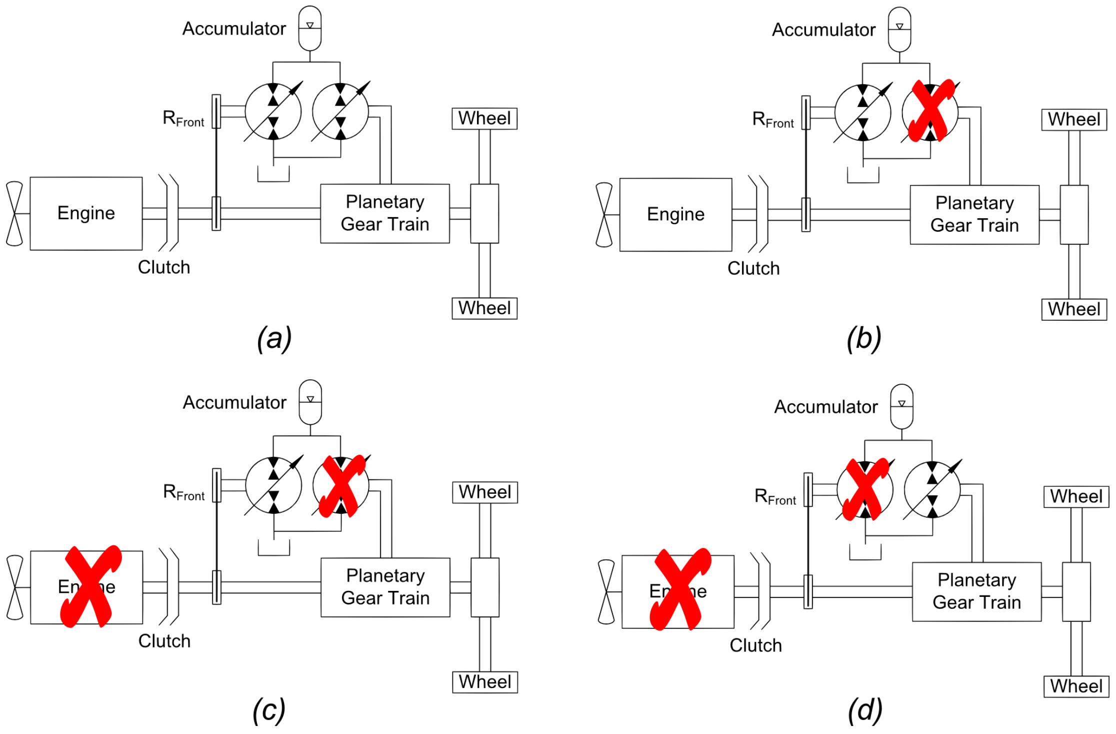


Figure 7.19: Schematic of four modes of operation. (a) HMT, (b) Parallel, (c) T-Only, (d) S-Only

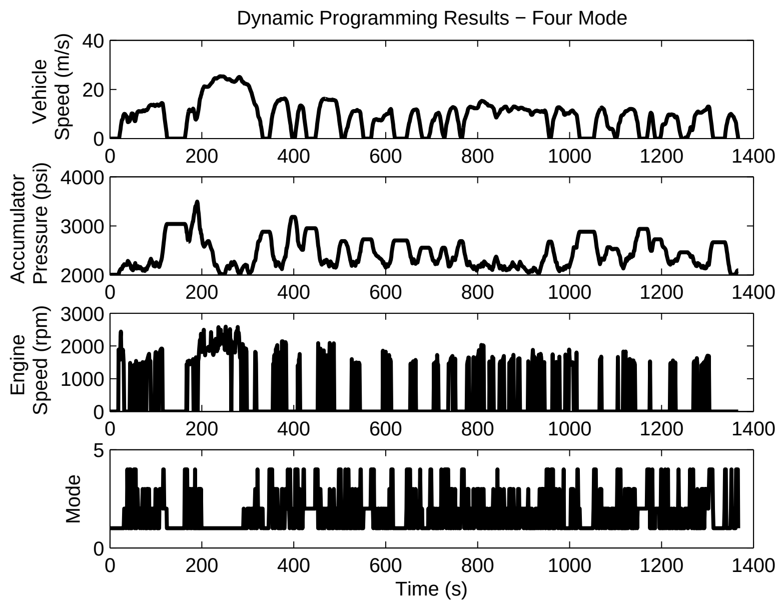


Figure 7.20: DP results for four mode operation

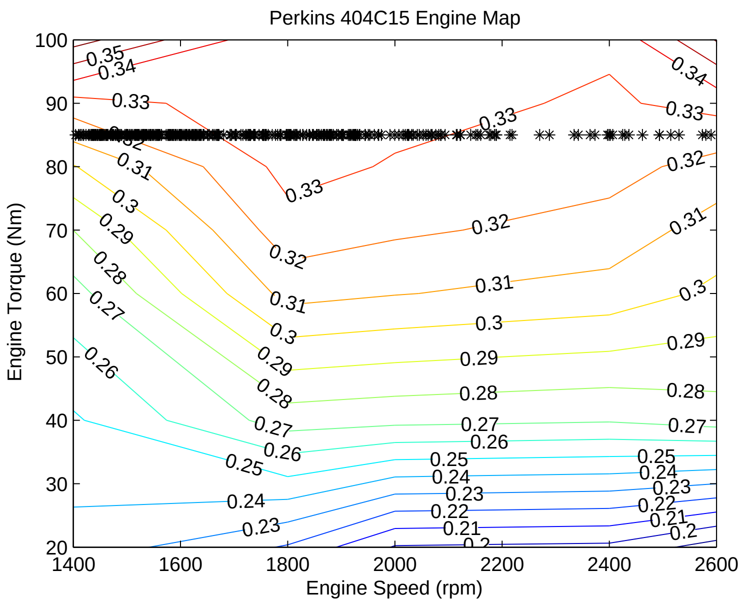


Figure 7.21: Engine operating points for four mode operation

Figure 7.22 shows a breakdown of the four mode operation during the drive cycle. The points when the vehicle is stopped are not included since it is not operating in any mode at these times. This shows that HMT mode is used the majority of the time, with parallel mode being the next most used mode. The resulting fuel economy using four modes is 63 mpg, which more than doubled the original baseline strategy.

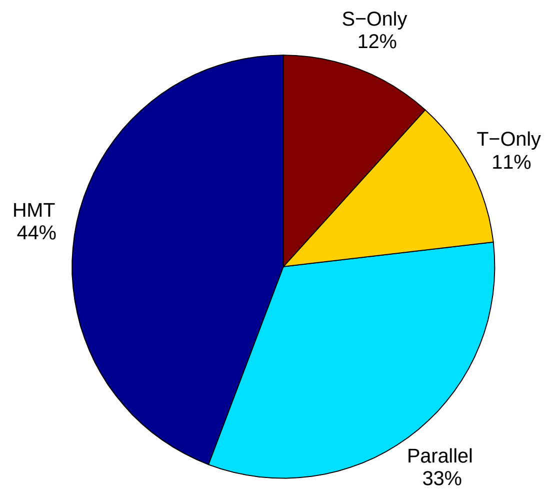


Figure 7.22: Percentage of operation in each mode

The dynamic programming results in Figure 7.20 show not only frequent changes with the engine between idling and providing power, but also changing between modes. To prevent this from happening, a penalty was added to the fuel consumption for both the engine changing states as well as when the mode changes. Just as in the engine penalty case, a penalty too high will lead to the system never changing states and little improvement for using four-modes. So an optimal penalty must be found as a balance. Even though the engine penalty and mode penalty could have different values, for this case each one having a value of one works the best. The results are shown in Figure 7.23.

The results show that the majority of the time during the drive cycle the engine is off. The engine comes on for short durations when the accumulator pressure is low. The

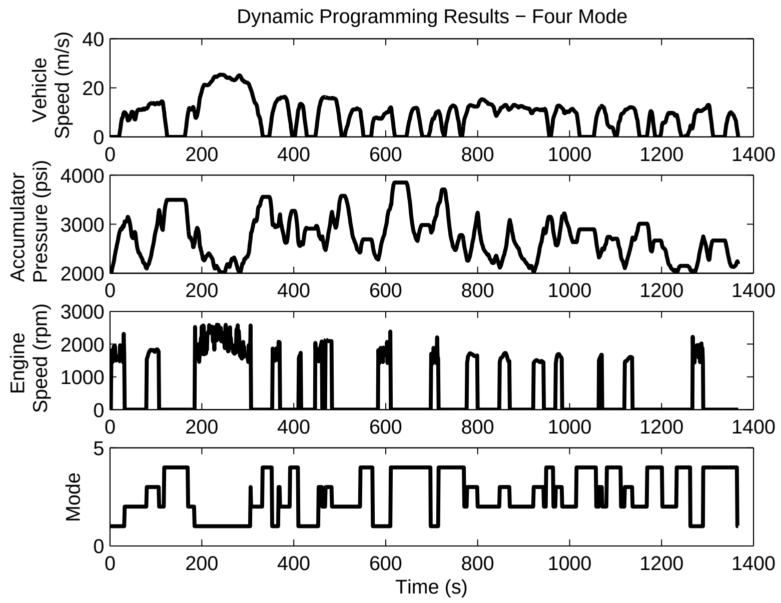


Figure 7.23: DP results for four mode operation with engine and mode penalties

engine is also on when the vehicle is traveling highway speeds, where it also operates in HMT mode.

Figure 7.24 shows the engine operating points using the engine and mode penalties. Event though the engine operation is quite different from the results with no penalties, the operating points are very similar to the results shown in Figure 7.21.

Figure 7.25 shows the percentage the system operates in each mode. As in the previous case, the times when the vehicle is stopped are not included. Comparing these results to those in Figure 7.22, the amount of time the system spends in parallel and T-Only modes is about the same. However, the time spent in S-Only mode is significantly more, while the time spent in HMT mode is less. This is because HMT is mostly used to charge the accumulator when the pressure is low, and S-Only for braking.

The results show a fuel consumption of 55 mpg. While this is about 12% less than the results without the penalty, it is an 80% improvement over the baseline strategy, and a 30% improvement over the next best optimized strategy of turning the engine off. This concludes that four-mode operation is the best way of operating the power-split architecture.

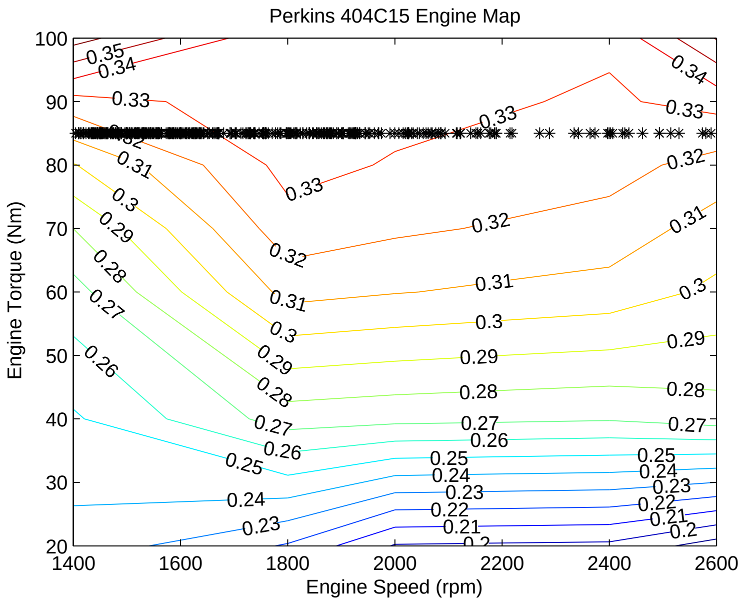


Figure 7.24: Engine operating points for four mode operation and engine and mode penalties

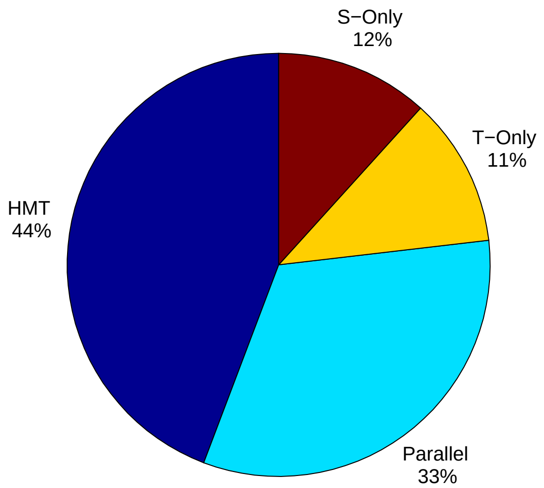


Figure 7.25: Percentage of operation in each mode with engine and mode penalties

7.1.5 Accumulator Sizing

One of the questions this research aimed to answer is which accumulator size is optimal to minimize fuel consumption. To do this, 100 random drive cycles are generated using the method described in Chapter 4. Three accumulator sizes are tried. The first is the size that is currently installed on the experimental vehicle - 38 liter. One was chosen that was half of this size (17 liter), and one was chosen that was double the size (76 liter). This gives a wide range from small to large. Along with the size, the mass was also varied. The 38 liter has a mass of approximately 100 pounds. For the 17 liter accumulator the mass was reduced by half to 50 pounds, and for the 76 liter accumulator the mass was doubled to 200 pounds.

Once the drive cycles are created, dynamic programming is used for each drivecycle for each accumulator size. The HMT architecture with engine always clutched to the drivetrain is used for these simulation. Since the constant engine torque gave slightly better fuel economy results in the previous studies, this strategy is chosen for the mid-level controller. Also, to ensure the engine would not frequently switch between idling and powering, a penalty of 10 is used each time the engine switches between modes.

The results for each accumulator size is shown in Figure 7.26. The bars represent the average fuel economy for the feasible drive cycles for that size, and the error bars represent the minimum and maximum values for each size. The results shows that the smallest accumulator has the highest average fuel economy at a little over 35 mpg, and as the accumulator size increases the fuel economy decreases. The smallest accumulator size also has the highest maximum and minimum values, while the largest accumulator size has the smallest maximum and minimum values. The 38 liter accumulator has a maximum value that is close to the maximum value for the smallest accumulator. Looking at these results, it would appear the best accumulator size is a small accumulator.

Table 7.1 summarizes the results from this study. The last column shows the number of feasible drive cycles out of the 100 random drive cycles generated. Even though the small accumulator size results in the best fuel economy, it could only complete 9 of the drive cycles, whereas the 38 liter and 76 liter accumulators could complete 84 and 90 of the drive cycles, respectively. This shows that the small accumulator size does not store enough energy for large accelerations, and since the engine was downsized, the engine is not able to provide enough supplemental power. Since one of the goals is to

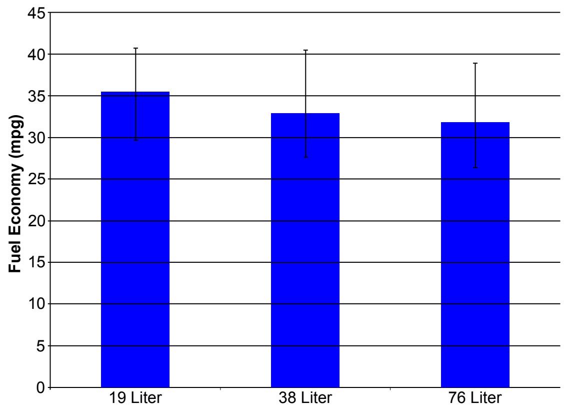


Figure 7.26: Accumulator size simulation results

Size	**Avg. Fuel Economy**	**Min. Fuel Economy**	**Max. Fuel Economy**	**No. Feasible Drivecycles**
17 Liter	35.5 mpg	29.7 mpg	40.8 mpg	9
38 Liter	32.9 mpg	27.7 mpg	40.5 mpg	84
76 Liter	31.8 mpg	26.4 mpg	38.9 mpg	90

Table 7.1: Summary of accumulator size simulation results

maintain the same performance of current non-hybrid vehicles on the market today, the current setup for the small accumulator would not be feasible. In order to use a smaller accumulator, a larger engine would be needed to be feasible for the majority of the drive cycles. One of the benefits of a hybrid vehicle is the ability to downsize the engine since the hydraulics are supplementing the engine power, so with a smaller accumulator size that benefit would be lost. Therefore, for a balance between performance and fuel economy, a 38 liter accumulator is the best for this architecture.

7.2 Augmented Earthmoving Vehicle Powertrain Simulator

With the simulation for each of the energy management strategies on the AEVPS validated with experimental results, the simulation was used to see how each strategy performed under different drive cycles. 100 random drive cycles were generated and each strategy was simulated for each drive cycle. The results are compared to a non-hybrid powertrain to obtain a fair comparison for how each would perform under different drive cycles. The fuel consumption and tracking error for each strategy is then compared.

7.2.1 Non-hybrid Model

To fairly compare the three strategies with different drive cycles, the results need to be compared to a non-hybrid powertrain. The non-hybrid model uses the same engine and vehicle loads as the AEVPS system. However, the hydraulic pump and motor are replaced with an ideal three gear transmission with a shifting policy based on motor speed. The ratios and policy were not optimized so better fuel economy may be achievable for the non-hybrid architecture, but the main objective of this simulation is for a baseline model. The gear policy is given by Equations (7.1)-(7.3).

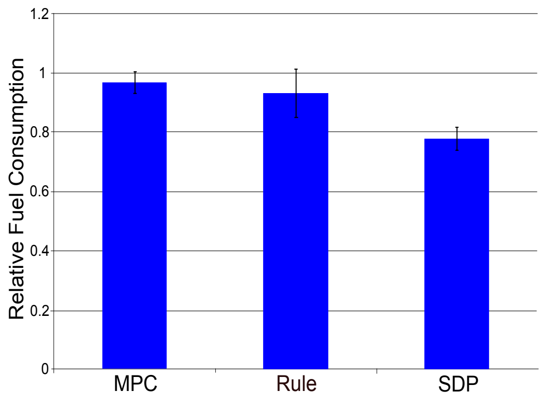
$$F_1(k) = \begin{cases} 2 & \text{For } \omega_m > 24.2 \\ 1 & \text{For } \omega_m < 21.0 \\ F_1(k-1) & \text{otherwise} \end{cases} \tag{7.1}$$

$$F_2(k) = \begin{cases} 1 & \text{For } \omega_m > 48.4 \\ 0 & \text{For } \omega_m < 41.9 \\ F_2(k-1) & \text{otherwise} \end{cases} \tag{7.2}$$

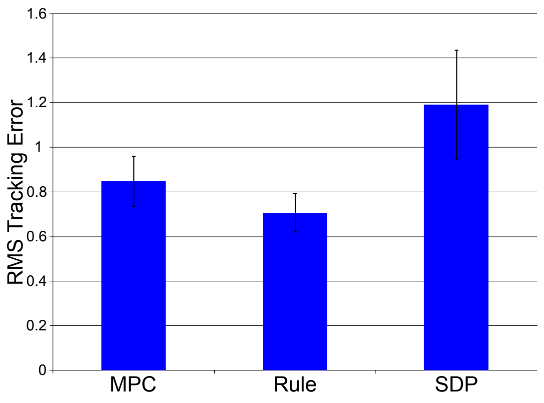
$$Gear(k) = \begin{cases} 7 & \text{If } F_1(k) + F_2(k) = 1 \\ 3.5 & \text{If } F_1(k) + F_2(k) = 2 \\ 1.75 & \text{otherwise} \end{cases} \tag{7.3}$$

7.2.2 Results

The results for the three different strategies for the 100 random drive cycles are shown in Figure 7.27. The error bars are ± 1 standard deviation. From the fuel consumption results, the SDP strategy achieved the best improvement, using about 25% less fuel than the non-hybrid case. The variance is also the smallest. This is expected since the random drive cycles are created using the same transition probability map that the SDP strategy uses. The rule-based strategy and MPC result in similar fuel consumption, but the variance for the rule-based strategy is greater. This is because the rule-based strategy is derived from one specific drive cycle, whereas the MPC strategy has no dependence on the drive cycle, so it makes sense that the rule-based strategy would be most sensitive to variations in the drive cycle. The RMS tracking error results show the opposite of the fuel consumption results. The rule-based strategy has the smallest tracking error with the smallest variance, whereas the SDP strategy had the largest tracking error with the largest variance. This was also seen in the results with the UDDS drive cycle by the oscillations in the control variables, and hence motor speed, at high speeds. Also, since the SDP strategy is developed offline in the form of a lookup table, it is more sensitive to model uncertainties. Based on these results, the rule-based appears to be the best strategy to balance fuel consumption and tracking error.



(a)



(b)

Figure 7.27: Mean fuel consumption (a) and mean RMS tracking error (b) relative to non-hybrid for urban driving

The same study is performed using 100 random highway drive cycles. As in the urban driving case presented in Chapter 5, the SDP strategy shows the least mean fuel consumption with the largest tracking error. The main difference is the rule-based strategy, resulting in the largest fuel consumption and poor tracking performance. This is expected since the rule-based strategy was developing using an urban drive cycle, so the performance in highway driving is expected to be worse. The MPC strategy has a mean fuel consumption between the two other strategies but much less tracking error with a smaller variance.

7.3 Concluding Remarks

In this chapter, variations in the control strategy were simulated to see what effect they have on the fuel economy. The accumulator size was also varied to determine the optimal size of accumulator to minimize fuel consumption. Finally, the different energy management strategies were tried over a variety of random drive cycles to determine the influence of driving characteristics on the control strategy. The results are summarized as follows.

7.3.1 Control Strategy Variations

For these studies, the power-split architecture was used to vary the original control strategy used in Chapter 5. The first strategy used was declutching the engine and running at idle when the vehicle was stopped. The second strategy was declutching and turning the engine off at any time in the drive cycle. Finally, the most complex strategy where the vehicle is able to operate in four different modes by not using the engine and/or the hydraulic units. Table 7.2 shows a summary of the different case studies.

Strategy	**Engine Penalty**	**Mode Penalty**	**Fuel Economy**	**% Over Baseline**
Baseline	N/A	N/A	30.5 mpg	N/A
DP Constant Engine Torque	0	N/A	40.6 mpg	33.1
Declutch When Stopped	10	N/A	37.2 mpg	22.0
DP Best BSFC Curve	0	N/A	40.2 mpg	31.8
Declutch When Stopped	1	N/A	38.0 mpg	24.6
Rule-based strategy with declutching the engine	N/A	N/A	36.8 mpg	20.7
DP Constant Engine Torque	0	N/A	50.7 mpg	66.2
Engine Off	10	N/A	42.0 mpg	37.7
DP Best BSFC Curve Engine	0	N/A	50.3 mpg	64.9
Off	1	N/A	39.5 mpg	29.5
DP Constant Engine Torque	0	0	63.0 mpg	106.6
Four Mode	1	1	55.0 mpg	80.3

Table 7.2: Summary of results for other operating modes using dynamic programming

The results from the different case studies are summarized as follows.

- By declutching the engine when the vehicle is stopped, a savings of over 13% is obtained over the case where the engine is always attached to the drivetrain. This shows the importance of being able to declutch the engine. Also, the results were very similar to the case where the engine is always clutched to the drivetrain, showing that the real-time implementable strategies for that case could also be applied to this case with little or no modification needed.

- By the engine having the capability to declutch at any time during the drive cycle and turn off resulted in another 13% improvement over the case when the engine declutched only when the vehicle is stopped. However, the results were different than the other strategies and therefore a new implementable strategy would need to be developed for this case.

- Operating the vehicle in four distinct modes gave the best fuel economy results out of all the strategies, and more than doubled the original baseline strategy. However, this strategy is also the most complex for developing a real-time implementable study since an additional control variable, mode, is added.

7.3.2 Accumulator Sizing

The effect of accumulator size on fuel economy was also studied using the power-split architecture by using three different accumulator sizes and running simulations through 100 random drivecycles that represent urban driving. The results from this study are summarized as follows.

- The smallest accumulator size resulted in the best fuel economy, while the largest had the worst fuel economy. The smallest size also had the highest maximum and minimum values, but the middle sized accumulator had a maximum value that was almost the same as the small accumulator size.

- The smallest accumulator size was only to complete 9% of the drive cycles, where the middle sized accumulator was able to complete 84% and the largest 90% of the drive cycles. In order for the small accumulator size to have the same performance as the larger sizes, a larger engine would be needed to provide more power. However, the fuel economy will then also decrease.

- The best is a trade-off between fuel economy and performance. Therefore, a middle sized accumulator should be used, which in this case is the 38 liter accumulator.

7.3.3 Drive cycle Effect on Control Strategies

Finally, the series architecture of the AEVPS system was simulated using different random drive cycle for each of the energy management strategies. The results show that each strategy performs differently under urban and highway driving. In both cases, the SDP strategy achieved the least fuel consumption with the highest tracking error. The rule-based strategy performed well in urban driving, with a fuel consumption slightly less than the MPC strategy but lowest tracking error, but used the most fuel in highway driving with poor tracking performance. Finally, the MPC strategy used more fuel in both urban and highway driving, but proved to be the most robust to changes in the drive cycle. Combining these observations, the following guidelines can be used when developing energy management strategies for hybrid vehicles.

- The rule-based strategy is best where the duty cycle is known with little variance. An example would be city buses that follow a prescribed route with stops along the way.

- The SDP strategy is best in predictable environments with models that have little uncertainties. An example would be a vehicle that is travelling in a city where stops will be frequent, but not exactly known when the vehicle will stop.

- The MPC strategy is best in uncertain environments where any type of driving could be encountered. An example might be a passenger vehicle where both highway and city driving will be present, but it is unknown when either will occur.

Chapter 8

Conclusions & Future Work

The goal of this research is to develop an energy management strategy for an input-coupled power-split architecture using a variety of methods and compare each strategy under different driving conditions and different hybrid architectures. From this research, the following contributions are made.

Operating the engine along an optimal curve may not lead to the most overall system operation. In this study, two different engine operating modes are used - the first operates the engine at constant torque, and the second operates the engine along the best BSFC curve. Dynamic programming is used to calculate the global optimal solution. The results from both are very similar, with the constant engine torque being slightly less than the best BSFC curve. That is because, for this case, the penalty for operating the engine a little less efficiently is less than the penalty of operating the hydraulic units inefficiently. Even though this may not be true for all systems, operating the engine as efficiently as possible should not be assumed for overall system efficient operation.

A systematic approach to developing an implementable real-time strategy. The dynamic programming results cannot be implemented directly, and therefore a real-time strategy must be developed. A rule-based strategy is developed by doing a linear regression of the dynamic programming results. This method is less time consuming than trying to find how operating points are grouped together. While the rule-based strategy was slightly less than the dynamic programming solution, an improvement was made over a baseline control strategy based on intuition, which constrained the engine to operating at the most efficient point except during high vehicle speeds.

Using stochastic dynamic programming to develop a real-time control strategy for a hydraulic hybrid. Stochastic dynamic programming differs from dynamic programming because the solution is directly implementable in real-time, whereas the dynamic programming solution requires that the future is exactly known. Transition probabilities are calculated from a combination of urban and highway drive cycles. These probabilities are used to form a lookup table of the control decision based on the states. However, through simulation it was found that all the results from the lookup table may not be feasible for the drive cycle because of using probabilities. With some modifications taking into account vehicle speed and accumulator pressure to allow the vehicle to complete the drive cycle, the strategy did better than the rule-based strategy. The results are summarized in Table 8.1 below.

Strategy	**Engine Penalty**	**Eng. Speed Weight**	**Fuel Economy**	**% Over Baseline**
Baseline	N/A	N/A	30.5 mpg	N/A
DP Constant Engine Torque	0	0	36.5 mpg	20.7
	10	0	32.8 mpg	7.5
DP Best BSFC Curve	0	0	36.6 mpg	20
	1	0	32.6 mpg	6.9
	1	0.0005	32.0 mpg	4.9
Rule-Based	N/A	N/A	32.4 mpg	6.2
SDP	N/A	N/A	31.4 mpg	2.8
SDP Modified When Stopped	N/A	N/A	33.1 mpg	8.7

Table 8.1: Summary of results from rule-based and SDP strategies

Importance of declutching the engine. The above results were simulated with the engine always clutched to the drivetrain. To see the effect this has on fuel economy, a two other scenarios are simulated. The first allows the engine to declutch and run at idle when the vehicle is stopped. The second allows the engine to declutch and turn off at any point in the drive cycle. The results show that by declutching the engine from the drivetrain just when the vehicle is stopped results in a fuel economy improvement of over 10% for both cases. This is a significant improvement for a simple control action. If the engine is allowed to declutch and turn off at any time, an additional 10-15% improvement can be achieved. However, this would require a more complex controller to ensure smooth operation between the engine clutching and declutching from the drivetrain.

The benefit of the power-split architecture to operate in four modes. An advantage of the power-split architecture is the ability to operate as a parallel architecture and series architecture by locking up different components. For example, by locking up the hydraulic unit attached to the planetary gearset, the vehicle is able to operate in parallel mode. The simulation results show that this four mode operation can more than double the fuel economy compared to the original baseline strategy, and an almost 25% improvement over the best solution for power-split only operation.

The effect of accumulator size on fuel economy, and a larger accumulator is not necessarily better. Different accumulator sizes are also studied in simulation. An accumulator that is half the size and double the size of the one currently installed are examined. For this study, 100 random drive cycles are generated that represent urban driving, and dynamic programming is used to find the optimal solution for each. The results show that a small accumulator size is the best for fuel economy, while a large accumulator is the worst. However, the performance decreases with a smaller accumulator, which would therefore be offset since a larger engine would then be needed to provide the same performance.

Comparison between different optimization methods. A series hybrid architecture is examined to determine how easy the algorithms are to change from one architecture to another with different components, along with a comparison of the different methods to different driving conditions. The rule-based and SDP solutions are used, along with a model predictive control solution. The results show that the SDP solution did the best in terms of fuel economy, while the MPC result had the highest fuel consumption over 100 random drive cycles. However, the SDP solution had the highest RMS tracking error to the reference drive cycle trajectory while the MPC strategy had the lowest. Therefore, the SDP strategy works well when the confidence level in the model is high with little disturbances, while the MPC strategy is the most robust to disturbances. Table 8.2 summarizes the results from this study for urban driving.

	Fuel Consumption Rel. to Non-Hybrid			RMS Error		
	MPC	Rule	SDP	MPC	Rule	SDP
Mean	0.9661	0.9307	0.778	0.8474	0.7076	1.1914
Std. Dev.	0.0359	0.0807	0.0383	0.1128	0.087	0.2443

Table 8.2: Summary of results from AEVPS for urban driving

8.1 Future Work

This research has laid the foundation for developing energy management strategies for hydraulic hybrid vehicles. One of the results from this study is that there is no one strategy that works best for every application. Therefore, a toolbox should be developed that makes it easy to design energy management strategies for different applications based on user input. For example, a user could specify models, efficiency maps, an objective function, and other inputs such as energy storage, knowledge of duty cycle, etc., and the toolbox would determine which method to use to develop the energy management strategy. An application where the duty cycle is well knows, such as a bus, would use the rule-based strategy, whereas an application with many unknowns about the duty cycle, such as a passenger vehicle for a family, would use the model predictive control strategy. This would automate the process of the energy management development. This would also allow the user to change different components in the model and simulate the performance using an optimal control strategy without having to worry about the development of that strategy.

Also, different case studies should be studied. This research looked at a passenger vehicle, but it should be used for other applications such as wheel loaders, excavators, and refuse trucks to compare their performance of each strategy to the passenger vehicle case.

Finally, in this research only one method was used during each trial. Since the different strategies are better suited for different parameters, the optimization techniques should be combined throughout the duty cycle. This is especially true since the system contains discrete states with engine on/off and continuous states with amount of power from the engine. As an example, from the AEVPS study, it was shown that the SDP algorithm gave the best fuel economy but the worst tracking performance. Meanwhile, the MPC formulation had the best tracking but the worst fuel economy. These could be combined into one optimization, using SDP to determine when the engine should be idling or providing power and the MPC method to determine the displacement and valve commands. This would give the fuel economy of the SDP algorithm while also achieving the tracking performance of the MPC algorithm.

References

[1] Annual energy review 2011. Technical report, U.S. Energy Information Administration, 2012.

[2] California Energy Commission. Consumer energy center. `http://www.consumerenergycenter.org/transportation/consumer_tips/vehicle_energy_losses.html`, 2013.

[3] U.S. Energy Information Administration. Gasoline and diesel fuel update. `http://www.eia.gov/petroleum/gasdiesel/`, March 2013.

[4] H. Tu, M. Rannow, J. Van de Ven, M. Wang, P. Li, and T. Chase. High speed rotary pulse width modulated on/off valve. In *Proceedings of the ASME International Mechanical Engineering Congress*, 2007.

[5] Peter A.J. Achten. Power density of floating cup axial piston principle. In *ASME-Fluid Power and Systems Technology Division*, Anaheim, CA, 2004.

[6] Tom Shelley. Digital pumps finally get wheels. *Eureka*, 26(10):65, 2006.

[7] P. Y. Li, J. D. Van de Ven, and C. Sancken. Open accumulator concept for compact fluid power storage. In *Proceedings of the ASME International Mechanical Engineering Congress*, 2007.

[8] J. H. Kress. Hydrostatic power-splitting transmission for wheeled vehicles - Classification and theory of operation. *SAE Paper 680549*, 1968.

[9] N. H. Beachley and F. J. Fronczak. Control of an accumulator energy-storage automobile using a single hydrostatic pump/motor. In *Proceedings of the International Conference on Fluid Power*, pages 1461–1475, September 1985.

[10] N. H. Beachley. Control strategies for an internal combustion engine/accumulator automobile. In *Proceedings of the Symposium on Advanced and Hybrid Vehicles*, pages 72–82, September 1984.

[11] N. H. Beachley. The "All-Hydraulic"car - Economy and performance through design integration. In *Proceedings of the 43rd National Conference on Fluid Power*, October 1988.

[12] F. J. Fronczak and N. H. Beachley. An integrated hydraulic drive train system for automobiles. In *Proceedings of the 8th International Symposium on Fluid Power*, 1988.

[13] F. T. Elder and D. R. Otis. Simulation of a hydraulic hybrid vehicle power train. *ASME Paper 73-ICT-50*, 1973.

[14] A. Lynn, E. Smid, M. Eshraghi, N. Caldwell, and D. Woody. Modeling hydraulic regenerative hybrid vehicles using AMESim and Matlab/Simulink. In *Proceedings of SPIE - Enabling Technologies for Simulation Science IX*, 2005.

[15] U.S. Department of Energy. Autonomie. `http://www.autonomie.net/index.html`, December 2013.

[16] D. E. Bowns, N. D. Vaughan, and R. E. Dorey. Design study of a regenerative hydrostatic split power transmission for a city bus. In *Hydrostatic Transmissions for Vehicle Application*, 1981.

[17] P. Buchwald, G. Christensen, H. Larsen, and P. Sunn Pedersen. Improvement of citybus fuel economy using a hydraulic hybrid propulsion system - A theoretical and experimental study. *SAE Preprints*, (790305), 1979.

[18] M. K. Vint and D. B. Gilmore. Simulation of transit bus regenerative braking systems. *Mathematics and Computers in Simulation*, 30(1-2):55–61, 1987.

[19] P. L. Matheson and J. S. Stecki. Development of hybrid diesel-hydraulic system for large commercial vehicles. In *Proceedings of the 8th Scandinavian International Conference on Fluid Power*, 2003.

[20] P. Matheson and J. Stecki. Modeling and simulation of a fuzzy logic controller for a hydraulic-hybrid powertrain for use in heavy commercial vehicles. *SAE Paper 2003-01-3275*, 2003.

[21] P. Matheson and J. Stecki. Development and simulation of a hydraulic-hybrid powertrain for use in commercial heavy vehicles. *SAE Paper 2003-01-3370*, 2003.

[22] P. L. Matheson and J. S. Stecki. Optimisation of a hybrid diesel-hydraulic automotive powertrain using ADVISOR, Matlab and Simulink. *Australian Journal of Mechanical Engineering*, 2(1):43–50, 2005.

[23] J. V. Svoboda and S. Sankar. Performance optimization of a hydraulic hybrid vehicular drive. In *Proceedings of the 5th International Fluid Power Symposium*, pages 41–50, September 1978.

[24] J. V. Svoboda, S. Sankar, and W. Blach. Hybrid computer-aided design of a hydrostatic vehicle drive with energy accumulator. *ASME Paper 80-DET-49*, 1980.

[25] S. Tollefson, N. H. Beachley, and F. J. Fronczak. Studies of an accumulator energy-storage automobile design with a single pump/motor unit. *SAE Paper 851677*, 1985.

[26] P. Wu, N. Luo, F. J. Fronczak, and N. H. Beachley. Fuel economy and operating characteristics of a hydropneumatic energy storage automobile. *SAE Paper 851678*, 1985.

[27] L. O. Hewko and T. R. Weber. Hydraulic energy storage based hybrid propulsion system for a terrestrial vehicle. In *Proceedings of the 25th Intersociety Energy Conversion Engineering Conference*, 1990.

[28] S. Martini. The M.A.N. hydrobus: A drive concept with hydrostatic brake energy recovery. In *Proceedings of the Symposium on Advanced and Hybrid Vehicles*, pages 227–234, September 1984.

[29] N. Nakazawa, Y. Kono, E. Takao, and N. Takeda. Development of a braking energy regeneration system for city buses. *SAE Paper 872265*, 1988.

[30] J. S. Stecki, F. Conrad, P. Matheson, and A. Rush. Development of a hydraulic drive for a novel hybrid diesel-hydraulic system for large commercial vehicles. In *Proceedings of the 5th JFPS International Symposium of Fluid Power*, volume 2, pages 425–430, 2002.

[31] E. J. Lewandowski. Design of a test rig for hydropneumatic accumulator energy storage vehicle research. Master's thesis, University of Wisconsin-Madison, 1987.

[32] E. Lewandowski, D. Benson, F. Fronczak, and N. Beachley. Test rig design for hydrostatic accumulator energy-storage automobile research. In *Trends in Vehicle Design Research - ASME Winter Annual Meeting*, 1987.

[33] S. A. Baum. Investigation of a dual pump/motor hydraulic accumulator energy storage automobile. Master's thesis, University of Wisconsin-Madison, 1987.

[34] D. J. Benson. Experimental evaluation of the controllability of a dual pump/motor hydraulic accumulator energy storage automobile. Master's thesis, University of Wisconsin-Madison, 1987.

[35] R. P. Kepner. Hydraulic power assist - A demonstration of hydraulic hybrid vehicle regenerative braking in a road vehicle application. *SAE Paper 2002-01-3128*, 2002.

[36] C.-C Lin, J.-M Kang, J. W. Grizzle, and H. Peng. Energy management strategy for a parallel hybrid electric truck. In *Proceedings of the American Control Conference*, 2001.

[37] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang. Power management strategy for a parallel hybrid electric truck. *IEEE Transactions on Control Systems Technology*, 11(6):839–849, 2003.

[38] L. Vermeiren, T. M. Guerra, and G. Paganelli. Application of fuzzy set theory to electric car control. *Cybernetics and Systems*, 28(8):675–693, 1997.

[39] A. Rajagopalan, G. Washington, G. Rizzoni, and Y. Guezennec. Development of fuzzy logic and neural network control and advanced emissions modeling for parallel hybrid vehicles. Technical Report NREL/SR-540-32919, National Renewable Energy Laboratory, 2003.

[40] J.-S. Won and R. Langari. Fuzzy torque distribution control for a parallel hybrid vehicle. *Expert Systems*, 19(1):4–10, 2002.

[41] N. Schouten, M. Salman, and N.A. Kheir. Fuzzy logic control for parallel hybrid vehicles. *IEEE Transactions on Control Systems Technology*, 10(3):460–468, May 2002.

[42] G. Paganelli, S. Delprat, T. M. Guerra, J. Rimaux, and J. J. Santin. Equivalent consumption minimization strategy for parallel hybrid powertrains. In *Proceedings of the 55th Vehicular Technology Conference*, May 2002.

[43] K. Ahn, S. W. Cha, J. M. Lee, and S. Cho. Three types of simulation algorithms for evaluating the HEV fuel efficiency. *SAE Paper 2007-01-1771*, 2007.

[44] K. Ahn, S. Cho, and S. W. Cha. Optimal operation of the power-split hybrid electric vehicle powertrain. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 222(5):789–800, 2008.

[45] A. Phillips M.L. Kuang I.V. Kolmanovsky H.A. Borhan, A. Vahidi.

[46] A. Vahidi A. Phillips M.L. Kuang S. Di Cairano H.A. Borhan, C. Zhang.

[47] T. Kaijiang K. Yu, M. Mukai. Model predictive control of a power-split hybrid electric vehicle system. *Artificial Life and Robotics*, 17(2):221–226, 2012.

[48] I. Kolmanovsky, I. Siverguina, and B. Lygoe. Optimization of powertrain operating policy for feasibility assessment and calibration: Stochastic dynamic programming approach. In *Proceedings of the American Control Conference*, May 2002.

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

[50] L. Johannesson, M. Asbogard, and B. Egardt. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):71–83, 2007.

[51] Z. Filipi, L. Louca, B. Daran, C.-C. Lin, U. Yildir, B. Wu, M. Kokkolaras, D. Assanis, H. Peng, P. Papalambros, J. Stein, D. Szkubiel, and R. Chapp. Combined

optimisation of design and power management of the hydraulic hybrid propulsion system for the 6 × 6 medium truck. *International Journal of Heavy Vehicle Systems*, 11(3-4):372–402, 2004.

[52] K. Wu, Q. Zhang, and A. Hansen. Modelling and identification of a hydrostatic transmission hardware-in-the-loop simulator. *International Journal of Vehicle Design*, 34(1):52–64, 2004.

[53] R. Kumar and M. Ivantysynova. An optimal power management strategy for hydraulic hybrid output coupled power-split transmission. In *Proceedings of the ASME Dynamic Systems and Control Conference 2009*, pages 299–306, 2009.

[54] R. Kumar and M. Ivantysynova. An instantaneous optimization based power management strategy to reduce fuel consumption in hydraulic hybrids. *International Journal of Fluid Power*, 12(2):15–25, August 2011.

[55] J. D. Van de Ven, M. W. Olson, and P. Y. Li. Development of a hydro-mechanical hydraulic hybrid drive train with independent wheel torque control for an urban passenger vehicle. In *Proceedings of the International Fluid Power Exposition*, March 2008.

[56] P.Y. Li T.R. Chase Z. Du, K.L. Cheong. Fuel economy comparisons of series, parallel and hmt hydraulic hybrid architectures. In *Proceedings of the 2013 American Control Conference*, pages 5954–5959, June 2013. Washington, D.C.

[57] P.Y Li T.P. Sim. Analysis and control design of a hydro-mechanical hydraulic hybrid passenger vehicle. In *Proceedings of the ASME Dynamic Systems and Control Conference 2009, DSCC2009*, pages 1579–1586, 2010.

[58] P.Y. Li K.L. Cheong, Z. Du and T.R. Chase. Hierarchical control strategy for a hybrid hydro-mechanical transmission (hmt) power-train. In *Proceedings of the 2014 American Control Conference*, June 2014. Portland, OR.

[59] P.Y. Li F. Mensing. Sizing and optimal operation of a power split hydraulic hybrid drive train. In *Proceedings of the International Fluid Power Exposition (IFPE)*, March 2011. Las Vegas, NV.

[60] F. Mensing P.Y. Li. Optimization and control of a hydro-mechanical transmission based hydraulic hybrid passenger vehicle. In *Proceedings of the 7th International Fluid Power Conference*, March 2010. Aachen, Germany.

[61] S. Sedler T.R. Chase K.L. Cheong, P.Y. Li. Comparison between input coupled and output coupled power-split configurations in hybrid vehicles. In *Proceedings of the International Fluid Power Exposition (IFPE)*, March 2011. Las Vegas, NV.

[62] T.R. Chase K.L. Cheong, P.Y. Li. Optimal design of power-split transmission for hydraulic hybrid passenger vehicles. In *Proceedings of the 2011 American Control Conference*, pages 3295–3300, June 2011. San Francisco, CA.

[63] A. Alleyne T. Deppen J. Meyer, K. Stelson. Power management strategy for a parallel hydraulic hybrid passenger vehicle using stochastic dynamic programming. In *Proceedings of the 7th International Fluid Power Conference*, 2010.

[64] A. Alleyne T. Deppen J. Meyer, K. Stelson. Energy management strategy for a hydraulic hybrid vehicle using stochastic dynamic programming. In *Proceedings of the 6th FPNI PhD Symposium*, 2010.

[65] D. Li D. Feng, D. Huang. Stochastic model predictive energy management for series hydraulic hybrid vehicle. In *IEEE International Conference on Mechatronics and Automation*, 2011.

[66] D. R. Grandall. The performance and efficiency of hydraulic pumps and motors. Master's thesis, University of Minnesota-Twin Cities, 2010.

[67] A. Pourmovahed, S.A. Baum, F.J. Fronczak, and N.H. Beachley. Experimental evaluation of hydraulic accumulator efficiency with and without elastometic foam. *Journal of Propulsion and Power*, 4(2):185–192, 1988.

[68] T.O. Deppen. *Optimal Energy Use In Mobile Applications With Storage.* PhD thesis, University of Illinois-Urbana/Champaign, 2013.

[69] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 & 2. Athena Scientific, Belmont, Massachusetts, 2005.

[70] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[71] Abhijit Gosavi. *Simulation-based optimization: Parametric Optimization Techniques and Reinforcement Learning.* Kluwer Academic Publishers, 101 Philip Drive, Norwell, MA, 2003.

[72] United States Department of Transportation Federal Highway Administration. Travel time to work. `http://www.fhwa.dot.gov/ohim/onh00/map1.htm`, 2011.

[73] K.A. Stelson J.J. Meyer T.O. Deppen, A.G. Alleyne. Optimal energy use in a light weight hydraulic hybrid passenger vehicle. *Journal of Dynamic Systems, Measurements, and Control*, 134(4), 2011.

[74] United States Environmental Protection Agency. EPA Urban Dynamometer Driving Schedule (UDDS). `http://www.epa.gov/otaq/standards/light-duty/udds.htm`, 2012.

[75] H.J. Kohring. Design and construction of a hydrostatic dynamometer for testing a hydraulic hybrid vehicle. Master's thesis, University of Minnestoa-Twin Cities, 2012.

[76] D. Carter and A. Alleyne. Load modeling and emulation for an earthmoving vehicle powertrain. In *Proceedings of the American Control Conference*, pages 4963–4968, 2003.

[77] E.A. Prasetiawan. Modeling, simulation and control on an earthmoving vehicle powertrain simulator. Master's thesis, University of Illinois-Urbana/Champaign, 2001.

[78] R. Zhang, E. Prasetiawan, and A. Alleyne. Modeling and H_2/H_∞ MIMO control of an earthmoving vehicle powertrain. *Journal of Dynamic Systems, Measurements, and Control*, 124(4):625–636, 2002.

Appendix A

Hydraulic Hybrid Vehicle Parameters

Parameter	**Value**
Accumulator expansion ratio	3
Accumulator precharge pressure	11 MPa
Accumulator size	38 liters
Drag coefficient	0.5
Efficiency of gears	1
Frontal area	1.784 m^2
Gear ratio - differential	3.45
Gear ratio - planetary gearset	0.25
Gear ratio - hydraulic unit S	2
Gear ratio - hydraulic unit T	1.3
Mass	1125 kg
Maximum displacement - hydraulic unit S	28 cc/rev
Maximum displacement - hydraulic unit T	28 cc/rev
Rolling resistant coefficient fo	0.0095
Rolling resistant coefficient fs	0.0035
Wheel diameter	0.610 m

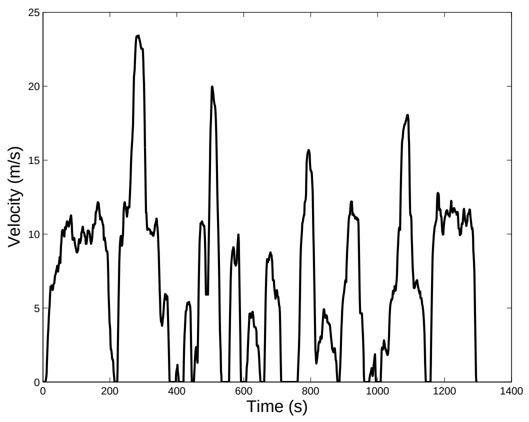
Appendix B

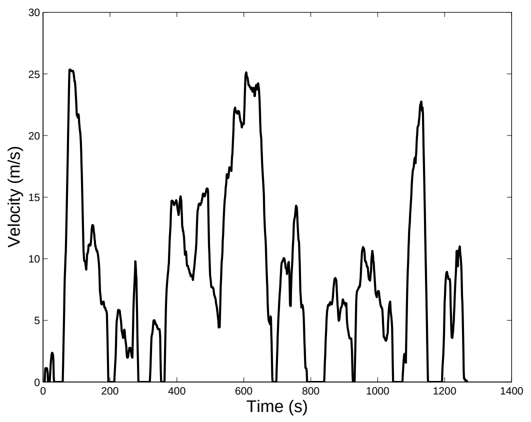
AEVPS Parameters

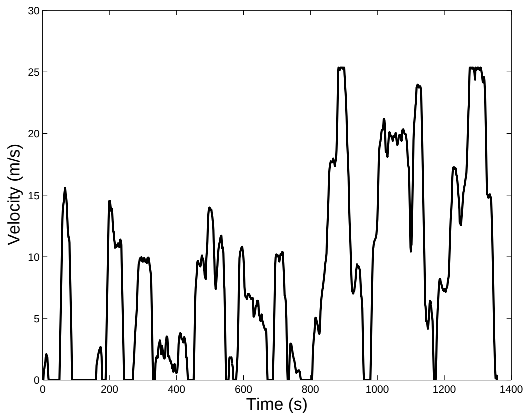
Parameter	**Value**
Accumulator precharge pressure	5.17 MPa
Accumulator size	1.89×10^{-4} cm^3
Bulk modulus downstream of valve	53.23 MPa
Bulk modulus upstream of valve	266.13 MPa
Drag coefficient	0.4
Engine inertia	0.383 kg-m^2
Engine friction coefficient	2.45×10^{-4} (N-m-s)/rad
Frontal area	2 m^2
Leakage coefficient downstream of valve	9.259 cm^3/(MPa-s)
Leakage coefficient upstream of valve	0.7 cm^3/(MPa-s)
Motor damping	0.0463 (N-m-s)/rad
Motor displacement	4.005 cm^3/rad
Motor inertia	0.0019 kg-m^2
Pump displacement	4.216 cm^3/rad
Pump flow gain	37.4 cm^3/rad^2
Specific heat ratio	1.4
Volume downstream of valve	1854 cm^3
Volume upstream of valve	2785 cm^3
Wheel damping	1 (N-m-s)/rad
Wheel radius	0.31 m

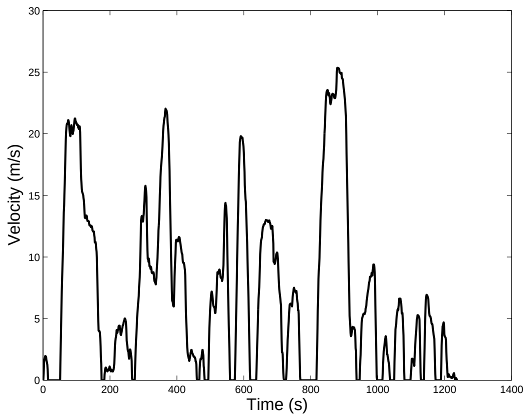
Appendix C

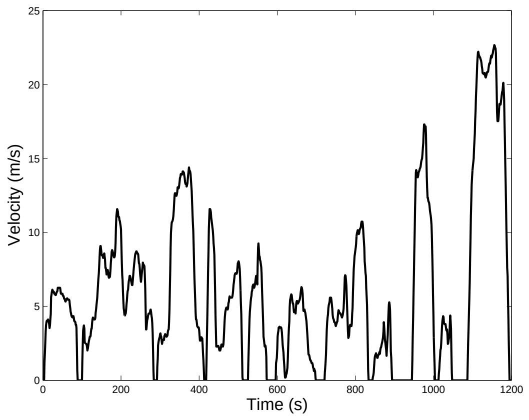
Random Drivecycles

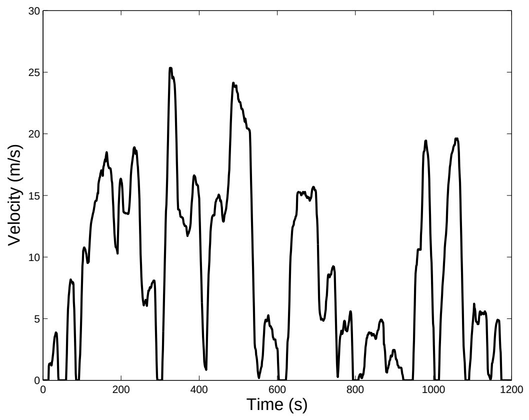


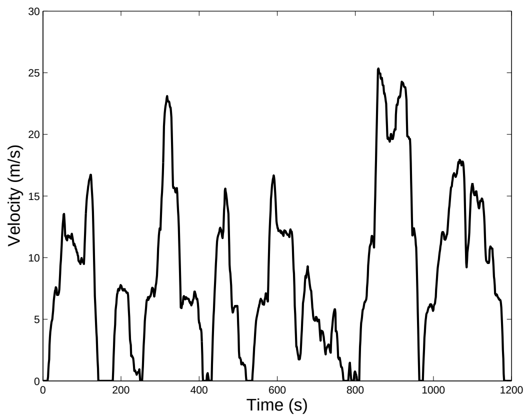


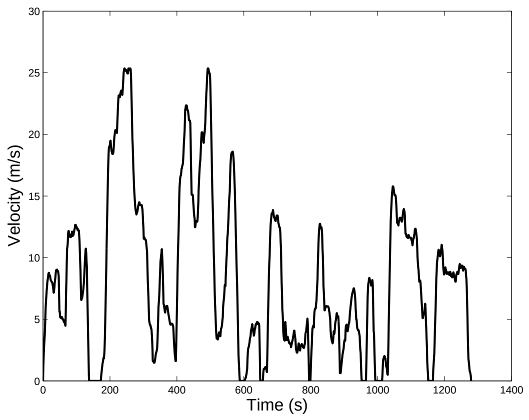

Velocity (m/s)
Time (s)

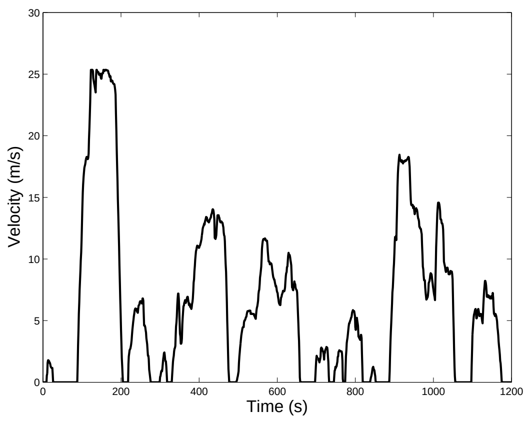

Velocity (m/s)
Time (s)

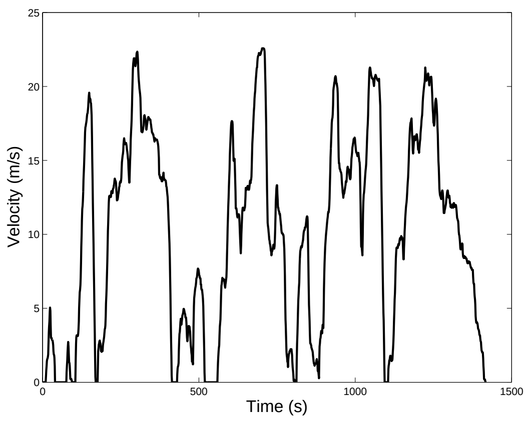

Velocity (m/s)
Time (s)

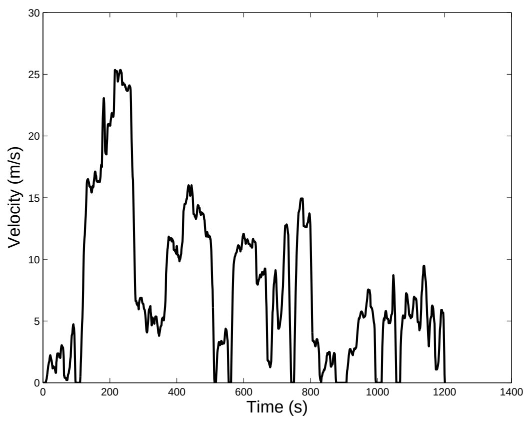

Velocity (m/s)
Time (s)

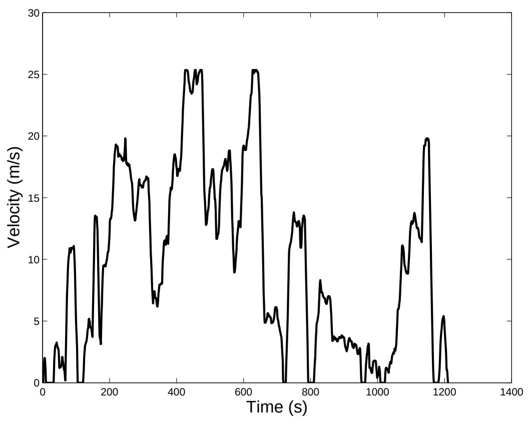

Velocity (m/s)
Time (s)

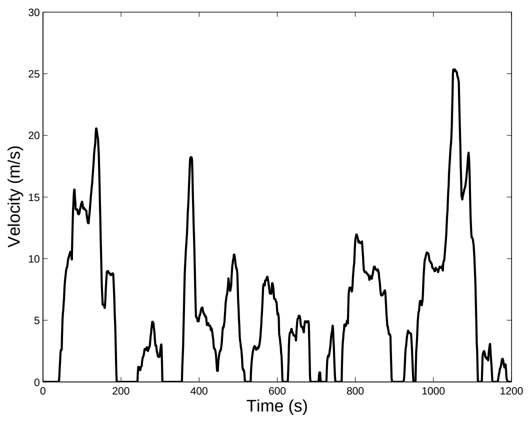

Velocity (m/s)
Time (s)

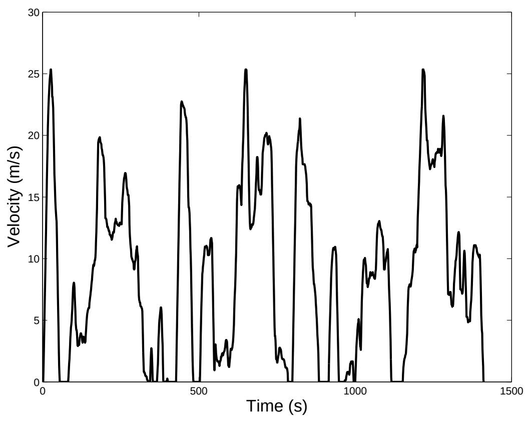

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

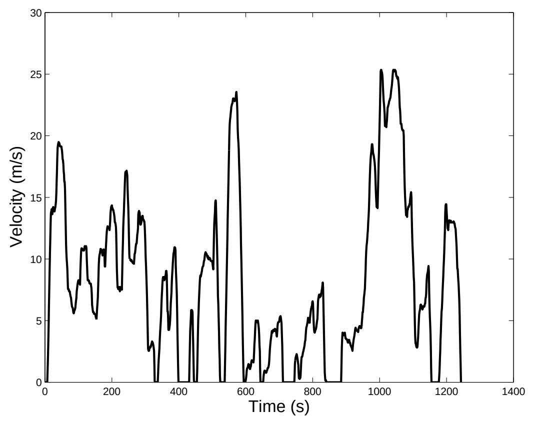

Velocity (m/s)
Time (s)
0 5 10 15 20 25
0 500 1000 1500

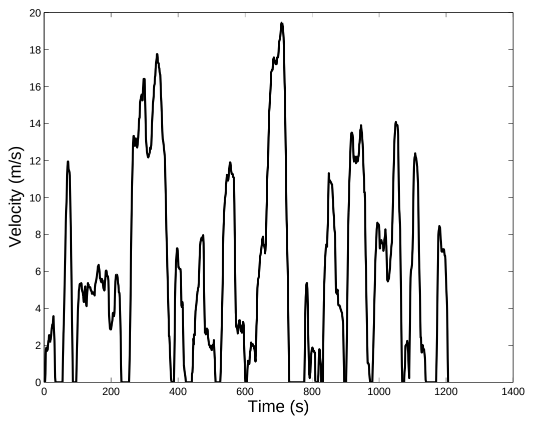

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

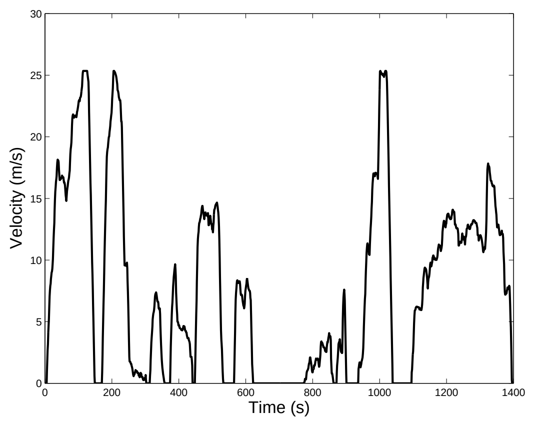

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

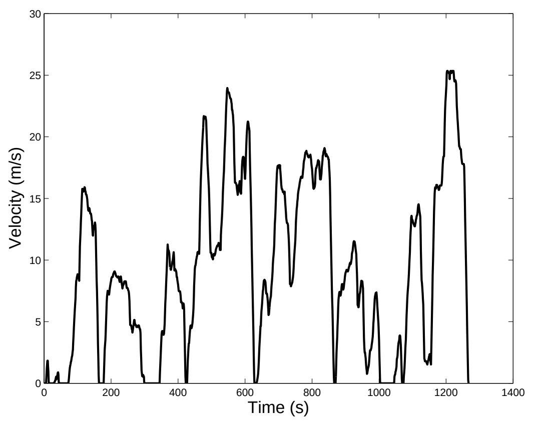

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

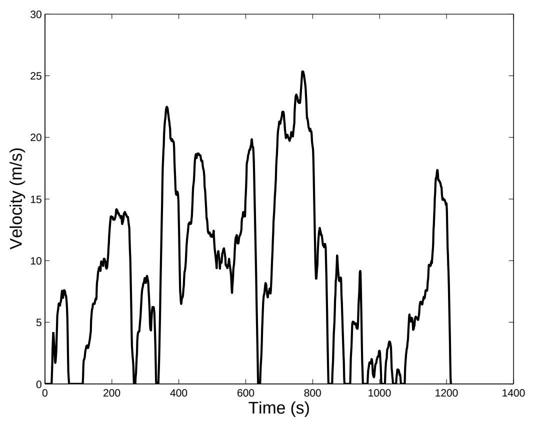

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 500 1000 1500

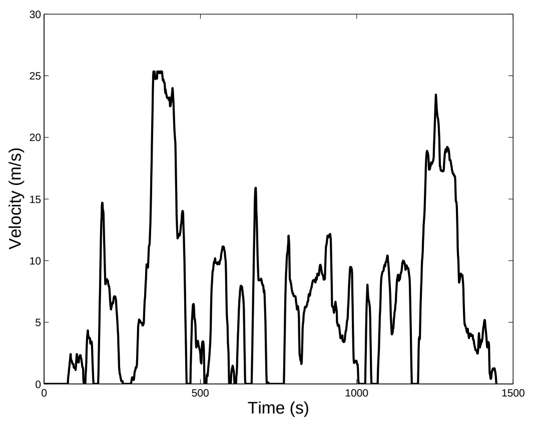

Velocity (m/s)
Time (s)

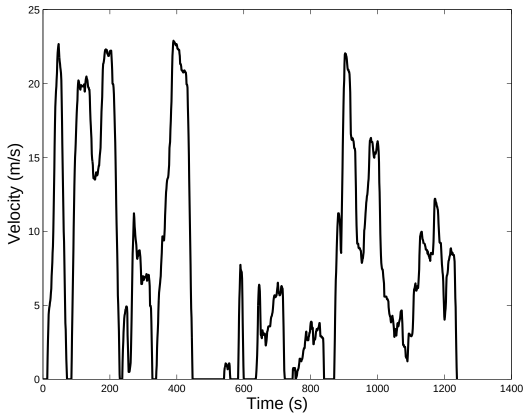

Velocity (m/s)
Time (s)

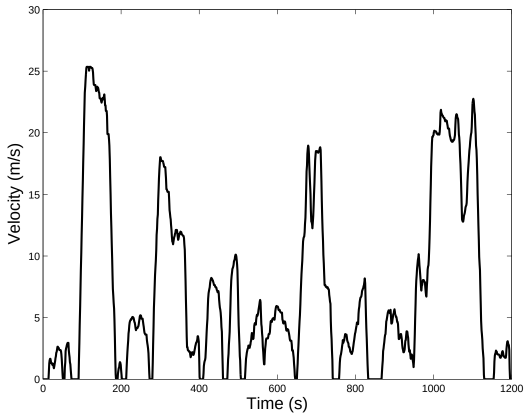

Velocity (m/s)
Time (s)

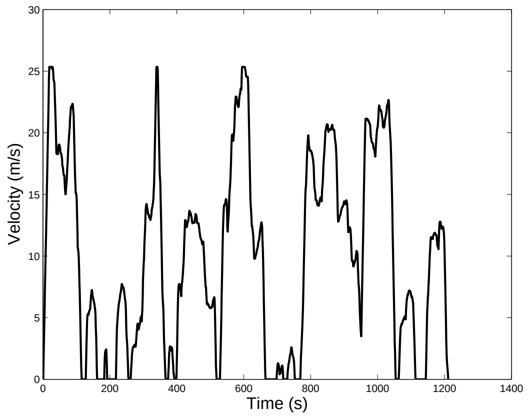

Velocity (m/s)
Time (s)

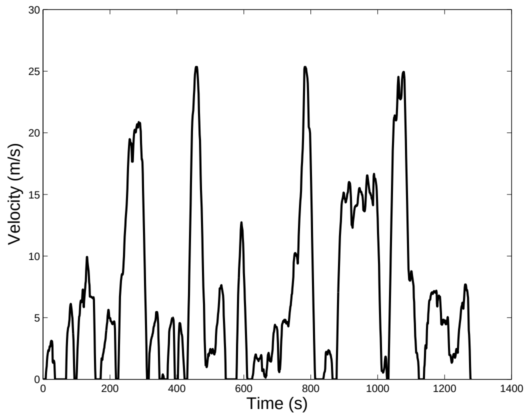

Velocity (m/s)
Time (s)

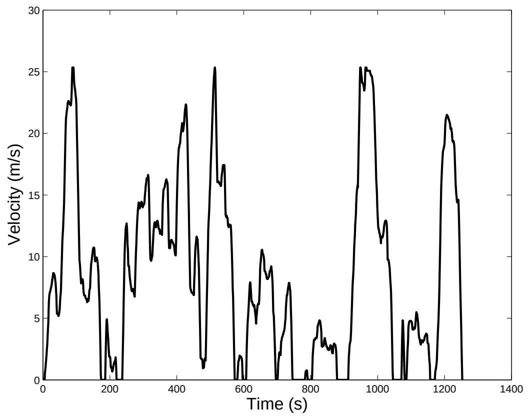

Velocity (m/s)
Time (s)

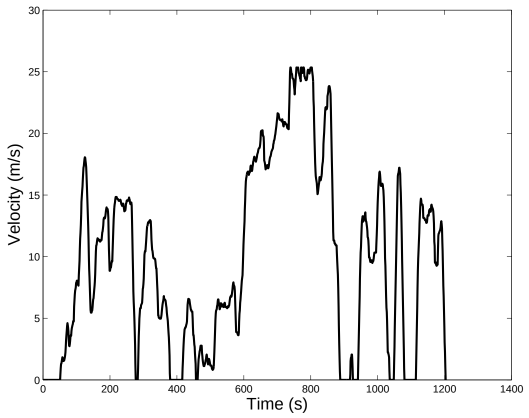

Velocity (m/s)
Time (s)

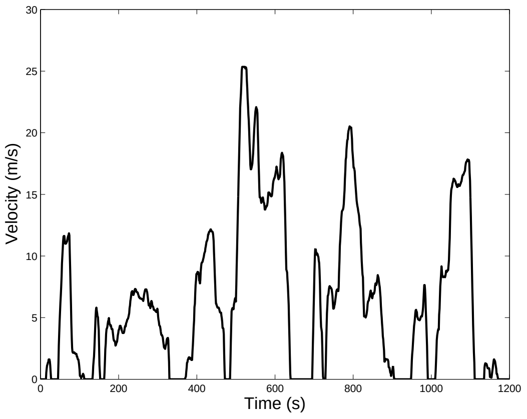

Velocity (m/s)
Time (s)

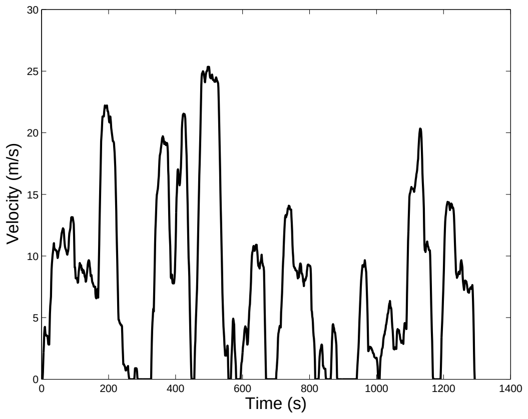

Velocity (m/s)
Time (s)

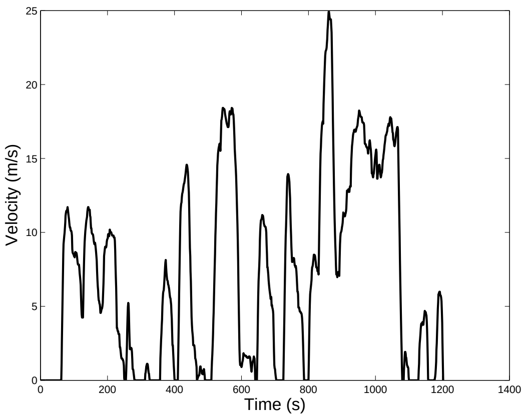

Velocity (m/s)
Time (s)

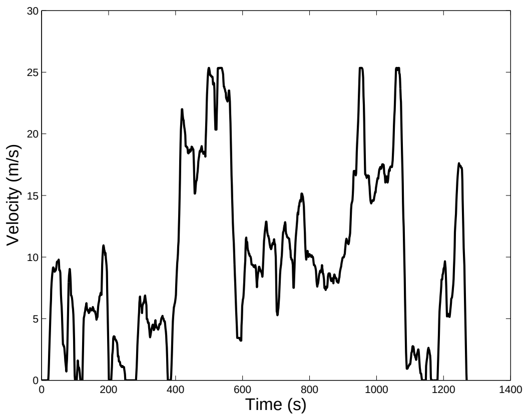

Velocity (m/s)
Time (s)

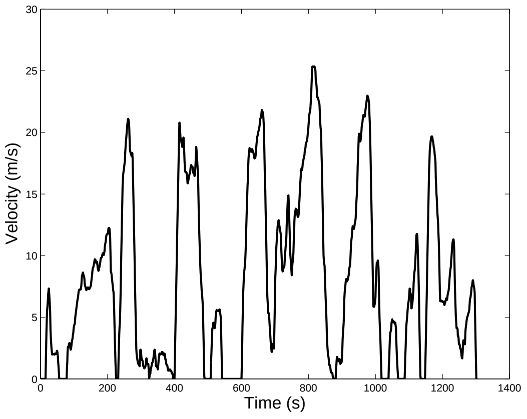

Velocity (m/s)
Time (s)

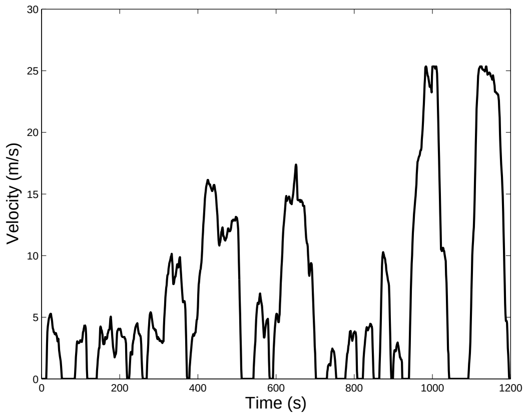

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

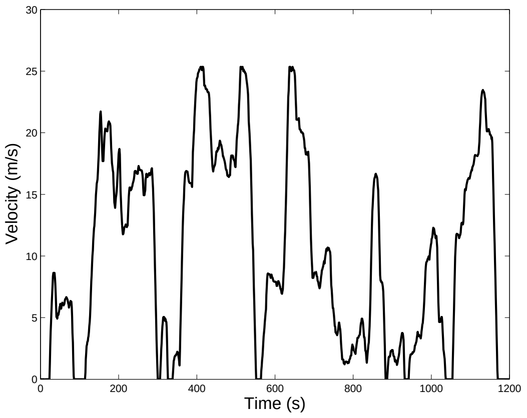

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

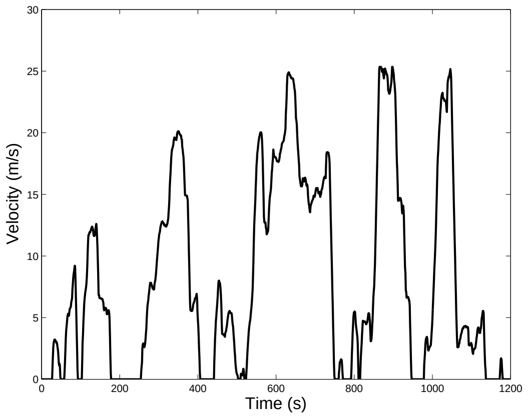

Velocity (m/s)
Time (s)
0
5
10
15
20
25
0
200
400
600
800
1000
1200
1400

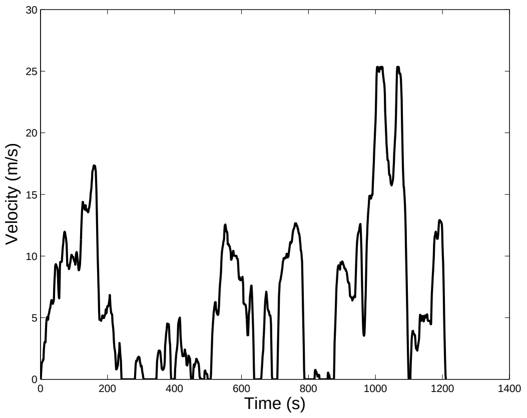

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

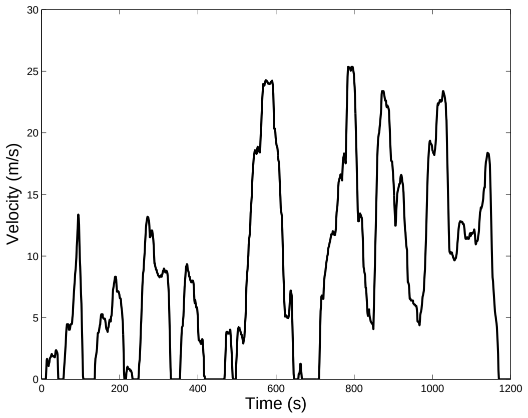

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

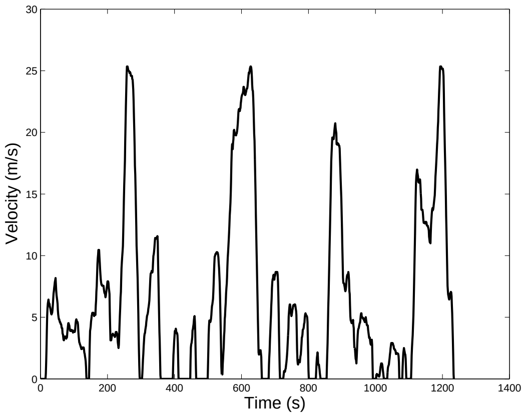

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

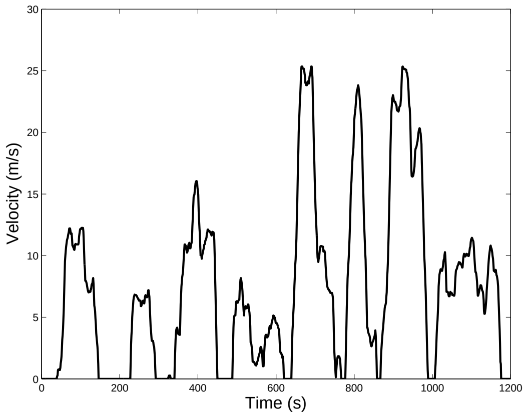

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

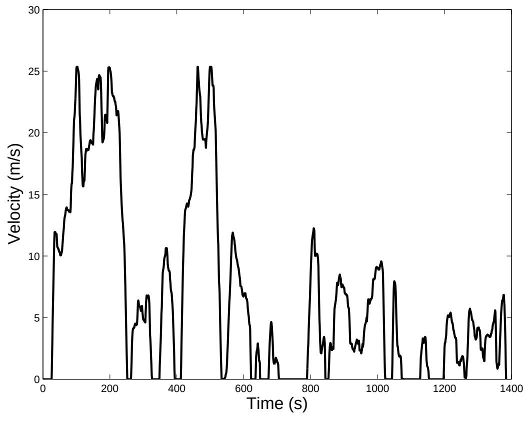

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

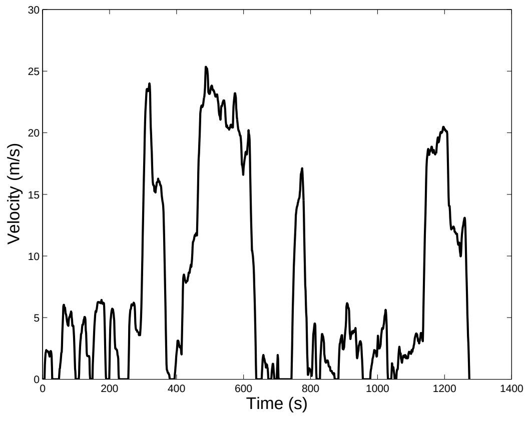

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

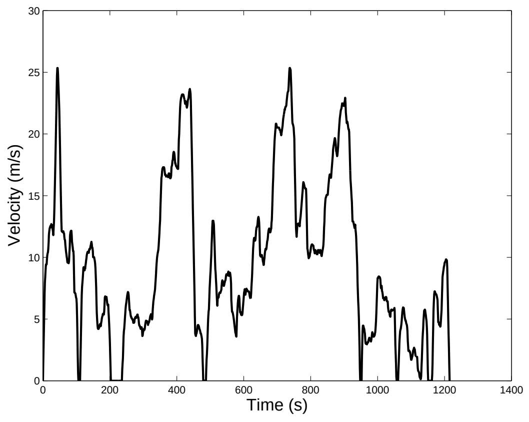

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

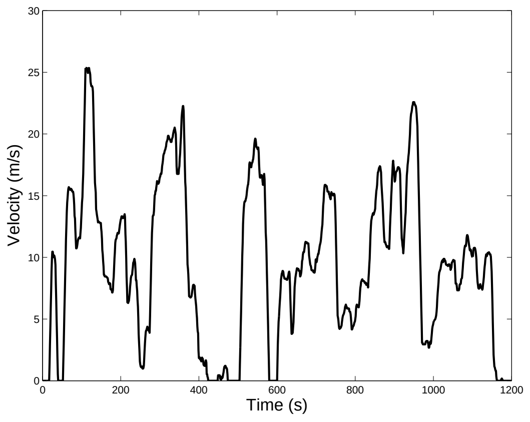

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

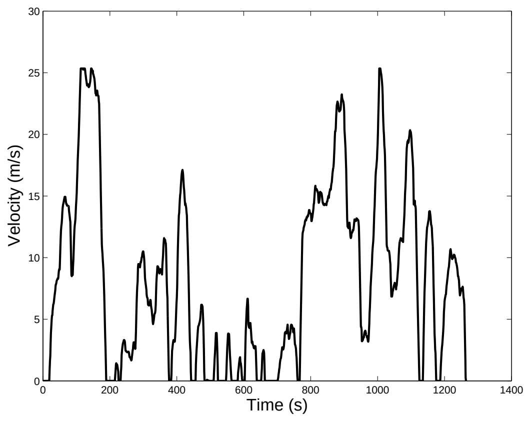

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

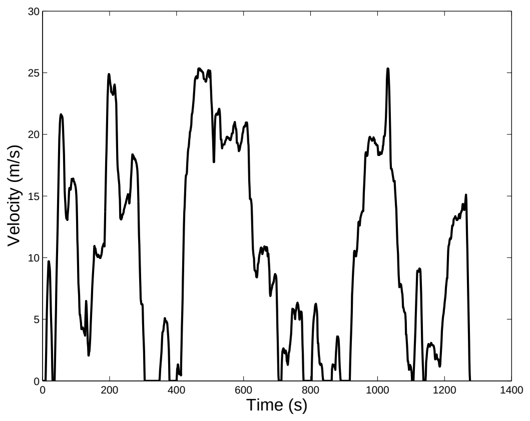

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

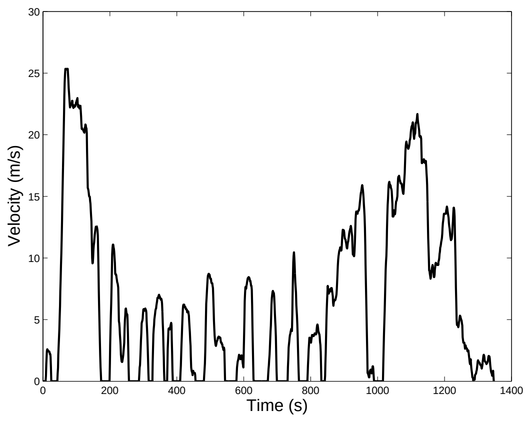

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

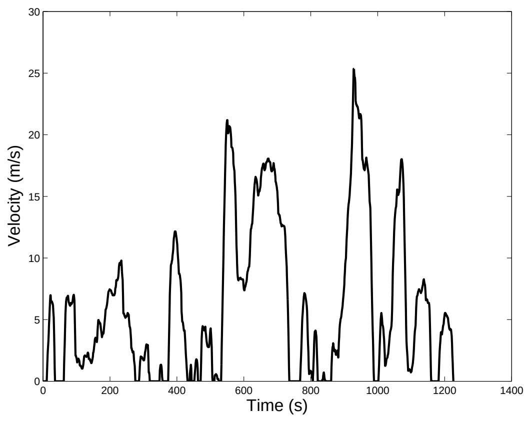

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

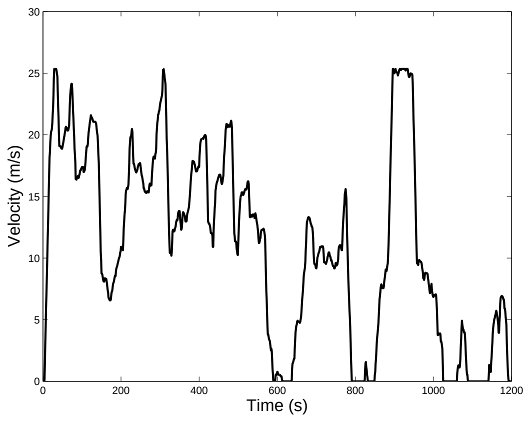

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

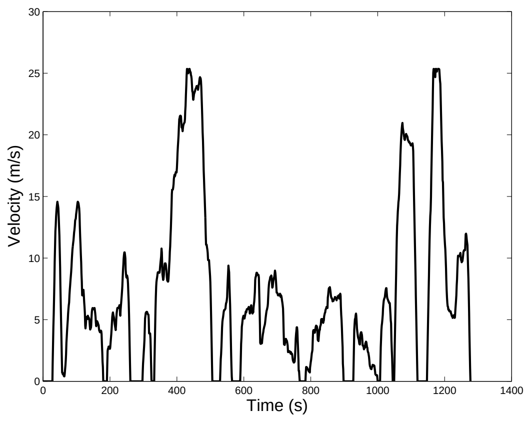

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

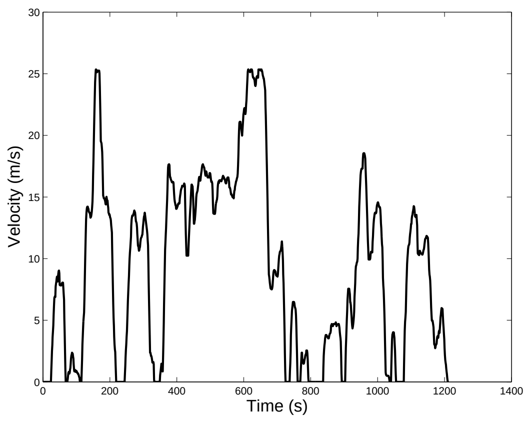

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

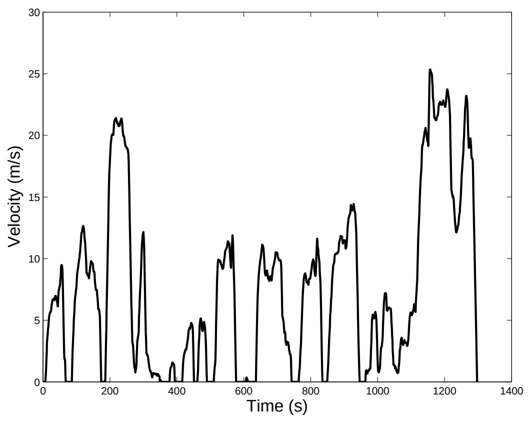

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

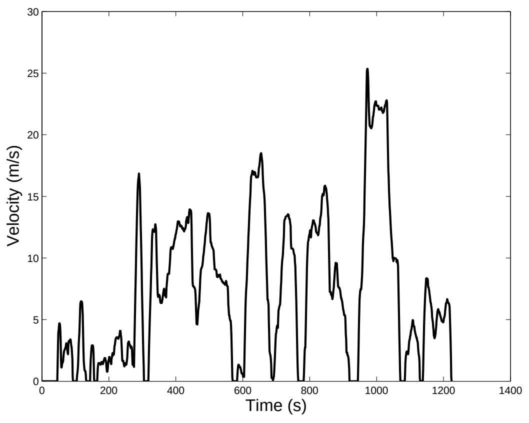

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

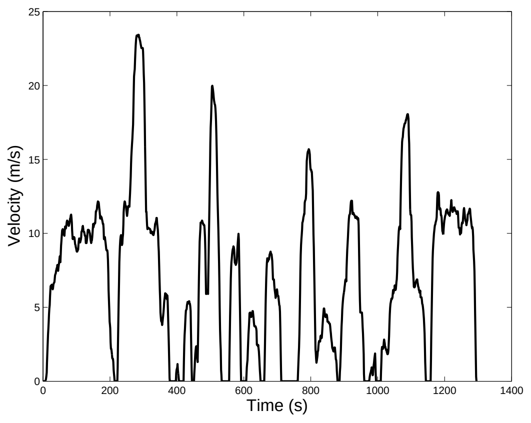

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200

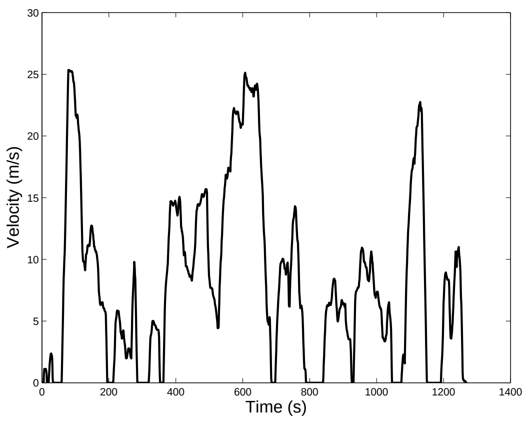

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

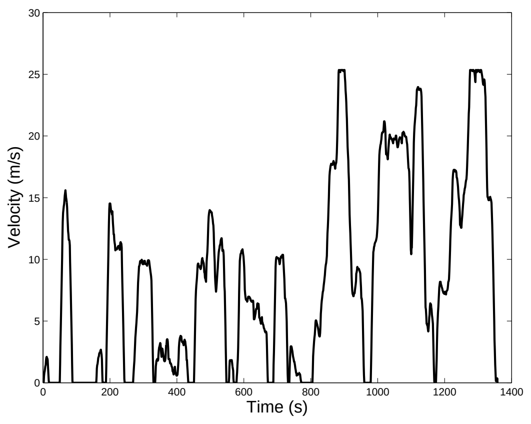

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

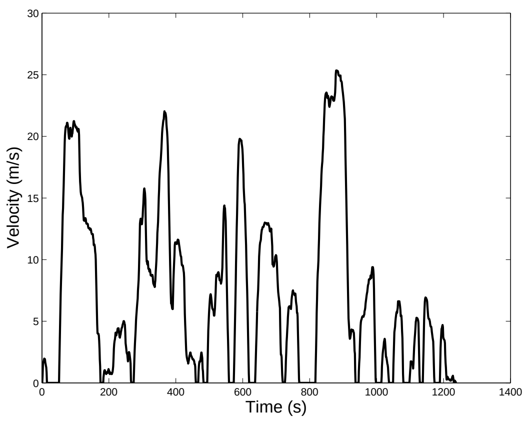

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

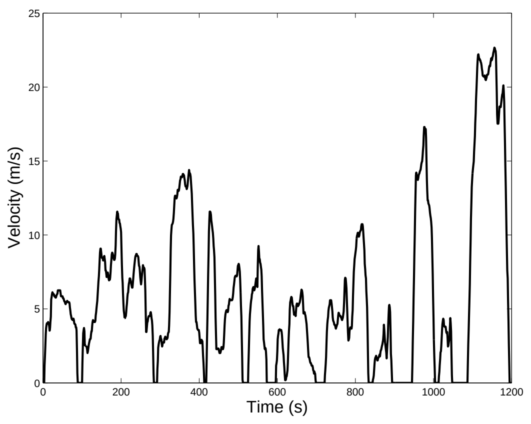

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

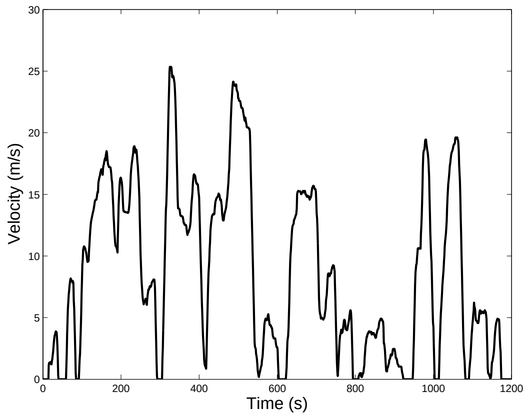

Velocity (m/s)
Time (s)
0
5
10
15
20
25
0
200
400
600
800
1000
1200
1400

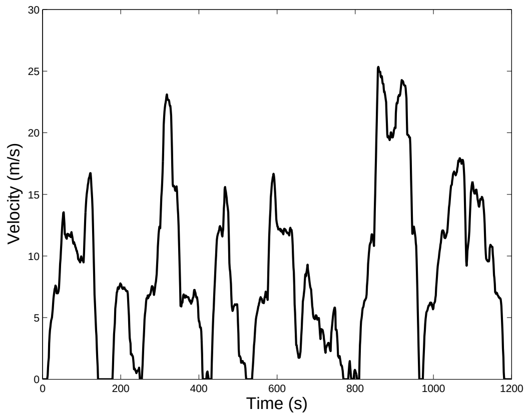

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

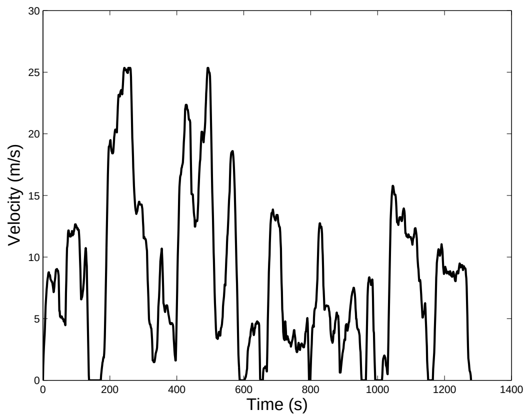

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

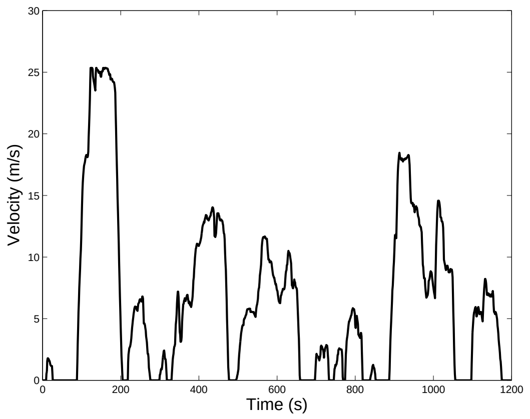

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

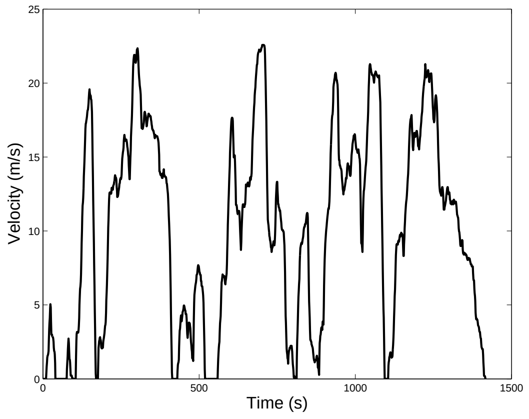

Velocity (m/s)
Time (s)
0
5
10
15
20
25
0
200
400
600
800
1000
1200

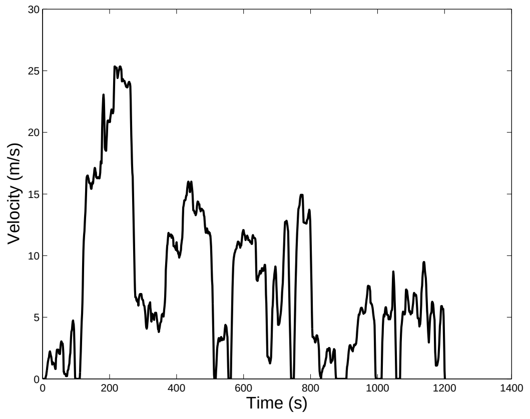

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

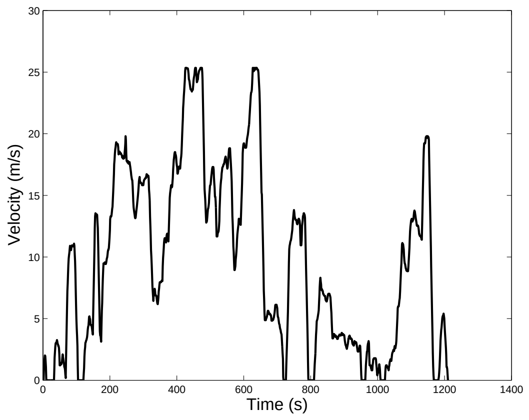

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

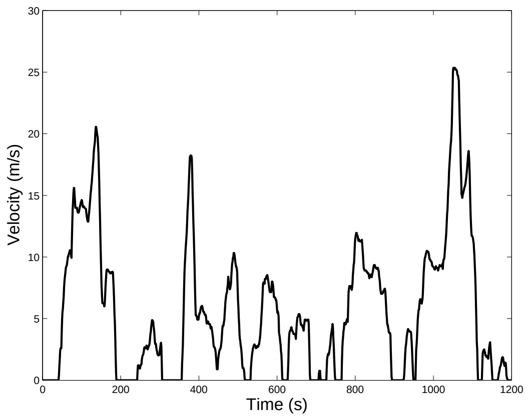

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

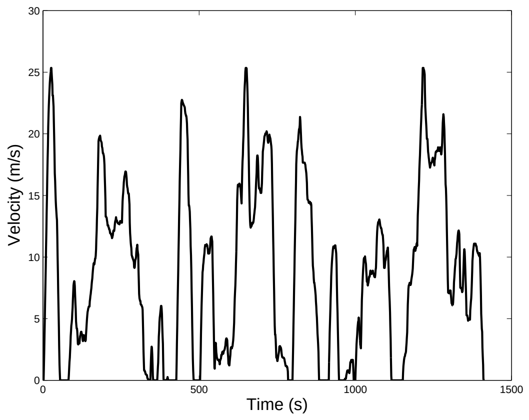

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200

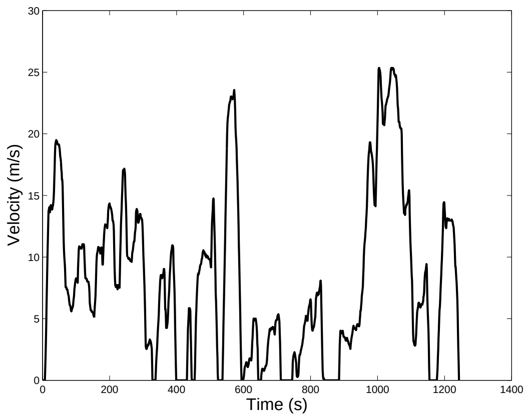

Velocity (m/s)
Time (s)
0
5
10
15
20
25
0
500
1000
1500

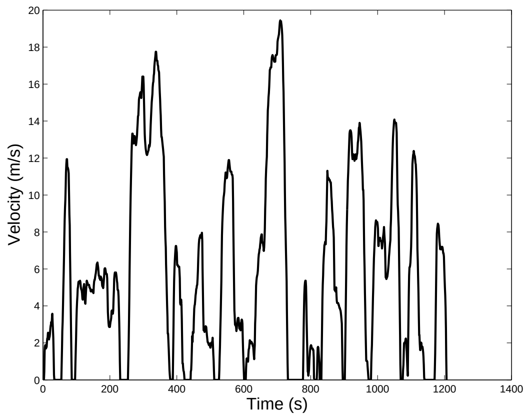

Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
0
200
400
600
800
1000
1200
1400

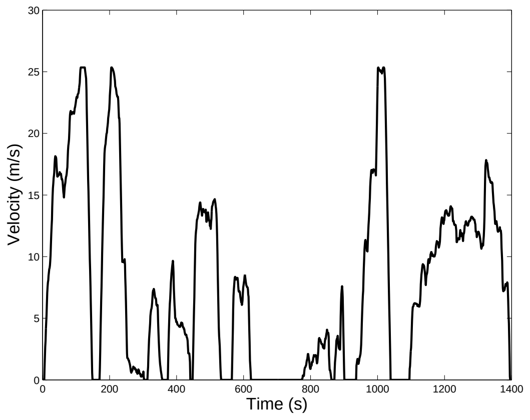

Velocity (m/s)
Time (s)

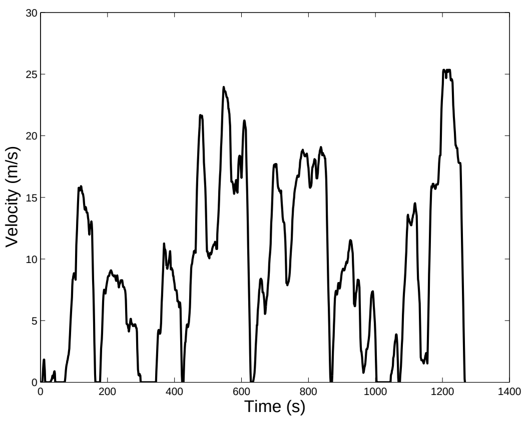

Velocity (m/s)
Time (s)

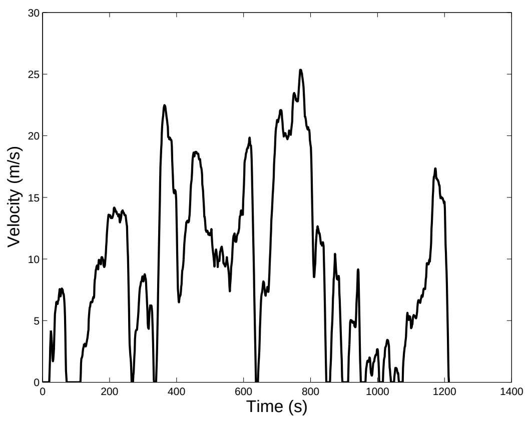

Velocity (m/s)
Time (s)

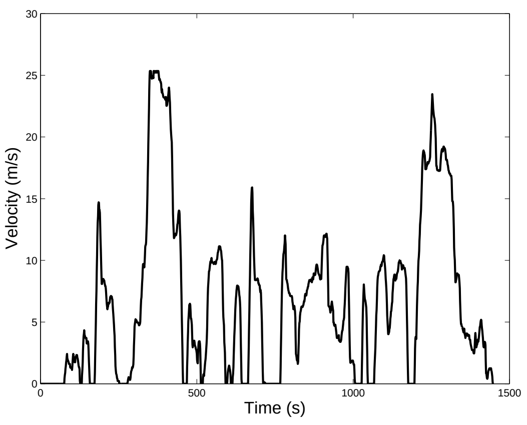

Velocity (m/s)
Time (s)

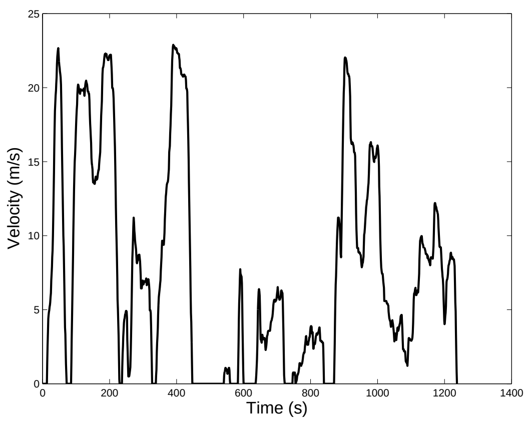

Velocity (m/s)
Time (s)

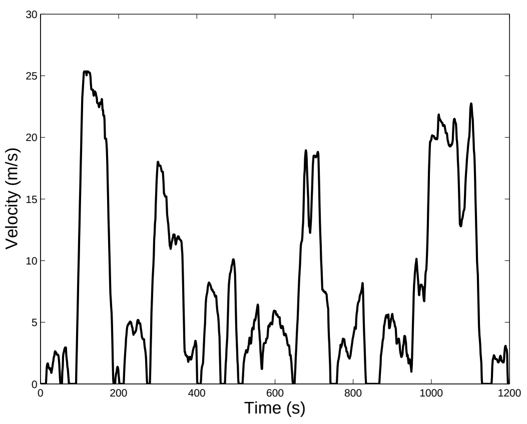

Velocity (m/s)
Time (s)

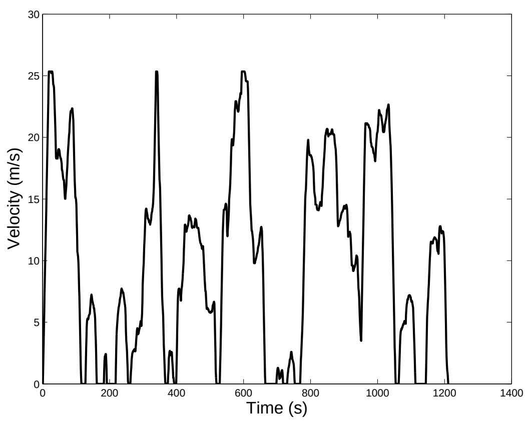

Velocity (m/s)
Time (s)

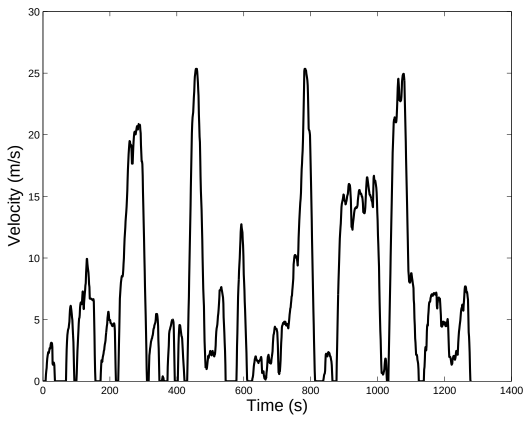

Velocity (m/s)
Time (s)

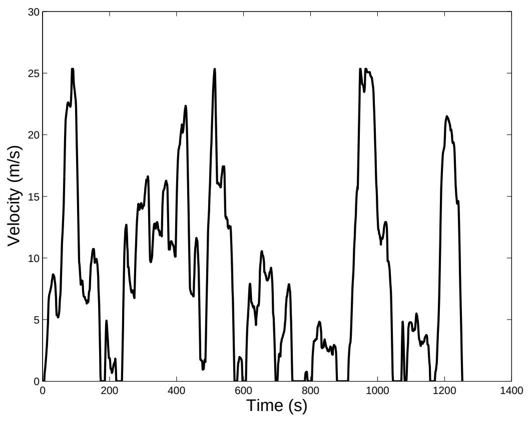

Velocity (m/s)
Time (s)

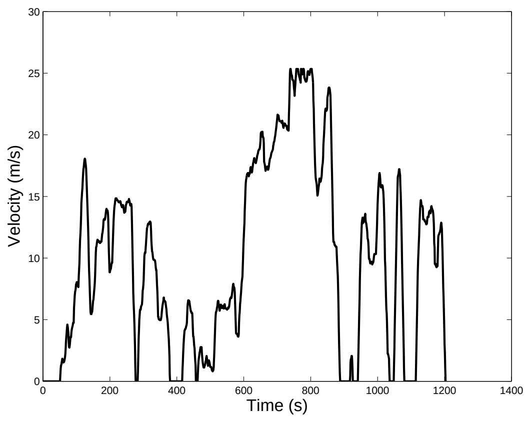

Velocity (m/s)
Time (s)

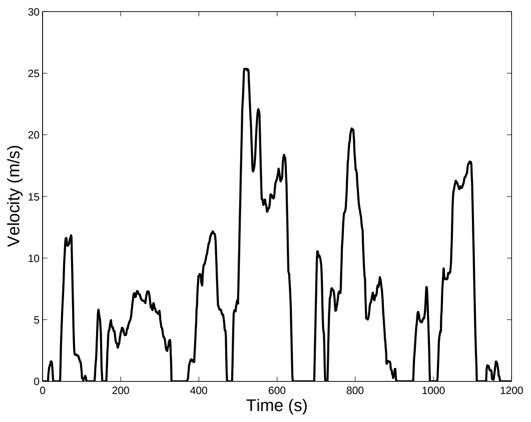

Velocity (m/s)
Time (s)

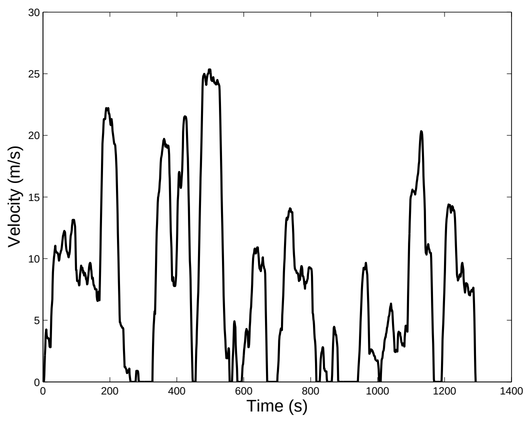

Velocity (m/s)
Time (s)

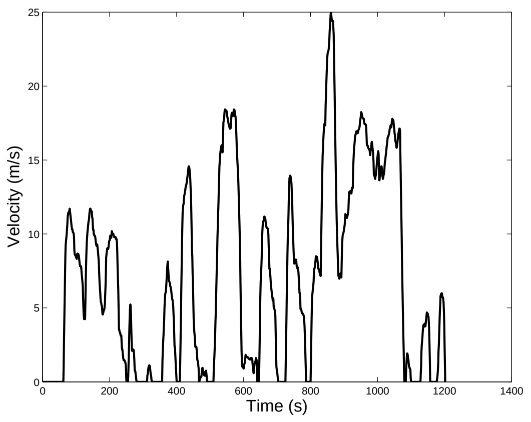

Velocity (m/s)
Time (s)

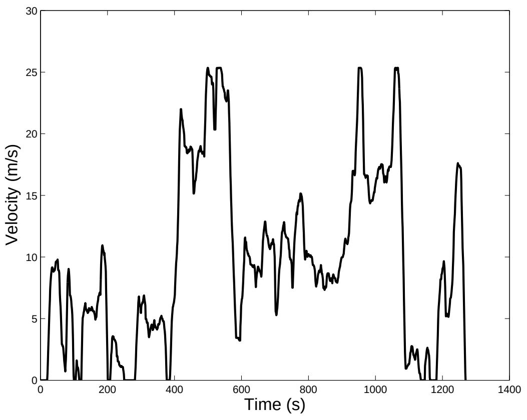

Velocity (m/s)
Time (s)

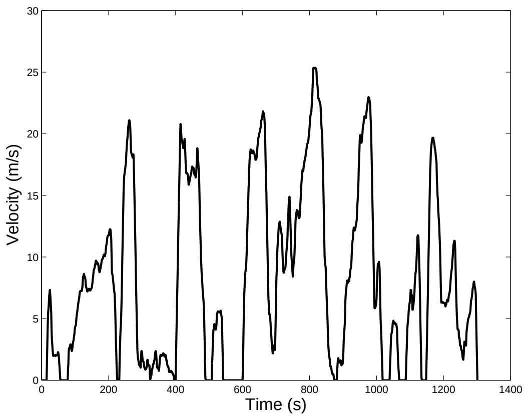

Velocity (m/s)
Time (s)

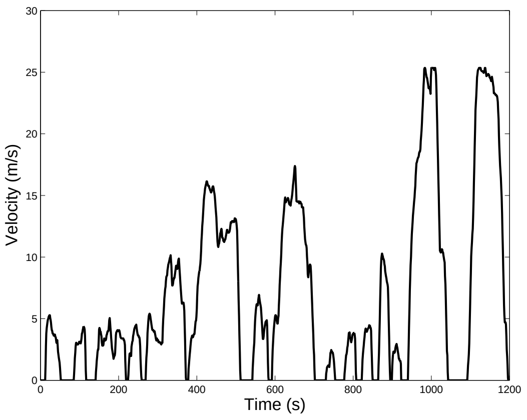

Velocity (m/s)
Time (s)

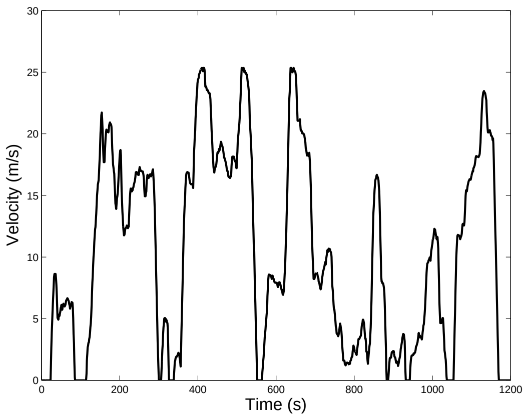

Velocity (m/s)
Time (s)

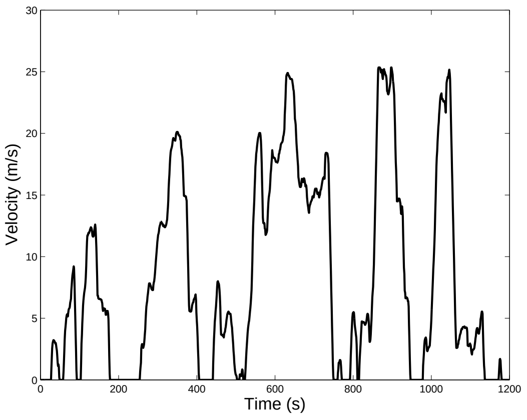

Velocity (m/s)
Time (s)

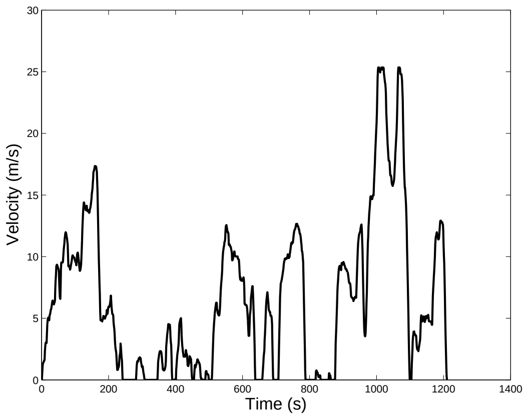

Velocity (m/s)
Time (s)

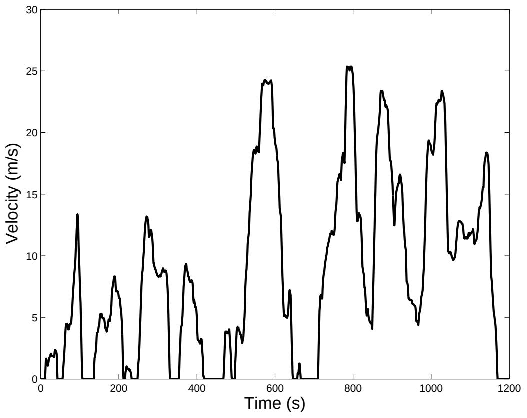

Velocity (m/s)
Time (s)

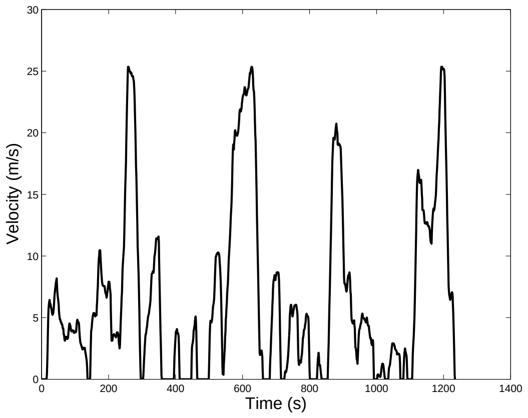

Velocity (m/s)
Time (s)

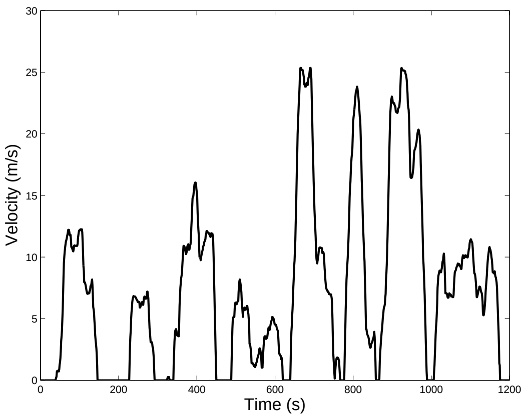

Velocity (m/s)
Time (s)

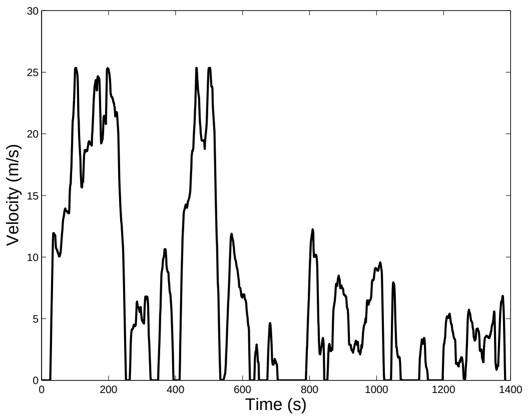

Velocity (m/s)
Time (s)

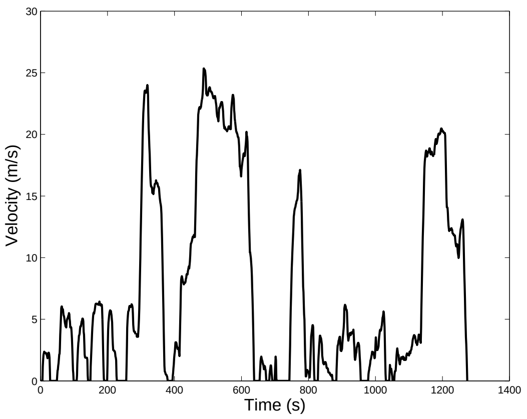

Velocity (m/s)
Time (s)

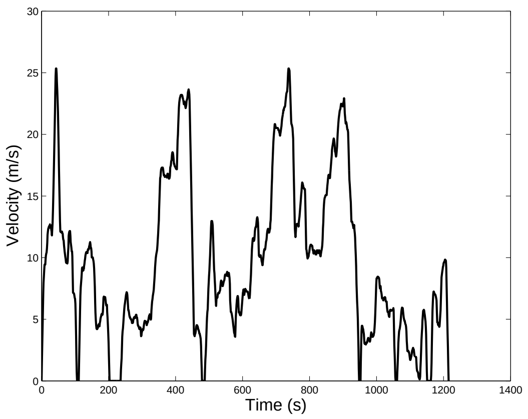

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

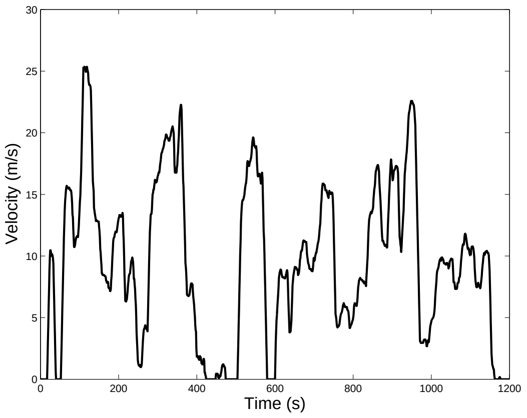

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

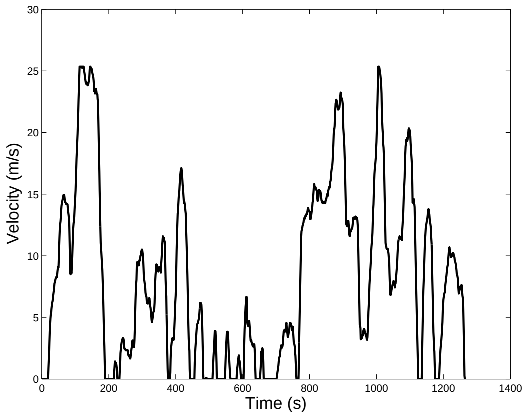

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200

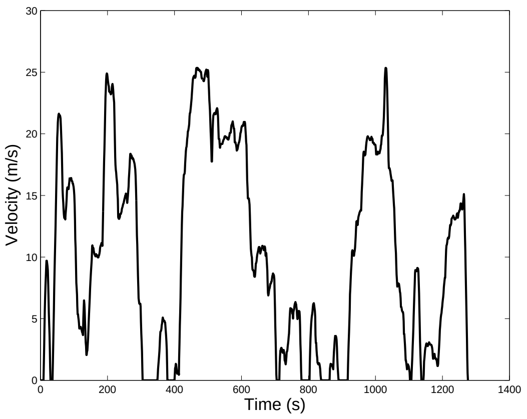

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

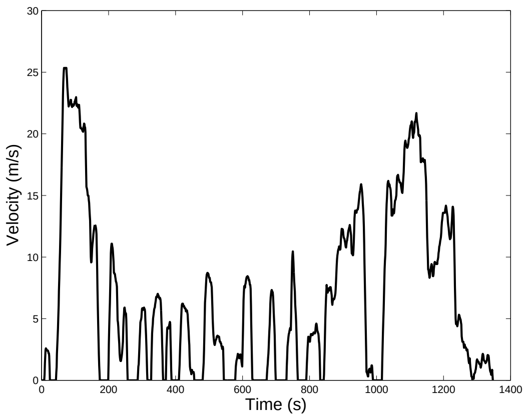

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

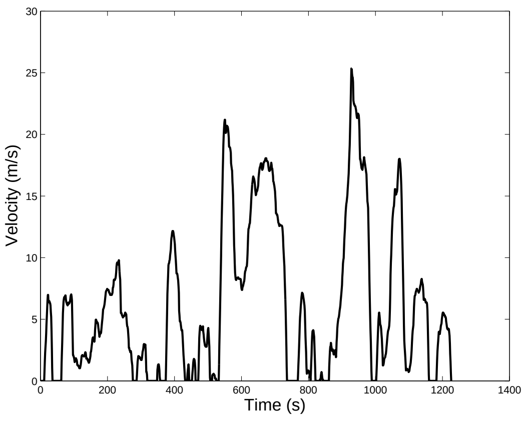

Velocity (m/s)
Time (s)
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200 1400

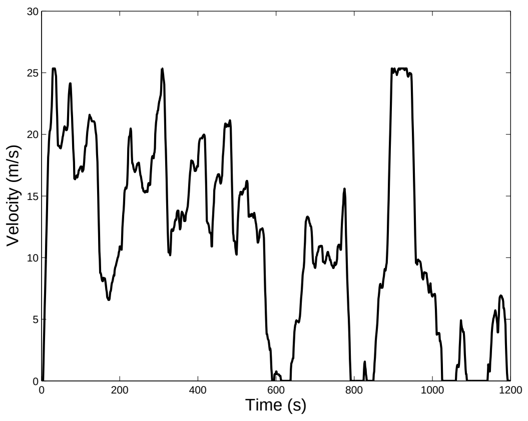

Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200

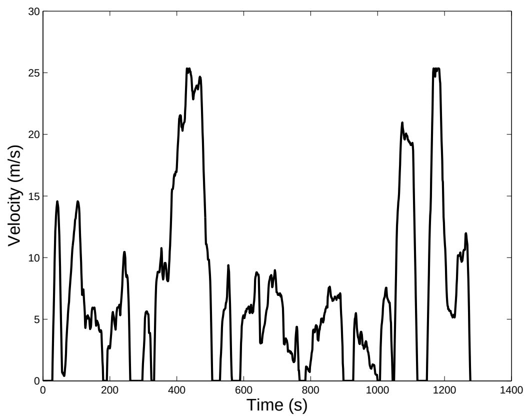

Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200
1400


Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200
1400

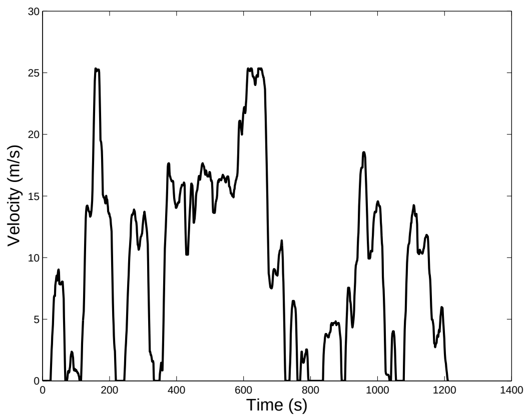

Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200
1400


Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200
1400


Velocity (m/s)
0
5
10
15
20
25
30
Time (s)
0
200
400
600
800
1000
1200



Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400



Velocity (m/s)
Time (s)
0
5
10
15
20
25
30
200
400
600
800
1000
1200
1400

Appendix D

MATLAB Code

HMT Parameters

```
%*********************************************************************
%***************   Hydraulic Drivetrain System Model   ***************
%*********   Center for Compact and Efficient Fluid Power   **********
%*********************************************************************

% script m-file sUV_HMT.m
% The purpose of this script is to define all parameters for the HMT
% hydraulic hybrid vehicle.

global airdens g slope psi2pa
global mass dia dragc fo fs fronta energyfuel Re_St_Penalty mu B
global AccumP_initial AccumV Vo_initial Vg_initial r AccumP_full
global AccumV_full AccumEff R_B H_Reduct R_Reduct gear1 gear2 e_tran e_rear
global e_hyd_red D_m D D_mE De V_r_P C_s_P C_v_P C_f_P V_r_M C_s_M C_v_M
global C_f_M RT RS rh Rdiff e_RT e_RS e_rh e_Rdiff speed torque eff fuel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%                 Physical Constants                      %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
airdens = 1.29;         %Air density STP    (kg/m^3)
```

```
g = 9.80665;                %acceleration due to gravity (m/s^2)
slope = 0.0;            %Grade angle              (rad)
psi2pa = 6894.75729;    %Psi to Pa conversion
gal2liter = 3.78;       %Gallons to liters conversion

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%                 Vehicle parameters                          %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mass = 1125;            %Mass of the vehicle      (kg)
dia = .619;             %Diameter of the wheel     (m)
dragc = .5;             %Drag Coefficient
fo = .0095;             %Rolling resistance coefficient fo
fs = .0035;             %Rolling resistance coefficient fs
fronta = 1.784;         %Frontal area   (m^2) (1.973)
energyfuel = 155000000; %Potential energy of Diesel (J/Gallon)
Re_St_Penalty = 0;%500;    %The restart engine penalty in Joules
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%                 Hydraulic Constants                         %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mu = .011;              %Oil Viscosity  (kg/m-sec)
B = 1.72e9;             %Bulk Modulus of oil(Pa)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%                 Accumulator parameters                      %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AccumP_initial = 17500000;  %Initial precharged accumulator pressure (Pa)
AccumP_initial = 1600*psi2pa; %Initial precharged accumulator pressure (Pa)
AccumV = 0.038;         %Total accumulator volume (m^3)
Vo_initial = 0.0;       %Initial volume of oil
Vg_initial = AccumV;    %Initial volume of gas in the accumulator
% r = 2;                %Maximum compression ratio of gas in accumulator
r = 3;                  %Maximum compression ratio of gas in accumulator
%Above this pressure, extra energy is dumped
```

```
AccumP_full = AccumP_initial*((Vg_initial*r)/AccumV);
AccumV_full = AccumV - ((AccumP_initial*Vg_initial)/AccumP_full);
AccumEff = 0.98;              %Accumulator efficiency
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%  Vehicle parameters specific to the transmission           %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R_B = 1.3;          %Speed reduction from P/M B to the engine output shaft
H_Reduct = 8.625;  %Reduction from hydraulic unit S to differential
R_Reduct = 8.625;  %Reduction from output of transmission to carrier
gear1 = 1;          %Transmission ratio (1st gear)
gear2 = 2.89;       %Transmission ratio (2nd gear)
e_tran = 1;         %Transmission efficiency
e_rear = 1;         %Rear sprocket efficiency to differentials
e_hyd_red = 1;      %Mechanical efficiency of the hydraulic branch
RT = 1.3;
RS = 2;
rh = 1/4;
Rdiff = 3.45;
e_RT = 1;
e_RS = 1;
e_rh = 1;
e_Rdiff = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                  Pump/Motor Parameters                     %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D_m = 2.8000e-005;      %Volumetric Displacement in m^3/rev
D = D_m/(2*pi);         %Volumetric Displacement in m^3/rad
D_mE = 2.8e-005;        %Volumetric Displacement of P/M E in m^3/rev
De = D_mE/(2*pi);       %Volumetric displacement of P/M E in m^3/rad
Dt_m = 2.8000e-005;     %Volumetric Displacement of tandom pump in m^3/rev
Dt = Dt_m/(2*pi);       %Volumetric Displacement of tandom pump in m^3/rad
V_r_P = 1.1113;         %Volume Ratio - Pump - Flow Loss
```

```
C_s_P = 1.8846e-009;   %Slip Coefficient -Pump - Flow Loss
C_v_P = 4.9098e+005;   %Viscous Drag Coefficient - Pump - Torque Loss
C_f_P = 0.0240;        %Coulomb Friction Coefficient - Pump - Torque Loss
V_r_M =  1.1113;       %Repeat for when motoring
C_s_M = 1.8846e-009;
C_v_M = 4.9098e+005;
C_f_M = 0.0240;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%                     Engine Parameters                       %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Engine speed (rpm)
speed = [0 1200 1400 1600 1800 2000 2100 2200 2400 2600 2800 3000];
%Max engine torque (Nm)
torque = [0 63.7 65.4 66.2 66.3 66.8 66.9 66.2 64 63.5 63.1 62.7];
eff = [0, 0.33, 0.30, 0.30, 0.305, 0.298, 0.29];
fuel = [0, 2985.6, 3864.2, 4140, 4384, 4748.9, 5097.6]./3600;
```

Calculation of required power

```
% Jonathan Meyer
% January 2007
%
% function m-file wheelpwr.m
% The purpose of this function is to import a drive cycle and vehicle
% parameters and then calculate the power requirements needed at the wheels
% as well as other parameters.
%

function [cycle] = wheelpwr(CurVehSpeed,NextVehSpeed,deltat,mass,dragc,...
                            fronta,airdens,fo,fs,g,dia,slope)

%Computes average velocity (m/s)
```

```
velocity = (NextVehSpeed+CurVehSpeed)/2;
%Computes acceleration at each pt. (m/s^2)
acceleration = (NextVehSpeed-CurVehSpeed)/deltat;
%Calculates acceleration force (N)
force = mass*acceleration;
%Calculates drag force (N)
drag = 0.5*(dragc*fronta*airdens*(velocity^2));
%If the vehicle is rolling then compute the rolling resistance
if velocity > 0
    %rolling resistance coefficient
    fr = (fo+(3.24*fs*((velocity*2.23693629)/100)^2.5));
else  %If the vehicle is stationary
    fr = 0;    %The rolling resistance contribution is zero
end
%Resistance force to rolling (N)
resistance = (fr*(mass)*(g));
%Slope Force
slopeforce = (mass*g)*(tan(slope));
%Ft: Total force seen by engine (N)
TotForce = (force+drag+resistance+slopeforce);
%Power required at the wheels (W)
power = (velocity*TotForce);
%Rotational speed of wheels (rpm)
SpeedRPM = (velocity/(pi*dia))*60;
%Calculates distance travelled
distance = velocity*deltat;

cycle = [deltat, velocity, acceleration, force, drag, fr, resistance, ...
         TotForce, power, SpeedRPM, distance, slopeforce];
```

Hydraulic hybrid vehicle model

```
function sol = HMTGen2_new_exp(cycle, deltat, percent, E, Te, V_o, ...
                              mode, PMTSpeed, wlspeed, wltorque)

% Jonathan Meyer
%
%**************************************************************************
%***************   Hydraulic Drivetrain System Model   ********************
%*********   Center for Compact and Efficient Fluid Power   ***************
%**************************************************************************

%The purpose of this script is to generate a theoretical fuel efficiency
%model of the redesigned Hydraulic Hybrid Passenger Vehicle test bed
%utilizing an engine, 2 hydraulic pump/motor units, and a planetary
%differential.
%
%This function takes 6 required input variables and 2 optional input
%variables.
%
%Required input variables:
%
%cycle: A 1x12 vector calculated from the function wheelpwr.m.  This
%calculates the power needed at the wheels for a certain speed and
%acceleration taking into account aerodynamic drag, rolling resistance,
%road slope, and inertial acceleration.
%
%deltat: A scalar value of the time step used to calculate acceleration (s)
%
%percent: A scalar or vector of percent engine load of full load at a given
%engine speed.
%
%E: A scalar or vector of engine speed (rpm).
%
%V_o: A scalar or vector of accumulator oil volume (m^3).
```

```
%
%mode: A number from 1-3 to determine the operating mode of the vehicle.
%Mode 1: HMT; if percent and E are zero the engine is off and declutched.
%Mode 2: Parallel/Pump T only; pump S is locked (speed = 0) with no flow.
%If percent and E are zero, the engine is off and decluched with pump T
%providing all the required torque.
%Mode 3: Pump S only; percent and E should be equal to zero for this mode.
%Pump T is locked (speed = 0) and all required torque is provided by pump S
%
%Optional input variables:
%
%These are only used when the wheel speed and wheel torque are known and it
%is to be determined if a certain mode is feasible.
%
%wlspeed: A scalar or vector of wheel speeds (rad/s).
%
%wltorque: A scalar or vector of torque required at the wheels (N-m).
%
%Output variable:
%
%volOilNew: Volume of oil in the accumulator at the end of the time step
%(m^3).  If a solution is not feasible (pump must provide too much torque,
%not enough oil in the accumulator, etc.) this will return a value near 1.
%
%Functions required (Note: These functions may have other dependent
%functions):
%
%HMTGen2Parameters.m: Loads all parameters for the vehicle
%RexrothA6.m: Pump/motor model for Rexroth A6 series hydraulic units.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%sUV hydromechanical transmission flow and pressure based model%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Add other directories where functions are located
S=pwd;
[PATH,NAME,EXT]=fileparts(S);
str=[PATH '\Rexroth A6'];
%addpath(str)

%% Load all variables related to vehicle parameters.
global psi2pa
global dia AccumP_initial AccumV Vg_initial
global RT RS rh Rdiff e_RT e_RS e_rh e_Rdiff

%% Initialize engine speed, percent load, oil volume, and accumulator
%% pressure variables in a format that can be used for table lookup in the
%% pump models.
if exist('wltorque','var')
    V_o = V_o*ones(size(wltorque));
    E = E*ones(size(wltorque));
else
    %     P = Accum_P*ones(length(E),length(percent));
    if length(E) > 1
        [E, percent_temp, V_o_temp] = ndgrid(E, percent, V_o);
        clear percent_temp V_o_temp
        [Te, percent, V_o] = ndgrid(Te, percent, V_o);
    else
        [PMTSpeed, percent, V_o] = ndgrid(PMTSpeed, percent, V_o);
        E = E*ones(size(PMTSpeed));
        Te = Te*ones(size(PMTSpeed));
    end
end
V_g = AccumV - V_o;
```

```
P = AccumP_initial.*(Vg_initial./V_g);  %Pressure in accumulator

%% If wheel speed and wheel torque variables are not present, calculate the
%% speed and torque based off of drivecycle information.
if ~exist('wlspeed','var')
    wlspeed = (cycle(1,10)*((2*pi)/60))*ones(size(E));
end
if ~exist('wltorque','var')
    wltorque = -(cycle(1,8)*(dia/2))*ones(size(E));
end

%% If in parallel mode, double check engine speed is correct.
if mode == 2
    E(E~=0) = wlspeed(E~=0)*(rh+1)*Rdiff*(60/(2*pi));
end

%% Initialize variables related to hydraulic subsystem
w_T = zeros(size(wltorque));
T_T = w_T;
x_T = w_T;
Q_T = w_T;
w_S = w_T;
T_S = w_T;
x_S = w_T;
Q_S = w_T;

%% Calculate flow rate for pump/motor T

%Angular velocity of pump/motor T
if mode == 1  %HMT Mode
    %     if wlspeed ~= 0
    w_T(E == 0) = PMTSpeed(E == 0)*(2*pi/60);
    %     end
    w_T(E ~= 0) = E(E ~= 0)*(2*pi/60)*RT;
elseif mode == 2  %Parallel/Pump T Only Modes
    w_T = wlspeed*(rh+1)*Rdiff*RT; %Angular velocity of P/M B in rad/s
end

%NOTE: Sign convention: Positive power = power added to system\
%Negative power = power taken away from the system

%Torque on output shaft of transmission
Tout = wltorque/Rdiff;
Tout(wltorque.*wlspeed>0) = Tout(wltorque.*wlspeed>0)*e_Rdiff;
Tout(wltorque.*wlspeed<0) = Tout(wltorque.*wlspeed<0)/e_Rdiff;

%Torque on ring gear
if mode == 1 || mode == 2
    Tring = -Tout/(rh+1);
    Tring(w_T.*Tring>0) = Tring(w_T.*Tring>0)/e_rh;
    Tring(w_T.*Tring<0) = Tring(w_T.*Tring<0)*e_rh;
else
    Tring = zeros(size(Tout));
end

%Torque on the shaft of the hydraulic unit
T_T(abs(w_T)>0) = ((Tring(abs(w_T)>0)-Te(abs(w_T)>0))/RT)+ ...
                    0.07*E(abs(w_T)>0)*pi/30;
T_T(w_T.*T_T>0) = T_T(w_T.*T_T>0)/e_RT;
T_T(w_T.*T_T<0) = T_T(w_T.*T_T<0)*e_RT;

%Fractional displacement and flow rate through pump/motor T

if mode == 1 || mode == 3
```

```
    w_S = w_T/(RS*RT*rh)-(((1+(1/rh))*Rdiff)/RS)*wlspeed;
end

%Torque on the shaft of the hydraulic unit
if mode == 1 || mode == 3
    T_S = (RS/(1+(1/rh)))*Tout;
    T_S(w_S.*T_S>0) = T_S(w_S.*T_S>0)/e_RS;
    T_S(w_S.*T_S<0) = T_S(w_S.*T_S<0)*e_RS;
end

volOilNew = V_o;
%Pressure in accumulator
P = AccumP_initial.*(Vg_initial./(AccumV-volOilNew));
dP = P;
[x_T(abs(w_T)>0),Q_T(abs(w_T)>0)] = RexrothA6_exp(T_T(abs(w_T)>0), ...
                                    dP(abs(w_T)>0),w_T(abs(w_T)>0));
x_T(T_T == 0 & w_T == 0) = 0;
Q_T(T_T == 0 & w_T == 0) = 0;

%Set flow rate = 1 and fuel consumption = Inf if solution is not feasible
%due to too high of displacement
Q_T(abs(x_T)>1) = 100;


%% Calculate flowrates for pump/motor S

%Fractional displacement and flow rate through pump/motor T
%NOTE: Since sign convention of the lookup table is opposite of the model,
%the torque is multiplied by -1
[x_S(abs(w_S)>0),Q_S(abs(w_S)>0)] = RexrothA6_exp(T_S(abs(w_S)>0), ...
                                    dP(abs(w_S)>0),w_S(abs(w_S)>0));
x_S(T_S == 0 & w_S == 0) = 0;
```

```
Q_S(T_S == 0 & w_S == 0) = 0;

%Set flow rate = 1 and fuel consumption = Inf if solution is not feasible
%due to too high of displacement
Q_S(abs(x_S)>1) = 100;



%% Calculate the new oil volume in the accumulator
volOilNew = volOilNew - (Q_T+Q_S)*deltat;
if length(V_o) > 1
    volOilNew(volOilNew<V_o(1)) = 1;
    volOilNew(volOilNew>V_o(end)) = 1;
end
volOilNew(isnan(volOilNew)) = 1;
volOilNew(isinf(volOilNew)) = 1;

sol = [volOilNew wlspeed wltorque E Te T_T w_T w_S T_S x_T x_S Q_T Q_S];
```

Hydraulic hybrid engine model

```
function [eng_loss,eff,fuel] = engine_404C15(Torque,W,goplot)

% [Loss, Efficiency, Fuel consumption] =
%                  ENGINE_404C15( Engine Torque, Engine Speed, Plot figure )
% Calculates Engine Losses, Engine Efficiency, Fuel Consumption
% Beyond max torque curve : Loss = inf
% Below idling speed (1400rpm) or above max speed (2600rpm) : Loss = inf
% goplot = 1 , plot engine operating points

global pp %fc_map_404C15.mat

rpm=2*pi/60;
% lbft2Nm=1.356;
```

```
Torque(Torque<1)=0;
eff=interpn(pp.fc_map_spd,pp.fc_map_trq,pp.fc_map_eff,W,Torque);

eng_loss=interpn(pp.fc_map_spd,pp.fc_map_trq,pp.fc_map_loss,W,Torque);
fuel=interpn(pp.fc_map_spd,pp.fc_map_trq,pp.fc_fuel_map,W,Torque);

eng_loss(W<1200*rpm)=inf;
eng_loss(Torque<0)=inf;
eng_loss(W>2700*rpm)=inf;
eng_loss(Torque==0)=0;

fuel(Torque<0)=inf;
fuel(W<1200*rpm)=inf;
fuel(W>2700*rpm)=inf;
fuel(Torque==0)=0;

% maxT = interp1(pp.fc_max_trq_spd,pp.fc_max_trq,W);
%
% eff(Torque>maxT) = nan;
% eng_loss(Torque>maxT) = inf;            % beyond max torque -> inf loss?
% fuel(Torque>maxT) = inf;

if goplot == 1
    figure
    [c,h]=contour(pp.fc_map_spd/rpm,pp.fc_map_trq,pp.fc_map_eff',[0:0.01:0.4]); clabel(c
    xlabel('Engine speed [rpm]')
    ylabel('Engine torque [Nm]')
    hold on
    plot(pp.fc_max_trq_spd/rpm,pp.fc_max_trq,'linewidth',2.5);
    plot(W/rpm,Torque,'kx','markersize',8);
    title('Perkins 404C15 engine map')
```

```
end
```

Hydraulic hybrid pump/motor model

```
function [displacement,flow] = RexrothA6(torque,pressure,speed)

load A6map

displacement = zeros(size(torque));
flow = displacement;
loss = displacement;

displacement = interp3(Pxlook,Txlook,wxlook,XX,pressure,torque,speed);

flow(isnan(displacement)) = 1;
flow(~isnan(displacement)) = interp3(xQlook,PQlook,wQlook,Q, ...
    displacement(~isnan(displacement)),pressure(~isnan(displacement)), ...
    speed(~isnan(displacement)));
```

textbfDynamic programming algorithm

```
clear;
clc;

global bbsfc_spd bbsfc_trq eng_pwr
load fc_map_404C15

filename1='UrbanDriveCycle.csv';                %File with drivecycle
drivecycle=csvread(filename1,2,0);
oilStep = 1e-4;
EngPenalty = 1;         %Fuel penalty for turning on/off engine
ModePenalty = 1;
EngSpeedPenalty = 0;
percent = 1;
```

```
EngOff = 1;   %Set to 1 for Engine Off operation, 0 for clutch always on
StopDeclutch = 1;  %1 to declutch when stopped, 0 to always be clutched
BSFCCurve = 0;  %Set to 1 to use BSFC curve, 0 for constant torque
ExpA6 = 1;      %Set to 1 to use experimental A6 map
FourMode = 1;   %Set to 1 for four mode operation

HMTGen2Parameters;        %Loads all parameters for HMT vehicle

if BSFCCurve
    load bbsfc_404C15
end

if FourMode
    [optimal, AccumStates] = DP_HMTGen2_4Mode(drivecycle,oilStep, ...
        EngPenalty,ModePenalty,percent,EngOff,StopDeclutch,BSFCCurve, ...
        ExpA6, EngSpeedPenalty);
else
    [optimal, AccumStates] = DP_HMTGen2_HMTOnly(drivecycle,oilStep, ...
        EngPenalty,ModePenalty,percent,EngOff,StopDeclutch,BSFCCurve, ...
        ExpA6, EngSpeedPenalty);
end

if FourMode
    if EngOff
        if BSFCCurve
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_'...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            end
        else
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            end
        end
    else
        if BSFCCurve
            if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        end
    else
        if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
```

```
                    'MLoil_4Mode'],AccumV,EngPenalty,ModePenalty);
                end
            end
            if EngOff == 0
                str = [str, '_ClutchOn'];
            end
            if StopDeclutch == 1
                str = [str, '_StopDeclutch'];
            end
        end
    else
        if EngOff
            if BSFCCurve
                if EngPenalty == 0
                    str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                        'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                        'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
                elseif EngPenalty < 1
                    str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                        'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                        'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
                elseif EngPenalty < 10
                    str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                        'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                        'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
                else
                    str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                        'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                        'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
                end
            else
                if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                'MLoil_HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        end
    end
else
    if BSFCCurve
        if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
        else
```

```
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_' ...
                    'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
            end
        else
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_' ...
                    'MLoil_HMTOnly'],AccumV,EngPenalty,ModePenalty);
            end
        end
        if EngOff == 0
            str = [str, '_ClutchOn'];
        end
        if StopDeclutch == 1
            str = [str, '_StopDeclutch'];
        end
    end
end
if ExpA6
```

```
    str = [str '_expA6'];
end
if e_rh == 1
    str = [str, '_IdealGears'];
end
if EngSpeedPenalty > 0
    if EngSpeedPenalty < 1
        stradd = sprintf('_EngSpeedPen%0.3f',EngSpeedPenalty);
        str = [str, stradd];
    elseif EngSpeedPenalty < 10
        stradd = sprintf('_EngSpeedPen%1.0f',EngSpeedPenalty);
        str = [str, stradd];
    else
        stradd = sprintf('_EngSpeedPen%2.0f',EngSpeedPenalty);
        str = [str, stradd];
    end
end
str=strrep(str,'0.','0');
if EngPenalty > 0
    str=strrep(str,'e-00','e-');
    str=strrep(str,'1e+000','1');
    str=strrep(str,'0e+000','0');
else
    str=strrep(str,'0e+000','0');
end
eval(['save ' str ' optimal AccumV EngPenalty oilStep percent AccumStates'])
```

Hydraulic hybrid DP time step calculation - HMT only

```
function [optimal, AccumStates] = DP_HMTGen2_HMTOnly(drivecycle, ...
    oilStep,EngPenalty,ModePenalty,percent,EngOff,StopDeclutch, ...
    BSFCCurve,ExpA6,EngSpeedPenalty)
```

```
global AccumV AccumP_initial Vg_initial
global RT mass dragc fronta airdens fo fs g dia slope
global psi2pa
global pp bbsfc_spd bbsfc_trq eng_pwr

VehSpeed = drivecycle(:,1);
deltat = drivecycle(1,3);          %Time step (s)
N = size(VehSpeed, 1);  %Number of time steps

lowerOilValue = ceil((AccumV-(AccumP_initial*Vg_initial)/ ...
    (2000*psi2pa))/oilStep)*oilStep;
upperOilValue = floor((AccumV-(AccumP_initial*Vg_initial)/ ...
    (4000*psi2pa))/oilStep)*oilStep;
AccumStates = lowerOilValue:oilStep:upperOilValue;

if BSFCCurve
    EPower = [0,3000:100:20000]';
    ESpeed = interp1(eng_pwr,bbsfc_spd,EPower)*60/(2*pi);
    ETorque = interp1(eng_pwr,bbsfc_trq,EPower);
else
    ESpeed = [0,1400:10:2600]';
    ETorque = 85*ones(length(ESpeed),1);
end

if EngOff
    BSpeed = [0,500:100:3500]';
    ESpeed(1) = 0;
    ETorque(1) = 0;
else
    ESpeed(1) = 1100;
    BSpeed = 1100;
    ETorque(1) = 2;
```

```
end

for k = N-1:-1:1
    tic
    k
    FuelCons = Inf(length(AccumStates));
    modeResult = FuelCons;
    EngSpeed = NaN(length(AccumStates));
    EngSpeedmode = EngSpeed;
    EngTorque = EngSpeed;
    PercentSol = EngSpeed;
    PercentSolmode = PercentSol;
    ModeSol = EngSpeed;
    ModeSolmode = ModeSol;
    BSpeedSol = ModeSol;
    Idle = zeros(length(AccumStates));
    if k ~= N-1
        str = '0000';
        str = num2str(k+1);
        if length(str) == 1
            str = ['000' str];
        elseif length(str) == 2
            str = ['00' str];
        elseif length(str) == 3
            str = ['0' str];
        end
        eval(['EngOnFuture = optimal.k' str '.Idle*' ...
            'ones(1,length(AccumStates)) > 0;']);
        EngOnFuture = EngOnFuture';
        eval(['ModeFuture = optimal.k' str '.Mode*' ...
            'ones(1,length(AccumStates));']);
        ModeFuture = ModeFuture';
        eval(['EngSpeedFuture = optimal.k' str '.EngSpeed;']);
        EngSpeedFuture=EngSpeedFuture*ones(1,length(AccumStates));
        EngSpeedFuture=EngSpeedFuture';
    end
    CurVehSpeed = VehSpeed(k);
    NextVehSpeed = VehSpeed(k+1);
    cycle = wheelpwr(CurVehSpeed,NextVehSpeed,deltat, mass,dragc, ...
        fronta,airdens,fo,fs,g,dia,slope);
    mode = 1;
        if EngOff == 0
            if StopDeclutch
                DeclutchFlag = 0;
                if CurVehSpeed == 0
                    DeclutchFlag = 1;
                end
                sol = HMTGen2_new_exp_StopDeclutch(DeclutchFlag, cycle, ...
                    deltat, percent, ESpeed, ETorque, AccumStates, mode, ...
                    BSpeed);
            elseif ExpA6
                sol = HMTGen2_new_exp(cycle, deltat, percent, ESpeed, ...
                    ETorque, AccumStates, mode, BSpeed);
            else
                sol = HMTGen2_new(cycle, deltat, percent, ESpeed, ETorque, ...
                    AccumStates, mode, BSpeed);
            end
        else
            if ExpA6
                sol = HMTGen2_new_exp(cycle, deltat, percent, ...
                    ESpeed(2:end), ETorque(2:end), AccumStates, mode, 1000);
                EngOffsol = HMTGen2_new_exp(cycle, deltat, percent, ...
                    ESpeed(1), ETorque(1), AccumStates, mode, BSpeed);
            else
```

```
                sol = HMTGen2_new(cycle, deltat, percent, ESpeed(2:end), ...
                    ETorque(2:end), AccumStates, mode, 1000);
                EngOffsol = HMTGen2_new(cycle, deltat, percent, ESpeed(1), ...
                    ETorque(1), AccumStates, mode, BSpeed);
            end
        end
    for ii = 1:length(AccumStates)
        volOil = sol(:,1,ii);
        if length(BSpeed) == 1
            volOilEngOff = volOil(1);
            columnEngOff = length(AccumStates)+1;
            if volOilEngOff >= AccumStates(1) && ...
                    volOilEngOff <= AccumStates(end)
                columnEngOff = round((volOilEngOff-AccumStates(1))/oilStep)+1;
                if EngOff
                    FuelCons(ii,columnEngOff) = 0;
                else
                    FuelCons(ii,columnEngOff) = 0.082;
                end
                EngSpeed(ii,columnEngOff) = ESpeed(1);
                EngTorque(ii,columnEngOff) = ETorque(1);
                PercentSol(ii,columnEngOff) = 0;
                ModeSol(ii,columnEngOff) = mode;
                Idle(ii,columnEngOff) = 1;
                BSpeedSol(ii,columnEngOff) = sol(1,7,ii)*(60/(2*pi));
            end
            volOil(1) = [];
        else
            volOilEngOff = EngOffsol(:,1,ii);
            volOilEngOff(volOilEngOff<-0.1) = 1;
            LowLimit = min(volOilEngOff);
            LowLimit(LowLimit<AccumStates(1)) = AccumStates(1);
```

```
volOilEngOff(volOilEngOff>0.1) = -1;
HighLimit = max(volOilEngOff);
HighLimit(HighLimit>AccumStates(end)) = AccumStates(end);
NextOilVolEngOff = ceil(LowLimit/oilStep):1: ...
    floor(HighLimit/oilStep);
NextOilVolEngOff = NextOilVolEngOff*oilStep;
volOilCur = volOilEngOff;
volOilCur(1) = round(volOilCur(1)/oilStep)*oilStep;
feasBSpeedCur = BSpeed;
nonfeasBSpeed = feasBSpeedCur;
nonfeasBSpeed(volOilCur>-oilStep & ...
    volOilCur<AccumStates(end)+oilStep) = 1;
nonfeasBSpeed(nonfeasBSpeed == 1) = [];
feasBSpeedCur(volOilCur<-oilStep) = [];
volOilCur(volOilCur<-oilStep) = [];
feasBSpeedCur(volOilCur>AccumStates(end)+oilStep) = [];
volOilCur(volOilCur>AccumStates(end)+oilStep) = [];
columnEngOff = zeros(length(NextOilVolEngOff),1);
for jj = 1:length(NextOilVolEngOff)
    if ~isempty(feasBSpeedCur)
        if length(feasBSpeedCur) == 1
            BSpeedCur = feasBSpeedCur;
        else
            BSpeedCur = interp1(volOilCur,feasBSpeedCur, ...
                NextOilVolEngOff(jj));
        end
        columnEngOff(jj) = round((NextOilVolEngOff(jj)- ...
            AccumStates(1))/oilStep)+1;
        if k == N-1
            if EngOff
                FuelConsCur = 0;
            else
            FuelConsCur = 0.082;
        end
        EngSpeedCur = ESpeed(1);
        if FuelConsCur < FuelCons(ii,columnEngOff(jj))
            EngSpeed(ii,columnEngOff(jj)) = EngSpeedCur;
            EngTorque(ii,columnEngOff(jj)) = 0;
            FuelCons(ii,columnEngOff(jj)) = FuelConsCur;
            PercentSol(ii,columnEngOff(jj)) = 0;
            Idle(ii,columnEngOff(jj)) = 1;
            ModeSol(ii,columnEngOff(jj)) = mode;
            BSpeedSol(ii,columnEngOff(jj)) = BSpeedCur;
        end
    else
        str = '0000';
        str = num2str(k+1);
        if length(str) == 1
            str = ['000' str];
        elseif length(str) == 2
            str = ['00' str];
        elseif length(str) == 3
            str = ['0' str];
        end
        if EngOff
            FuelConsCur = 0;
            EngSpeedCur = 0;
        else
            FuelConsCur = 0.082;
            EngSpeedCur = ESpeed(1);
        end
        if FuelConsCur < FuelCons(ii,columnEngOff(jj))
            EngSpeed(ii,columnEngOff(jj)) = EngSpeedCur;
            EngTorque(ii,columnEngOff(jj)) = ETorque(1);
```

```
                    FuelCons(ii,columnEngOff(jj)) = FuelConsCur;
                    PercentSol(ii,columnEngOff(jj)) = 0;
                    Idle(ii,columnEngOff(jj)) = 1;
                    ModeSol(ii,columnEngOff(jj)) = mode;
                    BSpeedSol(ii,columnEngOff(jj)) = BSpeedCur;
                end
                %                         end
            end
        end
    end
end
if ~isempty(volOil)
    volOil(volOil<-0.1) = 1;
    LowLimit = min(volOil);
    LowLimit(LowLimit<AccumStates(1)) = AccumStates(1);
    volOil(volOil>0.1) = -1;
    HighLimit = max(volOil);
    HighLimit(HighLimit>AccumStates(end)) = AccumStates(end);
    for mm = 1:length(percent)
        NextOilVol = ceil(LowLimit(mm)/oilStep):1: ...
            floor(HighLimit(mm)/oilStep);
        NextOilVol = NextOilVol*oilStep;
        if LowLimit(mm) == HighLimit(mm)
            NextOilVol = [];
        end
        volOilCur = volOil(:,mm);
        feasEngSpeedCur = ESpeed(2:end,1);
        feasEngTorqueCur = ETorque(2:end,1);
        nonfeasEngSpeed = feasEngSpeedCur;
        nonfeasEngSpeed(volOilCur>-oilStep & ...
            volOilCur<AccumStates(end)+oilStep) = 1;
        nonfeasEngSpeed(nonfeasEngSpeed == 1) = [];
```

```
feasEngSpeedCur(volOilCur<-oilStep) = [];
feasEngTorqueCur(volOilCur<-oilStep) = [];
volOilCur(volOilCur<-oilStep) = [];
feasEngSpeedCur(volOilCur>AccumStates(end)+oilStep) = [];
feasEngTorqueCur(volOilCur>AccumStates(end)+oilStep) = [];
volOilCur(volOilCur>AccumStates(end)+oilStep) = [];
Te = feasEngTorqueCur;
[eng_loss,eff,fuel] = engine_404C15(Te,feasEngSpeedCur* ...
    ((2*pi)/60),0);
feasEngPower = Te.*feasEngSpeedCur*(2*pi)/60;
for jj = 1:length(NextOilVol)
    if length(feasEngPower) > 1
        EngPwrSol = interp1(volOilCur,feasEngPower, ...
            NextOilVol(jj));
        EngOnSol = interp1(volOilCur,feasEngSpeedCur, ...
            NextOilVol(jj));
        EngTrqSol = interp1(volOilCur,Te,NextOilVol(jj));
    else
        EngOnSol = nonfeasEngSpeed(1);
    end
    if isempty(find(nonfeasEngSpeed == floor(EngOnSol/10)* ...
            10 | nonfeasEngSpeed == ceil(EngOnSol/10)*10,1))
        column = round((NextOilVol(jj)-AccumStates(1))/ ...
            oilStep)+1;
        if isempty(find(column == columnEngOff,1)) || k == N-1
            FuelConsCur = interp1(feasEngPower,fuel,EngPwrSol);
            if FuelConsCur < FuelCons(ii,column)
                EngSpeed(ii,column) = EngOnSol;
                EngTorque(ii,column) = EngTrqSol;
                FuelCons(ii,column) = FuelConsCur;
                PercentSol(ii,column) = percent(mm);
                ModeSol(ii,column) = mode;
```

```
                            BSpeedSol(ii,column) = EngOnSol*RT;
                        end
                    else
                        str = ’0000’;
                        str = num2str(k+1);
                        if length(str) == 1
                            str = [’000’ str];
                        elseif length(str) == 2
                            str = [’00’ str];
                        elseif length(str) == 3
                            str = [’0’ str];
                        end
                        eval([’EngStateNext = optimal.k’ str ...
                            ’.EngSpeed(column) > 0;’]);
                        if EngStateNext == 1
                            FuelConsCur = interp1(feasEngPower,fuel, ...
                                EngPwrSol);
                            if FuelConsCur < FuelCons(ii,column)
                                EngSpeed(ii,column) = EngOnSol;
                                EngTorque(ii,column) = EngTrqSol;
                                FuelCons(ii,column) = FuelConsCur;
                                PercentSol(ii,column) = percent(mm);
                                ModeSol(ii,column) = mode;
                                BSpeedSol(ii,column) = EngOnSol*RT;
                            end
                        end
                    end
                end
            end
        end
    end
end
```

```
if k ~= N-1
    EngOnNow = Idle > 0;
    idx = find(EngOnNow ~= EngOnFuture);
    FuelCons(idx) = EngPenalty + FuelCons(idx);
    idx = find(ModeSol == 1 & ModeFuture == 2);
    FuelCons(idx) = FuelCons(idx) + ModePenalty;
    idx = find(ModeSol == 1 & ModeFuture == 3);
    FuelCons(idx) = FuelCons(idx) + ModePenalty;
    diff=abs(EngSpeed-EngSpeedFuture);
    diff(EngOnNow==0 | EngOnFuture==0)=0;
    FuelCons=FuelCons+EngSpeedPenalty*diff;
end
cost = FuelCons;
if k ~= N-1
    eval(['FutureCost = optimal.k' str '.Cost;']);
    cost = bsxfun(@plus, cost, FutureCost');
end
[MinValue,MinInd] = min(cost,[],2);
str = '0000';
str = num2str(k);
if length(str) == 1
    str = ['000' str];
elseif length(str) == 2
    str = ['00' str];
elseif length(str) == 3
    str = ['0' str];
end
eval(['optimal.k' str '.Cost = MinValue;']);
for mm = 1:length(MinInd)
    eval(['optimal.k' str '.EngSpeed(mm,1) = EngSpeed(mm,MinInd(mm));']);
    eval(['optimal.k' str '.EngTorque(mm,1) = EngTorque(mm,MinInd(mm));']);
    eval(['optimal.k' str '.Mode(mm,1) = ModeSol(mm,MinInd(mm));']);
```

```
        eval(['optimal.k' str '.Percent(mm,1) = PercentSol(mm,MinInd(mm));']);
        eval(['optimal.k' str '.Idle(mm,1) = Idle(mm,MinInd(mm));']);
        eval(['optimal.k' str '.BSpeed(mm,1) = BSpeedSol(mm,MinInd(mm));']);
        eval(['optimal.k' str '.NextOilVol(mm,1) = AccumStates(MinInd(mm));']);
    end
    toc
end
```

Hydraulic hybrid DP time step calculation - Four mode

```
function [optimal, AccumStates] = DP_HMTGen2_4Mode(drivecycle,oilStep, ...
    EngPenalty,ModePenalty,percent,EngOff,StopDeclutch,BSFCCurve,ExpA6, ...
    EngSpeedPenalty)

global AccumV AccumP_initial Vg_initial
global RT mass dragc fronta airdens fo fs g dia slope
global psi2pa
global pp bbsfc_spd bbsfc_trq eng_pwr

VehSpeed = drivecycle(:,1);
deltat = drivecycle(1,3);           %Time step (s)
N = size(VehSpeed, 1);  %Number of time steps

lowerOilValue = ceil((AccumV-(AccumP_initial*Vg_initial)/ ...
    (2000*psi2pa))/oilStep)*oilStep;
upperOilValue = floor((AccumV-(AccumP_initial*Vg_initial)/ ...
    (4000*psi2pa))/oilStep)*oilStep;
AccumStates = lowerOilValue:oilStep:upperOilValue;

for k = N-1:-1:1
    tic
    k
    FuelCons = Inf(length(AccumStates));
```

```
modeResult = FuelCons;
EngSpeed = NaN(length(AccumStates));
EngSpeedmode = EngSpeed;
EngTorque = EngSpeed;
EngTorquemode = EngSpeed;
PercentSol = EngSpeed;
PercentSolmode = PercentSol;
ModeSol = EngSpeed;
ModeSolmode = ModeSol;
BSpeedSol = ModeSol;
BSpeedmode = ModeSol;
Idle = zeros(length(AccumStates));
Idlemode = Idle;
if k ~= N-1
    str = '0000';
    str = num2str(k+1);
    if length(str) == 1
        str = ['000' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 3
        str = ['0' str];
    end
    eval(['EngOnFuture = optimal.k' str ...
        '.Idle*ones(1,length(AccumStates)) < 0.5;']);
    EngOnFuture = EngOnFuture';
    eval(['ModeFuture = optimal.k' str ...
        '.Mode*ones(1,length(AccumStates));']);
    ModeFuture = ModeFuture';
    eval(['EngSpeedFuture = optimal.k' str '.EngSpeed;']);
    EngSpeedFuture=EngSpeedFuture*ones(1,length(AccumStates));
    EngSpeedFuture=EngSpeedFuture';
```

```
end
CurVehSpeed = VehSpeed(k);
NextVehSpeed = VehSpeed(k+1);
cycle = wheelpwr(CurVehSpeed,NextVehSpeed,deltat,mass,dragc,fronta, ...
    airdens,fo,fs,g,dia,slope);
mode = 1;
if BSFCCurve
    EPower = [0,3000:100:20000]';
    ESpeed = interp1(eng_pwr,bbsfc_spd,EPower)*60/(2*pi);
    ETorque = interp1(eng_pwr,bbsfc_trq,EPower);
else
    ESpeed = [0,1400:10:2600]';
    ETorque = 85*ones(length(ESpeed),1);
end

if EngOff
    BSpeed = [0,500:100:3500]';
    ESpeed(1) = 0;
    ETorque(1) = 0;
else
    ESpeed(1) = 1100;
    BSpeed = 1100;
    ETorque(1) = 2;
end
if EngOff == 0
    if StopDeclutch
        sol = HMTGen2_new_exp_StopDeclutch(cycle, deltat, percent, ...
            ESpeed, ETorque, AccumStates, mode, BSpeed);
    elseif ExpA6
        sol = HMTGen2_new_exp(cycle, deltat, percent, ESpeed, ...
            ETorque, AccumStates, mode, BSpeed);
    else
```

```
        sol = HMTGen2_new(cycle, deltat, percent, ESpeed, ETorque, ...
            AccumStates, mode, BSpeed);
    end
else
    if ExpA6
        sol = HMTGen2_new_exp(cycle, deltat, percent, ESpeed(2:end), ...
            ETorque(2:end), AccumStates, mode, 1000);
        EngOffsol = HMTGen2_new_exp(cycle, deltat, percent, ...
            ESpeed(1), ETorque(1), AccumStates, mode, BSpeed);
    else
        sol = HMTGen2_new(cycle, deltat, percent, ESpeed(2:end), ...
            ETorque(2:end), AccumStates, mode, 1000);
        EngOffsol = HMTGen2_new(cycle, deltat, percent, ESpeed(1), ...
            ETorque(1), AccumStates, mode, BSpeed);
    end
end
for ii = 1:length(AccumStates)
    volOil = sol(:,1,ii);
    if length(BSpeed) == 1
        volOilEngOff = volOil(1);
        columnEngOff = length(AccumStates)+1;
        if volOilEngOff >= AccumStates(1) && volOilEngOff <= ...
                AccumStates(end)
            columnEngOff = round((volOilEngOff-AccumStates(1))/oilStep)+1;
            if EngOff
                FuelCons(ii,columnEngOff) = 0;
            else
                FuelCons(ii,columnEngOff) = 0.082;
            end
            EngSpeed(ii,columnEngOff) = ESpeed(1);
            EngTorque(ii,columnEngOff) = ETorque(1);
            PercentSol(ii,columnEngOff) = 0;
            ModeSol(ii,columnEngOff) = mode;
            Idle(ii,columnEngOff) = 1;
            BSpeedSol(ii,columnEngOff) = sol(1,7,ii)*(60/(2*pi));
        end
        volOil(1) = [];
    else
        volOilEngOff = EngOffsol(:,1,ii);
        volOilEngOff(volOilEngOff<-0.1) = 1;
        LowLimit = min(volOilEngOff);
        LowLimit(LowLimit<AccumStates(1)) = AccumStates(1);
        volOilEngOff(volOilEngOff>0.1) = -1;
        HighLimit = max(volOilEngOff);
        HighLimit(HighLimit>AccumStates(end)) = AccumStates(end);
        NextOilVolEngOff = ceil(LowLimit/oilStep):1: ...
            floor(HighLimit/oilStep);
        NextOilVolEngOff = NextOilVolEngOff*oilStep;
        volOilCur = volOilEngOff;
        volOilCur(1) = round(volOilCur(1)/oilStep)*oilStep;
        feasBSpeedCur = BSpeed;
        nonfeasBSpeed = feasBSpeedCur;
        nonfeasBSpeed(volOilCur>-oilStep & volOilCur< ...
            AccumStates(end)+oilStep) = 1;
        nonfeasBSpeed(nonfeasBSpeed == 1) = [];
        feasBSpeedCur(volOilCur<-oilStep) = [];
        volOilCur(volOilCur<-oilStep) = [];
        feasBSpeedCur(volOilCur>AccumStates(end)+oilStep) = [];
        volOilCur(volOilCur>AccumStates(end)+oilStep) = [];
        columnEngOff = zeros(length(NextOilVolEngOff),1);
        for jj = 1:length(NextOilVolEngOff)
            if ~isempty(feasBSpeedCur)
                if length(feasBSpeedCur) == 1
                    BSpeedCur = feasBSpeedCur;
```

```
else
    BSpeedCur = interp1(volOilCur,feasBSpeedCur, ...
        NextOilVolEngOff(jj));
end
columnEngOff(jj) = round((NextOilVolEngOff(jj)- ...
    AccumStates(1))/oilStep)+1;
if k == N-1
    if EngOff
        FuelConsCur = 0;
    else
        FuelConsCur = 0.082;
    end
    EngSpeedCur = ESpeed(1);
    if FuelConsCur < FuelCons(ii,columnEngOff(jj))
        EngSpeed(ii,columnEngOff(jj)) = EngSpeedCur;
        EngTorque(ii,columnEngOff(jj)) = 0;
        FuelCons(ii,columnEngOff(jj)) = FuelConsCur;
        PercentSol(ii,columnEngOff(jj)) = 0;
        Idle(ii,columnEngOff(jj)) = 1;
        ModeSol(ii,columnEngOff(jj)) = mode;
        BSpeedSol(ii,columnEngOff(jj)) = BSpeedCur;
    end
else
    str = '0000';
    str = num2str(k+1);
    if length(str) == 1
        str = ['000' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 3
        str = ['0' str];
    end
```

```
                if EngOff
                    FuelConsCur = 0;
                    EngSpeedCur = 0;
                else
                    FuelConsCur = 0.082;
                    EngSpeedCur = ESpeed(1);
                end
                if FuelConsCur < FuelCons(ii,columnEngOff(jj))
                    EngSpeed(ii,columnEngOff(jj)) = EngSpeedCur;
                    EngTorque(ii,columnEngOff(jj)) = ETorque(1);
                    FuelCons(ii,columnEngOff(jj)) = FuelConsCur;
                    PercentSol(ii,columnEngOff(jj)) = 0;
                    Idle(ii,columnEngOff(jj)) = 1;
                    ModeSol(ii,columnEngOff(jj)) = mode;
                    BSpeedSol(ii,columnEngOff(jj)) = BSpeedCur;
                end
            end
        end
    end
end
if ~isempty(volOil)
    volOil(volOil<-0.1) = 1;
    LowLimit = min(volOil);
    LowLimit(LowLimit<AccumStates(1)) = AccumStates(1);
    volOil(volOil>0.1) = -1;
    HighLimit = max(volOil);
    HighLimit(HighLimit>AccumStates(end)) = AccumStates(end);
    for mm = 1:length(percent)
        NextOilVol = ceil(LowLimit(mm)/oilStep):1: ...
            floor(HighLimit(mm)/oilStep);
        NextOilVol = NextOilVol*oilStep;
        if LowLimit(mm) == HighLimit(mm)
    NextOilVol = [];
end
volOilCur = volOil(:,mm);
feasEngSpeedCur = ESpeed(2:end,1);
feasEngTorqueCur = ETorque(2:end,1);
nonfeasEngSpeed = feasEngSpeedCur;
nonfeasEngSpeed(volOilCur>-oilStep & volOilCur< ...
    AccumStates(end)+oilStep) = 1;
nonfeasEngSpeed(nonfeasEngSpeed == 1) = [];
feasEngSpeedCur(volOilCur<-oilStep) = [];
feasEngTorqueCur(volOilCur<-oilStep) = [];
volOilCur(volOilCur<-oilStep) = [];
feasEngSpeedCur(volOilCur>AccumStates(end)+oilStep) = [];
feasEngTorqueCur(volOilCur>AccumStates(end)+oilStep) = [];
volOilCur(volOilCur>AccumStates(end)+oilStep) = [];
Te = feasEngTorqueCur;
[eng_loss,eff,fuel] = engine_404C15(Te,feasEngSpeedCur* ...
    ((2*pi)/60),0);
feasEngPower = Te.*feasEngSpeedCur*(2*pi)/60;
for jj = 1:length(NextOilVol)
    if length(feasEngPower) > 1
        EngPwrSol = interp1(volOilCur,feasEngPower, ...
            NextOilVol(jj));
        EngOnSol = interp1(volOilCur,feasEngSpeedCur, ...
            NextOilVol(jj));
        EngTrqSol = interp1(volOilCur,Te,NextOilVol(jj));
    else
        EngOnSol = nonfeasEngSpeed(1);
    end
    if isempty(find(nonfeasEngSpeed == floor(EngOnSol/10)*10 ...
            | nonfeasEngSpeed == ceil(EngOnSol/10)*10,1))
        column = round((NextOilVol(jj)-AccumStates(1))/ ...
    oilStep)+1;
if isempty(find(column == columnEngOff,1)) || k == N-1
    FuelConsCur = interp1(feasEngPower,fuel,EngPwrSol);
    if FuelConsCur < FuelCons(ii,column)
        EngSpeed(ii,column) = EngOnSol;
        EngTorque(ii,column) = EngTrqSol;
        FuelCons(ii,column) = FuelConsCur;
        PercentSol(ii,column) = percent(mm);
        ModeSol(ii,column) = mode;
        BSpeedSol(ii,column) = EngOnSol*RT;
    end
else
    str = '0000';
    str = num2str(k+1);
    if length(str) == 1
        str = ['000' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 3
        str = ['0' str];
    end
    eval(['EngStateNext = optimal.k' str ...
        '.EngSpeed(column) > 0;']);
    if EngStateNext == 1
        FuelConsCur = interp1(feasEngPower,fuel, ...
            EngPwrSol);
        if FuelConsCur < FuelCons(ii,column)
            EngSpeed(ii,column) = EngOnSol;
            EngTorque(ii,column) = EngTrqSol;
            FuelCons(ii,column) = FuelConsCur;
            PercentSol(ii,column) = percent(mm);
            ModeSol(ii,column) = mode;
```

```
                                BSpeedSol(ii,column) = EngOnSol*RT;
                            end
                        end
                    end
                end
            end
        end
    end
end
if k ~= N-1
    EngOnNow = Idle == 0;
    idx = find(EngOnNow ~= EngOnFuture);
    FuelCons(idx) = EngPenalty + FuelCons(idx);
    idx = find(ModeSol == 1 & ModeFuture == 2);
    FuelCons(idx) = FuelCons(idx) + ModePenalty;
    idx = find(ModeSol == 1 & ModeFuture == 3);
    FuelCons(idx) = FuelCons(idx) + ModePenalty;
    diff=abs(EngSpeed-EngSpeedFuture);
    diff(EngOnNow==0 | EngOnFuture==0)=0;
    FuelCons=FuelCons+EngSpeedPenalty*diff;
end
mode = 2;
BSpeed = 1500;
if EngOff == 0
    ESpeed = [1100;1500];
    ETorque = [2;85];
    if StopDeclutch
        sol = HMTGen2_new_exp_StopDeclutch(cycle, deltat, percent, ...
            ESpeed, ETorque, AccumStates, mode, BSpeed);
    elseif ExpA6
        sol = HMTGen2_new_exp(cycle, deltat, percent, ESpeed, ...
            ETorque, AccumStates, mode, BSpeed);
    else
        sol = HMTGen2_new(cycle, deltat, percent, ESpeed, ETorque, ...
            AccumStates, mode, BSpeed);
    end
else
    ESpeed = [0;1500];
    ETorque = [0;85];
    if ExpA6
        sol = HMTGen2_new_exp(cycle, deltat, percent, ESpeed, ...
            ETorque, AccumStates, mode, BSpeed);
    else
        sol = HMTGen2_new(cycle, deltat, percent, ESpeed, ETorque, ...
            AccumStates, mode, BSpeed);
    end
end
for ii = 1:length(AccumStates)
    volOil = sol(:,:,ii);
    EngOffSol = volOil(1);
    columnEngOff = 0;
    if EngOffSol >= AccumStates(1) && EngOffSol <= AccumStates(end)
        columnEngOff = round((EngOffSol-AccumStates(1))/oilStep)+1;
        if EngOff
            modeResult(ii,columnEngOff) = 0;
        else
            modeResult(ii,columnEngOff) = 0.082;
        end
        EngSpeedmode(ii,columnEngOff) = ESpeed(1);
        EngTorquemode(ii,columnEngOff) = ETorque(1);
        PercentSolmode(ii,columnEngOff) = 0;
        Idlemode(ii,columnEngOff) = 1;
        ModeSolmode(ii,columnEngOff) = mode;
        BSpeedmode(ii,columnEngOff) = ESpeed(1)*RT;
```

```
end
volOil(1,:) = [];
EngOnSol = (cycle(1,10)/2)*8.625;
for mm = 1:length(percent)
    NextOilVol = volOil(mm);
    if NextOilVol > AccumStates(1) && NextOilVol < AccumStates(end)
        column = round((NextOilVol-AccumStates(1))/oilStep)+1;
        if column ~= columnEngOff || k == N-1
            Te = ETorque(2);
            [eng_loss,eff,fuel] = engine_404C15(Te,EngOnSol*(pi/30),0);
            FuelConsCur = fuel;
            if FuelConsCur < modeResult(ii,column)
                EngSpeedmode(ii,column) = EngOnSol;
                EngTorquemode(ii,column) = Te;
                modeResult(ii,column) = FuelConsCur;
                PercentSolmode(ii,column) = percent(mm);
                ModeSolmode(ii,column) = mode;
                BSpeedmode(ii,column) = EngOnSol*RT;
            end
        else
            str = '0000';
            str = num2str(k+1);
            if length(str) == 1
                str = ['000' str];
            elseif length(str) == 2
                str = ['00' str];
            elseif length(str) == 3
                str = ['0' str];
            end
            eval(['EngStateNext = optimal.k' str ...
                '.EngSpeed(column) > 0;']);
            if EngStateNext == 1
```

```
                    Te = ETorque(2);
                    [eng_loss,eff,fuel] = engine_404C15(Te, ...
                        EngOnSol*(pi/30),0);
                    FuelConsCur = fuel;
                    if FuelConsCur < modeResult(ii,column)
                        EngSpeedmode(ii,column) = EngOnSol;
                        EngTorquemode(ii,column) = Te;
                        modeResult(ii,column) = FuelConsCur;
                        PercentSolmode(ii,column) = percent(mm);
                        ModeSolmode(ii,column) = mode;
                        BSpeedmode(ii,column) = EngOnSol*RT;
                    end
                end
            end
        end
    end
end
if k ~= N-1
    EngOnNow = EngSpeedmode > 0;
    idx = find(EngOnNow ~= EngOnFuture);
    modeResult(idx) = EngPenalty + modeResult(idx);
    idx = find(ModeSolmode == 2 & ModeFuture == 1);
    modeResult(idx) = modeResult(idx) + ModePenalty;
    idx = find(ModeSolmode == 2 & ModeFuture == 3);
    modeResult(idx) = modeResult(idx) + ModePenalty;
end
mode = 3;
if ExpA6
    sol = HMTGen2_new_exp(cycle, deltat, percent, 0, 0, AccumStates, ...
        mode, 0);
else
    sol = HMTGen2_new(cycle, deltat, percent, 0, 0, AccumStates, mode, 0);
```

```
end
for ii = 1:length(AccumStates)
    if EngOff
        FuelConsCur = 0;
    else
        FuelConsCur = 0.082;
    end
    volOil = sol(:,:,ii);
    EngOffSol = volOil(1);
    if EngOffSol >= AccumStates(1) && EngOffSol <= AccumStates(end)
        columnEngOff = round((EngOffSol-AccumStates(1))/oilStep)+1;
        if k ~= N-1
            if EngOnFuture(1,columnEngOff) == 1
                FuelConsCur = FuelConsCur + EngPenalty;
            end
            if ModeFuture(1,columnEngOff) == 1
                FuelConsCur = FuelConsCur + ModePenalty;
            end
            if ModeFuture(1,columnEngOff) == 2
                FuelConsCur = FuelConsCur + ModePenalty;
            end
        end
        if FuelConsCur < modeResult(ii,columnEngOff)
            modeResult(ii,columnEngOff) = FuelConsCur;
            EngSpeedmode(ii,columnEngOff) = 0;
            EngTorquemode(ii,columnEngOff) = 0;
            PercentSolmode(ii,columnEngOff) = 0;
            Idlemode(ii,columnEngOff) = 1;
            ModeSolmode(ii,columnEngOff) = mode;
            BSpeedmode(ii,columnEngOff) = 0;
        end
    end
```

```
end
idx = find(modeResult < FuelCons);
FuelCons(idx) = modeResult(idx);
EngSpeed(idx) = EngSpeedmode(idx);
EngTorque(idx) = EngTorquemode(idx);
PercentSol(idx) = PercentSolmode(idx);
ModeSol(idx) = ModeSolmode(idx);
BSpeedSol(idx) = BSpeedmode(idx);
Idle = +(EngSpeed < 1150);
cost = FuelCons;
if k ~= N-1
    eval(['FutureCost = optimal.k' str '.Cost;']);
    cost = bsxfun(@plus, cost, FutureCost');
end
[MinValue,MinInd] = min(cost,[],2);
str = '0000';
str = num2str(k);
if length(str) == 1
    str = ['000' str];
elseif length(str) == 2
    str = ['00' str];
elseif length(str) == 3
    str = ['0' str];
end
eval(['optimal.k' str '.Cost = MinValue;']);
for mm = 1:length(MinInd)
    eval(['optimal.k' str '.EngSpeed(mm,1) = EngSpeed(mm,MinInd(mm));']);
    eval(['optimal.k' str '.EngTorque(mm,1) = EngTorque(mm,MinInd(mm));']);
    eval(['optimal.k' str '.Mode(mm,1) = ModeSol(mm,MinInd(mm));']);
    eval(['optimal.k' str '.Percent(mm,1) = PercentSol(mm,MinInd(mm));']);
    eval(['optimal.k' str '.Idle(mm,1) = Idle(mm,MinInd(mm));']);
    eval(['optimal.k' str '.BSpeed(mm,1) = BSpeedSol(mm,MinInd(mm));']);
```

```
        eval(['optimal.k' str '.NextOilVol(mm,1) = AccumStates(MinInd(mm));']);
    end
    toc
end
```

Dynamic programming forward algorithm

```
clear;
clc;

load fc_map_404C15

filename1='UrbanDriveCycle.csv';                %File with drivecycle
drivecycle=csvread(filename1,2,0);
% load 0to60_13s_T_accelOnly;
oilStep = 1e-4;
EngPenalty = 1;         %Fuel penalty for turning on/off engine
ModePenalty = 1;        %Fuel penalty for changing between modes
EngSpeedPenalty = 0;
percent = 1;
EngOff = 1;   %Set to 1 for Engine Off operation, 0 for clutch always on
StopDeclutch = 1;  %Set to 1 to declutch when stopped, 0 to always be clutched
BSFCCurve = 0;  %Set to 1 to use BSFC curve, 0 for constant torque
ExpA6 = 1;      %Set to 1 to use experimental A6 map
FourMode = 1;

HMTGen2Parameters;        %Loads all parameters for HMT vehicle

if FourMode
    if EngOff
        if BSFCCurve
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        end
    else
        if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                '4Mode_EngOff'],AccumV,EngPenalty,ModePenalty);
            end
        end
    else
        if BSFCCurve
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            end
        else
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                    'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                    '4Mode'],AccumV,EngPenalty,ModePenalty);
            end
        end
        if EngOff == 0
            str = [str, '_ClutchOn'];
        end
        if StopDeclutch == 1
            str = [str, '_StopDeclutch'];
        end
    end
else
    if EngOff
        if BSFCCurve
            if EngPenalty == 0
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 1
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
            elseif EngPenalty < 10
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                    'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                    'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
            else
                str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        end
    else
        if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 10
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        else
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
                'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
                'HMTOnly_EngOff'],AccumV,EngPenalty,ModePenalty);
        end
    end
else
    if BSFCCurve
        if EngPenalty == 0
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                'HMTOnly'],AccumV,EngPenalty,ModePenalty);
        elseif EngPenalty < 1
            str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
                'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
                'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    elseif EngPenalty < 10
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
            'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    else
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
            'ModePen%0.0f_engine404C15_bbsfc_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    end
else
    if EngPenalty == 0
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.0f_' ...
            'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    elseif EngPenalty < 1
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%0.2f_' ...
            'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    elseif EngPenalty < 10
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%1.0f_' ...
            'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    else
        str = sprintf(['DP_HMTGen2_AccumV%0.3f_EngPen%2.0f_' ...
            'ModePen%0.0f_engine404C15_torque85_vehparams_MLoil_' ...
            'HMTOnly'],AccumV,EngPenalty,ModePenalty);
    end
end
if EngOff == 0
    str = [str, '_ClutchOn'];
end
if StopDeclutch == 1
```

```
            str = [str, '_StopDeclutch'];
        end
    end
end
if ExpA6
    str = [str '_expA6'];
end
if e_rh == 1
    str = [str, '_IdealGears'];
end
if EngSpeedPenalty > 0
    if EngSpeedPenalty < 1
        stradd = sprintf('_EngSpeedPen%0.4f',EngSpeedPenalty);
        str = [str, stradd];
    elseif EngSpeedPenalty < 10
        stradd = sprintf('_EngSpeedPen%1.0f',EngSpeedPenalty);
        str = [str, stradd];
    else
        stradd = sprintf('_EngSpeedPen%2.0f',EngSpeedPenalty);
        str = [str, stradd];
    end
end
str=strrep(str,'0.','0');
if EngPenalty > 0
    str=strrep(str,'e-00','e-');
    str=strrep(str,'1e+000','1');
    str=strrep(str,'0e+000','0');
else
    str=strrep(str,'0e+000','0');
end
load(str)
```

```
if BSFCCurve
    load bbsfc_404C15
end

voilInit = AccumStates(2);

if FourMode
    DPData = DP_Forward_HMTGen2_FourMode(drivecycle,oilStep,voilInit, ...
        AccumStates,EngOff,optimal);
else
    DPData = DP_Forward_HMTGen2_HMTOnly(drivecycle,oilStep,voilInit, ...
        AccumStates,EngOff,optimal);
end
disp(['File Loaded: ' str])

fuel_used=sum(DPData.FuelCons);
m2mile = 100/(2.54*12*5280);
length_dv = sum(drivecycle(:,1))*m2mile;
rho_fuel = 832; %g/l
liter2gal = 0.264172;
rho_fuel = rho_fuel/liter2gal; %g/gal
mpg = length_dv/(fuel_used/rho_fuel)

Pressure_psi = ((AccumP_initial*Vg_initial)./ ...
    (AccumV-DPData.VolOil))/6894.75729;
Pressure_Pa = Pressure_psi*6894.75729/1e6;
DPData.Pressure = Pressure_psi;

if FourMode
    DPData.Mode(DPData.Mode==3)=4;
    DPData.Mode(DPData.Mode==2 & DPData.EngSpeed>1100)=3;
    figure
```

```
str = sprintf('Dynamic Programming Results - Four Mode');
subplot(4,1,1)
plot(DPData.Time,drivecycle(1:end-1,1),'k-','LineWidth',2)
title(str);
str = sprintf('Vehicle \n Speed (m/s)');
ylabel(str);
subplot(4,1,2)
plot(DPData.Time,Pressure_psi,'k-','LineWidth',2)
axis([0 1400 2000 4000])
str = sprintf('Accumulator \n Pressure (psi)');
ylabel(str);
subplot(4,1,3)
plot(DPData.Time,DPData.EngSpeed,'k-','LineWidth',2)
axis([0 1400 0 3000])
str = sprintf('Engine \n Speed (rpm)');
ylabel(str);
subplot(4,1,4)
plot(DPData.Time,DPData.Mode,'k-','LineWidth',2)
str = sprintf('Mode');
ylabel(str)
axis([0 1400 0 5])
xlabel('Time (s)');
figure
idx=find(DPData.WlSpeed>0);
Modes=DPData.Mode(idx);
idxHMT=find(Modes==1);
idxParallel=find(Modes==2);
idxTOnly=find(Modes==3);
idxSOnly=find(Modes==4);
percHMT=(length(idxHMT)/length(Modes))*100;
percParallel=(length(idxParallel)/length(Modes))*100;
percTOnly=(length(idxTOnly)/length(Modes))*100;
```

```
    percSOnly=(length(idxSOnly)/length(Modes))*100;
    strHMT = sprintf('HMT\n %2.0f%%',percHMT);
    strParallel = sprintf('Parallel\n %2.0f%%',percParallel);
    strTOnly = sprintf('T-Only\n %2.0f%%',percTOnly);
    strSOnly = sprintf('S-Only\n %2.0f%%',percSOnly);
    X = [length(idxHMT);length(idxParallel);length(idxTOnly);length(idxSOnly)];
    pie(X,{strHMT,strParallel,strTOnly,strSOnly})
    colormap(jet)
else
    figure
    str = sprintf('Dynamic Programming Results - HMT-Only Mode');
    subplot(4,1,1)
    plot(DPData.Time,drivecycle(1:end-1,1),'k-','LineWidth',2)
    title(str);
    str = sprintf('Vehicle \n Speed (m/s)');
    ylabel(str);
    subplot(4,1,2)
    plot(DPData.Time,Pressure_psi,'k-','LineWidth',2)
    axis([0 1400 2000 4000])
    str = sprintf('Accumulator \n Pressure (psi)');
    ylabel(str);
    subplot(4,1,3)
    plot(DPData.Time,DPData.EngSpeed,'k-','LineWidth',2)
    str = sprintf('Engine \n Speed (rpm)');
    ylabel(str);
    subplot(4,1,4)
    plot(DPData.Time,DPData.EngTorque,'k-','LineWidth',2)
    str = sprintf('Engine \n Torque (Nm)');
    ylabel(str)
    axis([0 1400 0 100])
    xlabel('Time (s)');
end
```

DP forward time step calculation - HMT Only

```
function DPData = DP_Forward_HMTGen2_HMTOnly(drivecycle, oilStep, ...
    VoilInit, AccumStates, EngOff, optimal)

global mass dragc fronta airdens fo fs g dia slope
global AccumP_initial Vg_initial AccumV
global Rdiff RT RS rh e_Rdiff e_RT e_RS e_rh

addpath('Rexroth A6')

VehSpeed = drivecycle(:,1);
deltat = drivecycle(1,3);          %Time step (s)
N = size(VehSpeed, 1);  %Number of time steps
DPsol = NaN(length(fieldnames(optimal)),19);



Voil = VoilInit;
EngTrqExist = isfield(optimal.k0001,'EngTorque');
for k = 1:N-1
    k
    DPData.Time(k,1) = k-1;
    str = num2str(k);
    if length(str) == 1
        str = ['000' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 3
        str = ['0' str];
    end
    CurVehSpeed = VehSpeed(k);
    NextVehSpeed = VehSpeed(k+1);
    cycle = wheelpwr(CurVehSpeed,NextVehSpeed,deltat,mass,dragc,fronta, ...
        airdens,fo,fs,g,dia,slope);
    DPData.WlTorque(k,1) = cycle(8)*(dia/2);
    DPData.WlSpeed(k,1) = (cycle(1,10)*((2*pi)/60));
    % cycle(8) = DesWheelTorqueDP(k)/(dia/2);
    row = round((Voil-AccumStates(1))/oilStep)+1;
    eval(['DPData.EngSpeed(k,1) = optimal.k' str '.EngSpeed(row,1);']);
    if EngTrqExist
        eval(['DPData.EngTorque(k,1) = optimal.k' str '.EngTorque(row,1);']);
    else
        DPData.EngTorque(k,1) = 70;
    end
    eval(['DPData.Mode(k,1) = optimal.k' str '.Mode(row,1);']);
    eval(['DPData.Idle(k,1) = optimal.k' str '.Idle(row,1);']);
    eval(['DPData.wT(k,1) = optimal.k' str '.BSpeed(row,1);']);
    DPData.VolOil(k,1) = Voil;
    eval(['DPData.NextOilVol(k,1) = optimal.k' str '.NextOilVol(row,1);']);
    Voil=DPData.NextOilVol(k,1);
end

wltorque = -DPData.WlTorque;
wlspeed = DPData.WlSpeed;

%Initialize Variables
w_T = zeros(size(wltorque));
T_T = w_T;
x_T = w_T;
Q_T = w_T;
T_S = w_T;
x_S = w_T;
Q_S = w_T;

w_T = DPData.wT;
```

```
Te = DPData.EngTorque;
E = DPData.EngSpeed;

%Torque on output shaft of transmission
Tout = wltorque/Rdiff;
Tout(wltorque.*wlspeed>0) = Tout(wltorque.*wlspeed>0)*e_Rdiff;
Tout(wltorque.*wlspeed<0) = Tout(wltorque.*wlspeed<0)/e_Rdiff;

%Torque on ring gear
Tring = -Tout/(rh+1);
Tring(w_T.*Tring>0) = Tring(w_T.*Tring>0)/e_rh;
Tring(w_T.*Tring<0) = Tring(w_T.*Tring<0)*e_rh;

%Torque on the shaft of the hydraulic unit
T_T(abs(w_T)>0) = ((Tring(abs(w_T)>0)-Te(abs(w_T)>0))/RT)+ ...
    0.07*E(abs(w_T)>0)*pi/30;
T_T(w_T.*T_T>0) = T_T(w_T.*T_T>0)/e_RT;
T_T(w_T.*T_T<0) = T_T(w_T.*T_T<0)*e_RT;
DPData.TT = T_T;

%Speed of P/M S
w_S = w_T/(RS*RT*rh)-(((1+(1/rh))*Rdiff)/RS)*wlspeed;
DPData.wS = w_S;

%Torque on the shaft of the hydraulic unit
T_S = (RS/(1+(1/rh)))*Tout;
T_S(w_S.*T_S>0) = T_S(w_S.*T_S>0)/e_RS;
T_S(w_S.*T_S<0) = T_S(w_S.*T_S<0)*e_RS;
DPData.TS = T_S;

P = AccumP_initial.*(Vg_initial./(AccumV-DPData.VolOil));  %Pressure
DPData.Pressure = P;
```

```
dP = P;

[x_T(abs(w_T)>0),Q_T(abs(w_T)>0)] = RexrothA6_exp(T_T(abs(w_T)>0), ...
    dP(abs(w_T)>0),w_T(abs(w_T)>0));
DPData.xT = x_T;
DPData.QT = Q_T;

[x_S(abs(w_S)>0),Q_S(abs(w_S)>0)] = RexrothA6_exp(T_S(abs(w_S)>0), ...
    dP(abs(w_S)>0),w_S(abs(w_S)>0));
DPData.xS = x_S;
DPData.QS = Q_S;

DPData.FuelCons = zeros(length(DPData.VolOil),1);
[eng_loss,eff,FuelCons] = engine_404C15(DPData.EngTorque, ...
    DPData.EngSpeed*((2*pi)/60),0);
if EngOff
    DPData.FuelCons(DPData.Idle==1) = 0;
else
    DPData.FuelCons(DPData.Idle==1) = 0.082;
end
DPData.FuelCons(DPData.Idle==0) = FuelCons(DPData.Idle==0);
```

DP forward time step calculation - Four mode

```
function DPData = DP_Forward_HMTGen2_FourMode(drivecycle, oilStep, ...
    VoilInit, AccumStates, EngOff, optimal)

global mass dragc fronta airdens fo fs g dia slope
global AccumP_initial Vg_initial AccumV
global Rdiff RT RS rh e_Rdiff e_RT e_RS e_rh

addpath('Rexroth A6')
```

```
VehSpeed = drivecycle(:,1);
deltat = drivecycle(1,3);           %Time step (s)
N = size(VehSpeed, 1);  %Number of time steps
DPsol = NaN(length(fieldnames(optimal)),19);


Voil = VoilInit;
EngTrqExist = isfield(optimal.k0001,'EngTorque');
for k = 1:N-1
    k
    DPData.Time(k,1) = k-1;
    str = num2str(k);
    if length(str) == 1
        str = ['000' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 3
        str = ['0' str];
    end
    CurVehSpeed = VehSpeed(k);
    NextVehSpeed = VehSpeed(k+1);
    cycle = wheelpwr(CurVehSpeed,NextVehSpeed,deltat,mass,dragc,fronta, ...
        airdens,fo,fs,g,dia,slope);
    DPData.WlTorque(k,1) = cycle(8)*(dia/2);
    DPData.WlSpeed(k,1) = (cycle(1,10)*((2*pi)/60));
    % cycle(8) = DesWheelTorqueDP(k)/(dia/2);
    row = round((Voil-AccumStates(1))/oilStep)+1;
    eval(['DPData.EngSpeed(k,1) = optimal.k' str '.EngSpeed(row,1);']);
    if EngTrqExist
        eval(['DPData.EngTorque(k,1) = optimal.k' str '.EngTorque(row,1);']);
    else
        DPData.EngTorque(k,1) = 70;
    end
    eval(['DPData.Mode(k,1) = optimal.k' str '.Mode(row,1);']);
    eval(['DPData.Idle(k,1) = optimal.k' str '.Idle(row,1);']);
    eval(['DPData.wT(k,1) = optimal.k' str '.BSpeed(row,1);']);
    DPData.VolOil(k,1) = Voil;
    eval(['DPData.NextOilVol(k,1) = optimal.k' str '.NextOilVol(row,1);']);
    Voil=DPData.NextOilVol(k,1);
end

wltorque = -DPData.WlTorque;
wlspeed = DPData.WlSpeed;

%Initialize Variables
w_T = zeros(size(wltorque));
T_T = w_T;
x_T = w_T;
Q_T = w_T;
T_S = w_T;
x_S = w_T;
Q_S = w_T;

w_T = DPData.wT;
Te = DPData.EngTorque;
E = DPData.EngSpeed;
mmode = DPData.Mode;

%Torque on output shaft of transmission
Tout = wltorque/Rdiff;
Tout(wltorque.*wlspeed>0) = Tout(wltorque.*wlspeed>0)*e_Rdiff;
Tout(wltorque.*wlspeed<0) = Tout(wltorque.*wlspeed<0)/e_Rdiff;

%Torque on ring gear
```

```
Tring = zeros(size(Tout));
Tring(mmode==1 | mmode==2) = -Tout(mmode==1 | mmode==2)/(rh+1);
Tring(w_T.*Tring>0) = Tring(w_T.*Tring>0)/e_rh;
Tring(w_T.*Tring<0) = Tring(w_T.*Tring<0)*e_rh;

%Torque on the shaft of the hydraulic unit
T_T(abs(w_T)>0) = ((Tring(abs(w_T)>0)-Te(abs(w_T)>0))/RT)+ ...
    0.07*E(abs(w_T)>0)*pi/30;
T_T(w_T.*T_T>0) = T_T(w_T.*T_T>0)/e_RT;
T_T(w_T.*T_T<0) = T_T(w_T.*T_T<0)*e_RT;
DPData.TT = T_T;

%Speed of P/M S
w_S = zeros(size(Tout));
w_S(mmode==1 | mmode==3) = w_T(mmode==1 | mmode==3)/(RS*RT*rh)- ...
    (((1+(1/rh))*Rdiff)/RS)*wlspeed(mmode==1 | mmode==3);
DPData.wS = w_S;

%Torque on the shaft of the hydraulic unit
T_S = zeros(size(Tout));
T_S(mmode==1 | mmode==3) = (RS/(1+(1/rh)))*Tout(mmode==1 | mmode==3);
T_S(w_S.*T_S>0) = T_S(w_S.*T_S>0)/e_RS;
T_S(w_S.*T_S<0) = T_S(w_S.*T_S<0)*e_RS;
DPData.TS = T_S;

P = AccumP_initial.*(Vg_initial./(AccumV-DPData.VolOil));  %Pressure
DPData.Pressure = P;
dP = P;

[x_T(abs(w_T)>0),Q_T(abs(w_T)>0)] = RexrothA6_exp(T_T(abs(w_T)>0), ...
    dP(abs(w_T)>0),w_T(abs(w_T)>0));
DPData.xT = x_T;
DPData.QT = Q_T;

[x_S(abs(w_S)>0),Q_S(abs(w_S)>0)] = RexrothA6_exp(T_S(abs(w_S)>0), ...
    dP(abs(w_S)>0),w_S(abs(w_S)>0));
DPData.xS = x_S;
DPData.QS = Q_S;

DPData.FuelCons = zeros(length(DPData.VolOil),1);
[eng_loss,eff,FuelCons] = engine_404C15(DPData.EngTorque, ...
    DPData.EngSpeed*((2*pi)/60),0);
if EngOff
    DPData.FuelCons(DPData.Idle==1) = 0;
else
    DPData.FuelCons(DPData.Idle==1) = 0.082;
end
DPData.FuelCons(DPData.Idle==0) = FuelCons(DPData.Idle==0);
```

Acceleration Markov chain

```
mph2ms = 0.44704;  %Conversion from mph to m/s

%Urban Dynamometer Driving Schedule (UDDS)
filename1='UrbanDriveCycle.csv';     %File with drivecycle
drivecycle=csvread([filename1],2,0); %read in drivecycle
temp = [0; drivecycle(:,1)];
temp(end,:) = [];
drivecycle(:,2) = temp;
clear temp
%West Virginia Interstate Driving Schedule
filename2='CYC_WVUINTER.mat';
load(filename2);
cycle(:,1)=cyc_mph(:,2)*mph2ms;
temp = [0; cycle(:,1)];
```

```
temp(end,:) = [];
cycle(:,2) = temp;
cycle(:,3) = [diff(cyc_mph(:,1));1];
drivecycle=[drivecycle; cycle];
clear temp cycle
%West Virginia Suburban Driving Schedule
filename3='CYC_WVUSUB.mat';
load(filename3);
cycle(:,1)=cyc_mph(:,2)*mph2ms;
temp = [0; cycle(:,1)];
temp(end,:) = [];
cycle(:,2) = temp;
cycle(:,3) = [diff(cyc_mph(:,1));1];
drivecycle=[drivecycle; cycle];
clear temp cycle
%West Virginia City Driving Schedule
filename4='CYC_WVUCITY.mat';
load(filename4);
cycle(:,1)=cyc_mph(:,2)*mph2ms;
temp = [0; cycle(:,1)];
temp(end,:) = [];
cycle(:,2) = temp;
cycle(:,3) = [diff(cyc_mph(:,1));1];
drivecycle=[drivecycle; cycle];
N = size(drivecycle,1);
time = 0:N-1;

Acceleration = -diff(drivecycle(:,1:2),1,2)./drivecycle(:,3);

% motorSpeed = drivecycle(:,1)/r_w;       %Motor Speed (rad/s)
vehicleSpeed = drivecycle(:,1);         %Vehicle Speed (m/s)
```

```
Na = 20;  %Number of discretized acceleration points
Nw = 20;  %Number of discretized wheel speed points

%Vector of discretized acceleration values
Accel = linspace(min(Acceleration), max(Acceleration), Na);

%Vector of discretized wheel speed values
Omega = linspace(min(vehicleSpeed), max(vehicleSpeed), Nw);

[Y, I] = min(abs(Accel));
Accel(I) = 0;   %Ensure lowest acceleration is 0

clear Y I

discreteSpeedMat = Omega'*ones(1,N);   %Matrix of discretized wheel speeds
SpeedMat = ones(Nw,1)*vehicleSpeed';   %Matrix of actual wheel speeds
%Take difference between discrete and actual values
diffMat = abs(SpeedMat-discreteSpeedMat);
[Y, I] = min(diffMat);   %Minimum of difference to find nearest-neighbor
discreteWheelSpeed = Omega(I)';  %Discrete wheel speed vector from drive cycle

clear discreteSpeedMat SpeedMat diffMat Y I

discreteAccelMat = Accel'*ones(1,N);   %Matrix of discretized accelerations
AccelMat = ones(Na,1)*Acceleration';    %Matrix of actual accelerations
%Take difference between discrete and actual values
diffMat = abs(AccelMat-discreteAccelMat);
[Y, I] = min(diffMat);   %Minimum of difference to find nearest-neighbor
%Discrete acceleration vector from drive cycle
discreteAcceleration = Accel(I)';
discreteAcceleration(end+1) = 0;
```

```
clear discretePowerMat PowerMat diffMat Y I

counts = zeros(Na,Na,Nw);
prob = zeros(Na,Na,Nw);
for j = 1:Nw
    % Find number of instances from current acceleration to next acceleration
    idx = find(discreteWheelSpeed == Omega(j));
    accls = [discreteAcceleration(idx), discreteAcceleration(idx+1)];
    counts(:,:,j) = hist3(accls, {Accel, Accel});
    totAccel = sum(counts(:,:,j),2);
    totAccel = totAccel*ones(1,Na);
    prob(:,:,j) = counts(:,:,j)./totAccel(:,:);
    idx = isnan(prob);
    prob(idx) = 0;
end
```

Build transition probability matrix - HMT

```
clear
clc

global pp
load fc_map_404C15.mat

S=pwd;
[PATH,NAME,EXT] = fileparts(S);
addpath(PATH)

HMTGen2Parameters

AccelerationMarkov

PressStep = 100;
```

```
Pressure = 2200:PressStep:4000;
Vo = AccumV-((Vg_initial*AccumP_initial)./(Pressure*psi2pa));
rowidx = 1;
states = zeros(length(Pressure)*Na,2);
% for k = 1:Nw
    for m = 1:Na
        for n = 1:length(Pressure)
%             states(rowidx,1) = Omega(k);
            states(rowidx,1) = Accel(m);
            states(rowidx,2) = Pressure(n);
            rowidx=rowidx+1;
        end
    end
% end

EngineSpeed = [1100,1400:100:2600];
rowidx = 1;
controls = EngineSpeed';
Te = 85*ones(size(controls));
Te(1) = 2;

deltat = 1;
for u = 1:size(controls,1)
    stru = num2str(u);
    if length(stru) == 1
        stru = ['0',stru];
    end
%     for velInd = 1:length(Omega)
    tic
    str = sprintf('Control %i of %i',u,size(controls,1));
    disp(str)
    for velInd = 1:Nw
```

```
strv = num2str(velInd);
if length(strv) == 1
    strv = ['0',strv];
end
str = sprintf('Velocity Index %i of %i',velInd,Nw);
disp(str);
TestSpeed = Omega(velInd);
temp = zeros(length(Pressure)*Na);
temp2 = temp;
sumNextAccel = sum(prob(:,:,velInd),1);
nonfeasNextAccel = find(sumNextAccel==0);
prob(:,nonfeasNextAccel,velInd) = 1/Na;
feasNextAccel = find(sumNextAccel ~= 0);
for accelInd = 1:Na
    TestAccel = Accel(accelInd);
    NextSpeed = TestSpeed+TestAccel;
    NextSpeed(NextSpeed<0) = 0;
    SpeedVector = linspace(TestSpeed,NextSpeed,(1/deltat)+1);
    cycle = wheelpwr(TestSpeed,NextSpeed,deltat,mass,dragc, ...
        fronta,airdens,fo,fs,g,dia,slope);
    sol = HMTGen2_new_exp(cycle, deltat, 1, controls(u), Te(u), ...
        Vo, 1, controls(u)*RT);
    if u == 1
        fuel = 0.082;
    else
        [eng_loss,eff,fuel] = engine_404C15(Te(u), ...
            controls(u)*(pi/30),0);
    end
    fuel = fuel*ones(size(Pressure));
    Vo_next = squeeze(sol(:,1,:));
    P_next = ((Vg_initial*AccumP_initial)./(AccumV-Vo_next))/psi2pa;
    fuel(abs(Vo_next)>0.5) = Inf;
    fuel(Vo_next<Vo(1)) = Inf;
    fuel(Vo_next>Vo(end)) = Inf;
    idx = [1:length(Vo)]';
        fuel(fuel==Inf) = 100;
        P_next = round(P_next/PressStep)*PressStep;
        feasAccel = find(prob(:,accelInd,velInd) > 0);
        newProb = prob(feasAccel,accelInd,velInd);
        accelRows = ((feasAccel-1)*length(Vo)*ones(1,length(idx)))';
        oilRows = idx*ones(1,length(feasAccel));
        rows = accelRows + oilRows;
        accelColms = ((accelInd-1)*length(Vo)* ...
            ones(1,length(idx)))'*ones(1,length(feasAccel));
        oilColms = (round((P_next(idx)-Pressure(1))/PressStep)+1)* ...
            ones(1,length(feasAccel));
        columns = accelColms + oilColms;
        probabilites = (newProb*ones(1,length(idx)))';
        cost = fuel(idx)'*ones(1,length(feasAccel));
        idx = find(rows<1);
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
        idx = find(columns<1);
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
        idx = find(rows>Na*length(Vo));
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
```

```
                idx = find(columns>Na*length(Vo));
                rows(idx) = [];
                columns(idx) = [];
                probabilites(idx) = [];
                cost(idx) = [];
            for ind = 1:numel(probabilites)
                temp(rows(ind),columns(ind)) = probabilites(ind);
                temp2(rows(ind),columns(ind)) = cost(ind);
            end
            clear probabilites cost
        end


        eval(['TPM.w' strv '.u' stru '= sparse(temp);']);
        eval(['TRM.w' strv '.u' stru '= sparse(temp2);']);
        clear temp temp2
    end
    eval(['save SolHHPVu' stru ' TPM TRM']);
    clear TPM TRM
    toc
end
```

Discounted policy iteration algorithm

```
function [policy x b] = pid2(tpm, trm, d_factor)

% Policy iteration for Discounted Reward Markov Decision Processes

NS = size(tpm.u01, 1);
NA = length(fieldnames(tpm));


policy = 2*ones(NS,1);   %Arbitrary policy
```

```
iteration = 0;
done = 1;

while done == 1
    iteration      %Uncomment to display iteration number
    G = zeros(NS);
    b = zeros(NS,1);
    % Policy Evaluation Stage

    for row = 1:NS
        str = ['u' int2str(policy(row))];
        if length(str) < 3
            str = ['u0' int2str(policy(row))];
        end
        eval(['G(row,:) = -d_factor*tpm.' str '(row,:);']);
        G(row,row) = 1-G(row,row);
    end

    for state = 1:NS
        str = ['u' int2str(policy(state))];
        if length(str) < 3
            str = ['u0' int2str(policy(state))];
        end
        eval(['tpmvec = tpm.' str '(state,:);']);
        eval(['trmvec = trm.' str '(state,:);']);
        b(state,1) = tpmvec*trmvec';
        clear tpmvec trmvec
    end

    tic           %Uncomment to display time to invert matrix
    x = inv(G)*b;
```

```
toc

% Policy improvement stage

done = 0;

tic                %Uncomment to display time to evaluate new decision
for state = 1:NS
    large = 1e12;
    best_action = 1;

    for action = 1:NA
        str = ['u' int2str(action)];
        if length(str) < 3
            str = ['u0' int2str(action)];
        end
        eval(['tpmvec = tpm.' str '(state,:);']);
        tpmvecfull = full(tpmvec);
        if sum(tpmvecfull) == 0 && action <= NA
            if action == NA
                break
            else
                continue
            end
        end
        eval(['trmvec = trm.' str '(state,:);']);
        trmvecfull = full(trmvec);
        idx = find(tpmvecfull == 0);
        tpmvecfull(idx) = [];
        trmvecfull(idx) = [];
        xsup = x;
        xsup(idx) = [];
```

```
            tot = tpmvecfull*trmvecfull' + d_factor*trmvecfull*xsup;
            if tot < large
                large = tot;
                best_action = action;
            end
        end

        if policy(state) ~= best_action;
            policy(state) = best_action;
            done = 1;
        end
    end
    toc

    if iteration == 15
        disp('Maximum Number of Iterations Reached!!')
        done = 0;
    end

    iteration = iteration + 1;

end

disp(sprintf('Number of iterations needed: %d\n', iteration));

return
```

Stochastic dynamic programming - HMT

```
clear
clc

S = pwd;
```

```
[PATH,FILE,EXT] = fileparts(S);
addpath(PATH)

HMTGen2Parameters

AccelerationMarkov

PressStep = 100;
Pressure = 2200:PressStep:4000;
Vo = AccumV-((Vg_initial*AccumP_initial)./(Pressure*psi2pa));
rowidx = 1;
states = zeros(length(Pressure)*Na,2);
% for k = 1:Nw
    for m = 1:Na
        for n = 1:length(Pressure)
            states(rowidx,1) = Accel(m);
            states(rowidx,2) = Pressure(n);
            rowidx=rowidx+1;
        end
    end
% end

EngineSpeed = [1100,1400:100:2600];
rowidx = 1;
controls = EngineSpeed';

%BuildTPM;
load SDP_HHPV_TPM_TRM

d_factor = 0.9;
for velInd = 1:Nw
    str = num2str(velInd);
    if length(str) == 1
        str = ['0',str];
    end
    disp(str);
    eval(['[policy.w' str ', x.w' str ', b.w' str '] = pid2(TPM.w' str ...
        ', TRM.w' str ', d_factor);'])
end

save SDP_HHPV_Results Accel Omega states controls policy x b
```

HHPV accumulator size study

```
global bbsfc_spd bbsfc_trq eng_pwr
load fc_map_404C15
load RandomDriveCycles

HMTGen2Parameters;        %Loads all parameters for HMT vehicle

% load 0to60_13s_T_accelOnly;
oilStep = 1e-4;
EngPenalty = 10;        %Fuel penalty for turning on/off engine
ModePenalty = 0;
EngSpeedPenalty = 0;
percent = 1;
EngOff = 0;   %Set to 1 for Engine Off operation, 0 for clutch always on
StopDeclutch = 0;  %Set to 1 to declutch when stopped, 0 to always be clutched
BSFCCurve = 0;  %Set to 1 to use BSFC curve, 0 for constant torque
ExpA6 = 0;      %Set to 1 to use experimental A6 map

if BSFCCurve
    load bbsfc_404C15
end
```

```
mpg = zeros(100,1);

for ii = 1:100
    str = sprintf('Drivecycle: %i',ii);
    disp(str);
    str = num2str(ii);
    if length(str) == 1
        str = ['00' str];
    elseif length(str) == 2
        str = ['0' str];
    end
    eval(['drivecycle = cyc.cycle' str ';'])

    [optimal, AccumStates] = DP_HMTGen2_HMTOnly(drivecycle,oilStep, ...
        EngPenalty,ModePenalty,percent,EngOff,StopDeclutch,BSFCCurve,ExpA6, EngSpeedPena

    voilInit = AccumStates(1);
    DPData = DP_Forward_HMTGen2_HMTOnly(drivecycle,oilStep,voilInit, ...
        AccumStates,optimal);

    fuel_used=sum(DPData.FuelCons);
    m2mile = 100/(2.54*12*5280);
    length_dv = sum(drivecycle(:,1))*m2mile;
    rho_fuel = 832; %g/l
    liter2gal = 0.264172;
    rho_fuel = rho_fuel/liter2gal; %g/gal
    mpg(ii) = length_dv/(fuel_used/rho_fuel);
end
```

Random drive cycle generator

```
clear
clc
```

```
Ns = 50;                              %Number of speed divisions
Na = 20;                              %Number of acceleration divisions
filename = 'UrbanDriveCycle.csv';     %Filename with drive cycle info
drivecycle = csvread(filename,2,0);   %Reads in drive cycle
MaxSpeed = max(drivecycle(:,1));      %Maximum vehicle speed
%Acceleration at each time step
accel = (drivecycle(:,2)-drivecycle(:,1))./drivecycle(:,3);
MinAccel = min(accel);
MaxAccel = max(accel);

idx = find(drivecycle(:,1)==0);       %Find where vehicle velocity is 0
previous = [0;idx(1:end-1)];          %Vector of previous indices
%Difference between current and previous index
difference = idx-previous;
idx2 = find(difference~=1);           %Find accelerations from zero speed
ProbAccel = length(idx2)/length(idx); %Probability of accelerating from zero
ProbStay = 1-ProbAccel;               %Probability of staying at zero
AccelZero = accel(idx(idx2-1));       %Accelerations from zero
[f.I_0, x.I_0] = ecdf(AccelZero);
ZeroIdx = find(drivecycle(:,1)==0);
VelRmvZeros = drivecycle(:,1);
VelRmvZeros(ZeroIdx) = [];
AclRmvZeros = accel;
AclRmvZeros(ZeroIdx) = Inf;
clear idx idx2

k = 0:Ns;
speeds = k.*[0, MaxSpeed/Ns*ones(1,Ns)]; %Vector of speeds
accels = cumsum([MinAccel, (MaxAccel-MinAccel)/Na*ones(1,Na)]);
for m = 1:Na
    idx = zeros(length(accel),1);
    idx(find(AclRmvZeros(:,1)>accels(m))) = 1;
    idx(find(AclRmvZeros(:,1)>accels(m+1))) = 0;
    idx2 = find(idx==1);
    acl.(['I_',num2str(m)]) = accel(idx2+1,1);
    [f.(['I_',num2str(m)]),x.(['I_',num2str(m)])] = ...
        ecdf(acl.(['I_',num2str(m)]));
    clear idx idx2
end

counter=1;
disp(['Counter = ' num2str(counter)]);
while counter < 101
    clear GenSpeed t
    m = 1;
    GenSpeed(m) = 0;
    GenAccel(m) = 0;
    while m < 1200 || GenSpeed(m) ~= 0
        if GenSpeed(m) == 0
            decision = rand(1,1);
            if decision > ProbAccel
                GenAccel(m) = 0;
                GenSpeed(m+1) = 0;
            else
                AccelDec = rand(1,1);
                idx = find(AccelDec < f.I_0,1,'first');
                GenSpeed(m+1) = x.I_0(idx);
                GenAccel(m) = x.I_0(idx);
            end
        else
            interval = find(accels<=GenAccel(m-1),1,'last');
            if interval == length(accels)
                interval = interval - 1;
            end
            decision = rand(1,1);
            idx = find(decision < f.(['I_',num2str(interval)]),1,'first');
            GenAccel(m) = x.(['I_',num2str(interval)])(idx);
            if GenAccel(m) < MinAccel
                GenAccel(m) = MinAccel;
            end
            GenSpeed(m+1) = GenAccel(m) + GenSpeed(m);
            if GenSpeed(m+1) > MaxSpeed
                GenSpeed(m+1) = MaxSpeed;
                GenAccel(m) = GenSpeed(m+1) - GenSpeed(m);
            end
            if GenSpeed(m+1) < 0
                GenSpeed(m+1) = 0;
            end
        end
        m = m + 1;
    end
    t=0:m-1;
    plot(t,GenSpeed)
    test1=0;
    test2=0;
    test3=0;
    idx=find(GenSpeed==0);
    DiffIdx=diff(idx);
    MoveIdx=DiffIdx;
    idx2=find(MoveIdx==1);
    MoveIdx(idx2)=[];
    MinTimeMoving=min(MoveIdx);
    if MinTimeMoving >= 6
        test1=1;
    end
```

```
    idx=find(GenSpeed~=0);
    StopIdx=diff(idx);
    idx2=find(StopIdx==1);
    StopIdx(idx2)=[];
    MinTimeStanding=min(StopIdx);
    if MinTimeStanding >= 6
        test2=1;
    end
    idx=find(GenSpeed>22);
    if length(idx) < 100
        test3=1;
    end
    if test1 && test2 && test3
        str=num2str(counter);
        if length(str)==1
            str=['00' str];
        elseif length(str)==2
            str=['0' str];
        end
        eval(['cyc.time' str '=transpose(t);'])
        eval(['cyc.cycle' str '=transpose(GenSpeed);'])
        counter=counter+1;
        disp(['Counter = ' num2str(counter)]);
    end
end

for ii = 1:100
    str=num2str(ii);
    if length(str) == 1
        str = ['00' str];
    elseif length(str) == 2
        str = ['0' str];
    end
    eval(['N = size(cyc.cycle' str ',1);'])
    eval(['temp = [cyc.cycle' str '(:,1); 0];'])
    temp(1) = [];
    eval(['cyc.cycle' str '(:,2) = temp;'])
    temp = ones(N,1);
    eval(['cyc.cycle' str '(:,3) = temp;'])
end

save RandomDriveCycles cyc
```

AEVPS parameters

```
%Drive Load Model Parameters%

%Wheel and Motor
J_wm = 2; %Hydraulic motor plus wheel moment of inertia (kg.m^2)
b_w = 1; %Hydraulic motor plus viscous damping ratio (Nm.s)
r_w = 0.31; %Effective radius of wheel (m)

%Vehicle
M = 1000; % 1/4 Vehicle mass (kg)
w=M*9.8;
rho_air = 1.2; %Air density (kg/m^3)
C_D = 0.4; %Drag Coefficient
A_V = 2; %Cross sectional area (m^2)
k = 0.8; %road friction coefficient

%MTF Coefficients for 8kN Normal Force
B = 0.214;
C = 1.78;
D = 7711;
E = 0.783;
```

AEVPS DP Time Step Calculation

```
function [pu, fuel_cons, torque_beg] = AEVPS_noeng_fuelcons_cu07(a, ...
    ne_sim,Qv,p_u_i,deltat)

load result_smooth_ibsfc_07_08_06
load Engin_Map_Torque_to_TC
eng_pwr_max = 18120;

[pu,ne_sim,a] = meshgrid(p_u_i,ne_sim,a);

% if size(p_u_i,1) > 1
%     p_u_i = p_u_i';
% end

K_f = 1.836e-7;

%%Accumulator Parameters%%
Ppr = 5.17; % gas precharge pressure (MP)
cap = 18900; % Accumulator Capacity (cm^3)
k_acc = 1.4; % specific heat ratio

%%Linearized Parameters%%
Beta_u = 266.13; % MPa
Beta_d = 53.23; % MPa
%K_P = 35.94; % cm^3/rad^2
K_P = 37.4;  % cm^3/rad^2
V_u = 2.67*1043; % cm^3
V_d = 1854; % cm^3
%c_u = 9.259; % cm^3/s/MPa
c_u = 0.7;   % cm^3/s/MPa
c_d = 0.04*231.475; % cm^3/s/MPa
```

```
N = length(Qv);


torque = NaN(size(ne_sim));
Qp = torque;
Qa_const = torque;
eng_eff = torque;

torque_constant = (121/(1185*0.8))*((2+0.048*(220/188.5)*(30/pi)*ne_sim)+ ...
    1.0336e-5*((220/188.5)*(30/pi)*ne_sim).^2*0.014*53.05)+0.2407*ne_sim;

for ii = 1:N-1
    torque = a.*pu*K_P+torque_constant;
    if ii == 1
        torque_beg = torque;
        eng_pwr_beg = ne_sim.*torque;
        throttle = interp2(WE_V,Torque_V,TC_Eng_Map,ne_sim,torque_beg)-1;
        fuel_cons_beg = K_f*ne_sim*(220/188.5)*(60/2/pi).*throttle*(100/80);
        eng_eff_beg=interp2(xi,yi,zi,ne_sim/188.5,eng_pwr_beg/eng_pwr_max);
        fuel_cons_beg(isnan(eng_eff_beg)) = Inf;
        idx = find(~isnan(eng_eff_beg));
    end
    Qp(idx) = K_P*a(idx).*ne_sim(idx);
    Qa_const(idx) = (cap/k_acc)*Ppr^(1/k_acc)*pu(idx).^(-(k_acc+1)/k_acc);
    pu(idx) = pu(idx) + ((bsxfun(@minus,Qp(idx),Qv(ii))-c_u*pu(idx))./ ...
        ((V_u/Beta_u)+Qa_const(idx)))*deltat;
end

eng_pwr_end = ne_sim.*torque;
throttle = interp2(WE_V,Torque_V,TC_Eng_Map,ne_sim,torque)-1;
fuel_cons_end = K_f*ne_sim*(220/188.5)*(60/2/pi).*throttle*(100/80);
eng_eff_end=interp2(xi,yi,zi,ne_sim/188.5,eng_pwr_end/eng_pwr_max);
```

```
fuel_cons_end(isnan(eng_eff_end)) = Inf;
fuel_cons = (fuel_cons_beg+fuel_cons_end)/2;
```

AEVPS DP algorithm

```
load DP_AEVPS_noeng_fuelcons_FUDS_newload

Drive_Load_param_passenger_vehicle

%load simple_cycle  % load in motor speed profile n_m
load FUDS_cycle
n_m = FUDS;
%n_m(:,2) = n_m(:,2)/r_w/0.3536;
n_m(:,2) = n_m(:,2)/r_w;
%load DP_AEVPS_Throttle2

D_m = 0.95*4.216; % cm^3/rad
J_m_tot = 0.0019; % kg.m^2
b_m = 0.9*0.0514; % N.m.s

%%Linearized Parameters%%
Beta_u = 266.13; % MPa
Beta_d = 53.23; % MPa
%K_P = 35.94; % cm^3/rad^2
K_P = 37.4;  % cm^3/rad^2
V_u = 2.67*1043; % cm^3
V_d = 1854; % cm^3
%c_u = 9.259; % cm^3/s/MPa
c_u = 0.7;   %cm^3/s/MPa
c_d = 0.04*231.475; % cm^3/s/MPa

b_w = 1;  % Nm/rad/sec
r_w = 0.31; % m
```

```
rho = 1.2; % kg/m^3
C_drag = 0.4;
A = 2; % m^2

N = size(n_m,1);  %Number of steps in cycle
deltat = n_m(N,1)-n_m(N-1,1);  %Time step (s)

%Load torque (divided by 4 for quarter model)
TL = (b_w*n_m(:,2)+(r_w/2)*rho*C_drag*A*(r_w*n_m(:,2)).^2)/4;
p_d = zeros(N,1);  %Initialize downstream pressure
for ii = N-1:-1:1
    %Solve for downstream pressure from motor speed and load torque
    p_d(ii,1) = ((n_m(ii+1,2)-n_m(ii,2))/deltat)*(J_m_tot/D_m)+ ...
        (b_m/D_m)*n_m(ii,2)+(TL(ii)/D_m);
end
p_d(N,1) = p_d(N-1,1);

Q_v = zeros(N,1);  %Initialize valve command
for ii = 1:N-1
    %Solve for valve command from motor speed and downstream pressure
    Q_v(ii,1) = (V_d/Beta_d)*((p_d(ii+1,1)-p_d(ii,1))/deltat)+ ...
        D_m*n_m(ii,2)+c_d*p_d(ii,1);
end
Q_v(N,1) = Q_v(N-1,1);

speed = [75:5:188.5, 188.5];  %Speed vector of engine (rad/s)
alpha = 0:0.2:18;  %Swashplate angle vector (deg)
alpha = alpha*(pi/180);  %Convert degrees to radians
PressStep = 0.01;  %Upstream pressure increment (MPa)
PressureStates = 6:PressStep:20;  %Upstream pressure vector (MPa)
N = n_m(end,1);
minPuNext = 0;
```

```
%% Dynamic Programming Algorithm
for ii = N-1:-1:0
    str = sprintf('ii = %i',ii);
    disp(str);
    tic
    if ii ~= N-1
        str = num2str(ii+1);
        if length(str) == 3
            str = ['0' str];
        elseif length(str) == 2
            str = ['00' str];
        elseif length(str) == 1
            str = ['000' str];
        end
        eval(['FutureCost = optimal.k' str '.cost;']);
    else
        %Initialize future cost vector
        FutureCost = zeros(length(PressureStates),1);
    end
    %% Initialize matricies
    %% Rows correspond to current state, columns correspond to future state
    cost = Inf(length(PressureStates));  %Cost (fuel consumption)
    EngSpeed = NaN(length(PressureStates));  %Engine speed
    DispAngle = EngSpeed;  %Swashplate angle
    %% Solve for portion of downstream pressure and valve command
    %Portion of downstream pressure vector for this 1 second time step
    pd = p_d(ii*100+1:ii*100+101);
    %Portion of valve command vector for this 1 second time step
    Qv = Q_v(ii*100+1:ii*100+101);
    %% Find largest downstream pressure for current time step
    maxPd = max(pd);
```

```
%% Solve for upstream pressure at next time step and fuel consumption
tic
[pu fuel] = AEVPS_noeng_fuelcons_cu07(alpha,speed,Qv,PressureStates, ...
    deltat);
minPu = ceil(maxPd/PressStep)*PressStep;
toc
%% Calculate fuel consumption for each swashplate angle and engine speed
for kk = 1:length(alpha)
    CurPressure = pu(:,:,kk);
    CurPressure(CurPressure < minPu) = NaN;
    CurPressure(CurPressure < minPuNext) = NaN;
    CurPressure(CurPressure < 6) = NaN;
    CurPressure(CurPressure > 35) = NaN;
    for zz = 1:length(speed)
        idx = find(~isnan(CurPressure(zz,:)));
        test = 1;
        if isempty(idx)
            test = 0;
        end
        if test
            for mm = 1:length(idx)
                CurRow = idx(mm);
                CurCol = round((pu(zz,idx(mm),kk)-6)/0.01)+1;
                if CurCol <= length(PressureStates)
                    if cost(CurRow,CurCol) > fuel(zz,idx(mm),kk)
                        cost(CurRow,CurCol) = fuel(zz,idx(mm),kk);
                        DispAngle(CurRow,CurCol) = alpha(kk);
                        EngSpeed(CurRow,CurCol) = speed(zz);
                    end
                end
            end
        end
    end
end
minPuNext = ceil(pd(1)/PressStep)*PressStep;
test = 1;
while test
    Uv = fzero(@(uv) (12*uv^3-40*uv^2+65*uv-4.5)* ...
        sqrt(minPuNext-pd(1))-Qv(1), 8);
    if Uv > 10
        minPuNext = minPuNext + PressStep;
    else
        test = 0;
    end
end
%% Find minimum cost for each state (upstream pressure)
ObjFunc = bsxfun(@plus,cost,FutureCost');
[MinValue,MinInd] = min(ObjFunc,[],2);
str = num2str(ii);
if length(str) == 3
    str = ['0' str];
elseif length(str) == 2
    str = ['00' str];
elseif length(str) == 1
    str = ['000' str];
end
eval(['optimal.k' str '.cost = MinValue;']);
for jj = 1:length(PressureStates)
    eval(['optimal.k' str '.EngSpeed(jj,1) = ' ...
        'EngSpeed(jj,MinInd(jj));']);
    eval(['optimal.k' str '.DispAngle(jj,1) = ' ...
        'DispAngle(jj,MinInd(jj));']);
    eval(['optimal.k' str '.NextState(jj,1) = ' ...
        'PressureStates(MinInd(jj));']);
    end
    toc
end

save DP_AEVPS_noeng_fuelcons_FUDS_newload optimal PressStep speed alpha
```

AEVPS DP forward

```
Drive_Load_param_passenger_vehicle

load FUDS_cycle  % load in motor speed profile n_m
n_m = FUDS;
n_m(:,2) = n_m(:,2)/r_w;
% Lookup table - Engine speed and torque to throttle command
load Engin_Map_Torque_to_TC
load DP_AEVPS_noeng_fuelcons_idle00043_FUDS_newload

pu_Cur = 10;
pu_Cur_NoRound = pu_Cur;

D_m = 0.95*4.216; % cm^3/rad
J_m_tot = 0.0019; % kg.m^2
b_m = 0.9*0.0514; % N.m.s

%%Linearized Parameters%%
Beta_u = 266.13; % MPa
Beta_d = 53.23; % MPa
%K_P = 35.94; % cm^3/rad^2
K_P = 37.4;  % cm^3/rad^2
V_u = 2.67*1043; % cm^3
V_d = 1854; % cm^3
%c_u = 9.259; % cm^3/s/MPa
c_u = 0.7;   % cm^3/s/MPa
```

```
c_d = 0.04*231.475; % cm^3/s/MPa

b_w = 1;  % Nm/rad/sec
r_w = 0.31; % m
rho = 1.2; % kg/m^3
C_drag = 0.4;
A = 2; % m^2

N = size(n_m,1);
deltat = n_m(N,1)-n_m(N-1,1);

TL = (b_w*n_m(:,2)+(r_w/2)*rho*C_drag*A*(r_w*n_m(:,2)).^2)/4;
p_d = zeros(N,1);
for ii = N-1:-1:1
    p_d(ii,1) = ((n_m(ii+1,2)-n_m(ii,2))/deltat)*(J_m_tot/D_m)+ ...
        (b_m/D_m)*n_m(ii,2)+(TL(ii)/D_m);
end
p_d(N,1) = p_d(N-1,1);

Q_v = zeros(N,1);
for ii = 1:N-1
    Q_v(ii,1) = (V_d/Beta_d)*((p_d(ii+1,1)-p_d(ii,1))/deltat)+ ...
        D_m*n_m(ii,2)+c_d*p_d(ii,1);
end
Q_v(N,1) = Q_v(N-1,1);
Q_v(Q_v < 0) = 0;

if ~exist('PressStep','var')
    PressStep = 0.01;
end
N = n_m(end,1);
```

```
%DPSol = NaN(N,9);
for ii = 0:N-1
    str = num2str(ii);
    if length(str) == 3
        str = ['0' str];
    elseif length(str) == 2
        str = ['00' str];
    elseif length(str) == 1
        str = ['000' str];
    end
    row = round((pu_Cur-6)/PressStep)+1;
    if row == 0
        row = 1;
    end
    eval(['a = optimal.k' str '.DispAngle(row,1);']);
    eval(['EngSpeed = optimal.k' str '.EngSpeed(row,1);']);
    eval(['Idle = optimal.k' str '.Idle(row,1);']);
    Uv = fzero(@(uv) (12*uv^3-40*uv^2+65*uv-4.5)* ...
        sqrt(pu_Cur-p_d(ii*100+1))-Q_v(ii*100+1), 3.5);
    Qv = Q_v(ii*100+1:(ii+1)*100+1,1);
    pd = p_d(ii*100+1:(ii+1)*100+1,1);
    [pu_Next fuel torque] = AEVPS_noeng_fuelcons_cu07(a,EngSpeed,Qv, ...
        pu_Cur,deltat);
    DPSol(ii+1,:) = [ii EngSpeed torque a Uv pd(1) pu_Cur fuel Idle];
    pu_Cur_NoRound = pu_Next;
    pu_Cur = round(pu_Next/PressStep)*PressStep;
end
DPSol(:,10) = DPSol(:,2).*DPSol(:,3);
DPSol(:,11) = interp2(WE_V,Torque_V,TC_Eng_Map,DPSol(:,2),DPSol(:,3))-1;
DPSol(:,12) = n_m(1:100:end-100,2);

idx = find(DPSol(:,2)==75 & DPSol(:,4)==0);
```

```
DPSol(idx,8)=0.00043;

figure
subplot(2,1,1)
plot(n_m(:,1),n_m(:,2),'k-','linewidth',2)
str = sprintf('Motor Speed\nProfile (rad/s)');
ylabel(str)
title('State Trajectories from DP Results')
subplot(2,1,2)
plot(DPSol(:,1),DPSol(:,7),'b-','linewidth',2)
str = sprintf('Pressure (MPa)');
ylabel(str)
hold on
plot(DPSol(:,1),DPSol(:,6),'r--','linewidth',2)
legend('Upstream Pressure','Downstream Pressure');
xlabel('Time (s)')

figure
subplot(3,1,1)
plot(DPSol(:,1),DPSol(:,4),'k-','linewidth',2)
str = sprintf('Swashplate\nAngle (rad)');
ylabel(str)
title('Control Trajectories from DP Results')
subplot(3,1,2)
plot(DPSol(:,1),DPSol(:,5),'k-','linewidth',2)
str = sprintf('Valve\nCommand (Volts)');
ylabel(str)
subplot(3,1,3)
plot(DPSol(:,1),DPSol(:,11),'k-','linewidth',2)
str = sprintf('Engine\nThrottle');
ylabel(str)
xlabel('Time (s)')
```

```
open Engine_Map.fig
hold on
plot((DPSol(:,2)/188.5)*100,((DPSol(:,3).*DPSol(:,2))/18120)*100,'k+')

p1 = 25.45;
p2 = 2124;
Xcor = 0:140;
Ycor = p1*Xcor + p2;
figure
plot(DPSol(:,10),DPSol(:,8),'k+');
hold on
plot(Xcor,Ycor,'r-')
xlabel('Motor Speed (rad/s)')
ylabel('Engine Power (W)')
legend('Data','Curve Fit')

p1 = 0.001491;
p2 = 0.002804;
Ycor = p1*Xcor + p2;
figure
plot(DPSol(:,10),DPSol(:,4),'k+');
hold on
plot(Xcor,Ycor,'r-')
xlabel('Motor Speed (rad/s)')
ylabel('Swashplate Angle (rad)')
legend('Data','Curve Fit')

p1 = 0.1257;
p2 = 7.032;
Ycor = p1*Xcor + p2;
figure
```

```
plot(DPSol(:,10),DPSol(:,9),'k+');
hold on
plot(Xcor,Ycor,'r-')
xlabel('Motor Speed (rad/s)')
ylabel('Engine Throttle')
legend('Data','Curve Fit')
```

AEVPS build transition matrix

```
clear
clc

Drive_Load_param_passenger_vehicle
AccelerationMarkov

D_m = 0.95*4.216; % cm^3/rad
J_m_tot = 0.0019; % kg.m^2
b_m = 0.9*0.0514; % N.m.s

%%Linearized Parameters%%
Beta_u = 266.13; % MPa
Beta_d = 53.23; % MPa
%K_P = 35.94; % cm^3/rad^2
K_P = 37.4;  % cm^3/rad^2
V_u = 2.67*1043; % cm^3
V_d = 1854; % cm^3
%c_u = 9.259; % cm^3/s/MPa
c_u = 0.7;   %cm^3/s/MPa
c_d = 0.04*231.475; % cm^3/s/MPa

b_w = 1;  % Nm/rad/sec
r_w = 0.31; % m
rho = 1.2; % kg/m^3
```

```
C_drag = 0.4;
A = 2; % m^2

oilStep = 1;
Vo = 6:oilStep:19;
rowidx = 1;
states = zeros(length(Vo)*Na,2);
% for k = 1:Nw
    for m = 1:Na
        for n = 1:length(Vo)
            states(rowidx,1) = Accel(m);
            states(rowidx,2) = Vo(n);
            rowidx=rowidx+1;
        end
    end
% end

EngineSpeed = 75:5:185;
Displacement = 0:0.01:0.314;
rowidx = 1;
controls = zeros(length(EngineSpeed)*length(Displacement),2);
for m = 1:length(EngineSpeed)
    for n = 1:length(Displacement)
        controls(rowidx,1) = EngineSpeed(m);
        controls(rowidx,2) = Displacement(n);
        rowidx=rowidx+1;
    end
end

deltat = 0.01;
for u = 1:size(controls,1)
    stru = num2str(u);
```

```
if length(stru) == 1
    stru = ['0',stru];
end
tic
str = sprintf('Control %i of %i',u,size(controls,1));
disp(str)
for velInd = 1:Nw
    strv = num2str(velInd);
    if length(strv) == 1
        strv = ['0',strv];
    end
    str = sprintf('Velocity Index %i of %i',velInd,Nw);
    disp(str);
    TestSpeed = Omega(velInd);
    temp = zeros(length(Vo)*Na);
    temp2 = temp;
    sumNextAccel = sum(prob(:,:,velInd),1);
    feasNextAccel = find(sumNextAccel ~= 0);
    for accelInd = feasNextAccel
        TestAccel = Accel(accelInd);
        NextSpeed = TestSpeed+TestAccel;
        NextSpeed(NextSpeed<0) = 0;
        SpeedVector = linspace(TestSpeed,NextSpeed,(1/deltat)+1);
        n_m(:,1) = 0:deltat:1;
        n_m(:,2) = SpeedVector/r_w;
        %Load torque (divided by 4 for quarter model)
        TL = (b_w*n_m(:,2)+(r_w/2)*rho*C_drag*A*(r_w*n_m(:,2)).^2)/4;
        N=length(SpeedVector);
        p_d = zeros(N,1);  %Initialize downstream pressure
        Q_v = zeros(N-1,1);  %Initialize valve command
        for ii = 1:N-1
            %Solve for downstream pressure
    p_d(ii,1) = ((n_m(ii+1,2)-n_m(ii,2))/deltat)* ...
        (J_m_tot/D_m)+(b_m/D_m)*n_m(ii,2)+(TL(ii)/D_m);
end
p_d(N,1) = p_d(N-1,1);
for ii = 1:N-1
    %Solve for valve command
    Q_v(ii,1) = (V_d/Beta_d)*((p_d(ii+1,1)-p_d(ii,1))/deltat)+ ...
        D_m*n_m(ii,2)+c_d*p_d(ii,1);
end
ne_sim = controls(u,1);
a = controls(u,2);
p_u_i = Vo;
[P_u,fuel,torque] = AEVPS_noeng_fuelcons_cu07(a,ne_sim,Q_v, ...
    p_u_i,deltat);
fuel(P_u<Vo(1)) = Inf;
fuel(P_u>Vo(end)) = Inf;
maxpd = max(p_d);
fuel(P_u < maxpd+0.5) = Inf;
idx = [1:length(Vo)]';
fuel(fuel==Inf) = 10;
P_u = round(P_u/oilStep)*oilStep;
feasAccel = find(prob(:,accelInd,velInd) > 0);
newProb = prob(feasAccel,accelInd,velInd);
accelRows = ((feasAccel-1)*length(Vo)*ones(1,length(idx)))';
oilRows = idx*ones(1,length(feasAccel));
rows = accelRows + oilRows;
accelColms = ((accelInd-1)*length(Vo)* ...
    ones(1,length(idx)))'*ones(1,length(feasAccel));
oilColms = (round((P_u(idx)-Vo(1))/oilStep)+1)'* ...
    ones(1,length(feasAccel));
columns = accelColms + oilColms;
probabilites = (newProb*ones(1,length(idx)))';
```

```
        cost = fuel(idx)'*ones(1,length(feasAccel));
        idx = find(rows<1);
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
        idx = find(columns<1);
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
        idx = find(rows>Na*length(Vo));
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];
        idx = find(columns>Na*length(Vo));
        rows(idx) = [];
        columns(idx) = [];
        probabilites(idx) = [];
        cost(idx) = [];

        for ind = 1:numel(probabilites)
            temp(rows(ind),columns(ind)) = probabilites(ind);
            temp2(rows(ind),columns(ind)) = cost(ind);
        end
        clear probabilites cost
    end

    eval(['TPM.w' strv '.u' stru '= sparse(temp);']);
    eval(['TRM.w' strv '.u' stru '= sparse(temp2);']);
    clear temp temp2
    end
    toc
end

save SDP_AEVPS_TPM_TRM TPM TRM
```