Modeling, Control Analysis, and Multi-Physics Co-Simulation Supporting High-Performance Hybrid-Electric Vehicles

A Thesis
Presented to
The Academic Faculty

By

Saeid Loghavi

In Partial Fulfilment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology

August 2017

Modeling, Control Analysis, and Multi-Physics Co-Simulation Supporting High-Performance Hybrid-Electric Vehicles

Approved by:
Dr. Michael Leamy, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Aldo Ferri
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Kenneth Cunefare
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 7/27/2017

DEDICATION

Dedicated to my father and mother.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor, Dr. Michael Leamy, for his excellent guidance, caring and patience. For the past 2 years, he has devoted countless hours to guide me through work submitted in this thesis and elsewhere. His technical and editorial advice was not only essential to the completion of this thesis but have also taught me innumerable lessons and insights for my work.

My special thanks goes to my committee members, Dr. Aldo Ferri and Dr. Kenneth Cunefare. Their comments and suggestions have been invaluable and as a result added significant value to my thesis. I would also like to thank Dr. Chris Paredis whose class in Modeling and Simulation introduced me to the Modelica object-oriented language and Dymola softwar,e which contributed greatly to my thesis.

I would also like to thank Ferrari S.p.A. for sponsoring my work. Special thanks goes to Mr. Andrea Canaparo, who provided me with many challenging projects as well as support, which has contributed greatly to my development as an engineer. Additionally, I would like to thank my lab mates Justin, Amir, Matt, Oscar and Douglas. They have all been instrumental to my success at Georgia Tech. Special thanks goes to Justin who provided me with great advice and support.

Lastly, I would like to thank those who are closest to me in life. I would not have been able to follow my dream of living and studying in the United States without the support of my parents. My sisters, Laleh and Mina inspired me to work hard and stay disciplined. I could not have asked for better role-models. Finally, my girlfriend Swarnika has been a constant source of support and encouragement during the challenges of graduate school.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CAA	Clean Air Act
CCCV	Constant Current, Constant Voltage
CFD	Computational Fluid Dynamics
CG	Center of Gravity
CSE	Co-simulation Engine
CVL	Charging Voltage Limit
DP	Dynamic Programming
DUT	Device under Testing
DVL	Discharge Voltage Limit
ECMS	Equivalent Consumption Minimization Strategy
EES	Electric Energy Storage
EM	Electric Machine
EV	Electric Vehicle
FBD	Free Body Diagram
FE	Fuel Economy
FEM	Finite Element Method
FMI	Functional Mock-up Interface

FMU	Functional Mock-up Unit
HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
MPG	Miles per gallon
MPGGE	Miles per gallon of gasoline equivalent
OCV	Open Circuit Voltage
SOC	State of Charge
SOH	State of Health
UDDS	Urban Dynamometer Driving Schedule
USEPA	United States Environmental Protection Agency
XML	eXtensible Mark-up Language

LIST OF SYMBOLS

G	Gravitational acceleration
A	Incline angle
M	Vehicle mass
H	Height of vehicle CG above the ground
a, b	Distance of front and rear axles, respectively, from the normal projection point of vehicle CG onto the common axle plane
V_x	Longitudinal vehicle velocity
V_w	Headwind speed
N	Number of wheels on each axle
F_{xf} , F_{xr}	Longitudinal forces on each wheel at the front and rear ground contact points, respectively
F_{zf} , F_{zr}	Normal load forces on each wheel at the front and rear ground contact points, respectively
A	Effective frontal cross-sectional area
C_d	Aerodynamic drag coefficient
P	Mass density of air
F_d	Aerodynamic drag force
T	Traction Torque
N	Effective gear ratio
R	Tire radius
v_f	Final velocity

T_{EM}	EM Torque
ω_{EM}	EM Speed
P_{EM}	EM power
$I_{Battery}$	Battery Current
$V_{Battery}$	Battery Voltage
$Ah_{consumed}$	Amp-hour energy consumed
$E_{Battery}$	Battery energy use
$E_{Electrical}$	Electrical energy stored in battery
E_{Fuel}	Fuel energy used
E_{Total}	Total energy used (sum of fuel and battery energy)
R_{Series}	Series impedance
$R_{Internal}$	Battery internal resistance
x_k	State variable
u_k	Control variable
F_k	Function defining state variables
$J_{0,\pi}$	Total cost
g_0	Cost associated with the initial step
g_N	Cost associated with the final step
ϕ_k	Cost associated with application of constraints
h_k	Incremental cost of application of controls
$\dot{m}_{fuel}$	Mass fuel rate
SOC_{ini}	Initial state of charge
$OCVtablecharge$	Look-up table for the OCV vs. SOC during charge

$OCVtabledischarge$	Look-up table for the OCV vs. SOC during discharge
Q_{ini}	Initial transferred charge
Q_{total}	Total transferable charge
C_0	Capacity at $T_{ref}$for $Q_{total} = 0$ and $t = 0$
$useHeatPort$	Boolean variable for using the heat port
T_{ref}	Reference temperature
$alphaRs$	Linear temperature coefficient for R_s
R_{sref}	Ohmic resistance at reference temperature
$K_{CQ_{abs}}$	Linear Q_{abs} dependency of capacity
$alphaC$	Linear temperature coefficient for capacity
Q_{Heat}	Cell heat generated
$Q_{cooling}$	Heat removed from the cell
C_{Heat}	Cell heat capacity

SUMMARY

This thesis presents a series of model-based studies and associated considerations supporting the development of a high-performance HEV. Due to increasingly strict governmental regulations and consumer demand, automakers have taken steps to reduce fuel consumption and greenhouse emissions. HEV's can provide a balance between fuel economy and vehicle performance by exploiting engine load-point shifting, regenerative braking, pure electric operation, and hybrid traction modes. The existence of a multitude of HEV architectures with different emissions and performance characteristics necessitates the development of simulation platforms which can assist in specifying and selecting critical components.

Recent advancements in the automotive industry, especially the introduction of hybrid technology, have resulted in lower emissions and improved fuel economy; however, hybrid technology can also be utilized in order to enhance the performance characteristics of traditional internal combustion high-performance vehicles. The complexity of the hybrid systems and high power demand of high-performance vehicles requires a detail analysis of critical system components, such as the energy storage systems, to ensure safe and optimal operation. The collaboration between Georgia Tech researchers and Ferrari S.p.A. is illustrative of the need for the further development of innovative and model-based tools to enhance the design and performance of high-performance hybrid electric vehicles.

This thesis also features a series hybrid electric vehicle model developed using Simulink modeling software as part of a tutorial which may be provided to students in order to teach the basic principles underlying the operation, control, and design of hybrid electric vehicles.

The final chapter of this thesis features a modeling approach developed in order to analyze the battery pack in high-performance hybrid-electric vehicles using a multi-physics co-simulation approach. This modeling capability can be extended to other multi-physics systems in order to develop high fidelity models while significantly decreasing computational costs.

Chapter 1: Introduction

Vehicle electrification is being driven both by growing consumer awareness as well as increasing efficiency standards. However both technical and market challenges remain. The development of hybrid electric vehicles requires access to a high skilled workforce as well as the necessary tools in order to reduce the development costs and expedite the evaluation of each design alternative. Virtual simulations can alleviate the product development cost and enable faster introduction of new vehicles. However, many new graduates lack the technical expertise to utilize the new simulation tools available to product development engineers. The second chapter of this thesis describes the development of a tutorial made available to engineering students in order to introduce the sequential process of vehicle development. The third chapter provides a brief description of the application of Bellman’s principle of optimality to evaluate the fuel economy of hybrid electric vehicles. Chapter four provides a multi-physics co-simulation to support development of energy storage systems.

1.1 Hybrid Electric Vehicle Technology

By the 1960’s, public awareness that improvements in quality of life are offset by increased emissions and detrimental effects on the atmosphere led to the passage of the Clean Air Act (CAA) and the establishment of the Environmental Protection Agency (USEPA). Since then, both government and industry have taken major steps to reduce emissions and improve the efficiency of our energy systems [1].

Since the early days of large scale vehicle manufacturing, which coincided with the development of new technology that provided easy access to gasoline, the majority of vehicles manufactured used an internal combustion power plant. The high energy density of gasoline makes it a good source to power vehicles, providing an ideal balance between range and power. However, the byproducts of combustion have been proven to have numerous negative consequences.

Advancements in manufacturing and fuel technology has dramatically increased the energy efficiency of internal combustion engines (ICE); however, even the most efficient engines manufactured today are limited by the Carnot efficiency of the engine. In addition, energy wasted due to friction of moving parts, cooling of engine block and incomplete combustion in cold temperatures reduce the overall efficiency of the vehicle.

One of the major advancements by the transportation industry has been the development of hybrid technology. With the addition of a secondary power plant to the vehicle, the ICE can be used more efficiently. During periods of high power demand, the engine management system can send power toward the secondary power source, which can significantly increase the efficiency of the system. During periods of low power demand, the ICE can operate at higher and more efficient operating points and the extra power is directed to energy storage devices such batteries or capacitors [1].

Hybrid electric vehicles on the market today feature superior fuel efficiency compared to their traditional competitors. While the hybrid technology can be used to dramatically increase the fuel economy and decrease the emissions, Ferrari S.p.A. has shown that the same technology can be used to dramatically increase vehicle performance. Unlike ICEs, electric machines (EM) provide maximum torque at low speeds, which can significantly increase vehicle performance [2-4].

With the addition of a secondary power plant, engineers need to carefully consider the layout of the vehicle powertrain. Each individual layout of the powertrain is referred to as the vehicle architecture. Traditionally HEVs have been classified into two categories: series and parallel. Figure 1.1 illustrates the difference between different hybrid architectures.

The series HEV is the simplest architecture, featuring an ICE powering a generator which charges a power storage device. The current from the generator, as well as the battery if needed, is then used to power the traction motor that drives the wheels. In this configuration, the engine does not provide torque directly to the wheels, rather it only produces electricity [1].

The advantage of the series architecture is in its simplicity as the engine can be designed to run at its most efficient operating point as the secondary on-board generator, and otherwise the vehicle is just an electric vehicle which simplifies the vehicle control strategy. Also a smaller engine can be used, as the role of the engine is to provide a steady source of power rather than provide a wide range of torques and speeds. The downside to this architecture, however, is in the number of energy conversions, each adding an associated efficiency factor. It should also be noted that as the traction motor is the only source of power to the wheels, it must be large enough to power the vehicle in a range of driving scenarios [1].

A parallel HEV architecture takes a different approach to traction by providing parallel energy paths. The ICE and any number of motors are connected directly to the wheels, and this availability of multiple sources of power enables the selection of any combination or all for different driving situations [1,5]. For example, the engine can be used for steady state highway speeds where its operating characteristics will provide effective combustion, while the motor can be used during stop-and-go urban driving where its regenerative braking capabilities will allow for much of the energy spent during acceleration to be recouped while braking [5]. At times when the driver demands high acceleration, all sources can be used to propel the vehicle, and if the battery state of charge gets too low, the motor will only provide regenerative braking during stopping events or even act against the engine, thus acting as a generator and forcing the engine into a more efficient operating point.

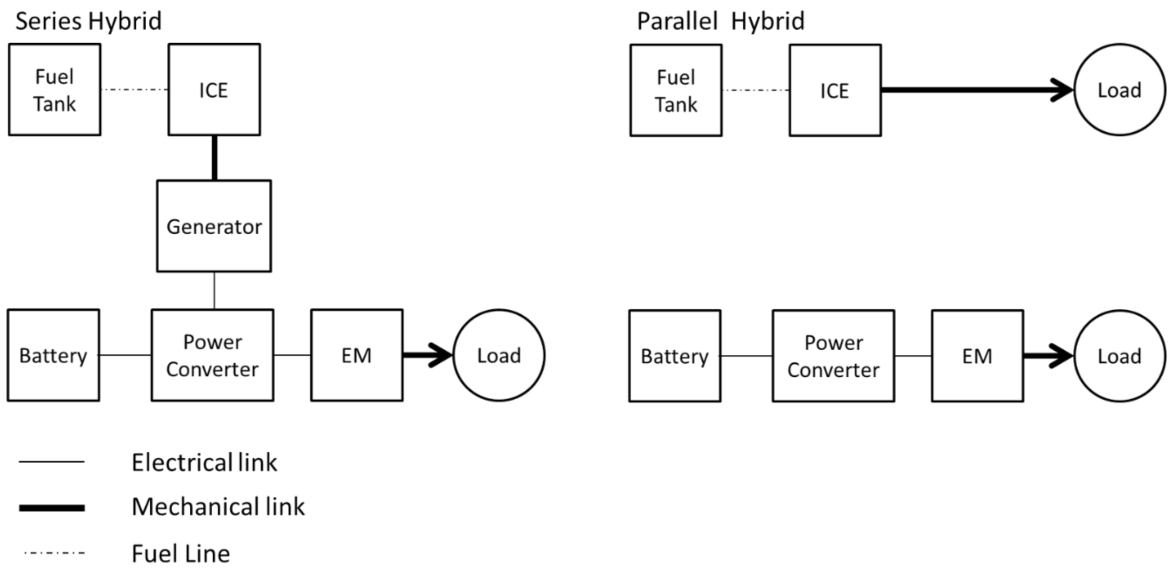


Figure 1.1: Classification of hybrid electric vehicles

Beyond these control options, another benefit of the parallel architecture is in component sizing [1,5]. As all power sources are available to provide power to the wheels, they can all be used for high torque demands, and therefore components can be downsized. The use of gears, shafts, and belts allow for this architecture to be laid out in numerous different configurations.

Due to the added complexity of HEVs and the growing demand for skilled talent, the automotive industry has invested heavily in collaborative research with universities. Examples of such work include the collaborative research between Georgia Tech and Ferrari S.p.A. One of the main challenges for students recruited into the field of automotive research is the knowledge gap between theoretical models for individual components and the overall system operation of vehicles. Providing students with an overall systems preview of vehicle powertrain has proven to significantly increase the performance of student teams. Chapter 2 of this thesis is dedicated to work done on developing a tutorial designed to familiarize students with system level operation of a vehicle. The tutorial features a sequential refinement approach to model development in order to introduce students to complex model components.

1.2 Evaluation of HEV Performance using Global Optimization Algorithms

The global optimization of a HEV architecture can be realized for a known time horizon and operating conditions, such as in the case of optimizing for a drive cycle. DP is used to find the optimal operation regime for a specific drive cycle, and it serves as a benchmark for other non-optimal control strategies [4,6-8]. Using DP multi-objective optimization is possible, as an example, for optimizing fuel economy (FE) and CO_2 emissions simultaneously [7]. However, as DP requires the future knowledge of the drive cycle conditions and large computational resources, vehicle online implementation has not materialized [4].

DP is applied using a backward-looking approach where the simulation requires knowledge of the complete drive cycle and simulation starts at the final simulation time and progresses using a backward step size [2,4]. At each time step the incremental cost of applying control is evaluated and penalty terms are applied to increase the cost of applying control. The optimal control strategy of the power cycle is determined by minimizing the overall cost [4,9]. Chapter 3 of this thesis provides details about DP and approach used to implement in HEV.

Providing benchmarking on performance and emissions results for vehicle architectures in the product development phase provides engineers with an unbiased metric to evaluate and recognize the differences between different HEV architectures. Ferrari S.p.A. used the benchmarking results to identify final vehicle architectures.

1.3 Model-based Design and Ferrari Partnership

In recent years, the development of automotive technology has changed from an exclusively electrical or mechanical engineering discipline to include software engineering. This shift has dramatically changed product development as well as methods, tools and engineering skills required. Due to the complexity of design in the development of hybrid electric vehicles, the need to use model-based design practices to evaluate the performance of each design alternative has increased dramatically.

Ferrari S.p.A. and Georgia Tech have formed a research partnership with a focus on the development of model-based analysis of HEVs. In the first stage of the partnership, focus was given to benchmarking the performance and fuel economy of each powertrain architecture. Fuel economy and CO_2 emissions were evaluated for each design alternative, using DP. Chapter 3 of this thesis features an overview of DP and work done in the first phase of the partnership.

After the completion of the first stage, focus has shifted toward providing model-based analysis of battery performance. One of the unique challenges when designing high-performance HEVs is the design of the battery pack and providing an accurate model of battery performance and aging under high load conditions. Due to high power demand, providing sufficient cooling is crucial to ensure safe operation and preventing damage to battery cells. The ultimate goal of this phase of the research partnership is to develop the capability to analyze the performance of different battery pack layouts and incorporate this capability in a complete vehicle model. Chapter 4 of this thesis is dedicated to work done on developing the battery model.

Chapter 2: Vehicle Model

The vehicle model presented in in this chapter was developed as part of a tutorial which may be provided to students in order to:

1) teach the basic principles underlying the operation, control, and design of hybrid electric vehicles,
2) educate students to model, formulate, and simulate the operation of hybrid vehicles, and
3) provide an overview of the modeling capabilities of Simulink software.

The vehicle model developed next uses a sequential refinement methodology, to create a complex system model using simple components and basic control strategies. The tutorial starts by developing a simple, but complete, vehicle model and then makes further component refinements in order to provide a more realistic model.

The vehicle model is developed using Matlab and Simulink 2016a software. Simulink is a graphical programming environment within which models are created by creating links between pre-defined function blocks and user-defined subcomponents. The predefined function blocks can be categorized under two categories: utility and functional. Utility blocks can be used in a wide array of applications such as applying simple and complex mathematical operations or perform logical operations. Functional blocks however, are designed for specific applications such as calculating the vehicle longitudinal dynamics, evaluating speed, torque and energy loss of a compound planetary gear, or estimating the slip and braking force in a tire. Examples of utility and functional blocks are presented in Figures 2.1 and 2.2.

The vehicle powertrain design is based on a series hybrid architecture. A diesel powered ICE is mechanically connected to a generator (EM-A) with a 1-to-1 gear ratio that converts engine power to electricity which can be stored in the battery or sent directly to EM-B. EM-B then sends power to the wheels. The architecture is designed to operate within five different power modes. Figure 2.3 provides a visual representation of the power modes and Table 2.1 describes the operating conditions as implement by the controller unit.

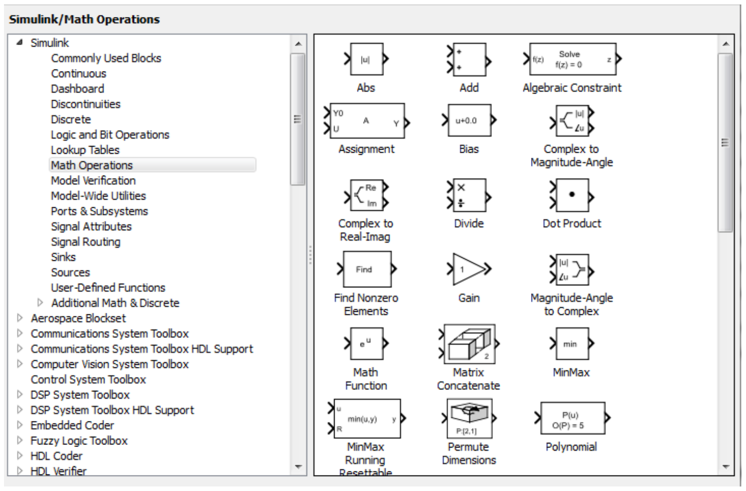


Figure 2.1: Simulink utility blocks

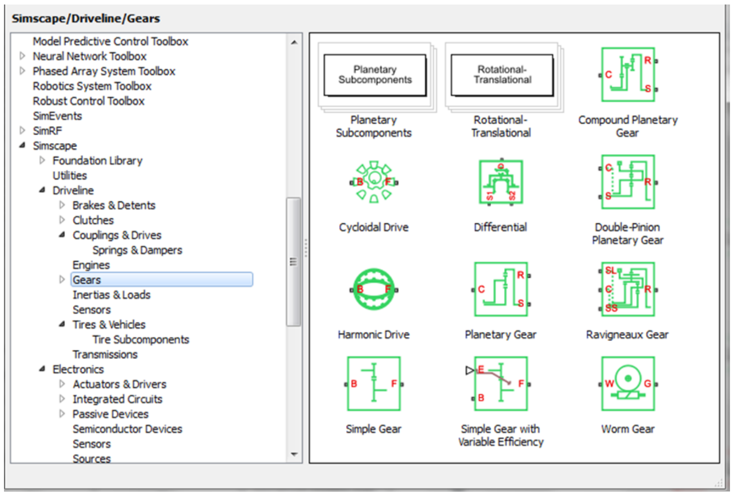


Figure 2.2: Simulink functional blocks

Table 2.1: Driving modes

Power Mode	Battery state of charge	Traffic conditions
Pure EV	High	Low power demand, Stop-and-go traffic
Pure ICE	Low	Constant high speed travel
Combined	Mid rang or high	High power demand, High performance setting
ICE – Charging	Low	Constant speed travel
Charging	As needed	Braking, cost down, hill descent

Based on the battery state of charge (SOC) and power demand, the vehicle controller selects the appropriate power mode. If the battery SOC is above a pre-determined value and the power demand can be supplied by the EM-B alone, the vehicle may operate in pure EV mode. Reducing the emissions to zero while allowing the EM-B to use energy stored in the battery. Under conditions that battery SOC is low, the controller will select the pure ICE driving mode to prevent damage to the battery. If the power demand is low, the controller may go into ICE-Charging mode and request additional power output from the ICE and store the extra energy in the battery. Under conditions when the SOC is within a pre-determined range, the controller may select the combined power model and reduce the fuel consumption by optimizing the power split between the ICE and EM-B. During periods of deceleration the controller may select the charging mode and allow EM-B to regenerate power and store the kinetic energy of the vehicle in the battery pack.

At the highest level, the model consists of three subsystems: Driver, Plant, and Controller. The Driver subsystem receives the speedometer and tachometer data and transmits torque request signals to the controller. The Plant subsystem includes system component models including ICE, EM-A and EM-B. The controller subsystem uses a state-flow controller which receives torque request data from the Driver subsystem and based on pre-determined state conditions sends appropriate control signals to the system components such as the ICE, EMs, and transmission.

The vehicle model can be used to simulate a pre-determined drive cycle and calculate the vehicle performance parameters such as fuel consumption, speed, acceleration, and battery state of charge. The Simulink control model can also be uploaded to a vehicle control unit and used for testing purposes in Hardware-in-the-loop applications.

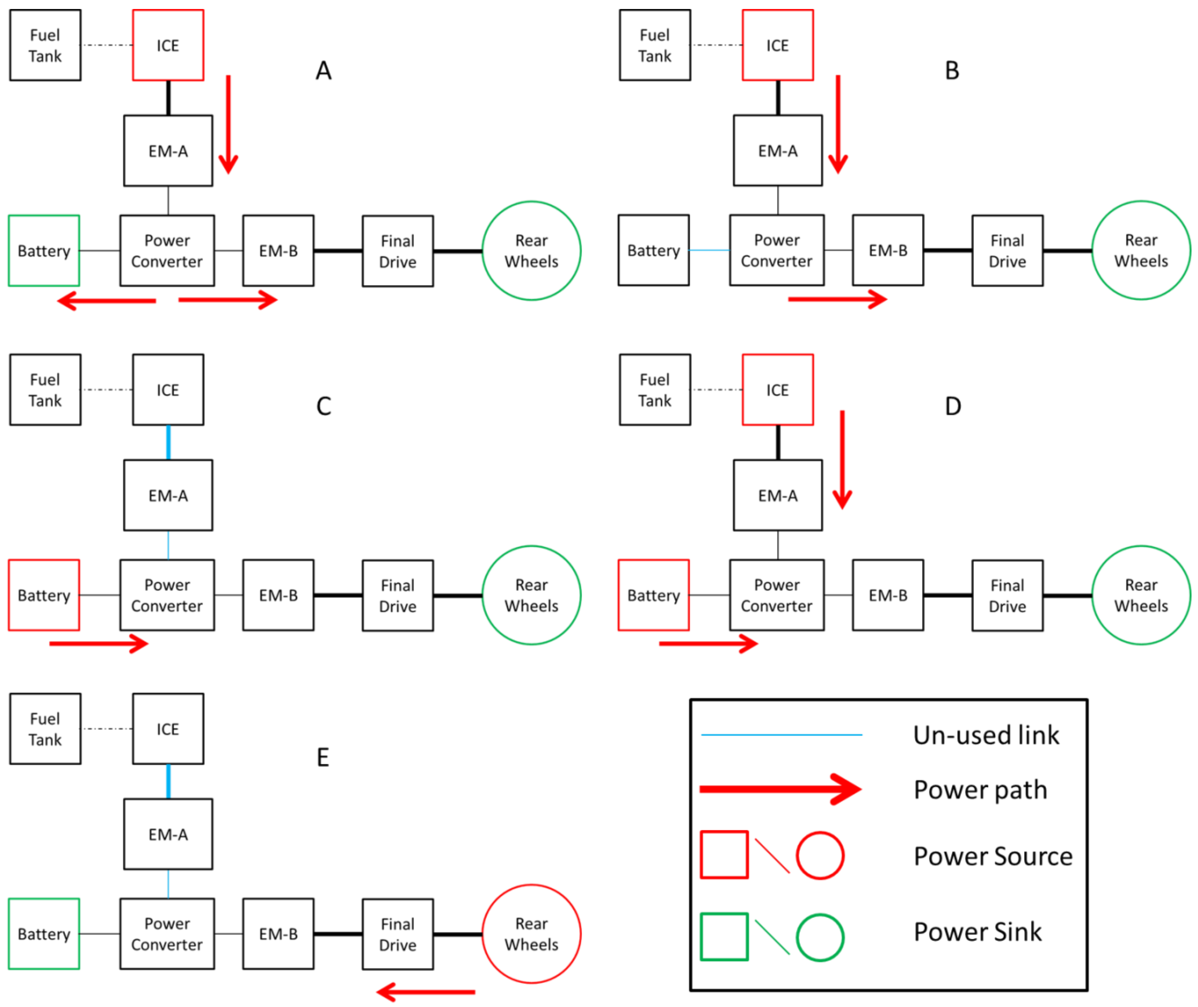


Figure 2.3: Hybrid vehicle power paths. A:ICE-Charging, B:Pure ICE, C:Pure EV, D:Combined, E:Charging

2.1 Simple Vehicle Model (EV-mode only)

The first step in the development of vehicle model is a simple vehicle subsystem with ideal components. The vehicle is propelled only by an ideal EM and a single gear pair is used to transmit power to the wheels. The main components of the simple vehicle subsystem are developed using the following functional Simulink blocks: Vehicle body, Tire (Magic formula), and Simple Gear blocks.

A vehicle body Simulink bock is used to model a vehicle with two axles in longitudinal motion. The vehicle wheels are assumed to be identical in size. The vehicle is assumed to be in normal equilibrium and does pitch or have any vertical movement. The vehicle axles are parallel and form a plane. The longitudinal x direction lies in this plane and is perpendicular to the axels. The model also includes traveling on an incline slope α, the normal z direction is not parallel to gravity but is always perpendicular to the axel longitudinal plane.

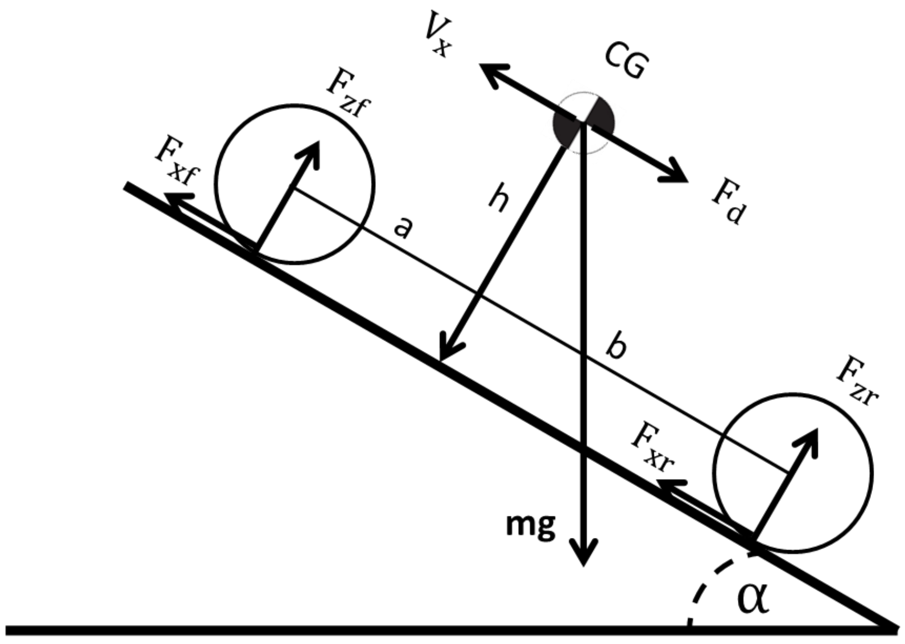


Figure 2.4 Vehicle free body diagram (FBD)

The vehicle motion is determined by the net effect of all forces and torques acting on it. The longitudinal tire forces push the vehicle forward or backward. The weight, mg, of the vehicle acts through its center of gravity (CG). Depending on the incline angle, the weight pulls the vehicle to the ground and pulls it either backward or forward. Whether the vehicle travels forward or backward, aerodynamic drag force acts in the opposite direction of travel. It is assumed that the drag force acts through the CG. Figure 2.4 and Table 2.2 define the vehicle motion model variables.

Table 2.2: Vehicle model variables

Symbol	Description and Unit
G	Gravitational acceleration = 9.81 $[{}^{m}/_{s^2}]$
A	Incline angle, [deg]
M	Vehicle mass, [kg]
H	Height of vehicle CG above the ground, [m]
a, b	Distance of front and rear axles, respectively, from the normal projection point of vehicle CG onto the common axle plane, [m]
V_x	Longitudinal vehicle velocity, $[{}^{m}/_{s}]$
V_w	Headwind speed, $[{}^{m}/_{s}]$
n	Number of wheels on each axle
F_{xf} , F_{xr}	Longitudinal forces on each wheel at the front and rear ground contact points, respectively, [N]
F_{zf} , F_{zr}	Normal load forces on each wheel at the front and rear ground contact points, respectively [N]
A	Effective frontal cross-sectional area $[m^2]$
C_d	Aerodynamic drag coefficient
ρ	Mass density of air = 1.18, $[{}^{kg}/_{m^3}]$
F_d	Aerodynamic drag force, [N]

The vehicle dynamics are described as follows:

$$\mathrm{m}\dot{\mathrm{V}}_{\mathrm{x}} = \mathrm{F}_{\mathrm{x}} - \mathrm{F}_{\mathrm{d}} - \mathrm{mgsin}\alpha \tag{1}$$

$$\mathrm{F}_{\mathrm{x}} = \mathrm{n}(\mathrm{F}_{\mathrm{xr}} + \mathrm{F}_{\mathrm{xf}}) \tag{2}$$

$$\mathrm{F}_{\mathrm{d}} = \frac{1}{2}\mathrm{C}_{\mathrm{d}}\rho\mathrm{A}(\mathrm{V}_{\mathrm{x}} - \mathrm{V}_{\mathrm{w}})^2 . \mathrm{sgn}(\mathrm{V}_{\mathrm{x}} - \mathrm{V}_{\mathrm{w}}) \tag{3}$$

In order to determine the normal force on each front and rear wheel, the normal force acting on the front and rear wheels are expressed by:

$$\mathrm{F}_{\mathrm{zf}} = \frac{-h\left(\mathrm{F}_{\mathrm{d}} + mgsin\alpha + \mathrm{m}\dot{\mathrm{V}}_{\mathrm{x}}\right) + b.mgcos\alpha}{n(a+b)} \tag{4}$$

$$\mathrm{F}_{\mathrm{zr}} = \frac{h\left(\mathrm{F}_{\mathrm{d}} + mgsin\alpha + \mathrm{m}\dot{\mathrm{V}}_{\mathrm{x}}\right) + a.mgcos\alpha}{n(a+b)} \tag{5}$$

while the wheel normal forces satisfy $\mathrm{F}_{\mathrm{zf}} + \mathrm{F}_{\mathrm{zr}} = \mathrm{mgcos}\alpha/\mathrm{n}$.

The Tire (Magic Formula) block is used to model a tire with longitudinal behavior given by the Magic formula[1]. The block can calculate the tire slip, however do to the nature of drive cycles used for analysis, it is assumed that tire slip is negligible. The effects of tire intertia, stiffness, and damping are ignored. The vehicle longitudinal velocity is calculate by $\mathrm{V}_{\mathrm{x}} = \mathrm{r}_{\mathrm{w}}\omega$. Where r_{w}, is the tire effective radius in meters and ω is the wheel angular velocity in ${rad}/{s}$.

A Simple Gear block represents a gearbox that constrains two connected driveline axles to co-rotate with a fixed ratio. The output shaft rotates in the same direction as the input shaft. It is assumed that energy loss in the gearbox due to meshing and viscous losses is negligible for the purposes of this analysis. In practice, based on the type of gears used, the quality of the gear meshing, and lubrication, the efficiency of the gear may be as low as 0.9. Gear inertia is

[1] Series of tire design models developed by Hans B. Pacejka. The term 'magic formula' refers to lack of physical basis for the models. Magic formula models are widely used in the industry in a variety of tire constructions and operating conditions.

assumed to be negligible compared to the engine and shaft inertia; therefore it is not included in the model. This assumption is only valid if the gearbox dynamics are not relevant to the study.

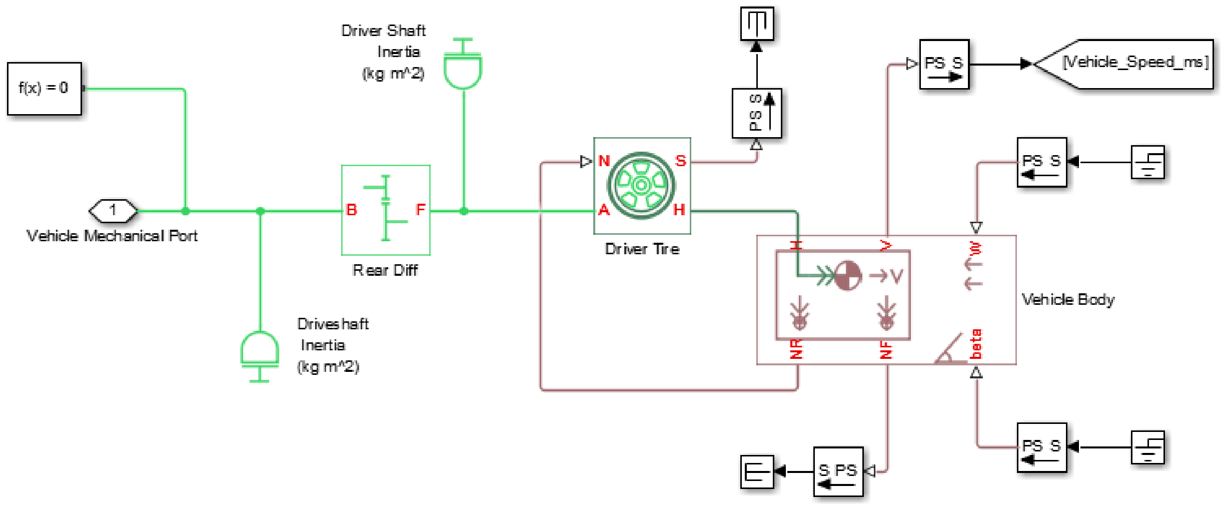


Figure 2.5: Simple vehicle model, Simulink diagram including functional blocks (vehicle body, Tire magic formula and simple gear)

Figure 2.5 illustrates the main components of the simple vehicle model developed using functional Simulink blocks and Table 2.3 includes the model parameters.

Table 2.3: Vehicle model parameters

System parameter	Value
Vehicle Mass	1200 kg
Tire Radius	0.3 m
Driveshaft inertia	$0.0015\,\frac{\mathrm{kg}}{\mathrm{m}^3}$
Driver shaft inertia	$0.003\,\frac{\mathrm{kg}}{\mathrm{m}^3}$
Gear Ratio	2

In order to test the validity of the model at this initial step, a constant torque of 200 N.m is supplied from EM-B for duration of 10 seconds. The velocity of the vehicle after 10 seconds can be calculated as follows:

$$\frac{dv_x}{dt} = \frac{1}{m}\left(\frac{TN}{r}\right) \tag{6}$$

$$\int_{v_i}^{v_f} dv_x = \int_{t_i}^{t_f} \left(\frac{TN}{mr}\right) dt \tag{7}$$

$$v_f = \frac{(200\ \mathrm{Nm})(2)}{(1200\ \mathrm{kg})(0.3\ \mathrm{m})} = 11.1\ {}^{m}/_{s} = 24.9\ \mathrm{mph} \tag{8}$$

Figure 2.6 illustrates the vehicle speed profile as described in Equations 6-8. The simple vehicle model does not capture effects such as tire slip, therefore the application of a constant torque results in a linear increase in speed with constant acceleration.

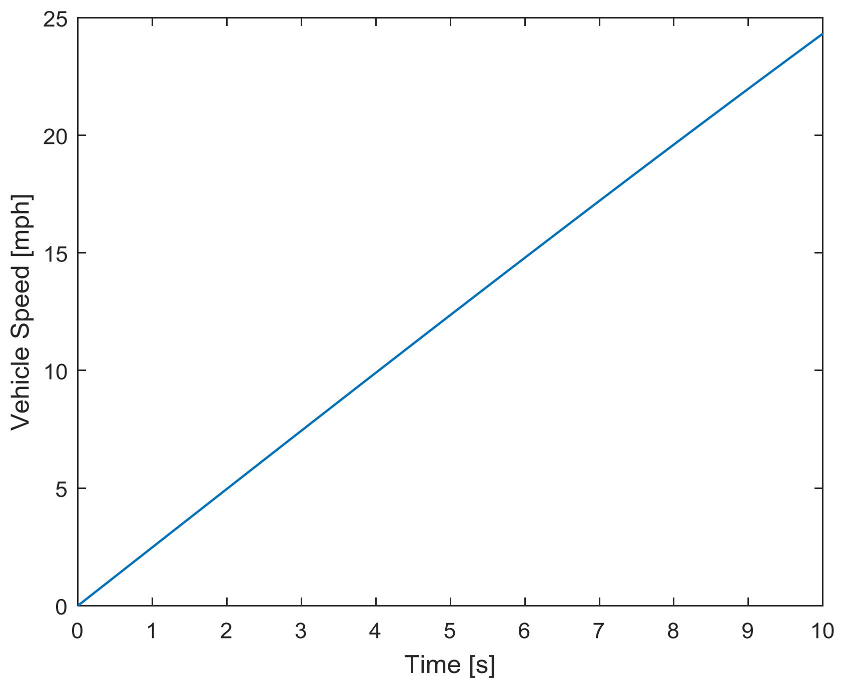


Figure 2.6: Vehicle speed profile

The Driver subsystem functions as a proportional feedback loop and sends a normalized torque signal to the controller. The proportional controller will send a maximum torque signal if the speed error is larger than 2 mph. Figure 2.7 illustrates the design of the proportional feedback loop which allows the vehicle to follow a drive cycle.

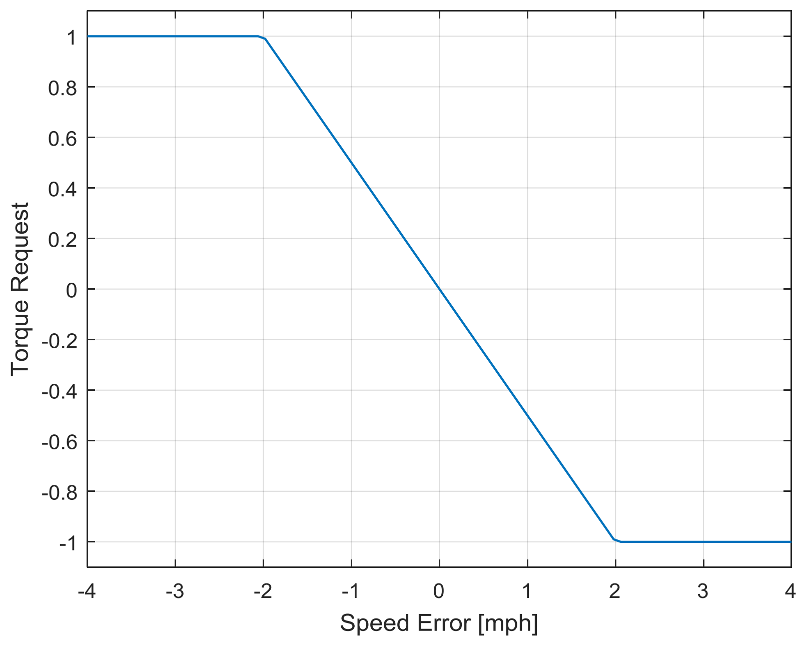


Figure 2.7: Normalized torque request

Using the first 500 seconds of the EPA Urban Dynamometer Driving Schedule (UDDS), it can be shown that the torque request signal from the proportional feedback controller will allow the vehicle to follow the drive cycle closely. Figure 2.8 illustrates the vehicle speed profile as well as the torque request as it follows the UDDS.

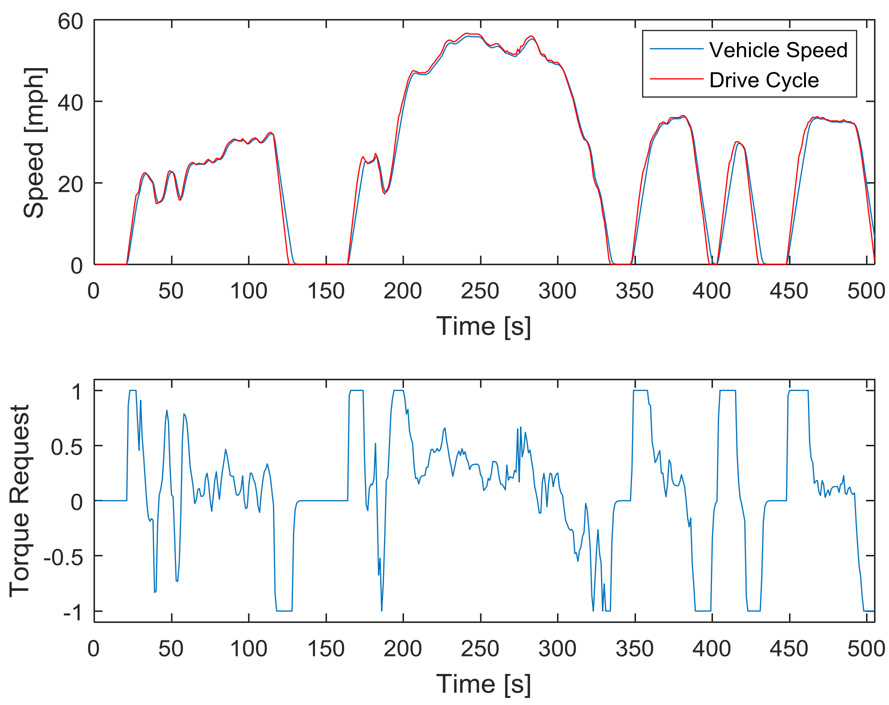


Figure 2.8: Vehicle speed profile, UDDS drive cycle

2.2 Simple battery and EM model

Following the development of the simple vehicle model, battery and EM models are developed using utility blocks as shown in Figure 2.9. The simple EM model features a constant maximum torque, no rpm limits, and is modulated by the controller. The battery model features a constant voltage power source with unlimited current and capacity.

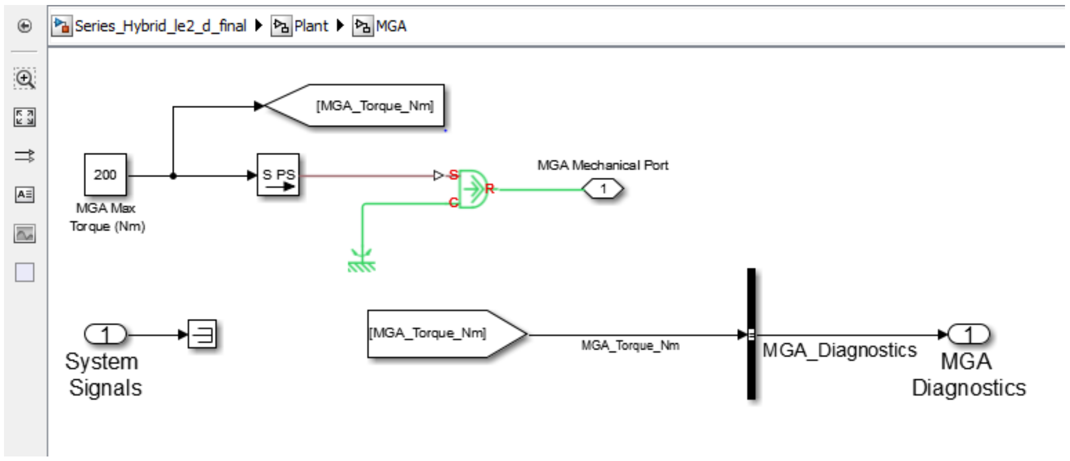


Figure 2.9: Simple EM model, utility blocks used in EM subcomponent

The power output of the EM is calculated by evaluating the speed and torque of the EM. The torque is directly proportional to the torque request from the driver subsystem as follows:

$$T_{EM} = T_{req} \mathrm{Max}(T_{EM}) \tag{9}$$

$$\omega_{EM} = \frac{V_x}{r} N \tag{10}$$

$$P_{EM} = T_{EM}\omega_{EM} \tag{11}$$

The power generated by the EM, is directly proportional to the battery current draw and voltage as described by:

$$I_{Battery} = -\frac{P_{EM}}{V_{Battery}} \tag{12}$$

and the battery state of charge (SOC) is calculated using:

$$SOC = \frac{1}{3600}\int_{t_0}^{t} I_{Battery}\, dt/\mathrm{Capacity} + SOC_{Initial} \tag{13}$$

In this calculation, the battery capacity is in the units of Amp-hour. Equations 12 and 13 were implemented using Simulink utility blocks as shown in Figure 2.10.

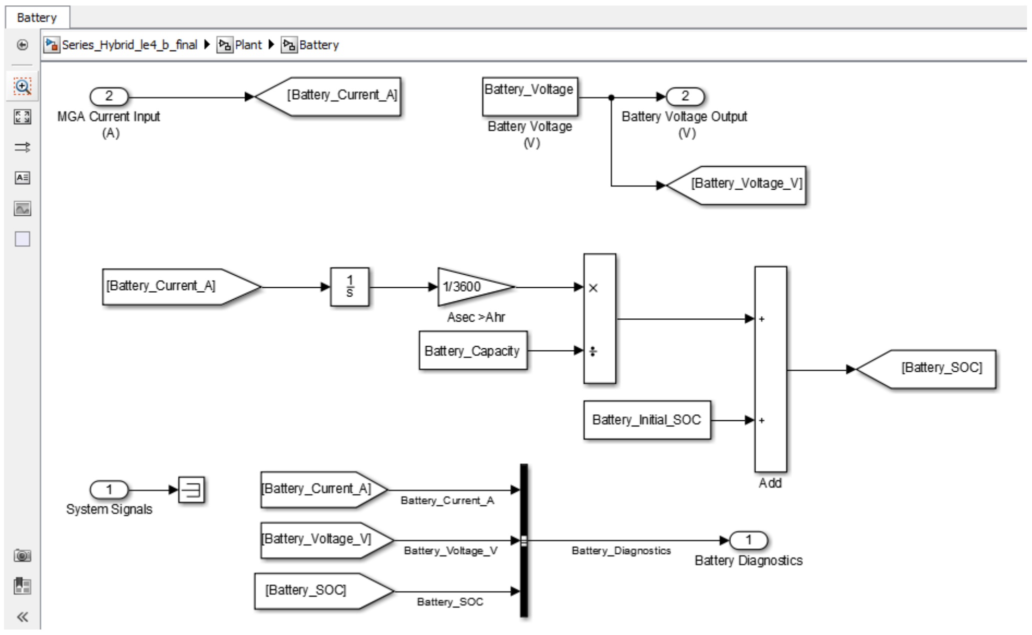


Figure 2.10: Simple battery model, Simulink diagram

As shown in Figure 2.11, during periods of acceleration, negative battery current results in lowering the SOC, whereas during deceleration, positive battery current results in an increase in the SOC. Most traditional HEVs are designed to maintain the battery SOC within an operable range. Continuous deep discharge and charge can damage the battery by lowering the battery life. Allowing the battery to achieve a high or low SOC can also prevent the vehicle controller from charging the battery when the SOC is high, or discharging the battery when the SOC is low.

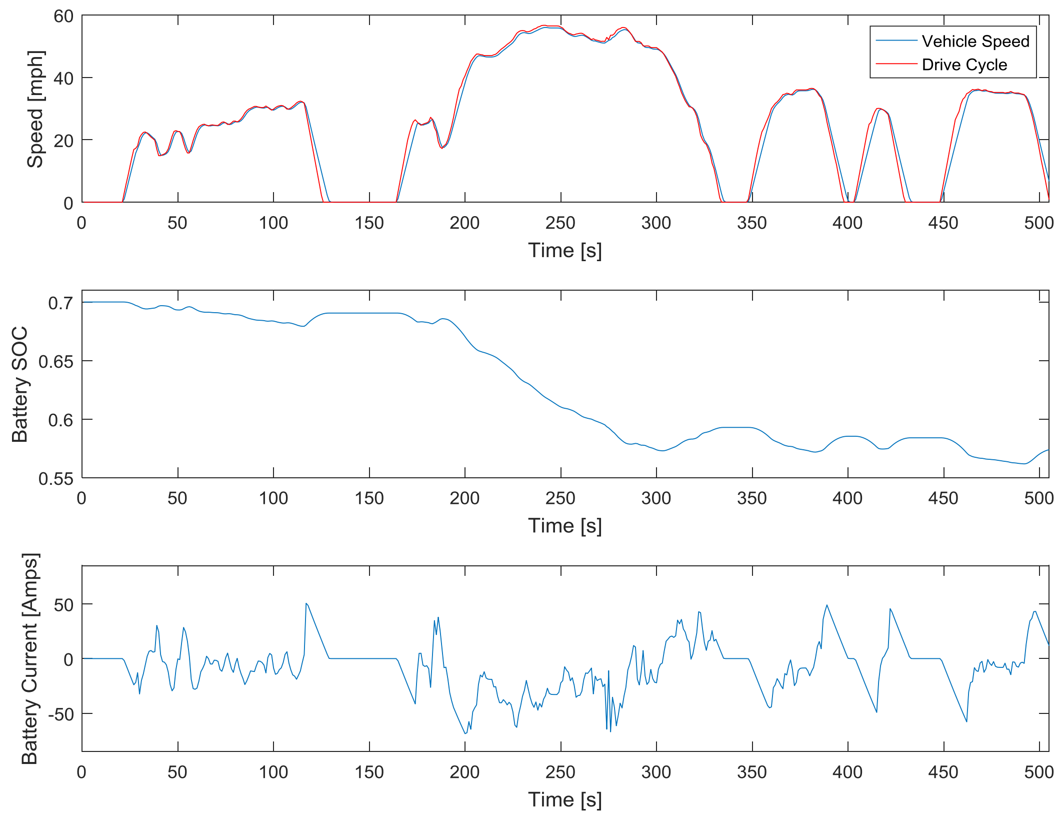


Figure 2.11: Vehicle performance with application of simple battery and EM model

2.3 Simple Charging Logic

AS shown in Figure 2.11, the vehicle SOC decreased which can lead to permanent damage to the battery and decrease the range. A secondary EM, EM-A which is identical to EM-B, is added to the model in order to charge the battery. EM-A is powered by an on-board generation unit in the form of a diesel engine.

A rule based controller unit is added to the controller subsystem to turn EM-A on and off based on the SOC. Applying this control strategy, referred to as charge sustaining, prevents the vehicle from depleting the battery. The effects of applying the charge sustaining strategy are shown in Figure 2.12. Controller prevents the SOC to drop below 60%, allowing EM-A to recharge the battery. Similarly the controller EM-A is turned off as the SOC reaches 70% to prevent overcharging.

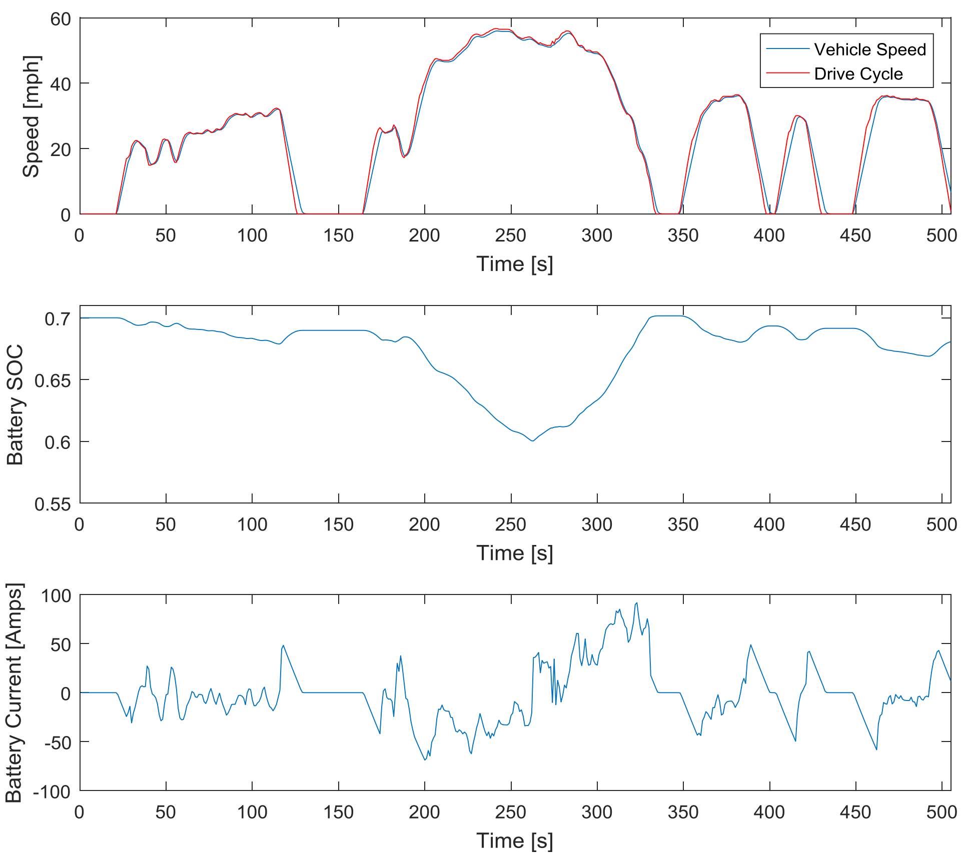


Figure 2.12: Vehicle performance with application of d simple charging logic

The Driver subsystem sends a normalized torque demand to EM-B, however the controller unit only sends an on and off signal to EM-A. In order to prevent damage to the EM, a feedback controller is implemented which sends a normalized torque request to EM-A. In order to power EM-A, a diesel engine is added to the Plant subsystem.

2.4 Improved Engine Model and Fuel Efficiency

In order to increase the engine model fidelity, an experimental torque curve for the engine is added to the engine model using look-up tables. The look-up table uses linear interpolation and extrapolation methods to evaluate the engine torque output based on the engine speed and throttle. An engine throttle request is provided from the controller unit. Figure 2.13 illustrates the experimental torque data included in the model.

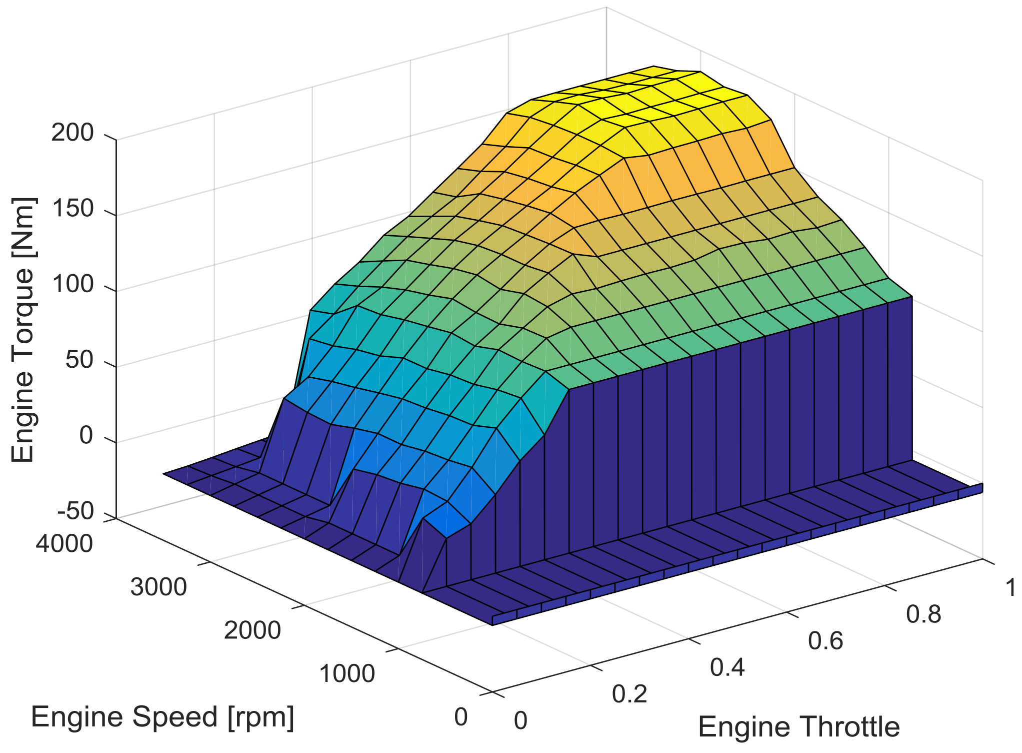


Figure 2.13: Engine Torque map

Similar to experimental engine torque data, fuel consumption data is added to the engine model using a look-up table. The look-up table uses linear interpolation and extrapolation methods to evaluate the engine fuel consumption based on the engine speed and throttle. Figure

2.14 presents the experimental fuel consumption data and Figure 2.15 illustrates the application of look-up blocks to incorporate the experimental data.

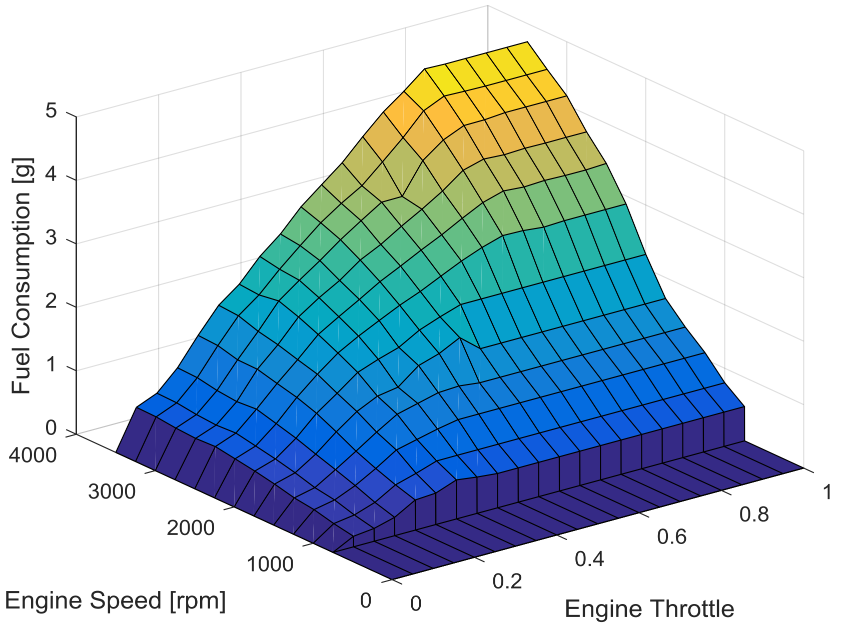


Figure 2.14: Engine fuel consumption map

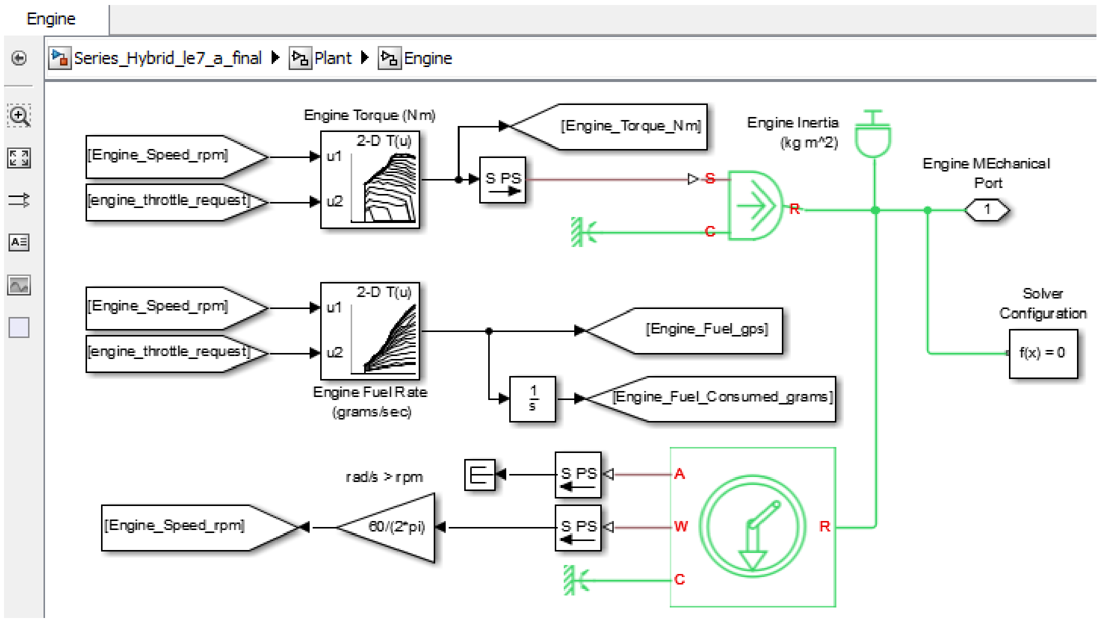


Figure 2.15: Improved engine model

The main advantage of using a series architecture in an HEV is the lack of direct mechanical connection between the engine and wheels which allows the engine to operate at a desired speed, independent from the wheels. The vehicle controller can operate the engine at the most efficient operating points, and reduce fuel consumption.

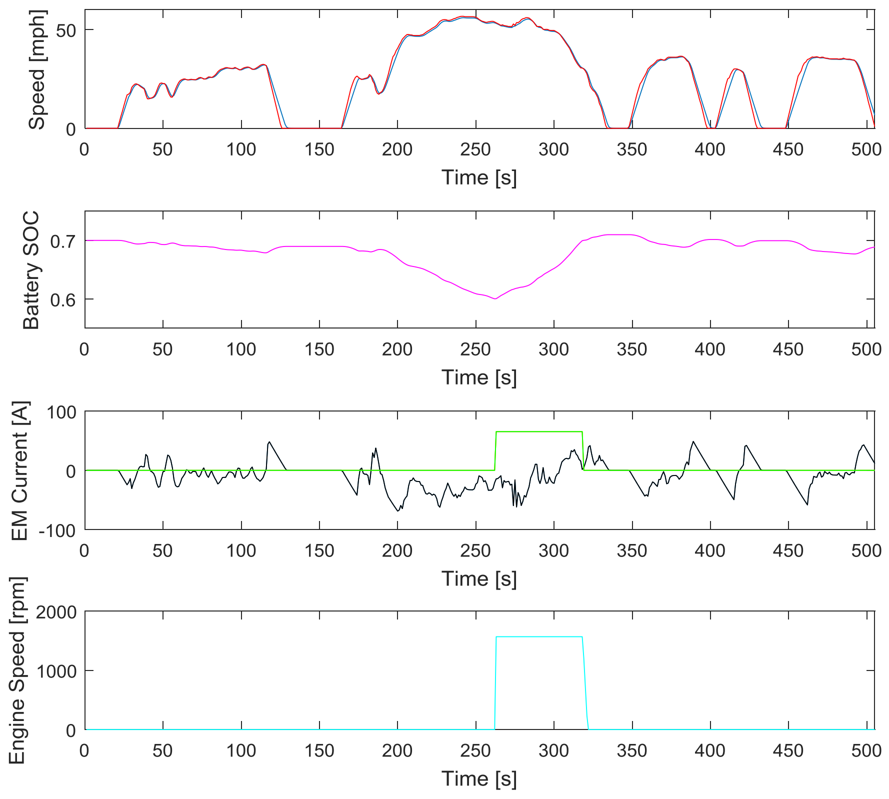


Figure 2.16: Vehicle performance, Engine charging enabled

Based on the engine fuel map and Brake specific fuel consumption (BSFC) map, the engine efficiency is maximized at a speed of 1800 rpm and engine throttle[2] of 0.5. The controller will operate the engine, exclusively at this operating point unless the battery SOC drops below a pre-defined threshold, at which point the engine throttle request increases to provide more torque

[2] Engine throttle is equal to the normalized torque request from the controller unit.

and to increase the SOC. Figure 2.16 presents the vehicle performance after the application of experimental engine data and Figure 2.17 illustrates the BSFC map.

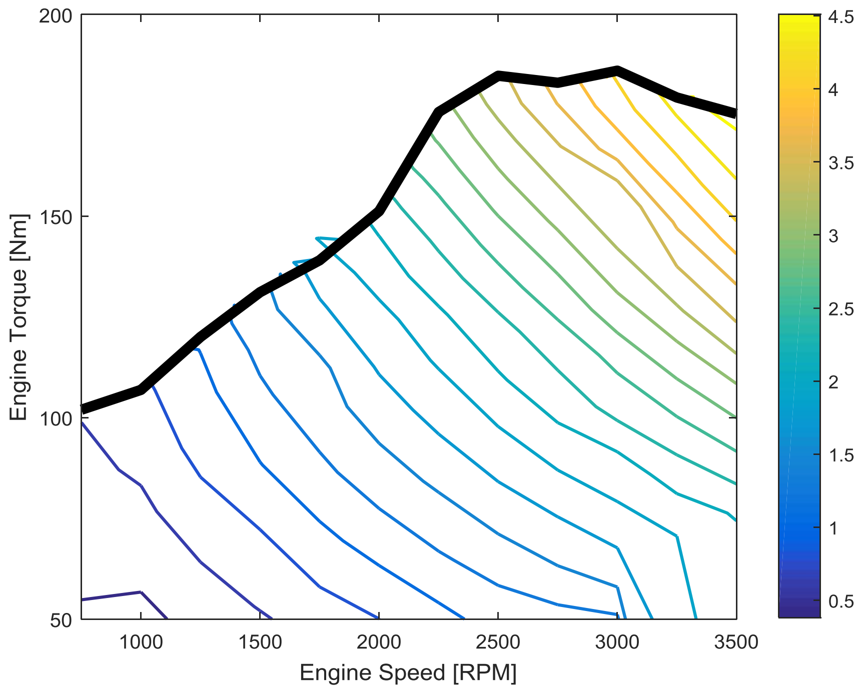


Figure 2.17: BSFC map

Fuel economy is traditionally reported by dividing the distance traveled by the volume of fuel consumed. However, in the case of HEVs, if the final battery SOC is lower than the initial SOC, fuel required to replenish the battery to the initial SOC should be considered in fuel economy calculations. Similarly if the final SOC is higher than the initial SOC, extra fuel consumed to charge the battery above the initial SOC should be subtracted.

First the electrical energy to change the battery SOC should be calculated as follows:

$$\mathrm{Ah_{consumed}[Amp.hour]} = (\mathrm{SOC_i} - \mathrm{SOC_f}) \times \mathrm{Capacity} \tag{14}$$

$$\mathrm{E_{Battery}[W.hour] = Ah_{consumed} \times OCV[V]} \tag{15}$$

and the efficiency of the system in converting fuel energy into electrical energy stored in the battery pack should also be considered by:

$$\mathrm{E_{Electrical}[BTU]} = \frac{3.412^{3} \times \mathrm{E_{Battery}}}{\eta_{\mathrm{conversion}}} \tag{16}$$

and the gas equivalent fuel economy with SOC correction can be calculated as described by:

$$\mathrm{E_{Fuel}[BTU] = Fuel_{consumed} \times 133393.1^{4}} \tag{17}$$

$$\mathrm{E_{Total}[BTU] = E_{Fuel}[BTU] + E_{Electrical}[BTU]} \tag{18}$$

$$\mathrm{Fuel_{consumed-RFG}} = \frac{\mathrm{E_{Total}[BTU]}}{114871.74^{5}} \tag{19}$$

$$\mathrm{MPGGE} = \frac{\text{Distance Traveled}}{\mathrm{Fuel_{consumed-RFG}}}. \tag{20}$$

The fuel economy reported for the vehicle model (MPGGE) is 103 mpg. The calculations are unrealistic as the inefficiencies of regenerative braking, the battery, and the EMs are not included in the model.

2.5 Improved Battery Model

Thus far the battery model featured: 1) a constant voltage equal to the open circuit voltage (OCV), 2) no energy conversion losses, 3) infinite current draw. In order to increase the model fidelity, a linear battery model as shown in Figure 2.18, is included where the battery voltage is a function of current.

[3] The 3.412 converts W.hour to BTU

[4] The 1.33393.1 $\left[{}^{BTU}/_{gal}\right]$ is the heating value of diesel fuel

[5] The 114871.74 $\left[{}^{BTU}/_{gal}\right]$is the heating value of RFG (Reformulated gasoline)

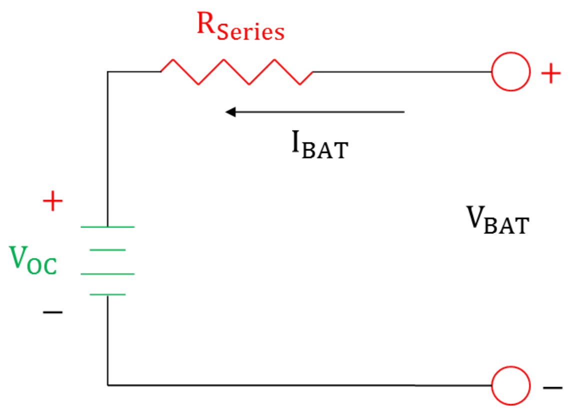


Figure 2.18: Improved battery model

$$V_{BAT} = V_{OC} + I_{BAT} \times R_{Series} \tag{21}$$

$$R_{Series} = R_{Internal} + R_o \tag{22}$$

Series resistance of the battery is the sum of battery internal resistance and the resistance of other electrical components such as the wiring. The battery OCV is a function of temperature and SOC. Experimental data, as shown in Figure 2.19, is used to include the temperature effects on the battery performance.

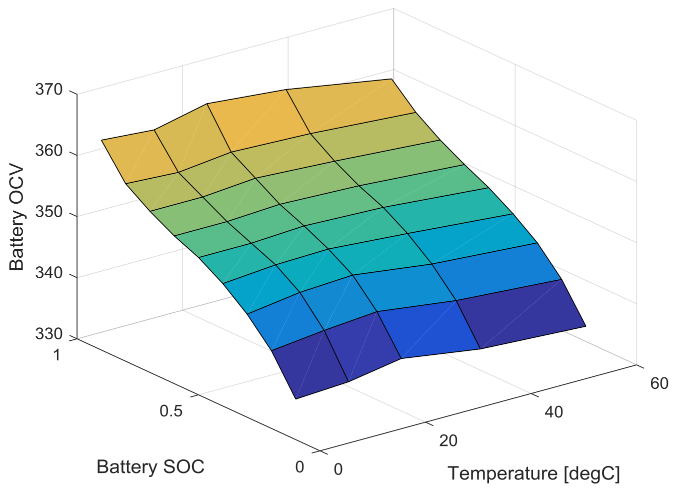


Figure 2.19: Battery Open Circuit Voltage experimental data

Experiments have also shown that the battery internal resistance is different during charge and discharge. Internal resistance is also a function of SOC and temperature. Figures 2.20 and 2.21 illustrate the internal resistance experimental data included in the model.

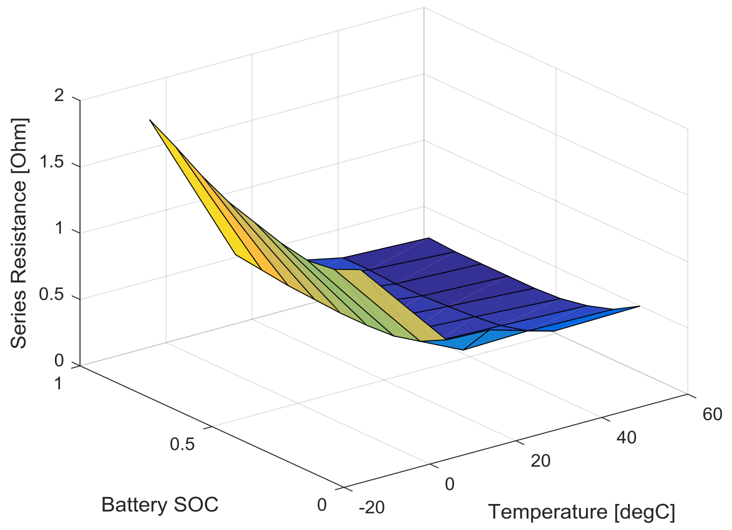


Figure 2.20: Total series resistance-Discharge

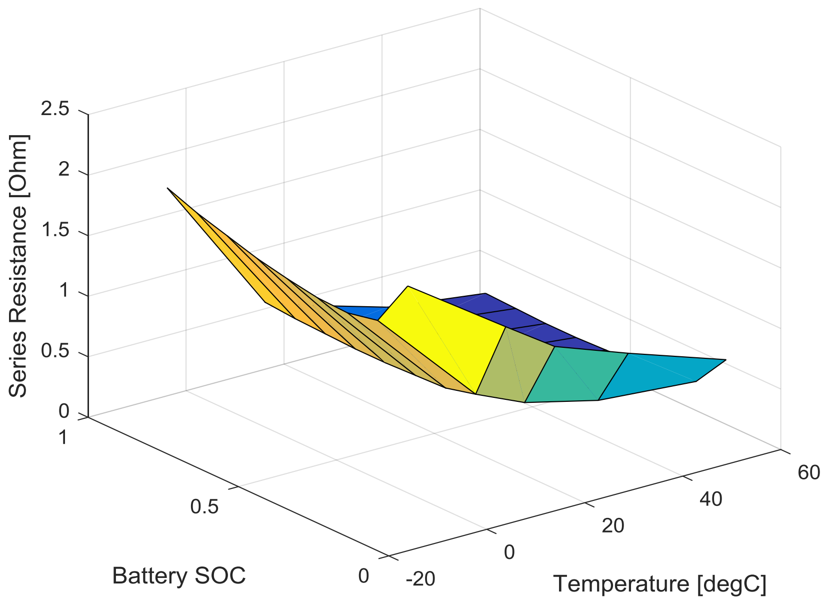


Figure 2.21: Total series resistance-Charge

The experimental data is added using a look-up table. Look-up table uses linear interpolation and extrapolation methods to evaluate the output based on the battery SOC and temperature. The improved battery model lowers the fuel efficiency of the vehicle model (MPGGE with SOC correction) to 90 mpg.

2.6 Improved EM Model

Thus far the EM model featured: 1) ideal conversion efficiency and, 2) constant maximum torque output. In order to improve the fidelity of the EM model, experimental torque-speed data is used which relates the maximum EM torque to the shaft speed as shown in Figure 2.22.

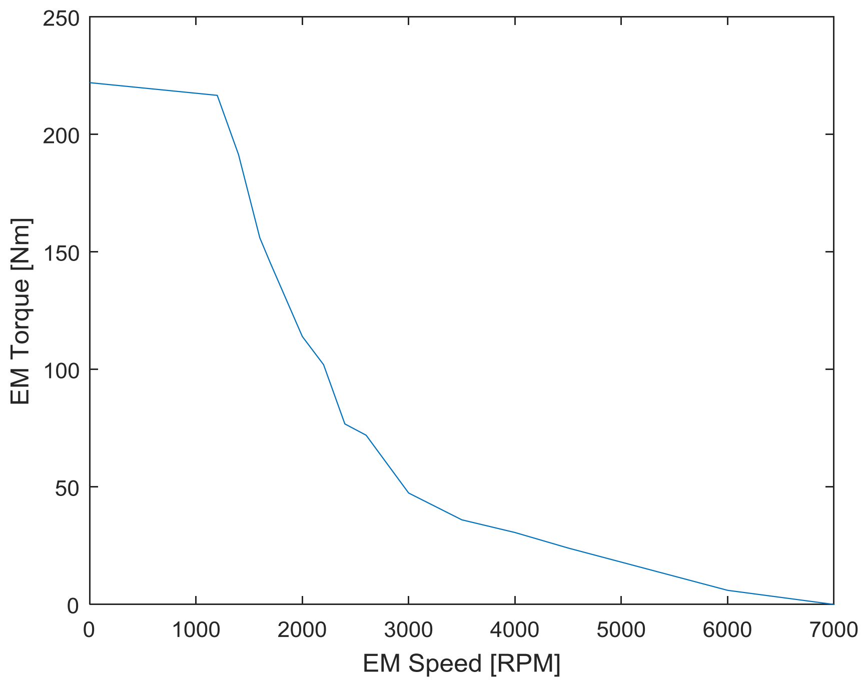


Figure 2.22: Electric machine torque curve

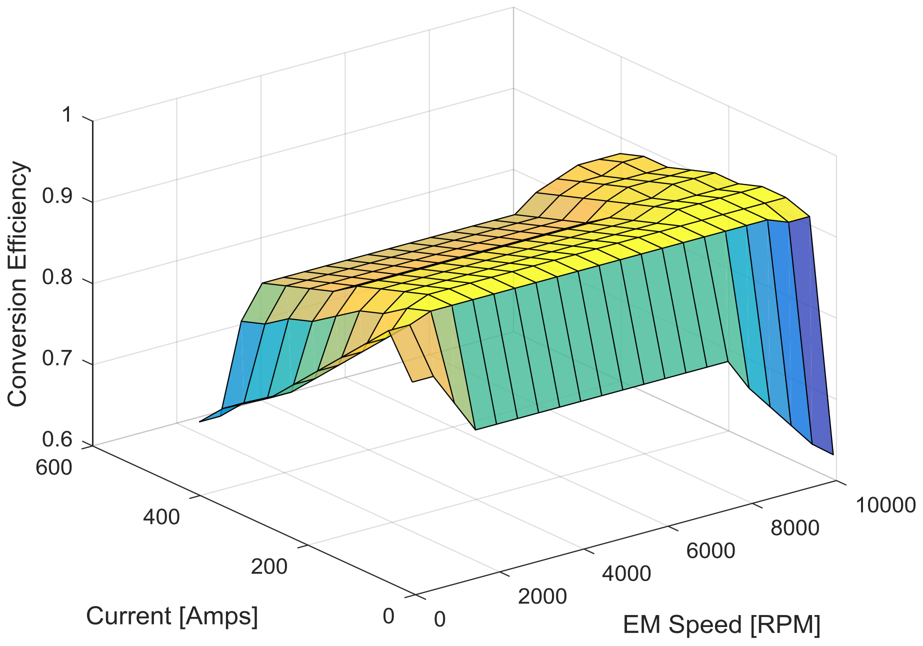


Figure 2.23: EM conversion efficiency map

Electromechanical energy conversion can go two ways: 1) electrical to mechanical and, 2) mechanical to electrical. Experimental results have shown that the conversion efficiency of the EM is dependent on shaft speed and current. Figure 2.23 illustrates the EM conversion efficiency data included in the model.

The addition of experimental data to the motor and generator model, increases the model fidelity. The fuel economy of the model decreases to 65.4 mpg.

2.7 Conclusion

Using the sequential refinement modeling method described in this chapter, students and engineers will be able to develop suitably complex vehicle models and develop the vehicle control strategy to operate the vehicle. In addition, the model can also be used in Hardware-in-the-loop tests where the model can be uploaded into the control unit and used for testing the vehicle operation in the next steps of vehicle development.

The hybrid vehicle model developed in this chapter is a series design which features a simple control strategy. However, the development of more complex control strategies, for example parallel and power-split architectures, can be accomplished using the instructions provided in this thesis and the supplementary tutorial in the Appendix.

In addition to the improvements to the vehicle model in Sections 2.4-2.6, further additions and improvements to the model can improve the model fidelity and ensure better operation in Hardware-in-the-loop scenarios. Simple additional improvements to the model can include the addition of current limits, engine emergency defueling at high engine speeds and limiting the auxiliary motor speed during engine startup.

Chapter 3: Evaluation of Vehicle Performance using Dynamic Programming

The introduction of HEVs into the fleet of performance vehicles manufactured by Ferrari S.p.A. requires a new design approach which ensures that new vehicles meet improved fuel economy and emissions standards while maximizing the utility and performance characteristics.

In order to evaluate each powertrain under consideration, a simulation-based design approach can ensure an unbiased comparison. DP can provide an optimally controlled performance metric for each architecture under consideration over pre-determined drive cycles. Therefore it evaluates each architecture using a consistent control strategy and thus avoids the uncertainty associated with different supervisory control approaches [4,9]. The DP algorithm has been developed and used in many applications in order to determine the optimal control strategy.

DP provides a globally optimal control path for a predetermined drive cycle using Bellman's Principle of Optimality. Since DP requires the future knowledge of the operation conditions and heavy computational loads, vehicle online implementation is not possible [9]. Nevertheless, attempts have been made to extract heuristic rules from dynamic programming results. Vehicle control units use alternative non-optimal control regimes such as Equivalent Consumption Minimization Strategy (ECMS) [4].

Bellman's Principle of Optimality states that the optimality of a future control action will not be affected by the optimality of any past control input [10]. DP uses this principle to progress backwards in time through a pre-determined drive cycle with identified states and control variables, and provides an optimal control path within constraints of the control space.

Chapter 3 provides a brief description of Dynamic Programming, followed by the method used to implement DP in HEVs and an example is provided.

3.1 Dynamic Programming

The DP algorithm is used on a class of discrete-time models in the following form,

$$x_{k+1} = F_k(x_k, u_k), \quad k = [0, N-1] \tag{23}$$

where k denotes the index of discretized time, x_k the state variable, u_k the control variable, and F_k the function defining the state variable. In addition, for application of DP, the state and control variables are discretized.

The total cost of employing the control strategy $\pi = \{u_0, u_1, \dots u_{N-1}\}$ with the initial state x_0 is defined by,

$$J_{0,\pi}(x_0) = g_0(x_0) + g_N(x_N) + \phi_N(x_N) + \sum_{k=0}^{N-1} [h_k(x_k, u_k) + \phi_k(x_k, u_k)] \tag{24}$$

where $J_{0,\pi}(x_0)$ represents the total cost, $g_0(x_0)$ and $g_N(x_N)$ include the cost of initial and final steps respectively, $\phi_k(x_k, u_k)$ the penalty function enforcing the constraints on the state and control variables, and $h_k(x_k, u_k)$ the incremental cost of applying the control at time k. The optimal control path is one that minimizes the total cost represented in Equation 24.

3.2 Application of DP to HEV Supervisory Control

The DP algorithm is implemented in MATLAB 2016a as a backward-looking simulation in which the vehicle follows a pre-determined drive cycle and the steady state kinematic and torque relationships are used to compute component operation states. The main advantage of implementing a backward-looking simulation is the faster computation time, which comes at the cost of overlooking energy due to transient effects.

The DP control problem of the parallel HEV is characterized as,

$$x = (\mathrm{SOC}, v, F_{req}) \tag{25}$$

$$u = (T_e, \omega_e, T_{EM}, \omega_{EM}, i_n) \tag{26}$$

$$h_k = \dot{m}_{fuel}(x, u) \tag{27}$$

where $\dot{m}_{fuel}$ is the mass rate of fuel consumed, v is the vehicle speed, and F_{req} refers to force required.

The DP algorithm applied to HEVs, seeks to minimize the forward fuel consumption at any point of discretized state-time space. This minimizing operation can be summarized in Equation 28,

$$J_k(\mathrm{SOC}_k^i) = \min\left[J_{k+1}(\mathrm{SOC}_{k+1}^i) + \dot{m}_{fuel} + \phi_k\right] \tag{28}$$

The system design constraints, based on the operational limit of each component, are summarized as,

$$\begin{array}{c}\text{Parallel HEV}\\ \text{Component capability}\\ \text{constraints}\end{array} : \begin{cases} T_{e,min} \leq T_e \leq T_{e,max} \\ \omega_{e,min} \leq \omega_e \leq \omega_{e,max} \\ T_{EM,min} \leq T_{EM} \leq T_{EM,max} \\ \omega_{EM,min} \leq \omega_{EM} \leq \omega_{EM,max} \\ i_n \epsilon [i_1, i_2, i_3, i_4, i_5] \end{cases} \tag{29}$$

where T_e and ω_e refer to engine torque and speed respectively, and T_{EM} and ω_{EM} refer to the EM torque and speed respectively, and i_n refers to the transmission gear ratio. Choices of control outside of the range specified in equation 29, results in application of a large penalty term.

In order to ensure the vehicle operates within a charge sustaining regime and the battery charge is within an allowable range, the following constraints are also applied,

$$\mathrm{SOC}_{min} \leq \mathrm{SOC} \leq \mathrm{SOC}_{max} \tag{30}$$

$$\mathrm{SOC}_0 = \mathrm{SOC}_N = \mathrm{SOC}_{ref} \tag{31}$$

using the penalty terms described earlier in Equation 24. The penalty terms are several orders of magnitude larger than the incremental cost of using fuel; otherwise they are zero.

In general, the computation time of a DP algorithm increases exponentially with the number of independent state and control variables since all permissible values of state and control variables are visited at each time step. Due to the nature of a backward-looking simulation and the prior knowledge of the vehicle speed at each time step, the drive cycle prescribes v and $\mathrm{F_{req}}$, leaving SOC as the only independent state variable. In addition, by applying the steady state constraints, Equations 25-27, the independent control variables are reduced to $\mathrm{u} = (\mathrm{T_e}, \omega_\mathrm{e})$. Following this method, the control variables will be reduced to the set meeting the speed and torque requirements at the wheels. Due to the transmission coupling constraint, ω_e choices are also limited to at most the number of gear ratios in the transmission due the kinematic constraint from Equation 27. Lastly, $\dot{\mathrm{m}}_{\mathrm{fuel}}$ is assumed to be only a function of engine operation points characterized by $\mathrm{T_e}$ and ω_e. During application of DP to an HEV, the discretized engine operation point ($\mathrm{T_e}$, ω_e) candidates are searched exhaustively to find the minimization operation point for Equation 25. An example Matlab code, for the application of the DP for a series architecture as shown in Chapter 2 is included in the appendix. For the purposes of the DP analysis a 9 speed transmission is added to the architecture to illustrate a more comprehensive application of DP.

Calculating the force required to power the vehicle using Equation 32,

$$\mathrm{F_{req}} = \delta\mathrm{Mma} + f_r mg + \frac{1}{2} C_d \rho_a A_{frontal} v^2 \qquad (32)$$

and estimating the incremental cost of using fuel by referring the experimental fuel data, the total cost of applying control is calculated by adding the fixed costs as stated above. The DP algorithm then determines the optimal control path by minimizing the total cost for the duration of the drive cycle.

Figure 3.1 represents the changes in the battery SOC for the duration of the drive cycle. In the design of hybrid electric vehicles, charge sustaining control strategies are utilized to ensure battery is available to power the vehicle at all times and store energy as needed. Another advantage of using charge sustaining control strategy is extended battery life. In order to ensure a charge sustaining operation, a high penalty term is applied when the SOC drops below or rises above the desirable SOC level.

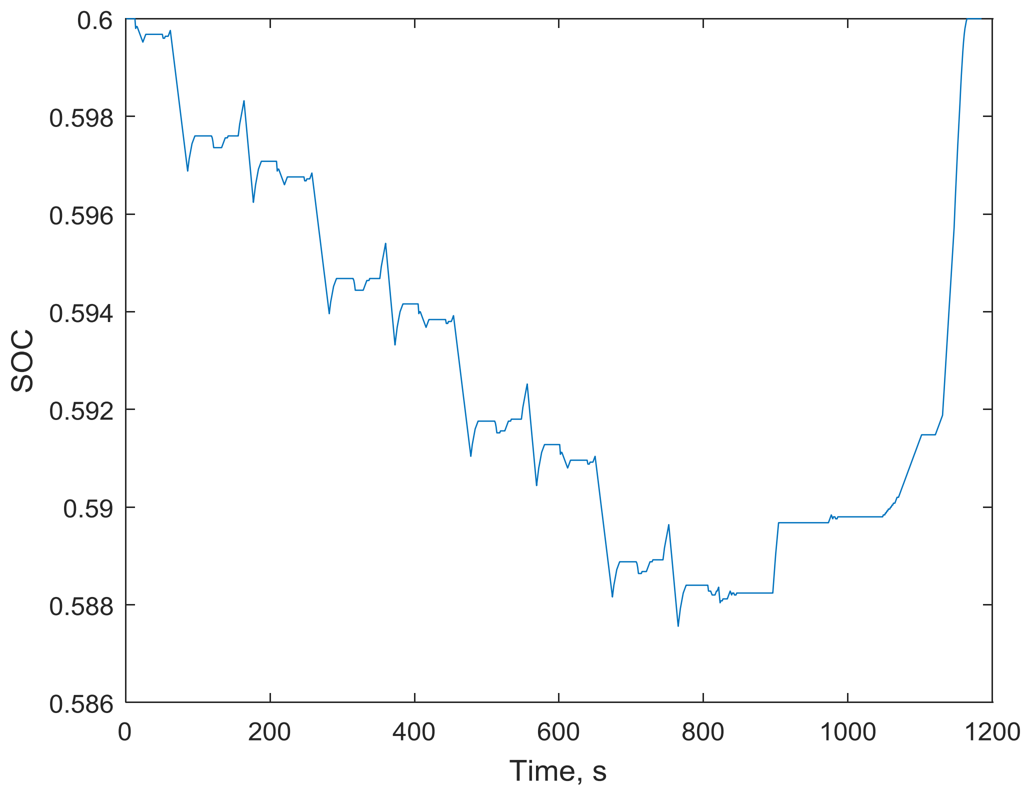


Figure 3.1: Battery SOC

Selecting the correct gear can also improve the efficiency of the drive as the engine can operate at a more desirable speed. The efficiency of the power transmission through the gearbox and other architecture components should also be considered when applying DP. Experimental data provided by Ferrari S.p.A. were used in order to estimate the energy loss through the powertrain.

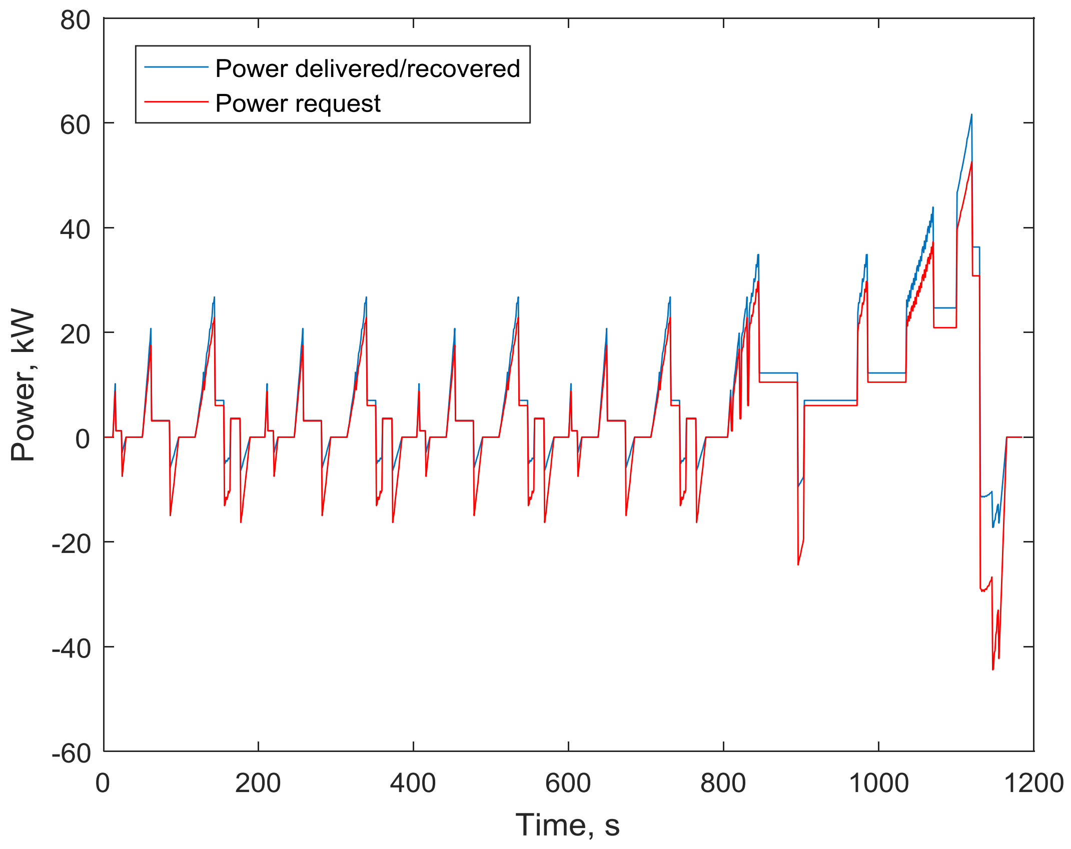


Figure 3.2: Power required vs. Power supplied/recovered

Figure 3.2 illustrates the effects of powertrain inefficiencies considered in the DP algorithm. During periods of deceleration due to the inefficacies of the EM and other powertrain components some of the power available for recovery in the energy storage unit is wasted.

Chapter 4: Multi-physics co-simulation of battery thermal model

The analysis and proper design of the energy storage system is a critical concern in the product development phase of HEVs. Rechargeable Li-ion batteries (LIBs) feature many advantages compared to alternative storage devices. LIBs provide high power and energy density over a broad operating temperature range, as well as low self-discharge, stable chemistry, and long battery life [11-12]. In addition, the growth in the market demand and production of high power density LIBs has led to a sharp drop in price of such batteries. However, challenges remain in the development of HEVs using LIBs due to stringent safety and performance requirements. Some of the limitations of the current LIBs include local underutilization, capacity drop, localized thermal degradation, and stress-induced material damage.

Significant efforts have been put forward to develop-simulation based tools to analyze battery performance and predict operational states of LIBs. The porous electrode theory, which predicts the lithium diffusion dynamics and charge transfer in an electrode system has been investigated to predict the electrical dynamics of a cell [12-13]. This approach can be used to investigate the microscopic behavior of the cell. Others have proposed and investigated an equivalent circuit model to estimate the electrical response and heat generation [12, 14-15]. Researchers have also focused on the structural response of the LIBs and effects on performance degradation. The volume change of electrode material has been investigated to estimate the electro-thermal induced stress and strain [12] and the potential effects on capacity have been thoroughly investigated. Several lumped parameter thermal models have been proposed and validated through experiments [16-17]. Great effort has been placed on developing coupled models between electro-chemical and heat transfer domains in order to investigate the coupled effects of discharge/charge current, voltage, and temperature on the SOC and battery state of health (SOH) [18-20].

Recent research has focused on development of coupled electro-thermal and heat transfer models in order to investigate the cell's temperature distribution under extreme conditions, such as over-charging/discharging, internal short circuit, and extrusion [16]. Coupled heat transfer and computational fluid dynamics models with a lumped value of heat generation were used to visualize the temperature gradient for both time-dependent and steady state simulations [16,21-22]. Multiple co-simulation platforms were used to simulate coupled thermo-electrical and heat transfer models. The effects of manufacturing variations and battery pack organization on cell thermal and power behavior were also investigated [21]. Moreover, researchers have been able to characterize the effects of localized high temperature regions on cell performance and SOH [22]. However, research on development of fully coupled electro-thermal, heat transfer, and computational fluid dynamics models to investigate temperature distribution at the battery pack level, which identify hot spots, is still in early stages. To the knowledge of the author, the coupled electro-thermal cell models have not been developed further to include the structural layout of battery pack with high capacity cooling units.

This chapter proposes a co-simulation approach to simulate modular and fully-coupled multi-physics models of a LIB battery with high fidelity thermal domain modeling using commercial off the shelf (COTS) software. The proposed multi-physics model couples a discrete electric model and continuous thermal model domains using the co-simulation capability of the SIMULIA simulation platform for rapid and scalable model development. Furthermore, the computational cost has been reduced by the development of a discrete electrical domain, as well as limiting the data exchange between co-simulation model components to discrete communication points. The main purpose of this multi-physics co-simulation is to predict the temperature distribution and identify temperature hot spots. The critical system parameters, which govern the electrical and thermal dynamics, were provided by the cell manufacturer. Experimental test data was used to calibrate the system response and ensure high model fidelity.

4.1 Model Development

One of the unique challenges facing Ferrari engineers and others in development of HEVs is the design and verification of the battery pack. Due to the high-performance characteristics of the vehicle, the battery pack should be designed to operate under high-frequency-high-load conditions, including high current charge and discharge which can lead to areas of high temperature within the battery pack. As such, the battery pack model includes a thermal domain in order to investigate:

1) The spatial distribution of temperature to identify hot spots,
2) The effects of high temperature on performance characteristics such as battery life,
3) The cooling power required to maintain battery temperature.

The modeling approach includes three simulation domains: electro-thermal, thermal-structural and fluid. The discrete electrical model refers to the mathematical representation of the battery dynamics, using ordinary differential equations, relating the transfer of chemical energy to electrical energy and vice versa, during discharge and charge, respectively. The model also relates the battery charge/discharge current to heat generated and uses ordinary differential equations to estimate the heat loss to the environment and cell core temperature. The continuous thermal-structural model, developed using COTS software, refers to the heat transfer from each individual cell through the battery structure, which includes a cooling plate, to the flow of coolant in the cooling system and the surroundings air through convection. The continuous fluid domain refers to the heat transfer to the cooling fluid within the cooling plate. Figure 4.1 illustrates the interactions between the three model domains.

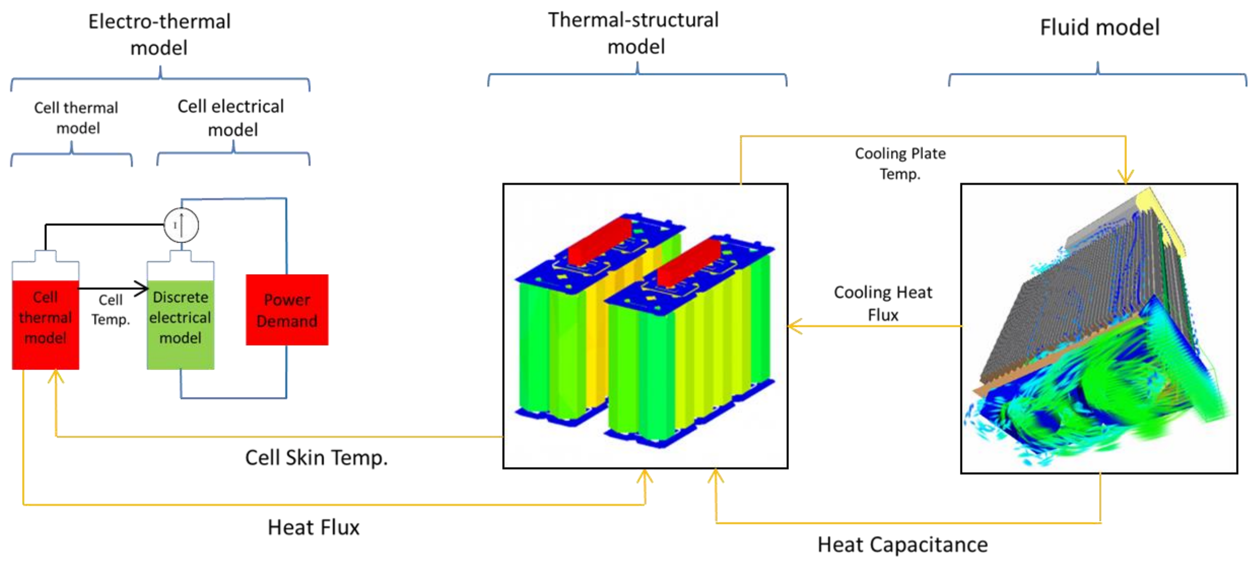


Figure 4.1: Battery model domains, orange arrows represent the flow of communication signals between co-simulation components. Note: Images representing the thermal-structural and fluid domain models only represent the hypothetical layout of the battery module and the coolant flow

The three simulation domains were coupled based on the conservation of energy. The heat loss from the electro-thermal model is imposed as heat flux input at the internal boundary of the cell thermal-structural model. In turn the temperature at the internal surface of the cell thermal-structural model is enforced as the temperature of the electro-thermal model. The thermal-structural model is coupled with the fluid model at an interface surface region. At this interface, the temperature of thermal-structure surface is imposed to the fluid model as a boundary condition. In turn, the fluid model heat capacitance and heat flux are enforced as the thermal-structural model boundary conditions and input heat flux.

Co-simulation requires the identification of master and slave components. The thermal-structural model is identified as the master component, while the fluid and electro-thermal domain function as slave components. The master initiates co-simulation by taking the first time step. The results and co-simulation time are communicated to the slave components and each slave takes appropriate steps to reach the co-simulation time, after which the results are communicated to the master and other slave components. The co-simulation progresses as the master takes the next step.

In order to establish communication between the three model components using a co-simulation approach, the electro-thermal model is translated into a tool-independent library, Functional Mock-up Unit (FMU), which can be used to execute the model in any Functional Mock-up Interface (FMI) compatible environment and establish communication between the electro-thermal model and other FMUs, or model components developed in the host, FMI compatible, modeling environment. The electro-thermal domain includes an individual cell (multiple cells can be arranged to form a battery module and/or pack), control unit, and power cycle in the form of a look-up table. The thermal-structural heat transfer model is developed using Finite Element Modeling (FEM) software, Abaqus 2016, and includes all the battery pack components. Similarly, the fluid domain model is developed using Abaqus 2016 and includes the model of cooling fluid that removes heat from the cooling plate. Abaqus 2016, which is part of the SIMULIA simulation platform, can establish communication with the electro-thermal domain FMU following the FMI standard. Development of high fidelity models for each domain is discussed in Section 4.1. In Section 4.2, the cell model was calibrated using experimental test results. Sections 4.3-4.7 focus on the next steps of the research partnership and potential applications of the battery pack model.

4.1.1 Electro-thermal Domain

In order to develop a modular model of the battery cell, the Modelica language is used. Modelica is an open-source, object-oriented, and equation-based modeling language which can be used to easily model complex physical systems containing electrical, mechanical, hydraulic, thermal, electronic, control and process oriented subcomponents [23,24]. A wide array of open-source Modelica Libraries have been developed, providing rapid and scalable model development capability.

Dymola is a commercial modeling and simulation environment based on the Modelica modeling language. Dymola has a built-in capability to simulate object-oriented models as well as export models as Functional Mock-up Units. FMUs are executable function files, generated following the Functional Mock-up Interface, which is a tool-independent standard to support model exchange and co-simulation of dynamic models. FMI allows for the coupling of several independent simulation tools to develop modular coupled models. Data exchange is restricted to discrete communication points, as specified by the host simulation tool, and each subsystem is solved independently between the communication points. More information about FMI and its functionality has been published elsewhere [25-26].

In order to streamline the development of a cell model, the open-source Electric Energy Storage (EES) library developed jointly at the Austrian Institute of Technology and Vienna University of Technology was used. The library contains sub-components of different complexity, such as individual cells and stacks interacting with loads, battery management components and charging devices.

The EES library components are designed as universal components allowing the end users to easily simulate specific scenarios by varying their parameterization. The use of the Modelica language allows seamless modifications of the equations used in each subcomponent. For the purposes of this analysis, the built-in LinearDynamicImpedance cell model was modified in order to properly define the cell model, based on experimental data. The EES library is structured as shown in Figure 4.2.

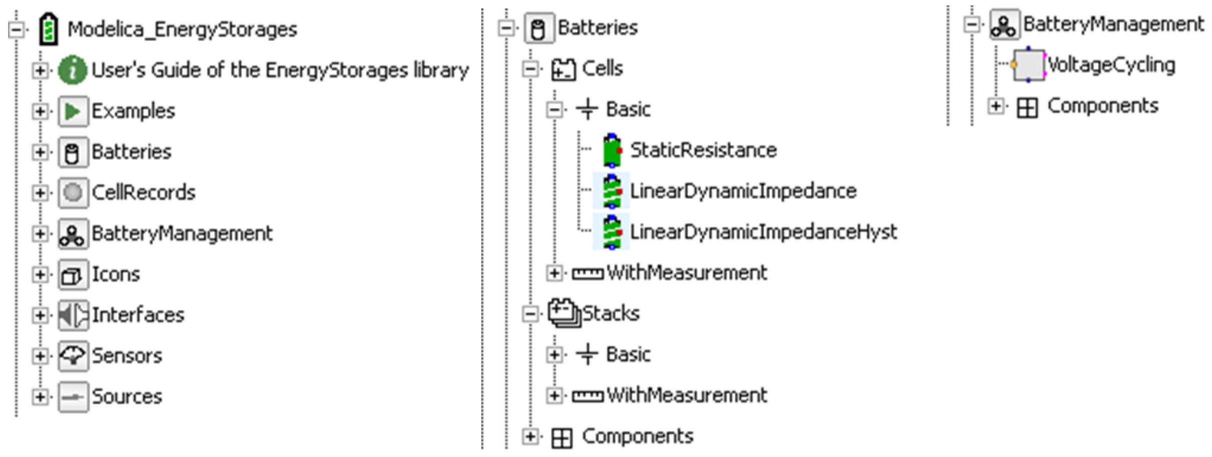


Figure 4.2: Electric Energy Storage library, Batteries and Battery Management subsystems

The Batteries package includes models for individual cells as well as stacks of n_s cells connected in series and n_p cells in parallel. A single cell component is selected for the purpose of this analysis. The single cell model features a positive (pin_p), a negative (pin_n) and an optional temperature connector (heatPort) which is activated herein as cell parameters were defined as a function of temperature [27]. The common parameters used in the cell model are given in Table 4.1.

Table 4.1: Common cell parameters [27]. Starred values* are calibrated

Name	Unit	Description	Value*
SOC_{ini}		Initial state of charge	0≤ SOC ≤1
$OCVtablecharge$	V	Look-up table for the OCV vs. SOC during charge	Figure 4.3
$OCVtabledischarge$	V	Look-up table for the OCV vs. SOC during discharge	Figure 4.3
Q_{ini}	C	Initial transferred charge	0
C_0	C	Capacity at T_{ref} for $Q_{ini} = 0$ and $t = 0$	2880
$useHeatPort$		Boolean variable for using the heat port	On/off
T_{ref}	K	Reference temperature	293
$alphaRs$	K^{-1}	Linear temperature coefficient for R_s	0.013*
R_{sref}	Ω	Ohmic resistance at reference temperature	0.014
$alphaC$	K^{-1}	Linear temperature coefficient for capacity	-1.6
C_{Heat}	J/K	Cell heat capacity	96*

The model output quantities are given in Table 4.2 and the calculations are presented next. The removed charge is given by,

$$\mathrm{Q} = \int_{\mathrm{t_{start}}}^{\mathrm{t_{stop}}} \mathrm{I(t)dt} \tag{33}$$

and the total transferred charge is,

$$\mathrm{Q_{abs}} = \mathrm{Q_{ini}} + \int_{\mathrm{t_{start}}}^{\mathrm{t_{stop}}} |\mathrm{I(t)}|\mathrm{dt} \tag{34}$$

The OCV of a battery cell changes with the SOC. Experimental results have shown that the OCV function is different during charge and discharge of the cell. As shown in Figure 4.3, OCV is provided using two linearly interpolated look-up tables for charge and discharge. The open circuit voltage is limited between the charging voltage limit CVL (SOC=1) and the discharge voltage limit DVL (SOC=0) [27].

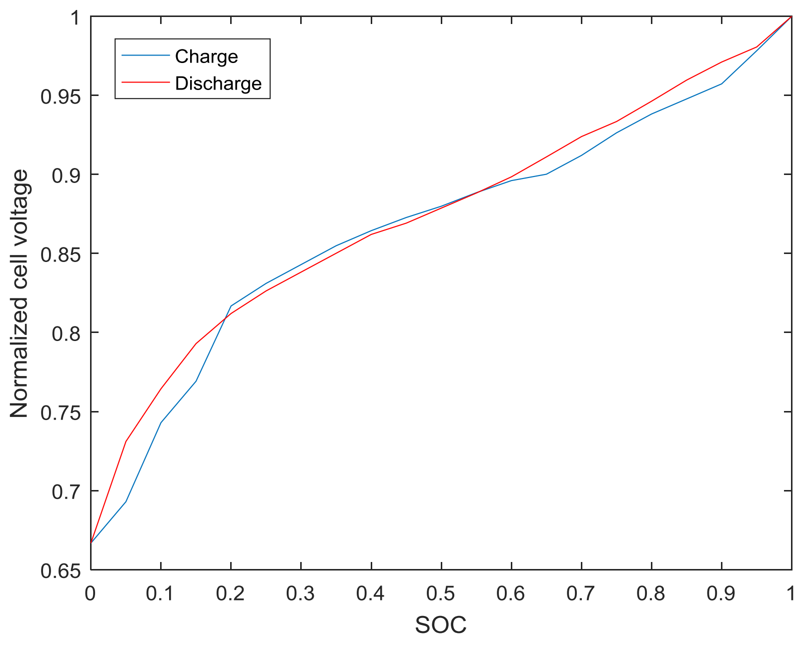


Figure 4.3: Normalized Cell Voltage

The model can consider the effects of aging by considering both calendaric aging and cycling. Calendaric aging is estimated from time, which for the purposes of the current analysis, is minimal and therefore removed from the model. Cycling is directly proportional to absolute transferred charge Q_{abs}. Aging of the cell mainly influences the performance by decreasing capacity and increasing the internal impedance [27].

The cell capacity calculated by,

$$C = \left(C_0 + K_{CQ_{abs}} \times Q_{abs}\right)\left[1 + alphaC \times \left(T_{heatPort} - T_{ref}\right)\right] \tag{35}$$

is temperature dependent and decreases with increased transferred charge due to cycling. The single cell model features a single, temperature dependent, ohmic impedance, modeled as,

$$R_s = R_{ref}\left[1 + alphaRs \times \left(T_{heatPort} - T_{ref}\right)\right] \tag{36}$$

which does not consider the increase in impedance due to aging.

The SOC is given at each time step as,

$$\mathrm{SOC} = \mathrm{SOC_{ini}} - \frac{\mathrm{Q}}{\mathrm{C}} \tag{37}$$

and the equivalent number of cycles

$$\mathrm{cycles} = \mathrm{cycles_{ini}} + \int_{\mathrm{t_{start}}}^{\mathrm{t_{stop}}} \frac{|\mathrm{I(t)}|}{2\mathrm{C}}\,\mathrm{dt} \tag{38}$$

with

$$\mathrm{cycles_{ini}} = \frac{\mathrm{Q_{ini}}}{\mathrm{C_0 + C}} \tag{39}$$

relates the total transferred charge to the cell capacity C. Therefore, one cycle is equivalent to the charge transfer of one full charge and one full discharge of the capacity C [27].

The Cell model includes *num* serially connected RC elements, as shown in Figure 4.4. The cell model was manually edited to include two sets of RC elements representing the cell dynamics in charge and discharge. Based on the direction of cell current, a switch is used to connect and disconnect the corresponding RC component.

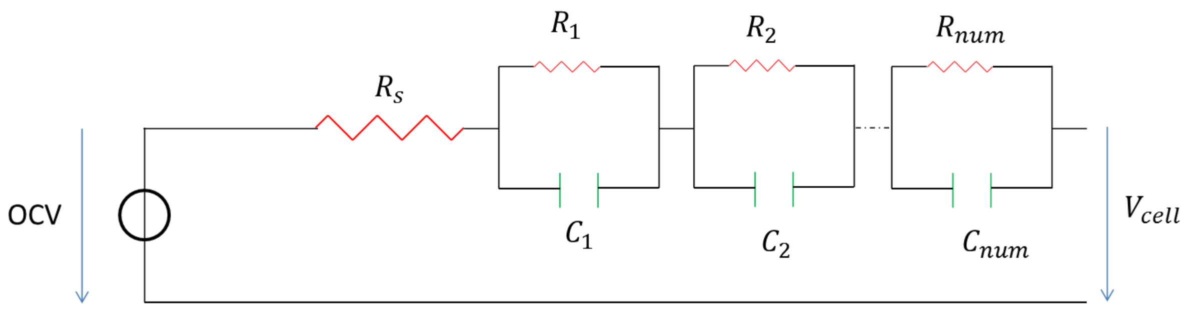


Figure 4.4: Cell model with one ohmic impedance and *num* serially connected RC elements

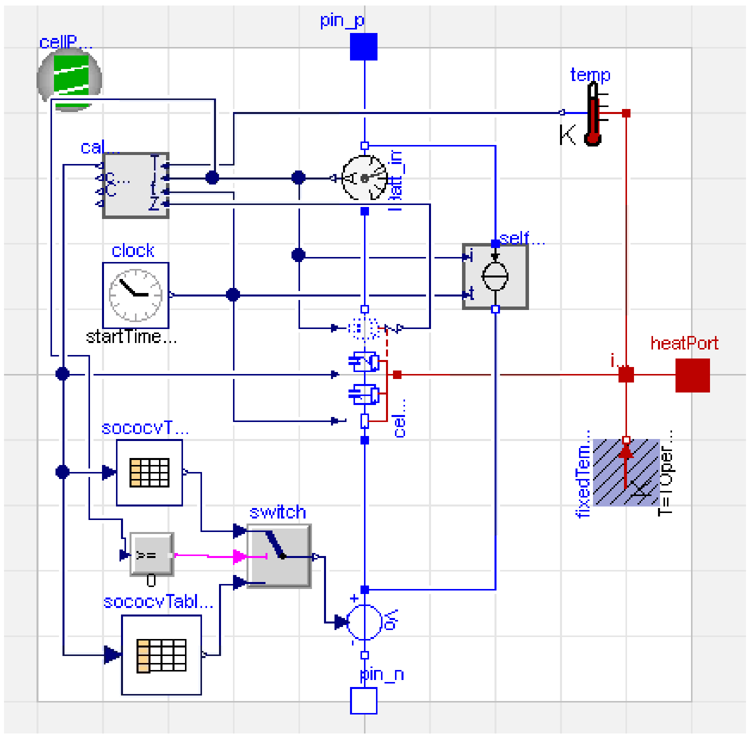


Figure 4.5: Cell model diagram

A list of calculated output variables of the modified *LinearDynamicImpedance* cell model used in the analysis is available in Table 4.2.

Table 4.2: Calculated output variables from cell model [27]

Name	Unit	Description
SOC		State of charge
OCV	V	Open circuit voltage
Q_{abs}	C	Total transferred charge
$cycles$		Number of equivalent cycles
t	S	Calendaric cell time
C	C	Cell capacity
V	V	Cell voltage

The EES library contains a battery management block (*Cycling)* which is used to implement a rule-based control of the battery cell. It has three Boolean outputs which control the operation of the charging unit, power cycle look-up table and the cell. The battery management block can be used to cycle the cell over a predefined voltage range [V_{min}, V_{max}], while limiting the maximum current draw during discharge and charging current. The EES also includes a constant current, constant voltage (CCCV) charging device which charges the cell with constant current until the constant voltage level is reached [27]. The battery module under study contains twelve individual cells arranged in two series sets of six parallel cells, as shown in Figure 4.6.

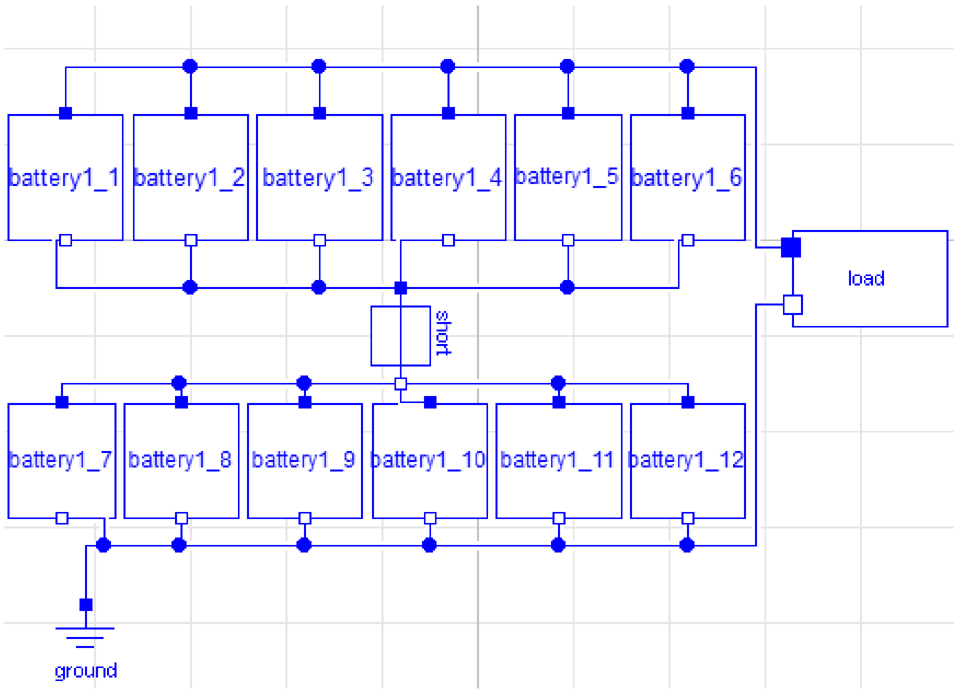


Figure 4.6: Diagram of module with 12 cells

The subcomponent labeled *load* includes the charging unit, power cycle look-up table, a switch and the cycling rule-based control unit. The switch works by disconnecting the power cycle when the cell/module drops below V_{min} and connecting the charger. The switch features a short delay to prevent numerical instability. Similarly, as the voltage reaches the constant voltage level, the switch disconnects the charger and connects the power cycle to continue the power cycle [27]. Figure 4.7 illustrates the construction of the *load* subcomponent.

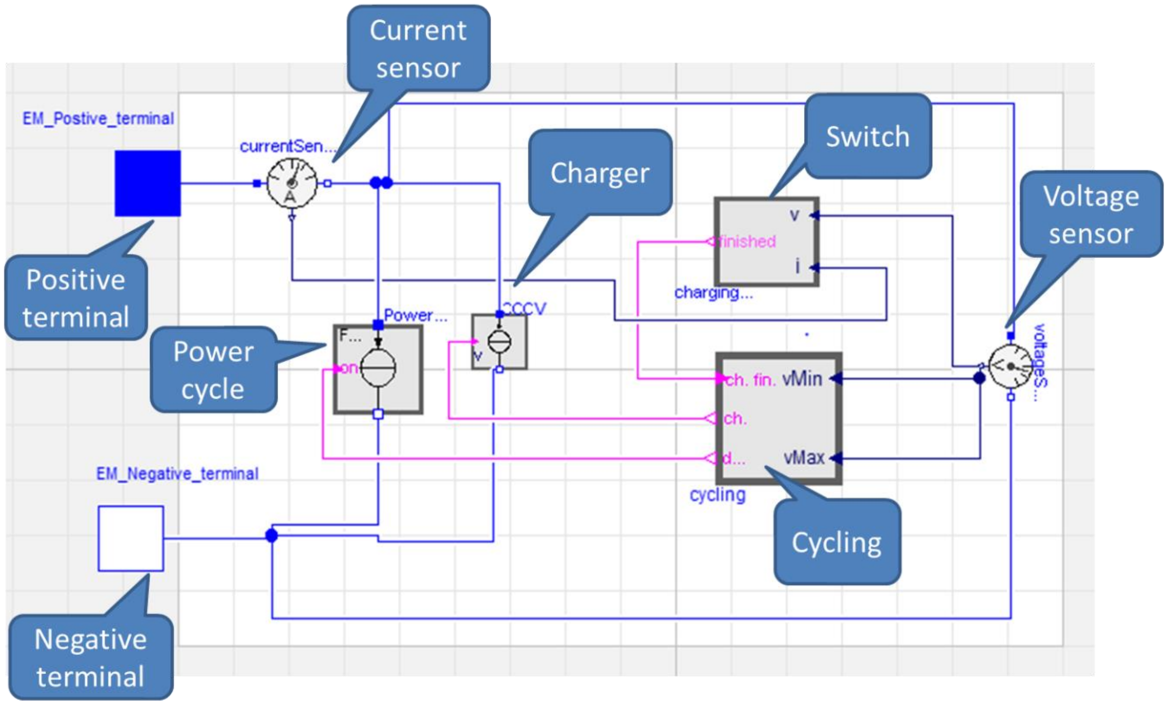


Figure 4.7: Charging unit and model controller

In order to study the effects of heating on battery performance and investigate the cooling power needed to ensure the safe operation of the battery pack, the heat generated during cycling is calculated by

$$Q_{Heat} = R_s I^2 \tag{40}$$

and the temperature of the cell is calculated by the heatCapacitor model by solving,

$$T = \int_{t_{start}}^{t_{stop}} \frac{Q_{Heat} - Q_{cooling}}{C_{Heat}} dt \tag{41}$$

where C_{Heat} is the heat capacity of the battery cell and $Q_{cooling}$ is the heat dissipated through the cooling plate. An additional thermal resistance is included in the model to represent the heat resistance of the skin of the cell. Figure 4.8 illustrates the application of Equations 40 and 41 to the electric domain model.

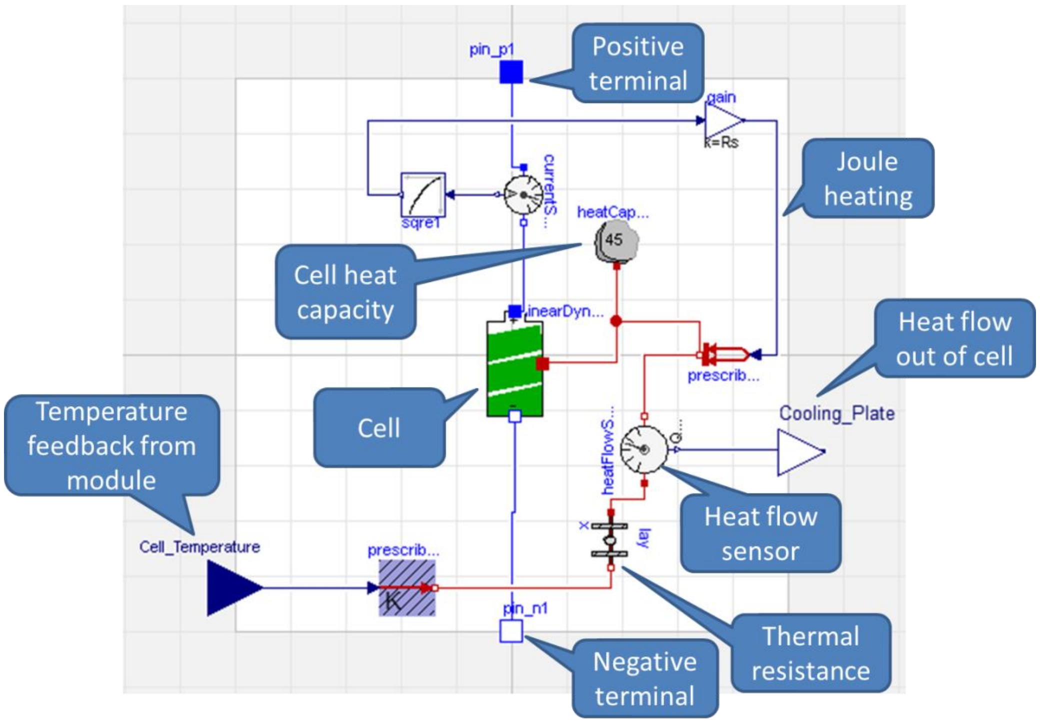


Figure 4.8: Single cell model interaction with the co-simulation engine

Due to the complex geometry of the battery module shown in Figure 4.9, the heat transfer model for the module and cooling plate is developed using FEM as discussed in Sections 4.1.2 and 4.1.3. The co-simulation approach used to communicate between the three domains is introduced in Section 4.1.4.

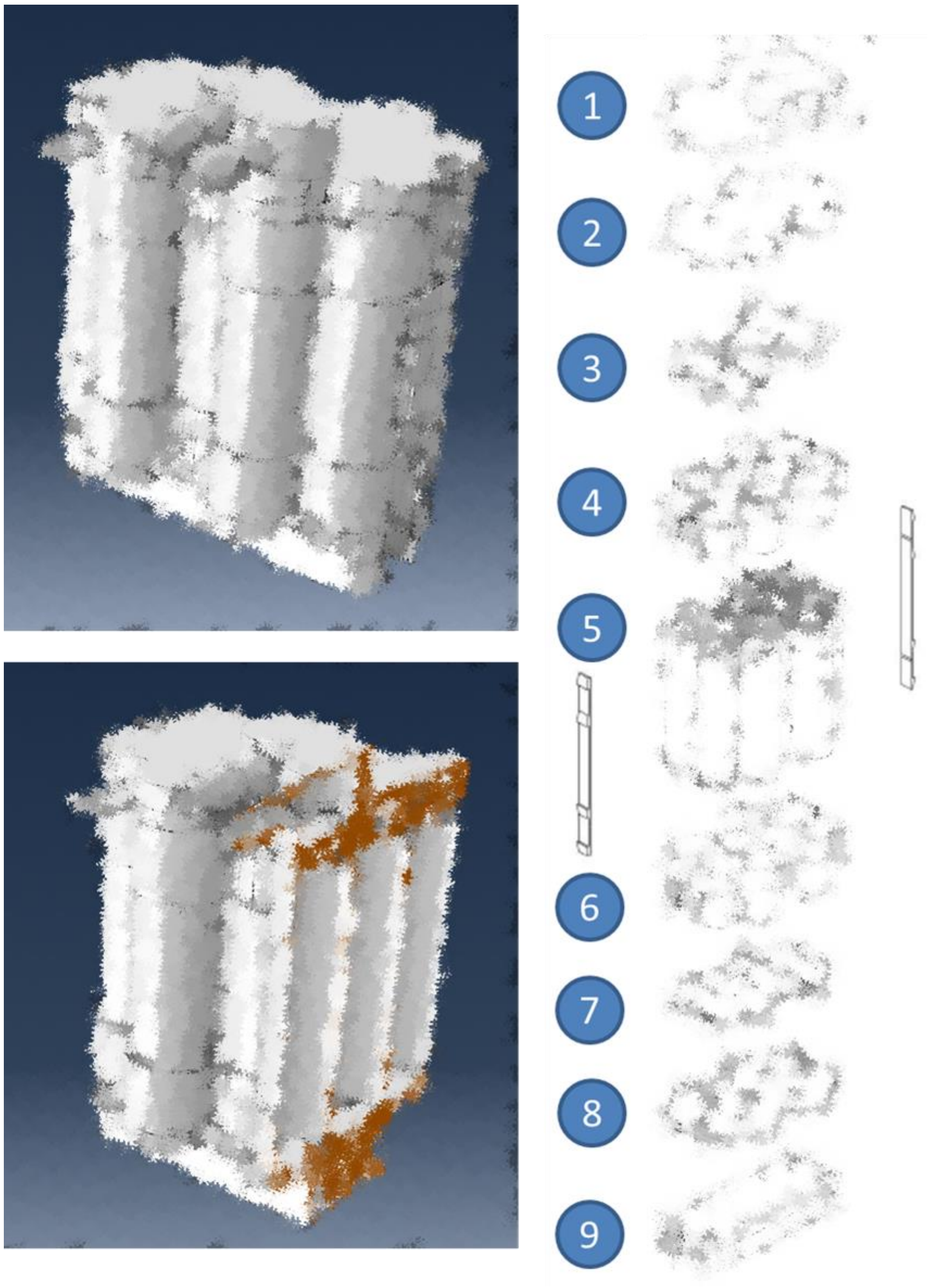


Figure 4.9: CAD drawing and assembly of the battery module. Module sections: 1: Cooling plate, 2&8: Cover, 3&7: Connecting plate, 4&6: Housing, 5: Cylindrical cells and clips, 9: Battery Management System (BMS) ** *Image obscured for proprietary reasons.*

4.1.2 Thermal-structural Domain

In order to create a high fidelity model of the battery module, a transient heat transfer model is also developed. The thermal properties of each component is assigned as shown in Table 4.3. Each part of the module was discretized on part using standard linear tetrahedron elements (DC3D4). Figure 4.10 provides an overview of the thermal-structural heat transfer model.

Table 4.3: Material properties of the thermal-structural model components

Sections	Conductivity $[\frac{\mathrm{W}}{\mathrm{m.K}}]$	Density $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	Specific Heat $[\frac{\mathrm{J}}{\mathrm{kg.K}}]$
Cooling Plate	167	2700	902
Cover, Housing, Cell, Clips, BMS cover	0.2	2000	1800
Connecting plate	400	8960	385
BMS circuit	149	2329	700

Under the assembly module a dependent instance was created containing all components and a surface of type geometry, named *INTERFACE*, was created using the internal surfaces of the cooling plate as shown in Figure 4.11 (marked in red). The surface is used to establish interaction with the fluid model. Temperature at the surface is defined as in output from the structural-thermal to the fluid model (applied as boundary condition in the fluid model). The surface was used to define a heat flux interaction with the fluid domain model developed in Section 4.1.3

In order to define heat transfer within the module, a conduction (contact) interaction property is defined as a function of clearance between surfaces, with conductance of 20000 $[W/m.K]$ with zero clearance and 0 conductance with 2mm clearance. Convection heat transfer is also defined as an interaction property of type film condition, with a film coefficient of $0.01[W/m.K]$. Convection is only defined on the external surface of the module.

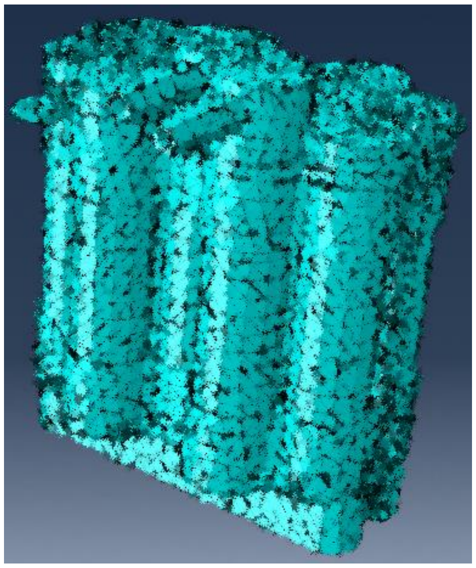


Figure 4.10: Thermal-Structural model ** *Image obscured for proprietary reasons.*

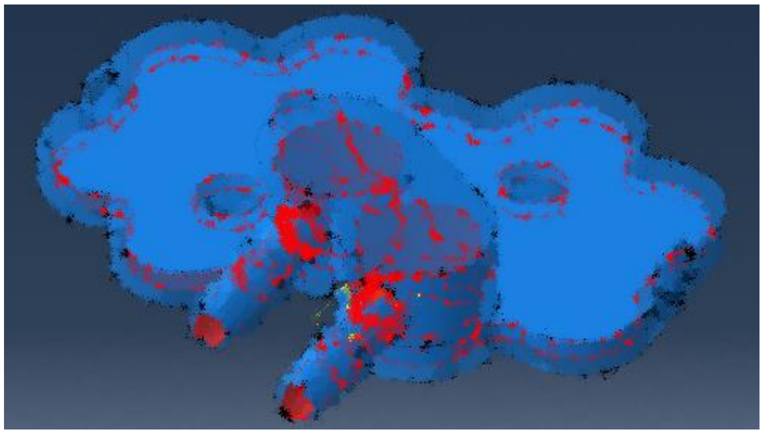


Figure 4.11: Fluid model interaction. Internal surface (marked in red) is identified as *INTERACTION* for interaction with fluid model** *Image obscured for proprietary reasons.*

Similar to interaction with the fluid model, an interface surface was created, named *INTERFACE2*, in order to establish communication with the electro-thermal model. Figure 4.12 displays a cut section of the thermal-structural model with the cell internal surfaces marked in red. Heat flux from the electro-thermal model, $Q_{cooling}$, is identified as an input to the thermal-structural model from the electro-thermal model. In turn, the surface temperature at *INTERFACE2* is reported back to the electro-thermal model.

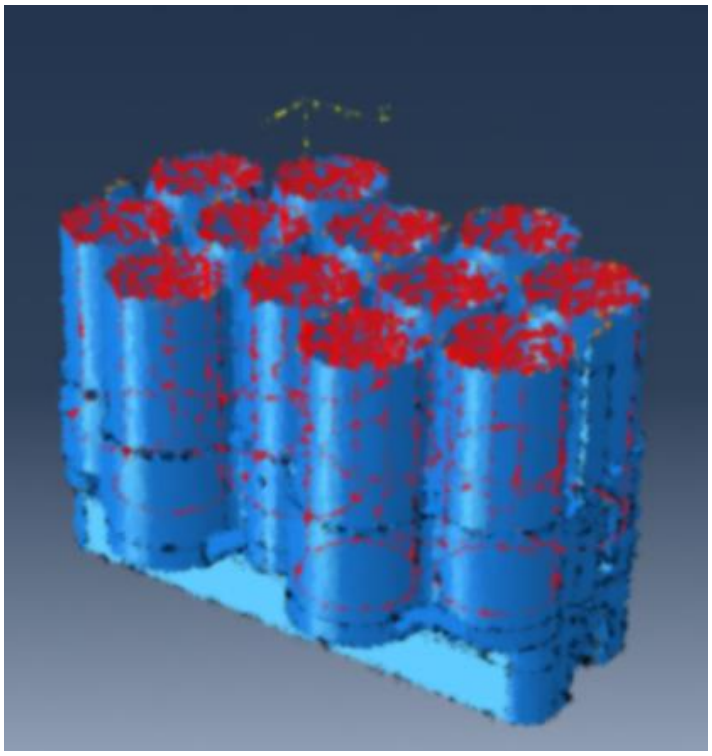


Figure 4.12: Electro-thermal model interaction. Internal surface of cells (marked in red) were identified as *INTERACTION2* for interaction with electro-thermal model ** *Image obscured for proprietary reasons.*

In order to ensure correct initialization of the analysis, a constant initial temperature is defined for the entire model at the initial time step. A transient step, named *Co-simulation*, of length $t_{simulation}$ is created with the maximum number of increments set at $t_{simulation} \times 50$ to ensure simulation is completed. A load of type *Body heat flux* is created and applied to all 12 cells representing input heat flux,$Q_{cooling}$ from the electro-thermal model. Finally an amplitude of type *Actuator*, named *BODYFLUX*, was defined and the amplitude of the load was updated.

In order to define the co-simulation interaction property between electro-thermal, thermal-structural, and fluid domain models, an input file, named *Standard.inp*, was generated using the thermal-structural heat transfer model created. Abaqus CAE interface does not allow the definition of the co-simulation interaction, thus the input file was edited as shown in Section 4.1.4.

4.1.3 Fluid Domain

In order to create a high fidelity model of the cooling fluid, a transient CFD model is developed. The thermal properties of the coolant, which is a mixture of 50% water and 50% Ethylene glycol, is assigned as shown in Table 4.4. The fluid model, which was generated to fill the void in the cooling plate part, was discretized on part using CFD linear tetrahedron elements (FC3D4), as shown in Figure 4.13.

Table 4.4: Coolant thermal properties

Fluid	Conductivity $[\frac{\mathrm{W}}{\mathrm{m.K}}]$	Density $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	Viscosity $[\frac{\mathrm{N.s}}{m^2}]$
Coolant	0.43	1056	0.0085

Under the assembly module a dependent instance was created and three surfaces of type mesh, named *INLET*, *OUTLET*, and *INTERFACE* were created as shown in Figure 4.14. The *INTERFACE* surface was used to define a heat flux interaction with the thermal-structural domain model as described in Section 4.1.2. The *INLET* surface was used to define the fluid inlet boundary conditions. Similarly, the outlet boundary conditions were applied to the *OUTLET* surface, as described in Table 4.5.

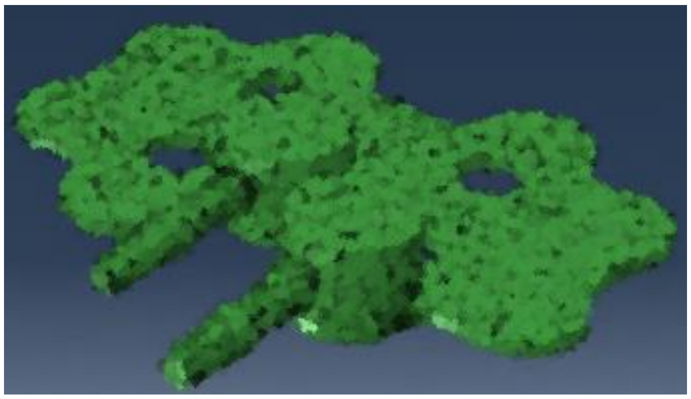


Figure 4.13: Fluid domain model (discretized) ** *Image obscured for proprietary reasons.*

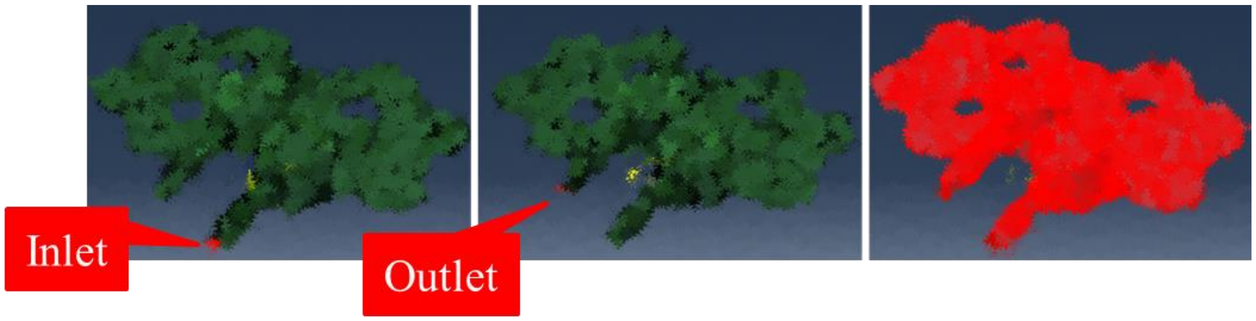


Figure 4.14: Fluid domain interactions (Marked in red). Left: Inlet, Middle: Outlet, Right: Interface ** *Image obscured for proprietary reasons.*

Table 4.5: Flow properties of the coolant

Boundary Region	Inlet	Outlet
Thermal properties	15 C^o	18 C^o
Flow properties	10 $\frac{Lit}{min}$	0 relative pressure

In order to ensure correct initialization of the analysis, a constant initial temperature is defined for the entire model at the initial time step. A transient step, named *Co-simulation*, of length $t_{simulation}$ (equal to duration of the thermal-structural heat transfer model) was created to ensure compatible simulation time between the co-simulation components.

Similar to the thermal-structural heat transfer model, in order to define the Co-simulation interaction property between electro-thermal, thermal-structural, and fluid domain models, an input file, named *Fluid.inp*, was generated using the CFD model created. The Abaqus CAE interface does not allow the definition of the co-simulation interaction, thus the input file was edited as shown in Section 4.1.4.

4.1.4 Co-simulation

Co-simulation between Abaqus/Standard (standard heat transfer model) and Abaqus/CFD (fluid domain model) is governed by an additional process, the SIMULIA Co-Simulation Engine (CSE) director. Typically, the CSE director is automatically invoked and the co-simulation parameters are stored in the co-simulation configuration file. The co-simulation as specified by the CSE director is illustrated in Figure 4.15. An overview of the Abaqus co-simulation procedure is described in Section 17.2.1 of the Abaqus User's Guide [28].

In order to execute co-simulation between models generated using Abaqus products and co-simulation format FMU files, the Abaqus command line should be used to invoke the CSE director and the co-simulation configuration file should be manually generated. Proper execution of co-simulation requires:

1) Proper definition of the interaction between the three models as outlined in the CSE User's Guide [29] and API reference Guide [30].

2) Compatible and consistent definition of co-simulation time and step size.

3) Definition of co-simulation parameters in the co-simulation configuration file as specified by the CSE director.

4) Definition of the co-simulation master and slave.

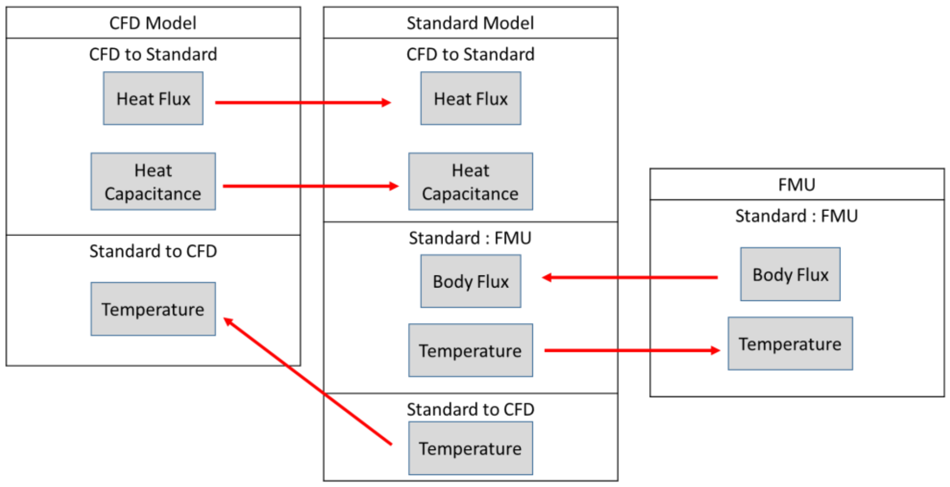


Figure 4.15: Co-simulation process

The CSE establishes communication between co-simulation models by assigning priority to individual models. The model designated as the master leads the co-simulation by taking the next time step, t_{next}, and results are communicated to slave models. The slave models will then individually take appropriate step sizes to advance to t_{next} and report the results the CSE director, which communicates the information to the master. The master will then take the next time step, t_{next+1}, and the co-simulation will continue, so on and such forth.

In preparing the Dymola model of the battery module for co-simulation, an FMU of type co-simulation and version 1.0 was generated. More information about generating an FMU using Dymola is available in the Dymola User's Guide [31]. The FMU file generated using Dymola is a compressed file which contains a Dynamic-link library (DLL) executable file as well as a model description file in XML (eXtensible Mark-up Language) format. The CSE director automatically assigns FMU model instances as slave models.

As stated previously in Section 4.1.2, definition of the interaction between the three co-simulation models required manually editing the thermal-structural model input file, *Standard.inp*. Lines, listed in Figure 4.16, were added to the input file within the definition of the transient time step in order to stablish proper communication between the standard heat transfer model and fluid model.

```
** STEP: Transient Step Name
...
...
...

** Interaction: Name
*Co-simulation, name=Name, controls=Name_Ctrls, program=MULTIPHYSICS
*Co-simulation Region, import, type=SURFACE
interface, CFL
interface, LUMPEDHEATCAPACITANCE
*Co-simulation Region, export, type=SURFACE
interface, NT
*Co-simulation Controls, name=Name_Ctrls, coupling scheme=gauss-seidel, time incrementation=subcycle, time
marks=yes,
step size=min, scheme modifier=lead
…
…
…
*End Step
```

Figure 4.16: Definition of co-simulation parameters (Abaqus Standard model). *Sections of the code referring to definition of boundary conditions and other step parameters are omitted to ensure generality and replaced with …

For the purpose of this analysis, communication between three models was defined using surface regions. For a complete description of fields available for exchange between co-simulation models interface regions, please refer to Section 17.2.1 of the Abaqus User's Guide [28]. As shown in Figure 4.15, the co-simulation controls, using the keyword *scheme modifier*, was used to assign the thermal-structural heat transfer model as the master. The keyword *incrementation*, was used to allow the master to independently take an appropriate step size.

Similar to the thermal-structural heat transfer model, definition of the interaction between the three co-simulation models required manually editing the thermal-structural model input file, *Fluid.inp*. Lines, listed in Figure 4.17, were added to the input file within the definition of the time step in order to stablish proper communication between the standard heat transfer model and fluid model.

```
** STEP: Fluid Step Name
...
...
...

** Interaction: Name
*Co-simulation, name=Name, controls=Name_Ctrls, program=MULTIPHYSICS
*Co-simulation Region, import, type=SURFACE
interface, TEMP
*Co-simulation Region, export, type=SURFACE
interface, HFL
interface, LUMPEDHEATCAPACITANCE
*Co-simulation Controls, name=Name_Ctrls, coupling scheme=gauss-seidel, time incrementation=lockstep, time
marks=yes,
step size=min, scheme modifier=lag
…
…
…
*End Step
```

Figure 4.17: Definition of co-simulation parameters (Abaqus Fluid model). *Sections of the code referring to definition of boundary conditions and other step parameters are omitted to ensure generality and replaced with …

The co-simulation controls, using the keyword *scheme modifier*, was used to assign the fluid model as a slave. The keyword *incrementation*, was used to force the slave to take appropriate step sizes and report the results at time t_{next}.

The co-simulation configuration file defines the simulation properties of the system and the numerical methods employed to simulate the system in its defined environment. This includes:

- **components**: subsystem simulator programs used in the co-simulation,
- **component instances**: subsystem simulations performed using identified simulators,
- **connectors**: available input and output simulation results from each subsystem,
- **connection sets**: pairing of subsystem simulation results,
- **connection categories**: time in the co-simulation when these pairings are relevant or active, and
- **model of computation**: numerical method used in the co-simulation.

In order for the configuration file to correctly identify the co-simulation parameters and be accepted by the CSE director, the following requirements must be satisfied:

1) XML well-formedness. The CSE director or any other XML authoring tool can confirm the well-formedness.
2) Abiding by the CSE schema definition. The rules and constraints are described formally in the CSE API kit.
3) Internal consistency of references.
4) Topological and algorithmic consistency. The definition of co-simulation and model parameters within the configuration file should be consistent with the problem definition in each subcomponent.
5) External consistency of references. The author of the configuration file should ensure consistency of names registered with the CSE director and each subcomponent.

The first four requirements can be verified by using the datacheck option. External consistency of references can only be verified at run time.

The configuration file contains six sections as described in the following paragraphs. The complete text of the configuration file created for the purpose of this analysis is include in the Appendix. For more details about the co-simulation configuration file refer to the CSE User's Guide and the CSE configuration schema documentation.

1) Document header

The first two lines, illustrated in Figure 4.18, in the file identify the document as an XML compliant and define the XML root element as <CoupledMultiphysicsSimulation>, which uniquely defines the document as relevant to the CSE. The next line segment will add the header. Various information can be included within the header however the only required information is the schema version.

```
<?xml version="1.0" encoding="UTF-8"?>
<CoupledMultiphysicsSimulation>
…
<header>
     <SchemaVersion>1.1</SchemaVersion>
  </header>
```

Figure 4.18: Co-simulation configuration file document header

2) Components

The components section of the configuration file defines each model included in the co-simulation (purple highlighting). As illustrated in Figure 4.19, using the keyword *<bottomUpImplementation>*, the XML file introduces co-simulation components that are developed within the SIMULIA package, and the phrase <topDownImplementation> identifies external components such as FMUs (green highlighting). In addition, a brief description of the master model, including the unit definitions (yellow highlighting) and co-simulation variables, will identify the master subcomponent to the CSE director. For example, the co-simulation parameters associated with the thermal-structural model, *BODYFLUX* and *TEMP*, are identified as well as their respective units. The CSE director can only recognize base units, therefore units such as kcal/kg and degC are further defined in regards to their respective base unit $\frac{m^2}{s^2}$ and K. For further details about the different implementation techniques refer to API User's kit [30].

```
<components>
    <component name="Abaqus/Standard">
      <bottomUpImplementation>
        <codeName>Abaqus/Standard</codeName>
        <modelDescription>
          <UnitDefinitions>
            <Unit name="kcal/kg">
              <BaseUnit factor="4186.8" m="2" s="-2"></BaseUnit>
            </Unit>
            <Unit name="degC">
              <BaseUnit K="1" factor="1.0" offset="273.15"></BaseUnit>
            </Unit>
          </UnitDefinitions>
          <ModelVariables>
            <ScalarVariable name="BODYFLUX">
              <Real unit="kcal/kg"></Real>
            </ScalarVariable>
            <ScalarVariable name="TEMP">
              <Real unit="degC"></Real>
            </ScalarVariable>
          </ModelVariables>
        </modelDescription>
      </bottomUpImplementation>
    </component>
    <component name="CFD">
      <bottomUpImplementation>
        <codeName>Abaqus/Cfd</codeName>
      </bottomUpImplementation>
    </component>
    <component name="BatteryCoSim_System">
      <topDownImplementation>
        <identifier>BatteryCoSim_System</identifier>
      </topDownImplementation>
    </component>
  </components>
```

Figure 4.19: Configuration file sections: Components

3) Component Instances

This section identifies the subsystem models participating in the co-simulation, highlighted by yellow, and associates these models with the simulation executables, highlighted purple, defined in the components section. Each co-simulation sub-component (Abaqus/Standard, CFD, and BatteryCoSim_System), can be used to identify *num* independent component instances which allows rapid and scalable model development. The XML text in Figure 4.20, *Fluid*, *Standard*, and *BatteryCoSim_System* instances are created using the *CFD*, *Abaqus/Standard*, and *BatteryCosim_System* subcomponents respectively.

```
…
<componentInstances>
    <componentInstance name="Fluid">
      <component>CFD</component>
      <timeIncrementation>
        <lockstep>false</lockstep>
      </timeIncrementation>
      <initialConditions>
        <sendBeforeReceive>false</sendBeforeReceive>
      </initialConditions>
    </componentInstance>
    <componentInstance name="Standard">
      <component>Abaqus/Standard</component>
      <timeIncrementation>
        <lockstep>false</lockstep>
      </timeIncrementation>
    </componentInstance>
    <componentInstance name="BatteryCoSim_System">
      <component>BatteryCoSim_System</component>
    </componentInstance>
  </componentInstances>
```

Figure 4.20: Configuration file sections: Component Instances

For the purpose of the co-simulation performed for this study, the two Abaqus simulation products each correspond to a single model (there is a one-to-one relation between the simulation executable and the simulation model). In a more general use, one-to-many relationships can be defined; for example, an Abaqus/CFD to Abaqus/Standard pairing of two separate models.

Using the initial conditions option and setting the *sendBeforeReceive* option to false, the Fluid model will not communicate its initial conditions until it receives the initial conditions from the master model (Standard). The default mode for the *sendBeforeReceive* option is true and therefore it is not included the Standard component instant definition.

4) Connectors

Connectors define the variables that enter or leave a component instance. The connector element provides the association between the component instance and the variable name and declares whether the variable is input or output. Text included in Figure 4.21 describes six connectors, highlighted in purple, and defines the model variables, highlighted in green, which stablish communication between the co-simulation components. The Connector named *CFD_to_STD_INPUT* defines the input variables to the Standard model instance while the connector named *CFD_to_STD_OUTPUT* assigns the same variable as outputs from the Fluid model instance. Table 4.6 includes the complete list of connectors as defined in the co-simulation configuration file.

The name attribute for the connector element is used for internal reference in the configuration file. For Abaqus simulation models and FMUs, there is no need to match the variables listed in input files with corresponding variables in the configuration file.

```
…
 <connectors>
    <connector name="CFD_to_STD_INPUT">
      <componentInstance>Standard</componentInstance>
      <variables>
        <input>
          <variable>heat_flux</variable>
          <variable>heat_capacitance</variable>
        </input>
      </variables>
    </connector>
    <connector name="CFD_to_STD_OUTPUT">
      <componentInstance>Fluid</componentInstance>
      <variables>
        <output>
          <variable>heat_flux</variable>
          <variable>heat_capacitance</variable>
        </output>
      </variables>
    </connector>
    <connector name="STD_to_CFD_INPUT">
      <componentInstance>Fluid</componentInstance>
      <variables>
        <input>
          <variable>temperature</variable>
        </input>
      </variables>
    </connector>
    <connector name="STD_to_CFD_OUTPUT">
      <componentInstance>Standard</componentInstance>
      <variables>
        <output>
          <variable>temperature</variable>
        </output>
      </variables>
    </connector>
    <connector name="STD_from_FMI">
      <componentInstance>Standard</componentInstance>
      <variables>
        <output>
          <variable>Cell_Temperature</variable>
        </output>
        <input>
          <variable>Cooling_Plate</variable>
        </input>
      </variables>
    </connector>
    <connector name="FMI_from_STD">
      <componentInstance>BatteryCoSim_System</componentInstance>
      <variables>
        <input>
          <variable>cell_temperature</variable>
        </input>
        <output>
          <variable>cooling_plate</variable>
        </output>
      </variables>
    </connector>
  </connectors>
```

Figure 4.21: Configuration file sections: Connectors

Table 4.6: Definition of Connectors in co-simulation configuration file

Connector Name	Model Instance	Connector Role	Variable(s)
CFD_to_STD_INPUT	Standard	Input	heat_flux heat_capacitance
CFD_to_STD_OUTPUT	Fluid	Output	heat_flux heat_capacitance
STD_to_CFD_INPUT	Fluid	Input	temperature
STD_to_CFD_OUTPUT	Standard	Output	temperature
STD_from_FMI	Standard	Output	Cell_Temperature
		Input	Cooling_Plate
FMI_from_STD	BatteryCoSim_System	Input	cell_temperature
		Output	cooling_plate

5) Connection Sets

Connection sets establish the associations between Connectors. As illustrated in Figure 4.22, pairing between all input and output variables is checked by the CSE director. In cases where connector elements have multiple variables defined, the connection elements must pair two connector elements that exactly complement each other in the sense that one connector's input variable count match the other's output variable count, and vice versa.

In the case of co-simulation performed for this analysis, three independent connection sets are defined. The first two connection sets, of type field, establish communication between the fluid and standard model. The first set defines the communication from the CFD to the Standard model, and the second set defines communication from the Standard to the CFD model. Type field connection ports will communicate both the variable magnitude as well as field units. The third connection set establishes communication of signals between the Standard model and the FMU.

```
…
<connectionSets>
    <connectionSet name="CFD_to_STD" type="FIELD">
      <connection>
        <connector>CFD_to_STD_INPUT</connector>
        <connector>CFD_to_STD_OUTPUT</connector>
      </connection>
    </connectionSet>
    <connectionSet name="STD_to_CFD" type="FIELD">
      <connection>
        <connector>STD_to_CFD_INPUT</connector>
        <connector>STD_to_CFD_OUTPUT</connector>
      </connection>
    </connectionSet>
    <connectionSet name="STD_and_FMI" type="SIGNAL">
      <connection>
        <connector>STD_from_FMI</connector>
        <connector>FMI_from_STD</connector>
      </connection>
    </connectionSet>
  </connectionSets>
```

Figure 4.22: Configuration file sections: Connection sets

6) Execution

The execution element describes all the details of the numerical method used to perform the co-simulation. As shown in Figure 4.23, execution is generalized to enable a hierarchical arrangement of an arbitrary number of component instances. The term *atomicActor*, highlighted in green, refers to each individual simulation participant in co-simulation; whereas, a *compositeactor*, highlighted in yellow, refers to a group of simulation participants. Model instance(s) defined within a single *compositeactor* will communicate using the assigned solver, highlighted in purple, and negotiation method, highlighted in gray.

In the case of the co-simulation performed for this analysis, the CSE uses the Gauss-Seidel algorithm and the CSE will negotiate the variable exchange time between participants by selecting the maximum of the time increments preferred by the three codes. Two connection categories, highlighted in blue, are defined, *InitialConditions* and *CouplingStep*. Each category defines the connection sets participating in the respective connection category. For more information about the algorithm types and exchange time negotiation options, please refer to the CSE configuration schema.

```
…
<execution>
    <compositeActors>
      <compositeActor name="twoCodeContinuousTime">
        <actors>
          <atomicActor>Standard</atomicActor>
          <atomicActor>Fluid</atomicActor>
          <atomicActor>BatteryCoSim_System</atomicActor>
        </actors>
        <modelOfComputation>
          <continuousTime>
            <algorithm>GAUSS-SEIDEL</algorithm>
            <negotiationMethod>MAX</negotiationMethod>
          </continuousTime>
        </modelOfComputation>
      </compositeActor>
    </compositeActors>
    <connectionGroups>
      <connectionCategory name="InitialConditions">
        <connectionSet>CFD_to_STD</connectionSet>
        <connectionSet>STD_to_CFD</connectionSet>
        <connectionSet>STD_and_FMI</connectionSet>
      </connectionCategory>
      <connectionCategory name="CouplingStep">
        <connectionSet>CFD_to_STD</connectionSet>
        <connectionSet>STD_to_CFD</connectionSet>
        <connectionSet>STD_and_FMI</connectionSet>
      </connectionCategory>
    </connectionGroups>
    <scenario>
      <duration>300.</duration>
    </scenario>
  </execution>
```

Figure 4.23: Configuration file sections: Execution

Finally, in order to perform the co-simulation, all input files (Abaqus input files and the FMU) as well as the co-simulation configuration file are stored in the current directory where the the Abaqus Command is issued. The commands, listed in Figure 4.24, will initiate the co-simulation on machine earth and communication with the CSE director is established through port 65533. For a complete list of available ports for co-simulation refer to the Abaqus User's Guide [28].

```
call abaqus cse -j Confige_File -config Confige_File.xml -listener 65533
call abaqus fmu -fmu Electrical_System.fmu -instance Electrical _System -csedirector earth:65533
call abaqus -j Standard -csedirector earth:65533
call abaqus -j Fluid -csedirector earth:65533
```

Figure 4.24: Co-simulation execution commands

After the first command prompt is used, the CSE director waits for the other co-simulation components, which can be invoked in an arbitrary order. During co-simulation, the CSE director generates individual log files for each co-simulation file, which can be used for debugging. A separate .msg file is also generated which provides a brief description of the co-simulation steps and possible errors encountered during co-simulation.

4.2 Calibration

The calibration of the battery model will be completed using experimental data in three phases: 1) Cell model calibration, 2) Battery module calibration, and 3) Battery pack calibration. The calibration of the cell model focuses on the calibration of the electro-thermal domain model. Cell parameters such internal resistance, OCV, and capacity are provided by the manufacturer. However, parameters such as cell thermal capacity, cell skin thermal conductance, temperature dependency of parameters such as internal resistance and capacity, and RC components are calibrated using experimental test results. Calibrating the battery module and battery pack models require further testing of the battery module and battery pack, which will be completed in the next phase of the research partnership.

In order to calibrate the cell model, a single battery cell was tested under high current charge and discharge conditions and experimental results were provided. For the purposes of the discharge test, the ambient temperature was at 23 ± 2^{o}C. The cell was initially charged fully. Afterwards, the sample was paused for 1 hour. The test cell was placed under a thermal camera and the device under testing (DUT) was discharged with a current of 40 A until the voltage dropped below the minimum cell voltage, or until the thermal camera detected a temperature of over 80^{o}C. For the purpose of the calibration, the distribution of temperature was constant over the surface.

During discharge the cell is discharged with a current of 40A and after 210 seconds the cell reaches 80^{o}C, at which point the discharge is stopped. In order to calibrate the model, system parameters were adjusted until good agreement between the cell temperature experimental results and the model output temperature was reached. Figure 4.25 illustrates the cell skin temperature as recorded by the thermal camera and the cell model temperature.

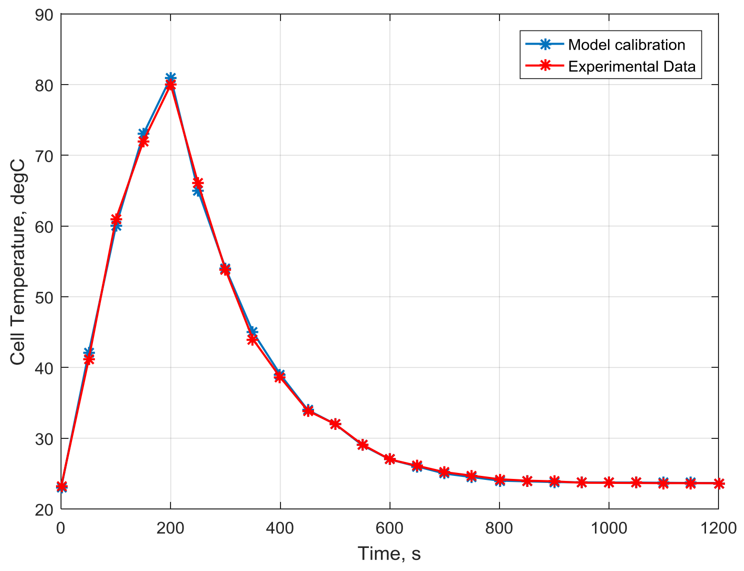


Figure 4.25: Cell calibration, discharge of fully charged cell

In order to test the effectiveness of the model calibration, the calibrated model was simulated at a discharge rate of 40 A. Figure 4.26 illustrates good agreement between the model predicted and experimental normalized cell voltage. During the cell discharge (first 210 seconds), the model predicted voltage drops at a slower pace compared to the experimental results. The cell electro-thermal domain model dynamic in fast discharge is governed by the cell capacity and internal resistance, both of which are defined as functions of temperature and calendric aging. The time constant of the cell thermal domain model (governed by the thermal capacity and conductance) is much larger compared to the electro-thermal domain. Therefore the slow change in temperature, and in turn cell internal resistance, results in the discrepancy between the model prediction and experimental results. After the discharge is completed, both experimental results and model-predicted results illustrate a fast voltage recovery (from 210 to 400 seconds). However, the model prediction fails to capture the slow continued voltage recovery beyond 400 seconds. Redefining the cell internal resistance and capacity as functions of current, temperature, and calendric aging may reduce the discrepancy between model prediction and experimental results.

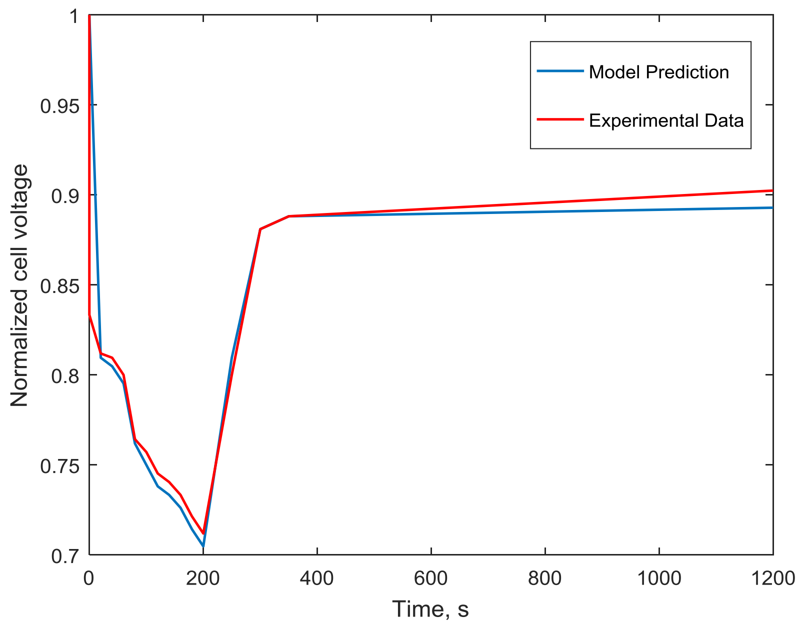


Figure 4.26: Verification of model calibration, normalized cell voltage - Discharge

Similar to the discharge test, during the charge test, the ambient temperature was at $23{\pm}2^{o}$C and the cell was discharged with a current of 2.6 A until the cell voltage dropped below the minimum cell voltage. After a 1 hour pause period the cell was charged with a maximum current of 30A for one hour or until the thermal camera detected a temperature of 80^{o}C.

In order to calibrate the model, system parameters were adjusted until good agreement between the cell temperature experimental results and the model output temperature was reached. Figure 4.27 illustrates the cell skin temperature as recorded by the thermal camera and the cell model temperature.

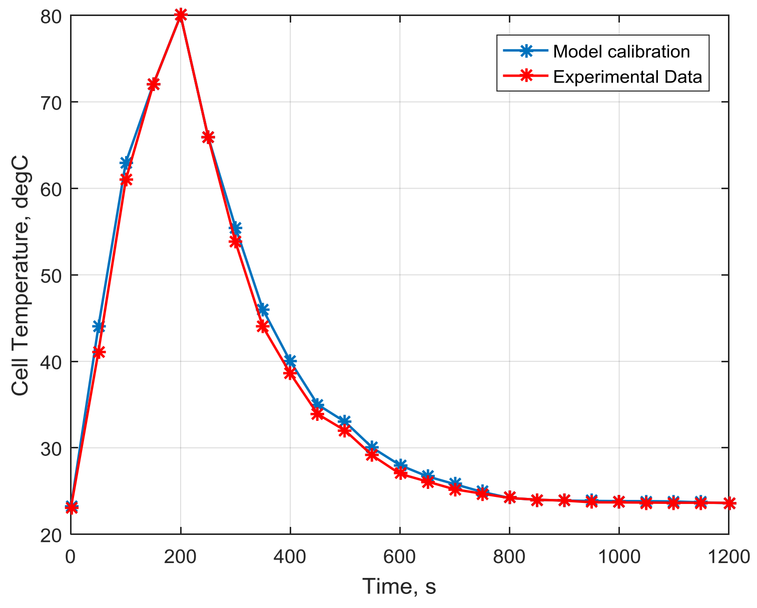


Figure 4.27: Cell calibration, Charging

Figures 4.27 and 4.28 illustrate a comparison between the model predicted and experimental current and normalized cell voltage respectively.

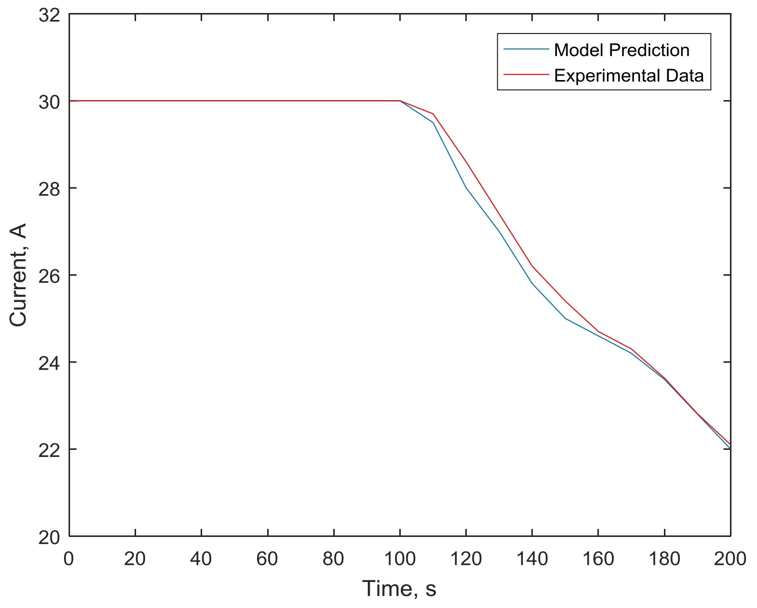


Figure 4.28: Cell current during charging

During the first 100 seconds of cell charging, both experimental and model-predicted discharge current is at the maximum allowed limit of 30 A. After the first 100 seconds, the charge current decreases rapidly until at 200 seconds the cell temperature reaches 80^{o}C. Figures 4.28 and 4.29 illustrate good agreement between the experimental results and model prediction.

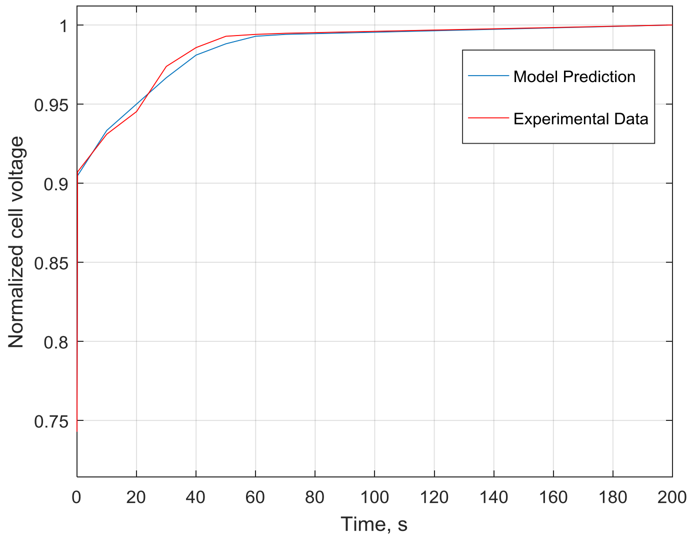


Figure 4.29: Cell normalized voltage during charging

After the first 100 seconds the cell model predicts a more rapid decline in charge current than experimental results. Similar to the discharge test, this discrepancy may be corrected by redefining the cell internal resistance and capacity as a function of charge current.

4.3 Battery module co-simulation

A battery module, following the model development and co-simulation execution procedure outlined in section 4.1, and using the calibrated cell parameters as described in 4.2, was created. Results presented in this section display the capability to identify hot spots and predict the system dynamics; however, the module has not yet been calibrated and thus predicted results are only of qualitative value. System parameters such as fluid film coefficient and thermal contact conductance coefficient were extracted from reference text [32]. Calibration of the battery module model using experimental data will be completed in the next phase of the research project.

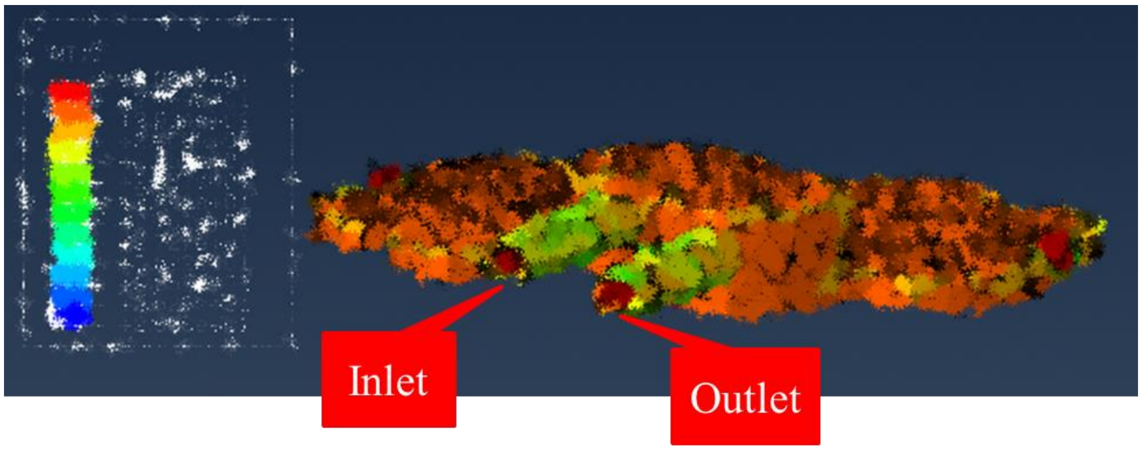


Figure 4.30: Cooling plate temperature distribution. Temperature contour displays areas of high temperature in red and low temperature in blue ** *Image obscured for proprietary reasons.*

Figure 4.30 illustrates the temperature distribution on the cooling plate. The initial module temperature was set at 30 C^0. The cooling fluid temperature at the inlet was fixed at 15 C^0 and the fluid temperature at the outlet is presented in Figure 4.31.

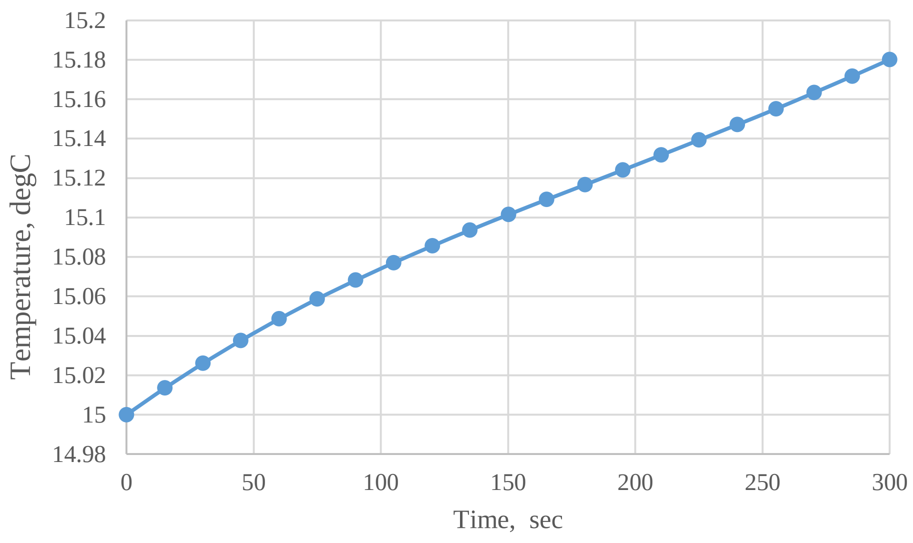


Figure 4.31: Coolant outflow temperature.

Similar to the cooling plate, hot spots can be identified on the cells as shown in Figure 4.32. The minimum cell temperature occurs at the contact points with the cooling plate at the top and bottom of each cell, while the heat loss through convection reduces the cell skin temperature on all sides. Maximum temperature regions, marked in red, occur at the top and bottom of the cell due to reduced heat transfer with the cooling plate and minimal heat loss through convection.

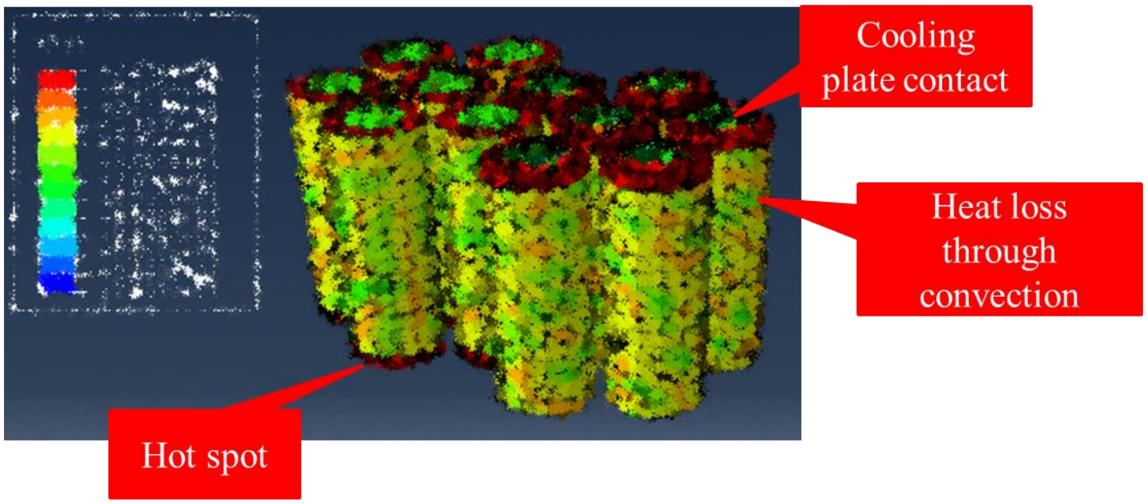


Figure 4.32: Cell temperature distribution. Temperature contour displays areas of high temperature in red and low temperature in blue. ** *Image obscured for proprietary reasons.*

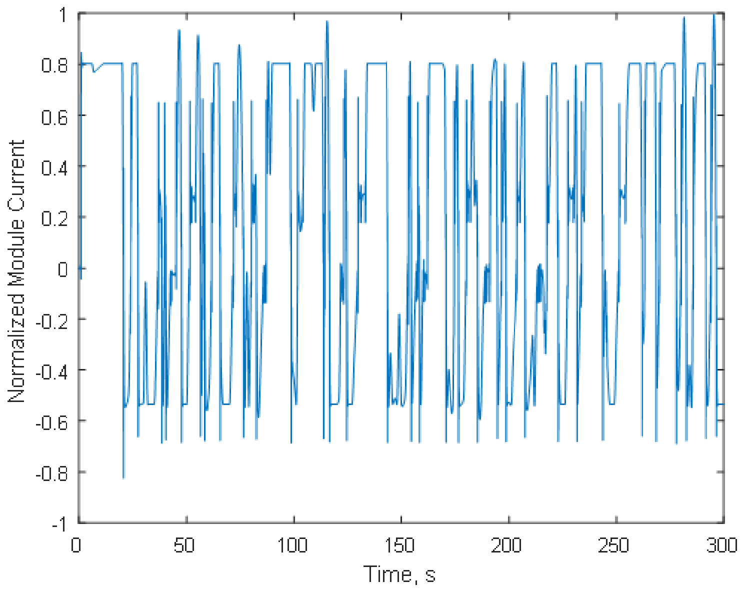


Figure 4.33: Normalized module current

In addition to the temperature distribution in the thermal-structural model, the electro-thermal model output is also calculated. Figure 4.33 displays the normalized module current operating under a high-frequency-high-load power cycle. Similarly, Figure 4.34 illustrates the module voltage during the co-simulation.

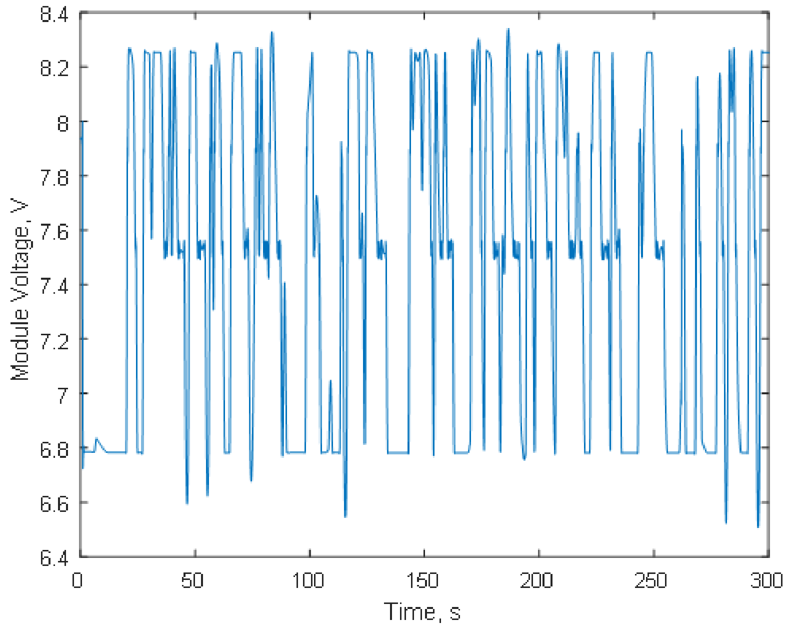


Figure 4.34: Module voltage under high-frequency-high-load conditions

4.4 Performance under high loading conditions

Performance of the battery pack under high loading conditions is of great concern in the development of high-performance HEVs. The current draw from the battery pack can be limited; however, extended high current draw can result in development of local high temperature regions which can significantly reduce the battery performance and battery life. The purpose of this study is to estimate the effects of high current draw on battery life and ensure the cooling system can efficiently remove heat and eliminate high temperature regions.

For the purpose of this study the model will undergo two types of loading conditions. In order to study the long term effects on battery life, the model will be executed with loading cycles that represent the long term use of the battery pack. In order to study the temperature profile in the battery pack, the loading cycle will feature long periods with maximum current draw. In order to complete this analysis, which will be completed in the next phase of research, a model of the battery pack to include hundreds of battery modules will be developed. This model will also include control logic that will monitor the temperature of the battery pack at predetermined locations and limit the current draw to prevent thermal or electrical damage to the battery pack.

The development of the battery pack model will also require further testing of an individual battery cell to better define the fast and slow transient behavior of the cell. The transient behavior is represented by several RC elements included in the cell model.

4.5 Battery life estimation

In order to study the battery life and characterize the battery aging, three battery parameters are studied. Battery current draw/ charging current, temperature and SOC significantly influence the battery life. A 2.5 year study documented in [32] has shown that cell temperature, depth-of-discharge, and current rates are the primary factors influencing battery state of health and its useful life.

The cell model included in the EES library has built-in functionality to estimate the battery life. The capacity, as well as the internal impedance of the cell, can change due to aging. The effects of aging on the battery state of health are identified by the battery state of health (SOH).

$$SOH = SOH_C \times SOH_Z \tag{42}$$

For each cell, SOH_C and SOH_Z are estimated by calibrating aging factors, x_c and x_Z, respectively [27]. Characterizing the battery life and the effects of high current draw and charging will be investigated in the next steps of the study.

$$SOH_C = \frac{1}{C_0(1 - x_C)} \times C - \frac{x_C}{1 - x_C} \tag{43}$$

$$SOH_Z = \frac{1}{Z_0(1 - x_Z)} \times Z - \frac{x_Z}{1 - x_Z} \tag{44}$$

4.6 High and low ambient temperature testing

Vehicle manufactures are required to ensure the vehicle can perform in both high and low ambient temperature conditions. A significant area of interest is performance in high ambient temperatures as the high ambient temperature can reduce the ability of the cooling system to remove heat from the battery pack, and additional cooling may be required in order to prevent long term damage to the battery pack.

Each battery cell model is temperature dependent and therefore the effects of ambient temperature will be investigated in later stages of the analysis. In order to ensure the safe operation of the battery pack, it is highly encouraged that each battery pack design is tested before final integration in the vehicle.

4.7 Vehicle integration

The final stage of this project will include the integration of the battery pack model into the vehicle model. The battery pack model developed in this chapter is an example of a high fidelity multi-physics model which can take supersede models of lower fidelity. The development of high fidelity models within a single modeling environment, such as Matlab or

Simulink, is often unfeasible, or at best can lead to a significant increase in the computational costs as many functional components of the vehicle feature multi-physics systems. It should also be noted that the development of finite element model of system components in junction with equation-based models and rule-based control is often impossible within a single modeling environment. Therefore the use of co-simulation in the development of subcomponent models and integration with the vehicle model can lead to a significant increase in model fidelity and reduced computational cost. It is highly encouraged that the battery pack model developed as part of this analysis be integrated with the vehicle model after model verification and development of the complete vehicle model.

Chapter 5: Concluding Remarks and Future Work

This thesis presented simulation tools, control analysis, and a multi-physics co-simulation approach supporting high-performance HEVs. It focused on the development of vehicle simulation and control of HEVs using a sequential approach, as well as a multi-physics modeling approach to develop high fidelity battery models. This chapter will provide concluding remarks for the thesis and suggest future work.

5.1 Adaptation of Vehicle Model to Simulate and Control Hybrid Electric Vehicles

A sequential modeling approach was introduced in Chapter 2 in order to develop a vehicle model for a series HEV. The same modeling approach can be adapted to develop vehicle supervisory control strategies for parallel and power-split HEVs.

The supervisory control strategy applied to the vehicle model in Chapter 2 was a rule-based strategy which provides a good compromise between vehicle performance and efficiency for series hybrid architectures. Parallel and power-split hybrid architectures, however, require the application of more complex supervisory control strategies such Equivalent Cost Minimization Strategy (ECMS), which assigns an equivalent fuel for battery power. Application of ECMS can be further enhanced by utilizing DP results to choose and update the equivalence factor.

In Chapter 3, DP and its application to a series hybrid architecture was introduced. DP is a Backward-Looking simulation which can be used as an unbiased tool to evaluate the controller performance. Applying DP to hybrid architectures and comparing the vehicle emissions and performance against the vehicle model can provide further insight in the development of the vehicle supervisory controller.

In both Chapters 2 and 3 the controller performance was evaluated based on fuel economy and CO_2 emissions. While the fuel economy and CO_2 emissions are of great importance in the design of vehicles, manufacturers require each new vehicle design to meet a long list of design goals, such as emission, drivability, and performance. One area for future work is to develop a robust supervisory control that can simultaneously minimize fuel economy in normal city driving conditions and maximize vehicle performance. Another possible improvement is to investigate the use of traffic light and congestion data, to interactively influence driver behavior and improve the fuel economy.

5.2 Battery Modeling and Multi-Physics Co-simulation

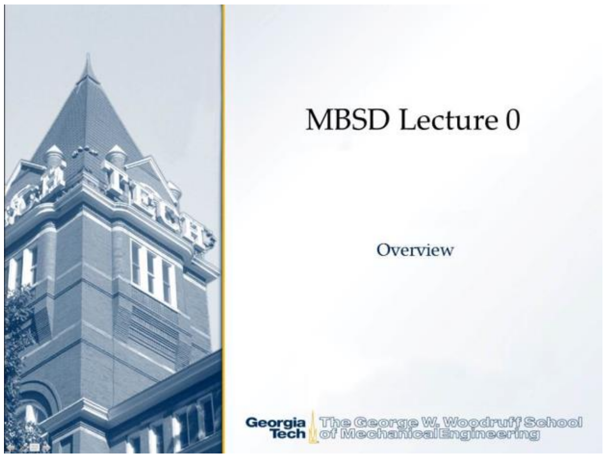
In Chapter 4 a multi-physics co-simulation approach was used to develop a high fidelity battery model in order to investigate the temperature distribution in the spatial framework of the battery pack. The co-simulation approach provides a fast computational time by incorporating a discrete electro-thermal domain model developed using Modelica, an open-source, object-oriented, and equation-based modeling language. The spatial context of the heat transfer problem was captured by the Finite Element Model (FEM) of battery components including a transient thermal-structural heat transfer model and a fluid heat transfer model of the cooling system.

Communication between the three model components was established following the Functional Mock-up Interface (FMI), a tool-independent standard to support model exchange and co-simulation of dynamic models. FMI allowed for the coupling of the discrete electro-thermal domain model and the FEM heat transfer models to develop a modular coupled model. Data exchange is established at discrete communication points, as specified by the host simulation tool, and each subsystem is solved independently between the communication points.

The battery model developed will be used in future studies to investigate the effects of high current charge/discharge in high performance applications to ensure safe operation of the battery pack. The model will also be used to design the cooling system and study the effects of high current on the battery life. Another area for future work is to incorporate the battery model into a vehicle model. Simulink software used to develop the vehicle model in Chapter 2 is an FMI compliant software which allows for the integration of the battery pack model. The high fidelity multi-physics models can replace the battery model developed using experimental results. In addition to improved fidelity, the model can be used to identify the areas of high temperature, design the cooling system to allow for proper cooling, investigate the effects of current draw on life, and in turn observe the effects of battery inefficacy on vehicle performance.

Appendix:

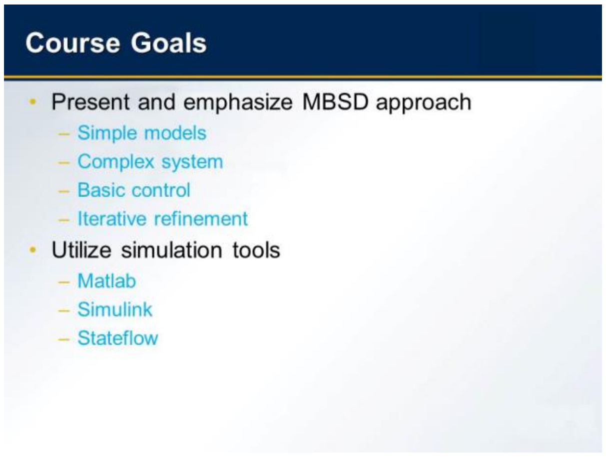
1) MBSD Tutorial



MBSD Lecture 0

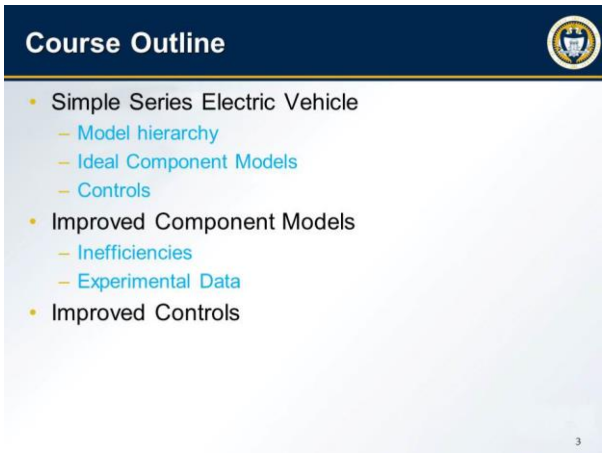
Overview

Georgia Tech | The George W. Woodruff School of Mechanical Engineering



Course Goals

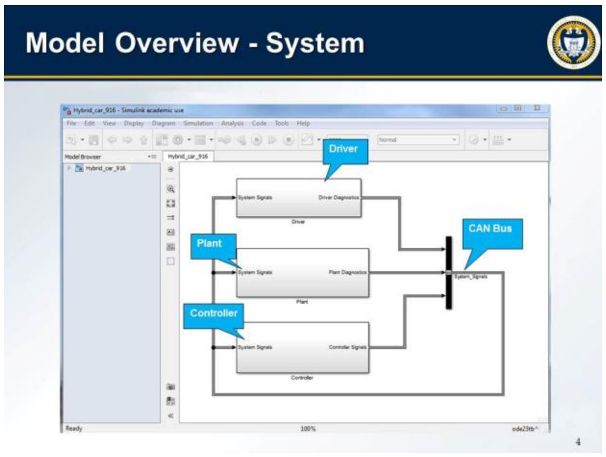
- Present and emphasize MBSD approach
  - Simple models
  - Complex system
  - Basic control
  - Iterative refinement
- Utilize simulation tools
  - Matlab
  - Simulink
  - Stateflow



Course Outline

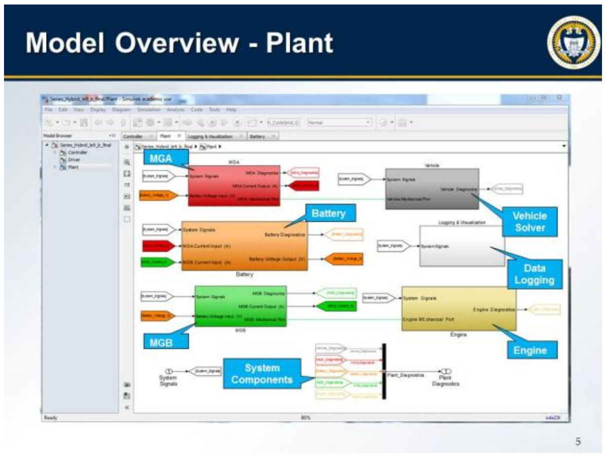
- Simple Series Electric Vehicle
  - Model hierarchy
  - Ideal Component Models
  - Controls
- Improved Component Models
  - Inefficiencies
  - Experimental Data
- Improved Controls

3



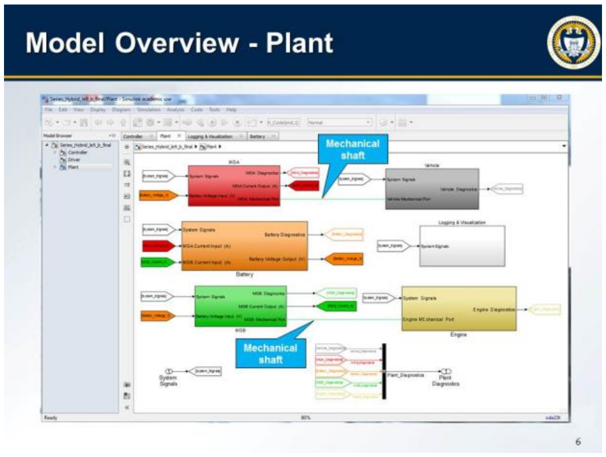
Model Overview - System

4



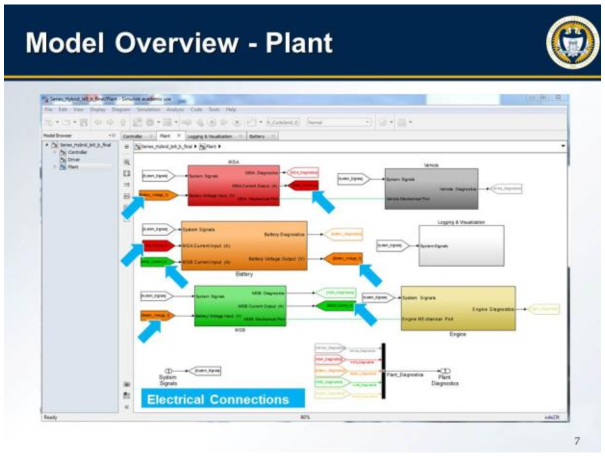
Model Overview - Plant

5



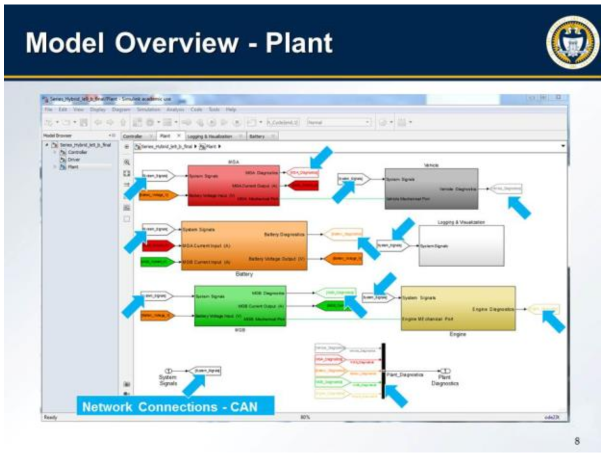
Model Overview - Plant

6



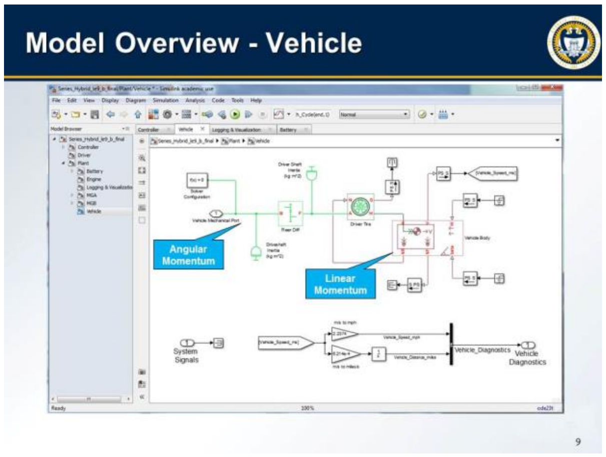
Model Overview - Plant

7



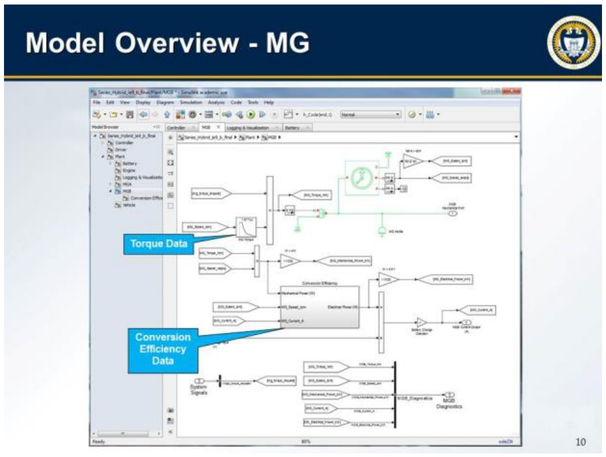
Model Overview - Plant

8



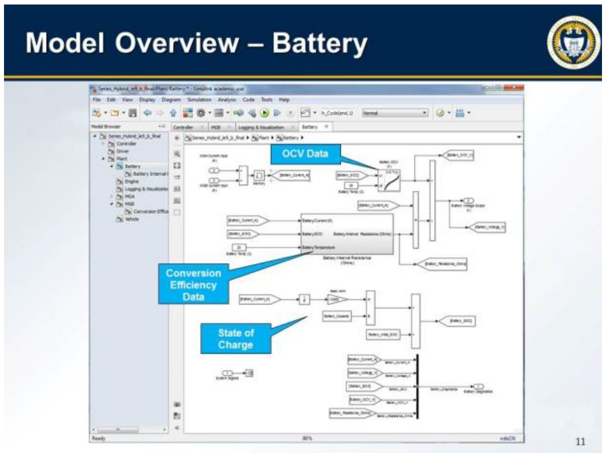
Model Overview - Vehicle

9



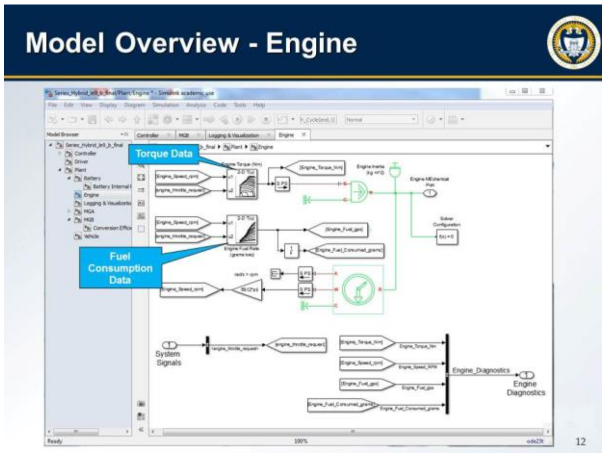
Model Overview - MG

10



Model Overview – Battery

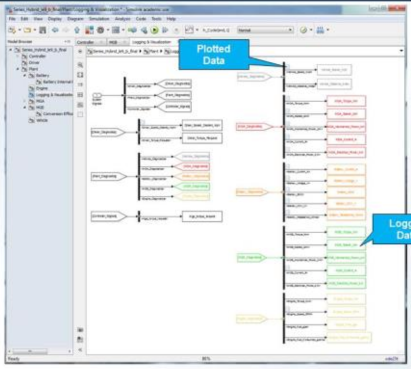
11



Model Overview - Engine

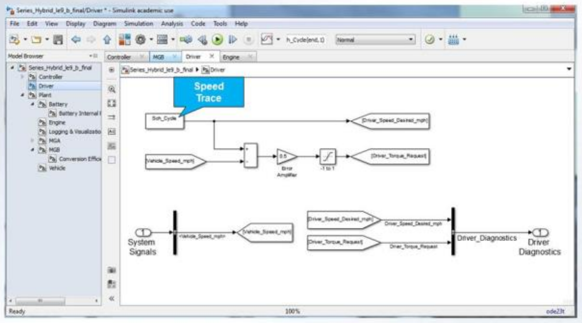
12

Model Overview - Logging



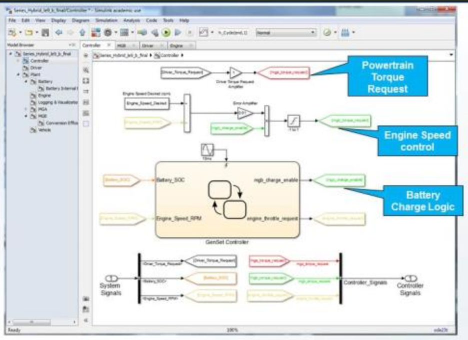
13

Model Overview - Driver



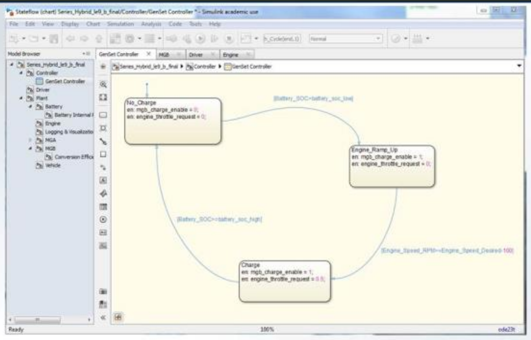
14

Model Overview - Controller



15

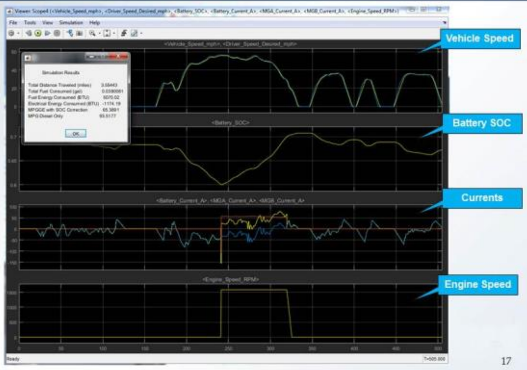
Model Overview - GentSet


Stateflow (chart) Series_Hybrid_le9_b_final/Controller/GenSet Controller * - Simulink academic use

```
No_Charge
en: mgb_charge_enable = 0;
en: engine_throttle_request = 0;

[Battery_SOC<battery_soc_low]

Engine_Ramp_Up
en: mgb_charge_enable = 1;
en: engine_throttle_request = 0;

[Engine_Speed_RPM>=Engine_Speed_Desired-100]

Charge
en: mgb_charge_enable = 1;
en: engine_throttle_request = 0.8;

[Battery_SOC>=battery_soc_high]
```

16

Results

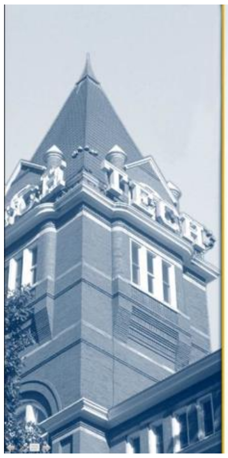


Simulation Results

Total Distance Traveled (miles)	3.58443
Total Fuel Consumed (gal)	0.0380061
Fuel Energy Consumed (BTU)	5070.02
Electrical Energy Consumed (BTU)	-1174.19
MPGGE with SOC Correction	65.3891
MPG Diesel Only	93.5177

OK

17



MBSD Lecture 1

Getting Started with the High Level System

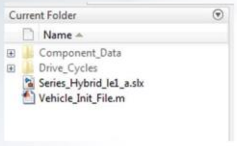
Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Lecture Goals

- Getting Started with Simulink
- Developing model hierarchy (subsystems)
- Establishing CAN bus

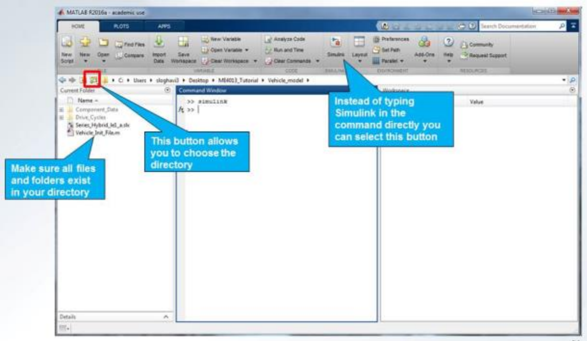
Getting Started

- Start Matlab
- Make sure you are in the correct directory

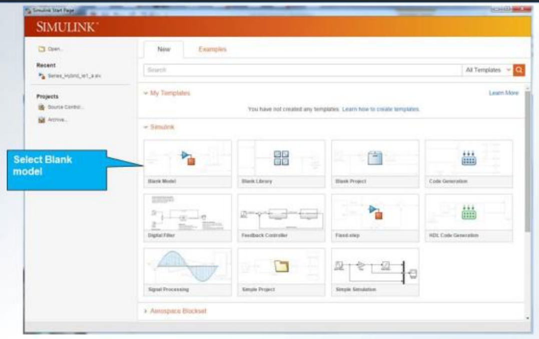


- Type **simulink** in the command window

Getting Started

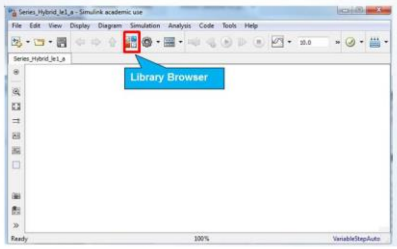

Make sure all files and folders exist in your directory
This button allows you to choose the directory
Instead of typing Simulink in the command directly you can select this button
>> simulink

Getting Started



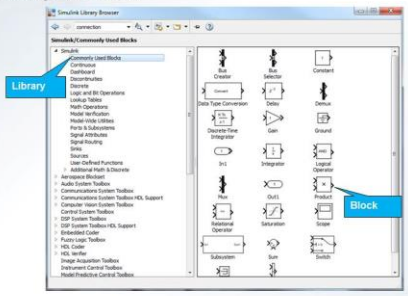
Getting Started

- Simulink is a graphical programming environment which enables you to link function blocks to create simulations
- In the Simulink model window, select library browser


Series_Hybrid_le1_a - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Library Browser
Ready
100%
VariableStepAuto

Getting Started

- Get yourself familiar with the libraries and blocks in the library browser



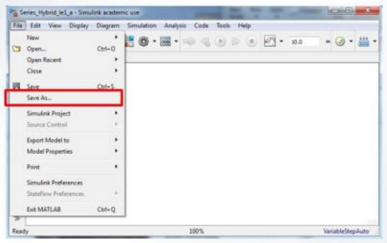
Getting Started

- We will primarily use the following libraries
  - Simulink \ Commonly Used Blocks
  - Simulink \ Signal Routing
  - Simulink \ Lookup Tables
  - Simulink \ Sinks
  - Simulink \ Sources
  - Simscape \ Driveline
  - Stateflow

25

Getting Started

- Save the Simulink model as :
  - Series_Hybrid_le1_a
- Do not use spaces or variable names in file names



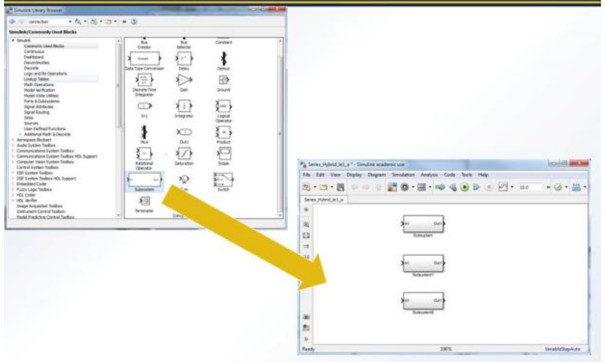
26

High Level Model

- We will first create the high level plant and controller model
- From
  - Simulink \ Commonly Used Blocks
- Drag three Subsystem blocks into the model
  - Driver block
  - Controller block
  - Vehicle block

27

High Level Model



28

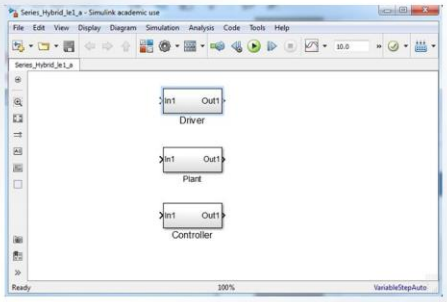
High Level Model

- Rename the upper subsystem
  - Driver
- The middle subsystem
  - Plant
- The lower subsystem
  - Controller

29

High Level Model

- You can change the font size by right clicking on each subsystem/Format/Font Style…


Series_Hybrid_le1_a - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Series_Hybrid_le1_a
In1 Out1
Driver
In1 Out1
Plant
In1 Out1
Controller
Ready
100%
VariableStepAuto

30

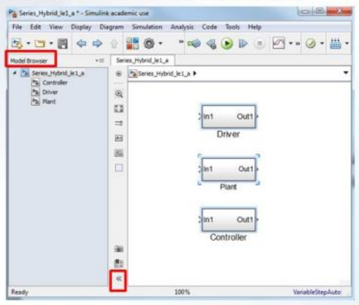
High Level Model

- At this high level, signals will be sent out by the Driver, Plant, and Controller
  - Driver Torque Request
  - Battery State of Charge
  - Engine Throttle Request
- These signals will be put onto a common bus so that all three subsystems can receive all signals
  - Only desired signals will be selected

31

High Level Model

- Open Model Browser by selecting the arrow
  - Shows an outline form of the model


Series_Hybrid_le1_a * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
Series_Hybrid_le1_a
Controller
Driver
Plant
In1 Out1
Driver
Plant
Controller
Ready
100%
VariableStepAuto

32

Driver Subsystem

- The Driver receives information
  - Speedometer
  - Tachometer
  - Idiot lights (engine hot, …..)
- The Driver sends information
  - Torque request (gas/brake pedal position)
  - Gear selection

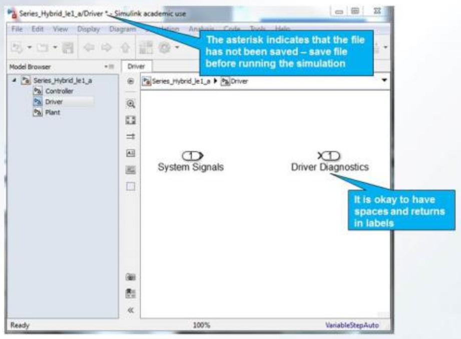
33

Driver Subsystem

- Open the Driver subsystem by:
  - clicking on Driver in Model Browser or
  - double click on the Driver block
- Delete the wire that connects the In1 block to out1 block
- Rename
  - In1 to System Signals
  - Out1 to Driver Diagnostics

34

Driver Subsystem


Series_Hybrid_le1_a/Driver * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
Driver
Series_Hybrid_le1_a
Controller
Driver
Plant
System Signals
Driver Diagnostics
Ready
100%
VariableStepAuto

The asterisk indicates that the file has not been saved – save file before running the simulation

It is okay to have spaces and returns in labels

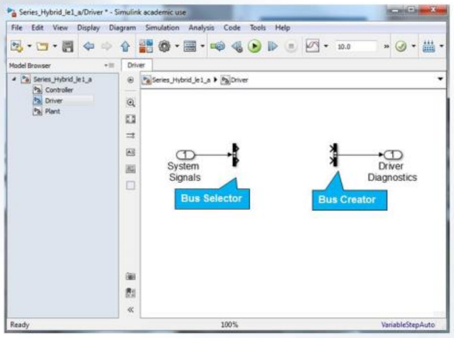
35

Driver Subsystem - CAN

- From
  - Simulink \ Commonly Used Blocks
- Drag in a Bus Selector and a Bus Creator
  - The Bus Creator puts signals on the bus
  - The Bus Selector extracts them from the bus
- Connect the System Signals input to the Bus Selector
- Connect the Driver Signals input to the Bus Creator

36

Driver Subsystem - CAN



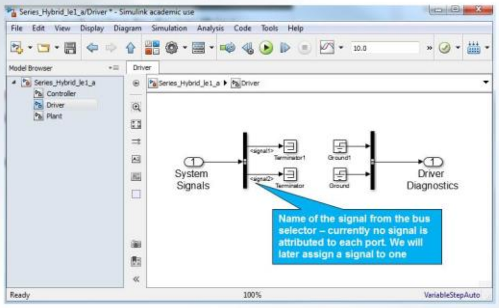
37

Driver Subsystem - CAN

- For now, the Driver is not going to do anything
- From
  - Simulink \ Commonly Used Blocks
- Drag in two Terminators and two Grounds
- Connect the two outputs of the Bus Selector to the Terminators
- Connect the two inputs of the Bus Selector to the Grounds

38

Driver Subsystem - CAN



Name of the signal from the bus selector – currently no signal is attributed to each port. We will later assign a signal to one

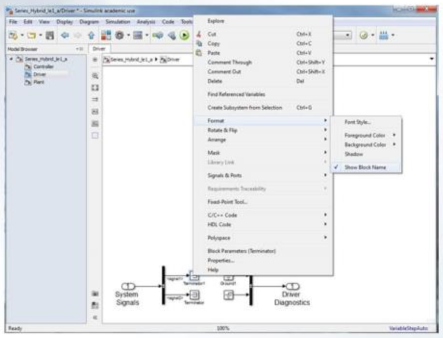
39

Driver Subsystem - CAN

- The symbols for the ground and terminator are pretty obvious – there is no need for them to be labeled
- On one of the Terminators
  - Right Click
  - Select Format
  - Deselect Show Block Name
- We will do this for many of the blocks
- Hide the name of all terminator and ground blocks

40

Driver Subsystem - CAN



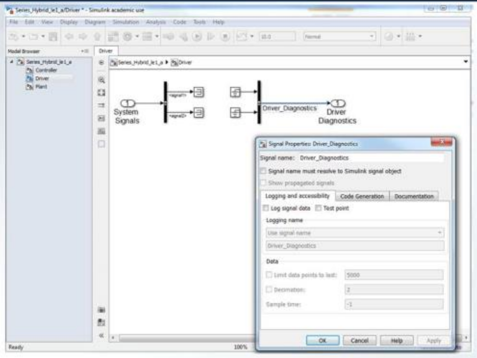
41

Driver Subsystem - CAN

- While no signals are being sent by the Driver, the Driver Diagnostics signal should be identified on the bus
- On the wire connecting the Bus Creator to the Driver Diagnostics output
  - Right Click
  - Select Properties
  - For the Signal name enter Driver_Diagnostics
  - Click OK

42

Driver Subsystem - CAN



Signal Properties: Driver_Diagnostics

Signal name: Driver_Diagnostics

☐ Signal name must resolve to Simulink signal object

☐ Show propagated signals

Logging and accessibility | Code Generation | Documentation

☐ Log signal data ☐ Test point

Logging name: Use signal name — Driver_Diagnostics

Data

☐ Limit data points to last: 5000

☐ Decimation: 2

Sample time: -1

OK Cancel Help Apply

43

Driver Subsystem - CAN

- Move the blocks around, enlarge if necessary, and make things look nice

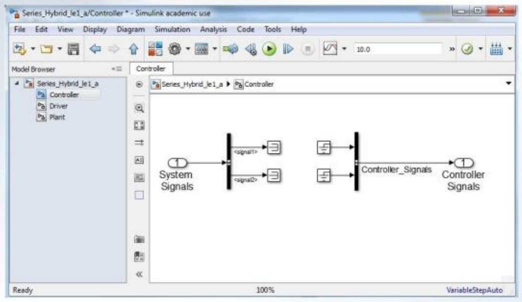
44

Controller Subsystem

- For now, the controller is not going to do anything
- Repeat everything done to the Driver subsystem for the Controller subsystem
  - Rename the Out1 to Controller Signals
  - Rename the CAN signal to Controller_Signals

45

Controller Subsystem



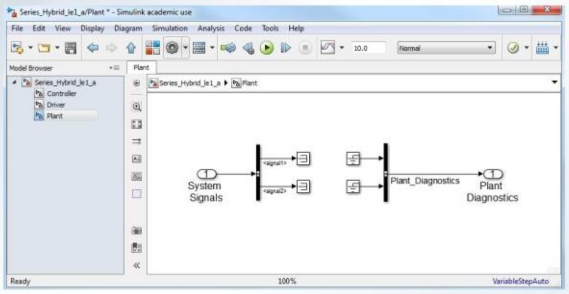
46

Plant Subsystem

- For now, the Plant is not going to do anything
- Repeat everything done to the Controller subsystem for the Plant subsystem
  - Rename the Out1 to Plant Diagnostics
  - Rename the CAN signal to Plant_Diagnostics

47

Plant Subsystem



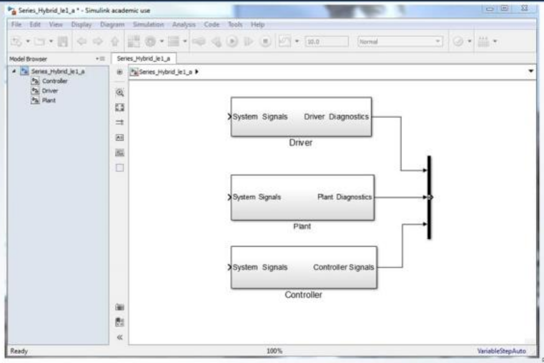
48

High Level

- Return to the top level of the model
- Increase the size of the subsystems to make them legible
- Drag in a Bus Creator
  - Double click of the Bus Creator
  - Change the number of inputs to 3
  - Click OK
- Connect each subsystem to the Bus Creator
- If you Double click on the Bus Creator after connecting the subsystem you can view the input signals

49

High Level



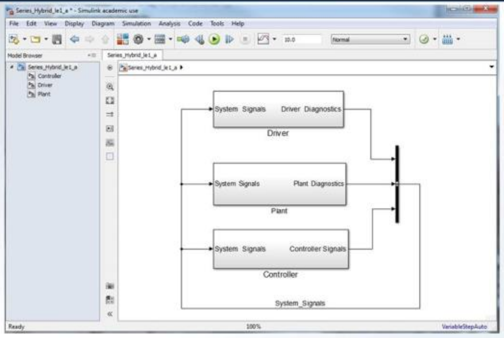
50

High Level

- Connect the output of the Bus Creator to each of the subsystem inputs
- Rename the output signal of the Bus Creator to System_Signals

51

High Level

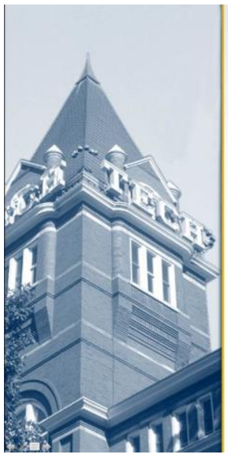


52

High Level

- Congratulations! You have
  - Created a high level hierarchical system
  - Established communication between the three key subsystems
  - Gained experience with Simulink

53



MBSD Lecture 2

Simple Vehicle Model

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Lecture Goals

- Create the Vehicle subsystem
- Create the MGA subsystem with a preposterous motor model
- Observe and assess the results

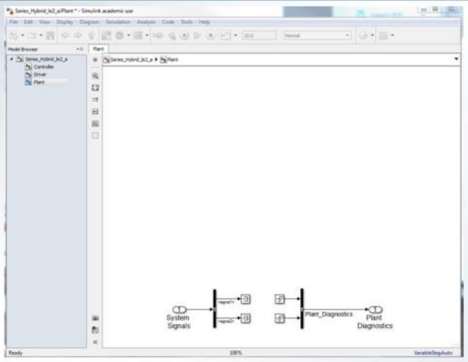
55

Simple Vehicle

- Save your previous model (Series_Hybrid_le1_a.slx) as Series_Hybrid_le2_a
- In the Model Browser window, click on the Plant subsystem
- Move the buses to the bottom of the window

56

Simple Vehicle



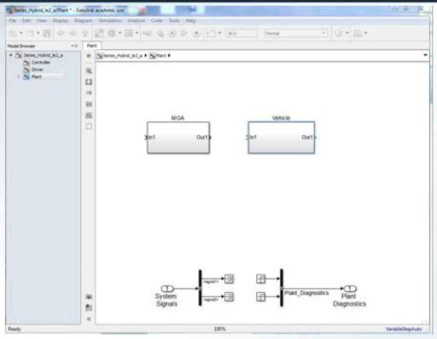
57

Simple Vehicle

- Drag two Subsystems into the Plant window
- Rename them
  - Vehicle
  - MGA
- Right click on the Vehicle subsystem
  - Select Rotate & Flip
  - Flip Block Name
- Repeat for the MGA subsystem

58

Simple Vehicle



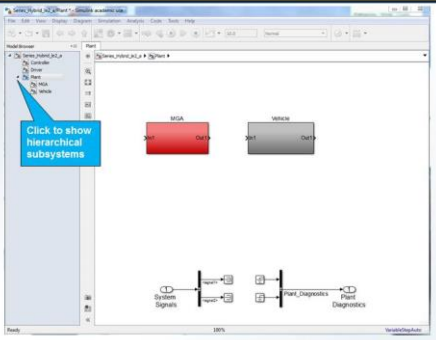
59

Simple Vehicle

- Let's also color code our subsystems to make things easy to identify
  - Right click on the subsystem
  - Format/Background Color
- Make the vehicle gray and MGA red

60

Simple Vehicle



61

Vehicle Subsystem

- The Vehicle subsystem will mechanically receive torque via a shaft from MGA
- We will skip mechanical brakes and use exclusively regenerative braking as applied by MGA
- The Vehicle subsystem will contain
  - Posi-trac rear axle
  - Rear tires
  - Vehicle solver
  - Inertias for the driveshaft and axle half shafts

62

Vehicle Subsystem

- Delete the In1 and Out1 blocks as well as the connecting wire
- Copy the In1, Out1, Bus, Terminator, and Ground blocks from the Plant subsystem
- Rename Plant to Vehicle
- From
  - Simscape / Utilities
- Drag in a Connection Port and rename it Vehicle Mechanical Port

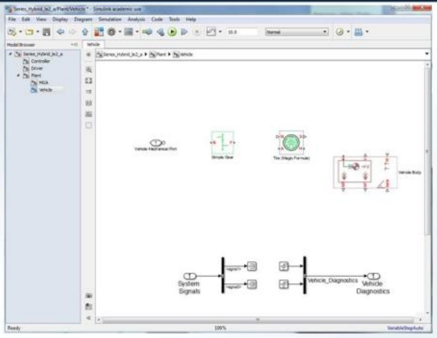
63

Vehicle Subsystem

- From
  - Simscape / Driveline / Tires & Vehicles
- Drag in the Vehicle Body and one Tire(Magic Formula) blocks
- From
  - Simscape / Driveline / Gears
- Drag in a Simple Gear block

64

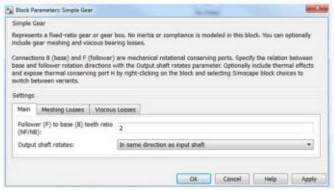
Vehicle Subsystem



65

Vehicle Subsystem

- Double click on the Simple Gear
- We can specify the gear ratio and relative rotational directions
  - Change the output shaft rotation to the same direction as input shaft
  - Leave the ratio as 2


Block Parameters: Simple Gear

Simple Gear

Represents a fixed-ratio gear or gear box. No inertia or compliance is modeled in this block. You can optionally include gear meshing and viscous bearing losses.

Connections B (base) and F (follower) are mechanical rotational conserving ports. Specify the relation between base and follower rotation directions with the Output shaft rotates parameter. Optionally include thermal effects and expose thermal conserving port H by right-clicking on the block and selecting Simscape block choices to switch between variants.

Settings

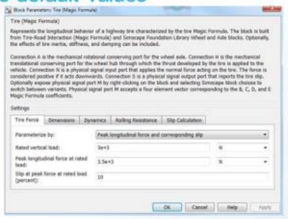
Follower (F) to base (B) teeth ratio (NF/NB): 2

Output shaft rotates: In same direction as input shaft

66

Vehicle Subsystem

- Double click on the Tire block
  - This block converts the applied torque to force in the x-direction
  - It also determined whether or not the tire slips
  - Keep the default values



Block Parameters: Tire (Magic Formula)

Tire (Magic Formula)

Represents the longitudinal behavior of a highway tire characterized by the tire Magic Formula. The block is built from Tire-Road Interaction (Magic Formula) and Simscape Foundation Library Wheel and Axle blocks. Optionally, the effects of tire inertia, stiffness, and damping can be included.

Connection A is the mechanical rotational conserving port for the wheel axle. Connection H is the mechanical translational conserving port for the wheel hub through which the thrust developed by the tire is applied to the vehicle. Connection N is a physical signal input port that applies the normal force acting on the tire. The force is considered positive if it acts downwards. Connection S is a physical signal output port that reports the tire slip. Optionally expose physical signal port M by right-clicking on the block and selecting Simscape block choices to switch between variants. Physical signal port M accepts a four element vector corresponding to the B, C, D, and E Magic Formula coefficients.

Settings

Tire Force | Dimensions | Dynamics | Rolling Resistance | Slip Calculation

Parameterize by: Peak longitudinal force and corresponding slip

Rated vertical load: 3e+3 N

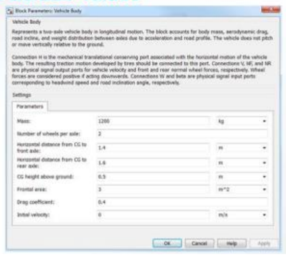
Peak longitudinal force at rated load: 3.5e+3 N

Slip at peak force at rated load (percent): 10

67

Vehicle Subsystem

- Double click on the Vehicle Body block
  - This block solves linear momentum (with drag and pitch) from the tire force
  - Keep the default values



Block Parameters: Vehicle Body

Vehicle Body

Represents a two-axle vehicle body in longitudinal motion. The block accounts for body mass, aerodynamic drag, road incline, and weight distribution between axles due to acceleration and road profile. The vehicle does not pitch or move vertically relative to the ground.

Connection H is the mechanical translational conserving port associated with the horizontal motion of the vehicle body. The resulting traction motion developed by tires should be connected to this port. Connections V, NF, and NR are physical signal output ports for vehicle velocity and front and rear normal wheel forces, respectively. Wheel forces are considered positive if acting downwards. Connections W and beta are physical signal input ports corresponding to headwind speed and road inclination angle, respectively.

Settings

Parameters

Mass: 1200 kg

Number of wheels per axle: 2

Horizontal distance from CG to front axle: 1.4 m

Horizontal distance from CG to rear axle: 1.6 m

CG height above ground: 0.5 m

Frontal area: 3 m^2

Drag coefficient: 0.4

Initial velocity: 0 m/s

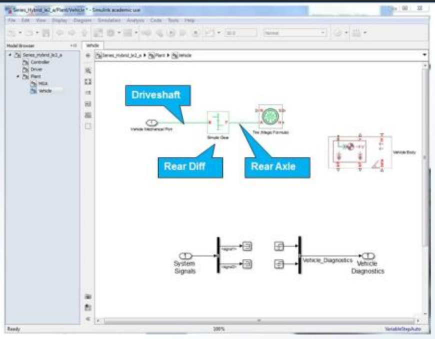
68

Vehicle Subsystem

- Connect the Vehicle Mechanical Port to the Base side of the Simple Gear
- Connect the Follower side of the Simple Gear to the A port on the Tire
- We now have a model of the driveshaft, rear differential, and axle shaft

69

Vehicle Subsystem



70

Vehicle Subsystem

- Note the difference in between the Driveline connection ports and the Simulink connection ports
- They will not connect to each other
- Driveline blocks are used to solve the Conservation of Angular Momentum principle during each time step

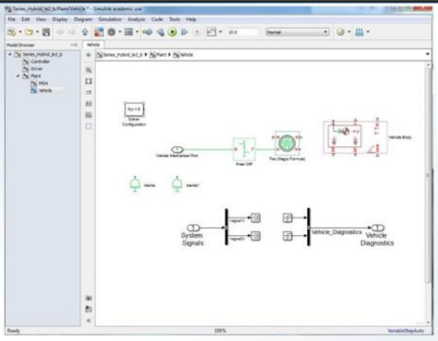
71

Vehicle Subsystem

- Every shaft must have a mass moment of inertia
- Also, a special solver must be used
- From
  - Simscape / Foundation Library / Mechanical / Rotational Elements
- Drag in two Inertia blocks
- From
  - Simscape / Utilities
- Drag in a Solver Configuration block

72

Vehicle Subsystem



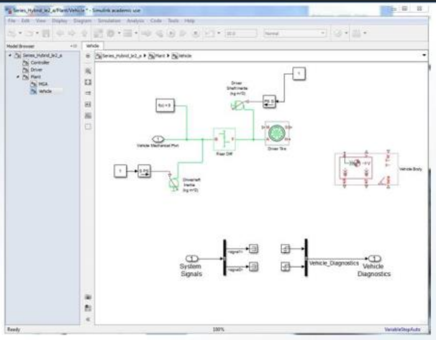
73

Vehicle Subsystem

- Connect the Inertias to the driveshaft and rear axle and Rename them:
  - Driveshaft Inertia (kg m^2)
  - Driver Halfshaft Inertia (kg m^2)
- Connect the Solver Configuration to the driveshaft
- Hide its name
- Rename the Tire to Driver Tire
- Rename the Simple Gear to Rear Diff

74

Vehicle Subsystem



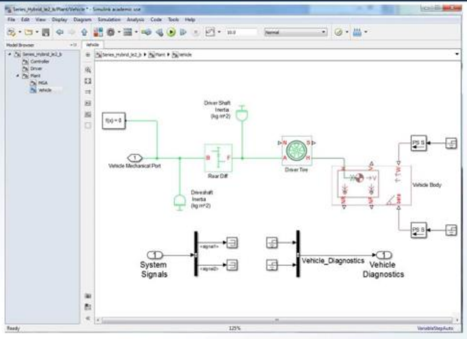
75

Vehicle Subsystem

- Rename and save you your model as Series_Hybrid_le2_b
- Now we can connect the H port of the Drive Tire block to the H port of the Vehicle Body block
- From
  - Simscape / Utilities
- Drag in two Simulink-PS Converter blocks
- These blocks transfer Simulink signals to physical signals
- Drag in two Ground blocks
- Hide their names
- Connect the Ground blocks to the Simulink-PS Converters
- Connect the Simulink-PS Converters to the beta and W ports of the Vehicle Body block

76

Vehicle Subsystem



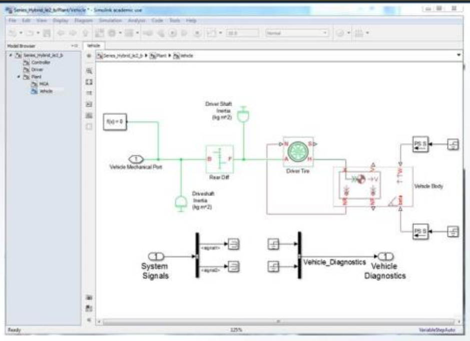
77

Vehicle Subsystem

- The NR port of the Vehicle Body block is a physical signal reporting the normal force applied on the rear wheels
- Connect the NR port of the Vehicle Body block to the N port of the Driver Tire block
- Rearrange the block and wire to make model look good
- S port of the Drive Tire block is a physical signal reporting tire slip

78

Vehicle Subsystem



79

Vehicle Subsystem

- S port of the Drive Tire block is a physical signal reporting tire slip
- NF port of the Vehicle Body block reports the normal load on the front tires
- We are not interested in tire slip or the normal force on the front tire (this is a rear wheel vehicle)
- From
  - Simscape / Utilities
- Grab two PS-Simulink Converter blocks (opposite of Simulink-PS Converter)
- Grab two Terminator blocks
- Hide name of PS-Simulink Converter and Terminator both blocks
- Connect the PS-Simulink Converter blocks to the Terminators
- Connect the S port of the Drive Tire block and NF port of the Vehicle Body block to the PS-Simulink Converter blocks
- Rearrange the block to make the model look good and compact

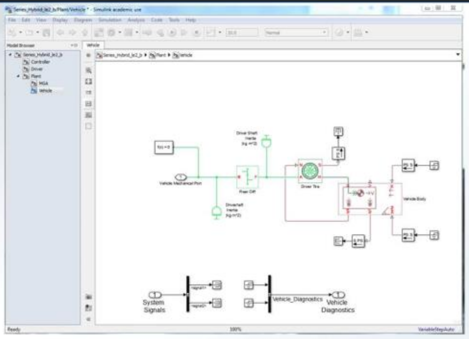
80

Vehicle Subsystem

- It is preferred for the wires not to cross as much as possible. You should place your blocks in position to avoid the wire crossing as much as possible

81

Vehicle Subsystem



82

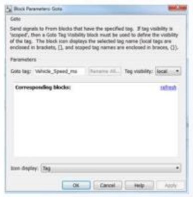
Vehicle Subsystem

- We need to send the vehicle velocity signal (from the V port of the Vehicle Body block) to other parts of the model
- Grab a PS-Simulink Converter block and connect the V port of the Vehicle Body block to the PS-Simulink Converter
- From
  - Simulink / Signal Routing
- Grab a Goto block
- Connect the PS-Simulink Converter to the Goto block

83

Vehicle Subsystem

- Double click on the Goto block and change the tag to Vehicle_Speed_ms
- Make sure the visibility is set to local
- Click OK
- Drag the sides of the Goto block to make the name visible
- Hide the name of Goto and PS-Simulink Converter blocks



Goto

Send signals to From blocks that have the specified tag. If tag visibility is 'scoped', then a Goto Tag Visibility block must be used to define the visibility of the tag. The block icon displays the selected tag name (local tags are enclosed in brackets, [], and scoped tag names are enclosed in braces, {}).

Parameters

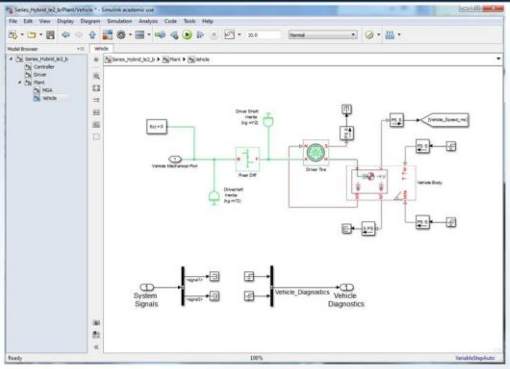
Goto tag: Vehicle_Speed_ms Tag visibility: local

Corresponding blocks:

Icon display: Tag

84

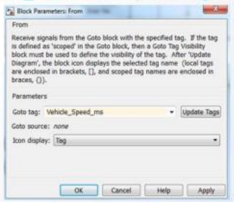
Vehicle Subsystem



85

Vehicle Subsystem

- From
  - Simulink / Signal Routing
- Grab a From block
- Double click on the From block
- Open the Goto tag bar and select Vehicle_Speed_ms
  - Note: if you don't see Vehicle_Speed_ms, click on Update Tags and try again
- Click OK
- Hide the name of the From block


Block Parameters: From

From

Receive signals from the Goto block with the specified tag. If the tag is defined as 'scoped' in the Goto block, then a Goto Tag Visibility block must be used to define the visibility of the tag. After 'Update Diagram', the block icon displays the selected tag name (local tags are enclosed in brackets, [], and scoped tag names are enclosed in braces, {}).

Parameters

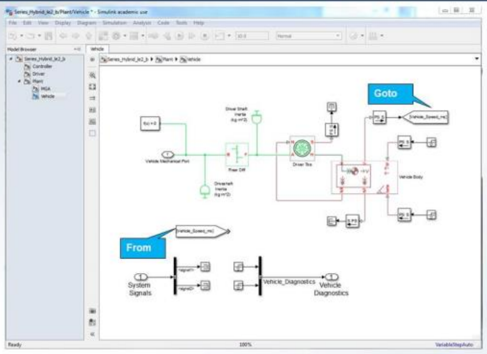
Goto tag: Vehicle_Speed_ms

Goto source: none

Icon display: Tag

86

Vehicle Subsystem



87

Vehicle Subsystem

- Congratulations – you have a model of a vehicle that will (hopefully) move when subjected to a torque
- From a vehicle diagnostics perspective, we are interested in two things
  - Speed (mph)
  - Distance Traveled (miles)
- Move the Vehicle_Speed_ms From block down near the Vehicle Diagnostics Bus
- From
  - Simulink / Commonly Used Blocks
- Drag in two Gain blocks and one Integrator block
- Hide the name of the Integrator block

88

Vehicle Subsystem

- Congratulations – you have a model of a vehicle that will (hopefully) move when subjected to a torque
- From a vehicle diagnostics perspective, we are interested in two things
  - Speed (mph)
  - Distance Traveled (miles)
- Move the Vehicle_Speed_ms From block down near the Vehicle Diagnostics Bus
- From
  - Simulink / Commonly Used Blocks
- Drag in two Gain blocks and one Integrator block
- Hide the name of the Integrator block

89

Vehicle Subsystem

- Set the first Gain to 2.2374 (m/s to mph)
- Set the second Gain to 6.214 e-4 (m/s to miles/s)
- Name them accordingly
- Connect the output of the From block to the input of the Gain blocks
- Connect the second Gain block to the Integrator block
- Delete the Grounds going into the Bus
- Connect the two signals
- Name them: (Double click on the wires)
  - Vehicle_Speed_mph
  - Vehicle_Distance_miles

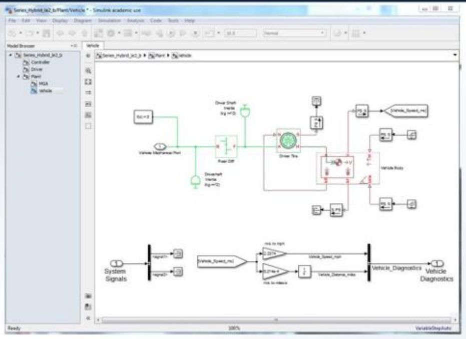
90

Vehicle Subsystem

- The Vehicle subsystem is done
  - Note that we did not use any of the system signals
  - When we get around to mechanical brakes we will need a brake signal
- Take some time to move things around and make it look nice

91

Vehicle Subsystem



92

MGA Subsystem

- Our motor model is going to be preposterously simple
  - Constant torque output
  - No RPM limits
  - No input power required
- Rename and save you model as:
  - Series_Hybrid_le2_c
- Use the Model Browser to go to the MGA subsystem

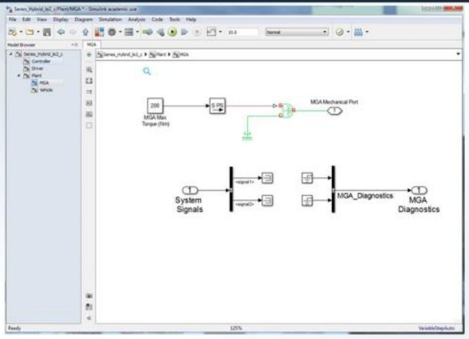
93

MGA Subsystem

- Delete the In1, Out1, and the connecting wire
- Copy the Bus from the Plant subsystem
  - Rename accordingly
- Drag in a Constant block and rename it MGA Max Torque (Nm)
- Set the constant value at 200 Nm
- From
  - Simscape / Foundation Library / Mechanical / Mechanical Sources
- Drag in an Ideal Torque Source block and hide its name
- Drag in a Connection Port and rename it MGA Mechanical Port
- Drag in a Simulink-PS Converter and hide its name
- From
  - Simscape / Foundation Library / Mechanical / Rotational Elements
- Grab a Mechanical Rotational Reference block and hide its name
- Wire everything together as shown in the next page

94

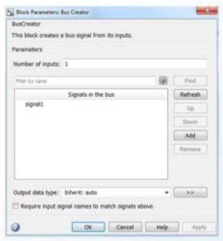
MGA Subsystem



95

MGA Subsystem

- The only MGA diagnostic is the torque
- Delete the Terminators
- Double click on the Bus Creator and change the Number of inputs to 1
- Click OK


Block Parameters: Bus Creator

BusCreator

This block creates a bus signal from its inputs.

Parameters

Number of inputs: 1

Signals in the bus: signal1

Output data type: Inherit: auto

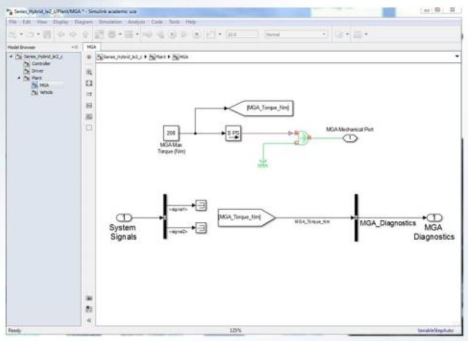
☐ Require input signal names to match signals above

96

MGA Subsystem

- Drag in a pair of Goto and From blocks
- Name them accordingly
- Use them to route the MGA torque signal to the Bus Creator block
- Name the signal going into the Bus
- Compare with next slides

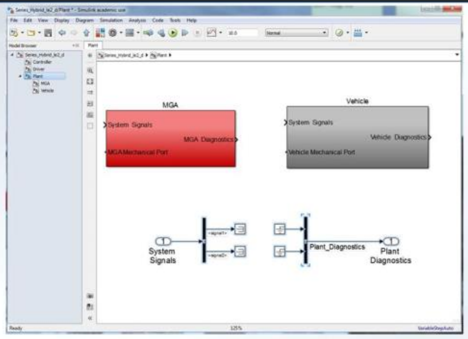
MGA Subsystem



MGA Subsystem

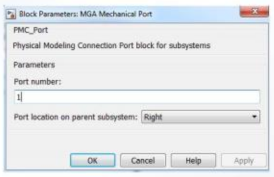
- Congratulations – you've made a motor model
- It is now time to connect MGA to the Vehicle and see what happens
- Save your model as
  - Series_Hybrid_le2_d
- Use the Model Browser to go to the Plant level of the model
- Make the MGA and Vehicle subsystems larger

Plant Level



Plant Level

- The MGA Mechanical Port is on the wrong side of the block
- Go into the MGA subsystem and double click on the Port
- Change the location from left to right
- Click OK


Block Parameters: MGA Mechanical Port

PMC_Port

Physical Modeling Connection Port block for subsystems

Parameters

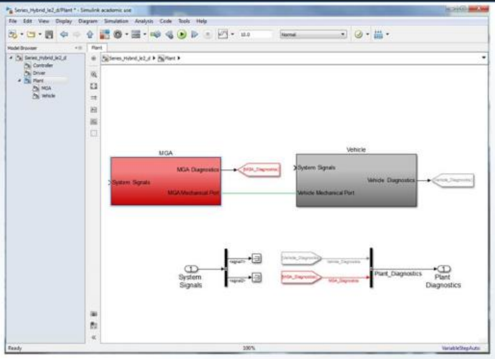
Port number: 1

Port location on parent subsystem: Right

Plant Level

- Connect the two Mechanical Ports
- Route the Vehicle and MGA Diagnostics to the Plant Signal CAN Bus
  - Name the signal inputs accordingly
- Change the foreground of the Vehicle routing blocks to gray
- Change the foreground of the MGA routing blocks to red

Plant Level



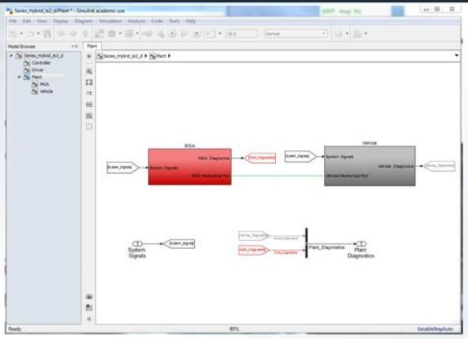
103

Plant Level

- All that is left is to route the System Signals to the Vehicle and MGA
- Delete the Terminators and the Bus Selector
- Use the Goto and From blocks to route the System Signals to MGA and the Vehicle

104

Plant Level



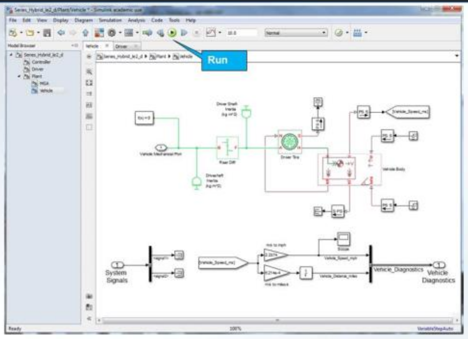
105

Running the Model

- It is now time to run the model
- Go to the Vehicle subsystem
- From
  - Simulink / Sinks
- Drag in a Scope
- Connect it to the Vehicle_Speed_mph signal
- Click on the Run button to start the simulation

106

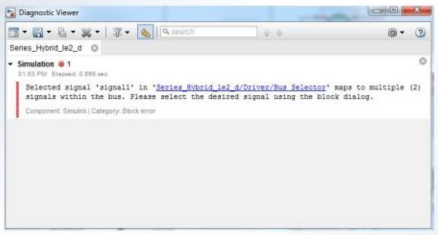
Running the Model



107

Running the Model

- Oh! There is an error



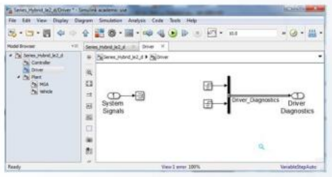
```
Selected signal 'signal1' in 'Series_Hybrid_le2_d/Driver/Bus Selector' maps to multiple (2)
signals within the bus. Please select the desired signal using the block dialog.
```

Component: Simulink | Category: Block error

108

Running the Model

- Since we do not yet need any signals from the System Diagnostics bus, replace the Bus Selector with a Terminator in the
  - Driver
  - Controller
  - Plant / Vehicle
  - Plant / MGA
- subsystems



109

Running the Model

- Rerun the model
- If the scope is not open, double click on the block
- Observe the results
  - 0 to 25 mph in 10 seconds
- You can check the results by hand

$$m\frac{dv_x}{dt} = \sum F_x$$

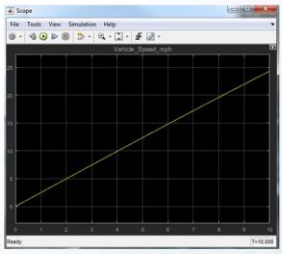
$$\frac{dv_x}{dt} = \frac{1}{m}\left(\frac{TN}{r}\right)$$

$$\int_{v_i}^{v_f} dv_x = \int_{t_i}^{t_f}\left(\frac{TN}{mr}\right)dt$$

$$v_f = \frac{(200\ Nm)(2)}{(1200\ kg)(0.3\ m)} = 11.1\ {}^{m}/{}_{s} = 24.9\ mph$$

Gear ratio

Tire radius

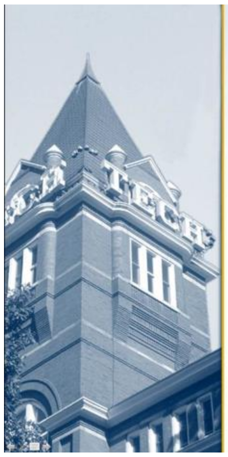


110

Simple Vehicle

- Review
  - Vehicle subsystem
    - Receives torque from MGA
    - Solves vehicle velocity
    - Sends/Receives CAN messages
  - MGA subsystem
    - Provides torque to Vehicle
    - Sends/Receives CAN messages

111



MBSD Lecture 3

Initialization, Solver, Logging & Visualization, and Driver

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Lecture Goals

- Create an initialization file for data management
- Select the correct ODE solver
- Create a Data Logging and Visualization subsystem
- Create a Driver subsystem
- Have the vehicle follow a drive cycle
- Experiment with feedback gain

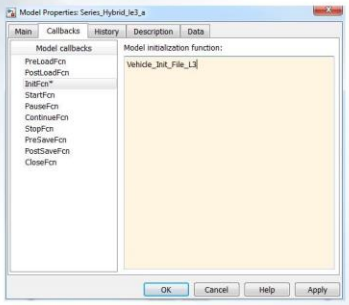
113

Initialization File

- Save your previous model as
  - Series_Hybrid_le3_a
- Go to the System Level of the model
- Right click on any vacant white space on the model
  - Select Model Properties
  - Select the Callbacks tab
  - Click on the InitFcn
  - Type Vehicle_Init_File_L3 in the second column

114

Initialization File


Model Properties: Series_Hybrid_le3_a

Main | Callbacks | History | Description | Data

Model callbacks: PreLoadFcn, PostLoadFcn, InitFcn*, StartFcn, PauseFcn, ContinueFcn, StopFcn, PreSaveFcn, PostSaveFcn, CloseFcn

Model initialization function:

```
Vehicle_Init_File_L3
```

OK | Cancel | Help | Apply

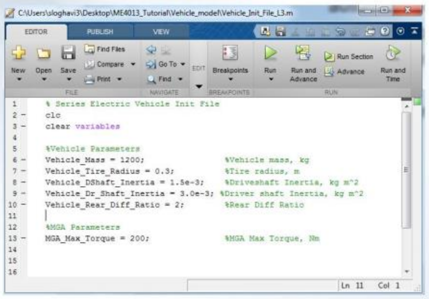
115

Initialization File

- This is going to be a Matlab M-file which will automatically execute each time the model is run
- All of our component parameters will go in this file
- Start by going to the Matlab window
- Open file
  - Vehicle_Init_File_L3.m

116

Initialization File


C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

```
% Series Electric Vehicle Init File
clc
clear variables

%Vehicle Parameters
Vehicle_Mass = 1200;                    %Vehicle mass, kg
Vehicle_Tire_Radius = 0.3;              %Tire radius, m
Vehicle_DShaft_Inertia = 1.5e-3;        %Driveshaft Inertia, kg m^2
Vehicle_Dr_Shaft_Inertia = 3.0e-3; %Driver shaft Inertia, kg m^2
Vehicle_Rear_Diff_Ratio = 2;            %Rear Diff Ratio

%MGA Parameters
MGA_Max_Torque = 200;                   %MGA Max Torque, Nm
```

Ln 11 Col 1

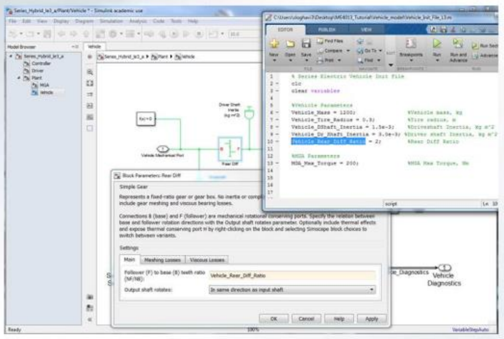
117

Initialization File

- We will replace the numeric values in the blocks with these variable names
  - This consolidates all key parameters into one location
  - Help prevent making mistakes
  - Tip: copy the variable names from the M-file and paste into the block to prevent spelling errors
- Update all relevant blocks

118

Initialization File


Block Parameters: Rear Diff

Simple Gear

Represents a fixed-ratio gear or gear box. No inertia or compl… include gear meshing and viscous bearing losses.

Connections B (base) and F (follower) are mechanical rotational conserving ports. Specify the relation between base and follower rotation directions with the Output shaft rotates parameter. Optionally include thermal effects and expose thermal conserving port H by right-clicking on the block and selecting Simscape block choices to switch between variants.

Settings

Main | Meshing Losses | Viscous Losses

Follower (F) to base (B) teeth ratio (NF/NB): Vehicle_Rear_Diff_Ratio

Output shaft rotates: In same direction as input shaft

OK | Cancel | Help | Apply

119

ODE Solver

- While the Solver Configuration calls the hooks for solving the driveline angular momentum, an overall ODE solver must be specified
- Our model contains a set of stiff ODEs and as recommended by The MathWorks, best solver suited for this type of ODEs is ODE23t

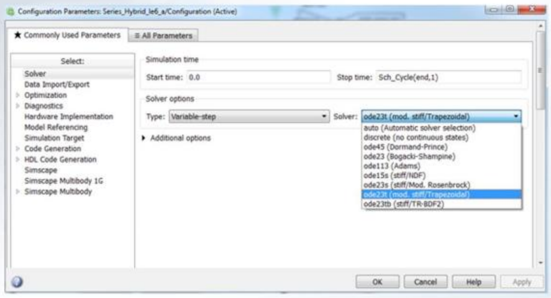
120

ODE Solver

- As with the Init file, right click on the model background
- Select
  - Model Configuration Parameters
  - Solver: ode23t (mod. stiff/Trapezoidal)
  - Click OK
- Run the model and note how much faster it runs!

121

ODE Solver


Configuration Parameters: Series_Hybrid_le6_a/Configuration (Active)

Simulation time
Start time: 0.0 Stop time: Sch_Cycle(end,1)

Solver options
Type: Variable-step Solver: ode23t (mod. stiff/Trapezoidal)

122

Logging & Visualization

- Let's now create a subsystem to represent the on-board data logger
- This will also be a good place for having all the visualization
- First, go to the Vehicle subsystem and delete the Scope block
  - Having scopes in the model is bad for auto-code generation

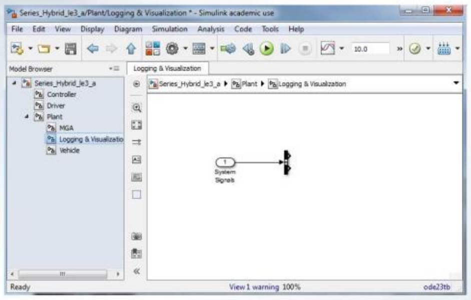
123

Logging & Visualization

- Drag a new Subsystem into the Plant level of the model
- Rename it Logging & Visualization
- Rename the In1 block to System Signals
- Delete the Out1 block
  - The logger does not send any diagnostics
- Drag a Bus Selector into the subsystem
- Go back into the Plant level of the model
- Copy a System Signals From block and connect it to System Signals of Logging & Visualization subsystem

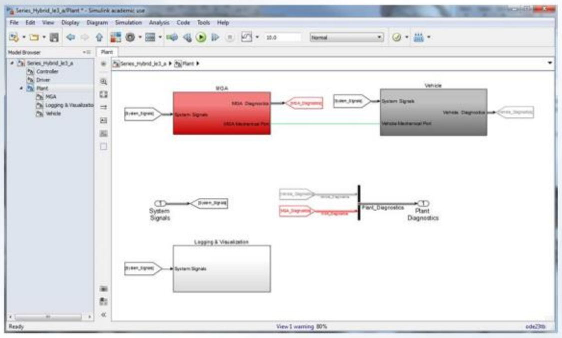
124

Logging & Visualization


Series_Hybrid_le3_a/Plant/Logging & Visualization * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
Series_Hybrid_le3_a
Controller
Driver
Plant
MGA
Logging & Visualization
Vehicle
System Signals
Ready
View 1 warning 100%
ode23tb

125

Logging & Visualization



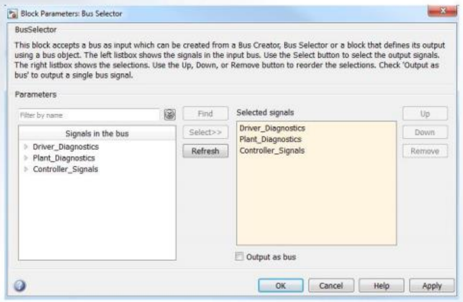
126

Logging & Visualization

- In the Logging & Visualization subsystem
- Double click on the Bus Selector block
- In the Selected signals column
  - Click on signal1 and then Remove
  - Click on signal1 and then Remove
- In the Signals in the bus column
  - Click on Driver_Diagnostics and then Select
  - Click on Plant_Diagnostics and then Select
  - Click on Controller_Diagnostics and then select
- Click OK

127

Logging & Visualization


Block Parameters: Bus Selector

BusSelector

This block accepts a bus as input which can be created from a Bus Creator, Bus Selector or a block that defines its output using a bus object. The left listbox shows the signals in the input bus. Use the Select button to select the output signals. The right listbox shows the selections. Use the Up, Down, or Remove button to reorder the selections. Check 'Output as bus' to output a single bus signal.

Parameters

Filter by name

Signals in the bus:
- Driver_Diagnostics
- Plant_Diagnostics
- Controller_Signals

Selected signals:
- Driver_Diagnostics
- Plant_Diagnostics
- Controller_Signals

☐ Output as bus

128

Logging & Visualization

- Drag in three Goto blocks and name them Driver_Diagnostics, Plant_Diagnostics, and Controller_Signals
- Wire them to the Bus Selector
- Drag in three From blocks
- Terminate the Driver_Diagnostics and the Controller_Signals From blocks
- Drag in a Bus Selector and wire it to the Plant_Diagnostics From block

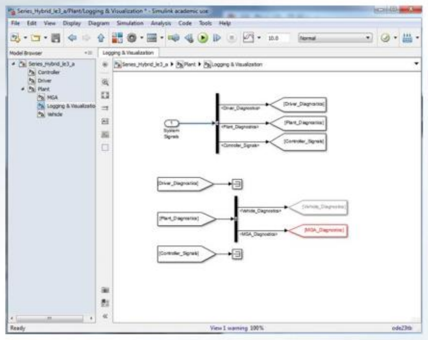
129

Logging & Visualization

- Double click on the Bus Selector
- Remove signal1 and signal2
- Select the Vehicle_Diagnostics and the MGA_Diagnostics signals
- Drag in two more Goto blocks
  - Connect them to the Bus Selector
  - Name them accordingly
  - Color them accordingly

130

Logging & Visualization



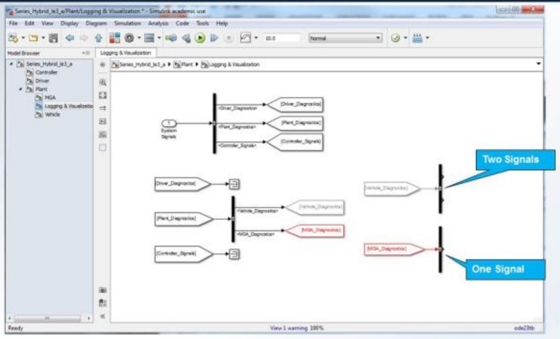
131

Logging & Visualization

- Drag in two more From and Bus Selector blocks
- Select
  - Vehicle_Diagnostics - first From block
  - MGA_Diagnostics – second From block
- Connect the From blocks to the Bus Selector blocks
- Use the Bus Selector blocks to select all signals

132

Logging & Visualization



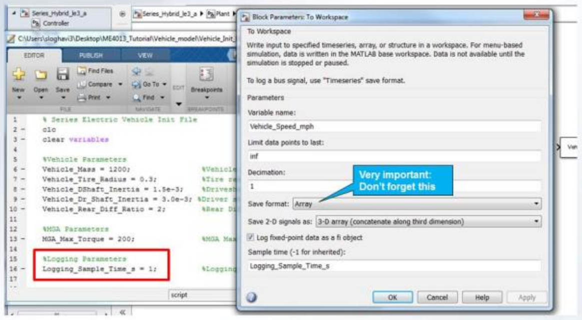
133

Logging & Visualization

- From
  - Simulink / Sinks
- Drag in a To Workspace block
  - This will write the desired signal values to an array for review when the simulation is done
- Double click on the To Workspace block
  - Make the following changes
    - Variable name - Vehicle_Speed_mph
    - Sample time - Logging_Sample_Time_s
    - Save format – Array
- Update the input file

134

Logging & Visualization



C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_

```
% Series Electric Vehicle Init File
clc
clear variables

%Vehicle Parameters
Vehicle_Mass = 1200;                    %Vehicle
Vehicle_Tire_Radius = 0.3;              %Tire r
Vehicle_DShaft_Inertia = 1.5e-3;        %Drivesh
Vehicle_Dr_Shaft_Inertia = 3.0e-3;      %Driver s
Vehicle_Rear_Diff_Ratio = 2;            %Rear Di

%MGA Parameters
MGA_Max_Torque = 200;                   %MGA Max

%Logging Parameters
Logging_Sample_Time_s = 1;              %Logging
```

Block Parameters: To Workspace

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: Vehicle_Speed_mph

Limit data points to last: inf

Decimation: 1

Save format: Array

Save 2-D signals as: 3-D array (concatenate along third dimension)

[✓] Log fixed-point data as a fi object

Sample time (-1 for inherited): Logging_Sample_Time_s

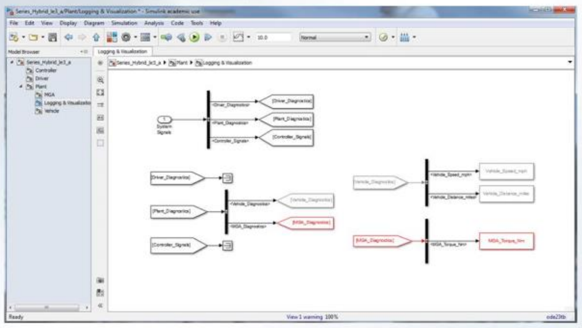
135

Logging & Visualization

- Connect the To Workspace block to the Bus Selector
- Repeat for the remaining signals coming from the Plant
- Color everything accordingly

136

Logging & Visualization



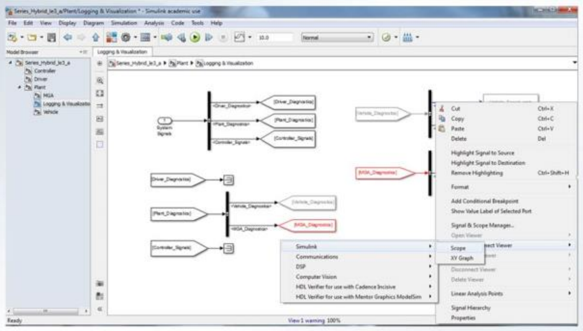
137

Logging & Visualization

- Rather than using Scope blocks, we will use the Signal and Scope Manager to view the signals
- Right click on the Vehicle_Speed_mph signal and select
  - Create & Connect Viewer / Simulink / Scope

138

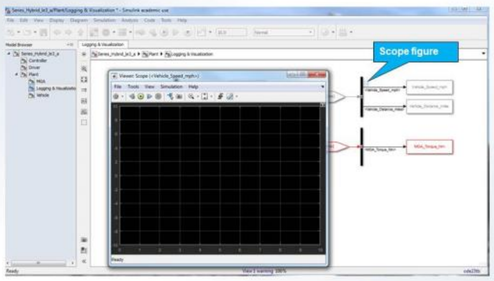
Logging & Visualization


Series_Hybrid_le3_a/Plant/Logging & Visualization * - Simulink academic use
System Signals
Driver_Diagnostics
Plant_Diagnostics
Controller_Signals
Vehicle_Diagnostics
MGA_Diagnostics
Cut
Copy
Paste
Delete
Highlight Signal to Source
Highlight Signal to Destination
Remove Highlighting
Format
Add Conditional Breakpoint
Show Value Label of Selected Port
Signal & Scope Manager...
Open Viewer
Connect Viewer
Disconnect Viewer
Delete Viewer
Linear Analysis Points
Signal Hierarchy
Properties
Simulink
Communications
DSP
Computer Vision
HDL Verifier for use with Cadence Incisive
HDL Verifier for use with Mentor Graphics ModelSim
Scope
XY Graph
Ready

139

Logging & Visualization

- Note the scope figure on the Vehicle_Speed_mph signal – that denotes that the signal is being scoped



140

Driver Subsystem

- The Driver subsystem will be our first proportional feedback loop
  - The vehicle will be at a certain speed
  - The driver will want to go a different speed
  - The driver will send a torque request (-1 to 1) to the controller
  - The vehicle will change speed
  - Repeat for the duration of the drive cycle

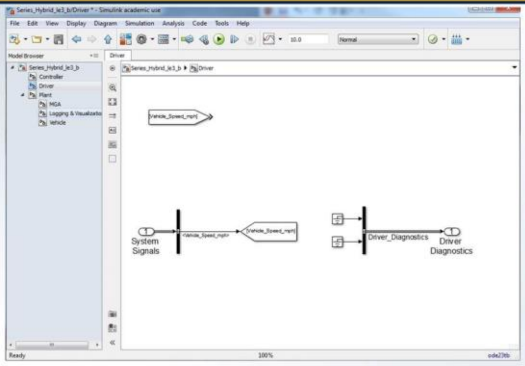
141

Driver Subsystem

- Save the model as Series_Hybrid_le3_b
- Use the Model Browser to go into the established Driver subsystem
- Drag in a Bus Selector, a Goto, and a From block
- Delete the Terminator and connect the Bus Selector block
- Select the Vehicle_Speed_mph signal
- Connect this signal to the Goto block

142

Driver Subsystem



143

Driver Subsystem

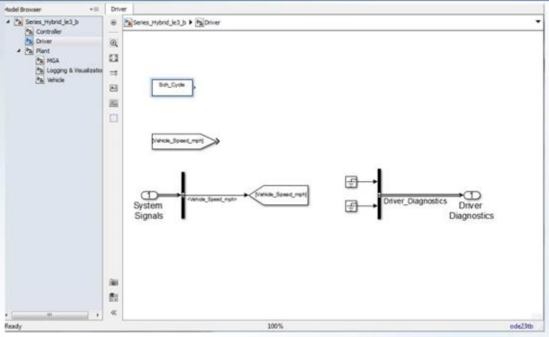
- The drive cycle will be read in from the workspace
  - It will contain time (s) and desired speed (mph) at that time
  - We will use the standard FU505 drive cycle
    - 505 seconds long
    - Collected data
  - The data is stored as a .mat file
  - The .mat file will be loaded with the init file

144

Driver Subsystem

- From
  - Simulink / Sources
- Drag in a From Workspace block
  - Double click on it
  - Rename the Data to: Sch_Cycle
  - Click OK

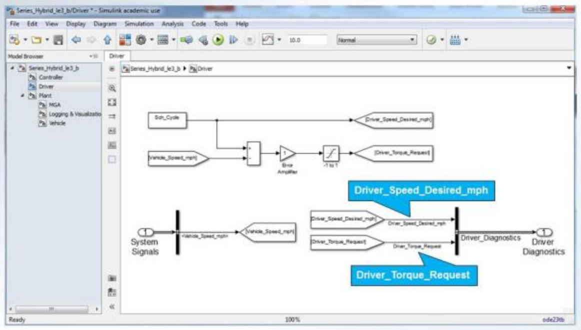
Driver Subsystem



Driver Subsystem

- From
  - Simulink / Commonly Used Blocks
- Drag in a Sum, Gain, and Saturation block
  - Make the Sum block rectangular with a +-
- Drag in another two Goto and From blocks
- Arrange, wire, and name as shown on the next slide

Driver Subsystem



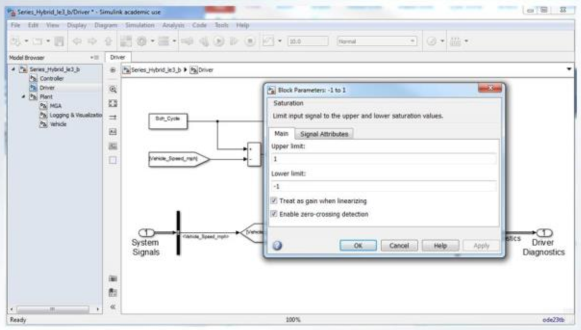
Driver Subsystem

- We must now determine the value of the error amplifier
- Classical controls theory would have us generate a transfer function and look for poles
- This system will soon become so complex that classic approaches do not work
- Instead use some experience
- If you are driving a car and want to go
  - 1 mph faster
  - 10 mph faster
  - 20 mph faster
- You step on the gas (torque request)
  - Very little
  - Half way
  - To the floor

Driver Subsystem

- The torque request will vary from
  - +1
    - full positive torque requested
    - 20 mph+ too slow
  - -1
    - Full negative torque requested
    - 20 mph+ too fast
- A gain of 0.05 will send a ± 1 at 20 mph error
- A Saturation will limit the request to ± 1

Driver Subsystem


Block Parameters: -1 to 1

Saturation

Limit input signal to the upper and lower saturation values.

Upper limit: 1

Lower limit: -1

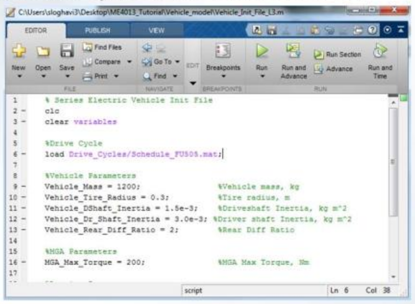
☑ Treat as gain when linearizing

☑ Enable zero-crossing detection

151

Driver Subsystem

- The final step in the Driver subsystem is to add the desired drive cycle to the initialization file

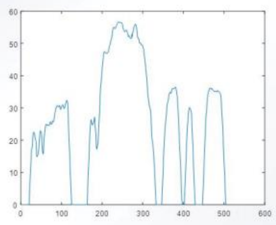

C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

```
% Series Electric Vehicle Init File
clc
clear variables

%Drive Cycle
load Drive_Cycles/Schedule_FU505.mat;

%Vehicle Parameters
Vehicle_Mass = 1200;                    %Vehicle mass, kg
Vehicle_Tire_Radius = 0.3;              %Tire radius, m
Vehicle_DShaft_Inertia = 1.5e-3;        %Driveshaft Inertia, kg m^2
Vehicle_Dr_Shaft_Inertia = 3.0e-3;  %Driver shaft Inertia, kg m^2
Vehicle_Rear_Diff_Ratio = 2;            %Rear Diff Ratio

%MGA Parameters
MGA_Max_Torque = 200;                   %MGA Max Torque, Nm
```

152

Driver Subsystem

- Run the init file manually to load the drive cycle
- At the Matlab command line type
  - plot(Sch_Cycle(:,1),Sch_Cycle(:,2))



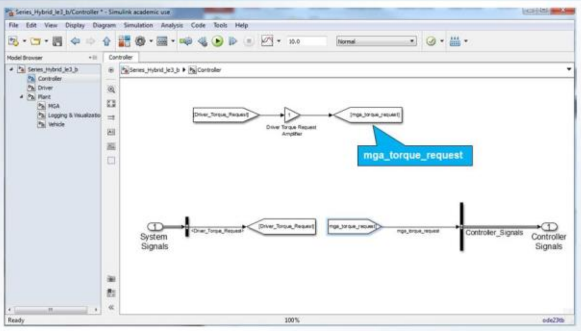
153

Controller Subsystem

- The Driver torque request will next be sent through the CAN bus to the Controller subsystem
- Go to the Controller subsystem, delete the Terminators and Grounds, and drag in a Gain, a Bus Selector, two Goto, and two From blocks
- On the Bus Selector, select the Driver_Torque_Request signal
- Wire and label as shown on the next slide

154

Controller Subsystem



155

Controller Subsystem

- The driver torque request passes through the controller where it is sent to MGA
  - Just like a real vehicle
- The Gain block can be modified
  - Sporty : increase request
  - Economy : decrease request
- Naming convention
  - All Control signals will be lowercase
  - End with either "request" or "enable"

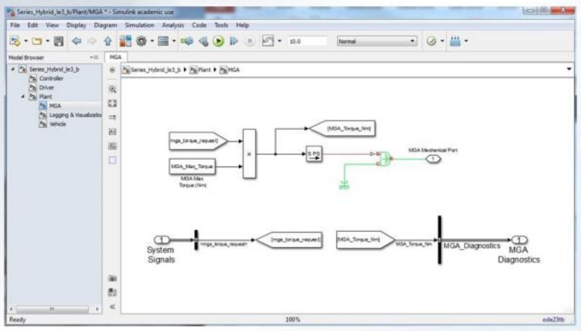
156

MGA Subsystem

- The last step is to pass the torque request to MGA
- Drag in a Bus Selector, a Goto, a From, and a Product block
- Delete the Terminator
- Select the mga_torque_request signal
- Wire and label as shown on the next slide

157

MGA Subsystem



158

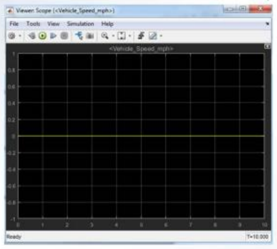
Run the simulation

- Go to the model level and Run the model

159

Results

- Run the model and observe the scope



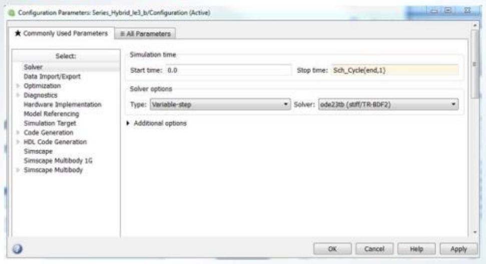
Viewer: Scope (<Vehicle_Speed_mph>)
File Tools View Simulation Help
<Vehicle_Speed_mph>
Ready
T=10.000

- It only ran for 10 seconds

160

Results

- Open the Configuration Parameters
- Change the Stop Time to
  - Sch_Cycle(end,1)


Configuration Parameters: Series_Hybrid_le3_b/Configuration (Active)

Simulation time

Start time: 0.0 Stop time: Sch_Cycle(end,1)

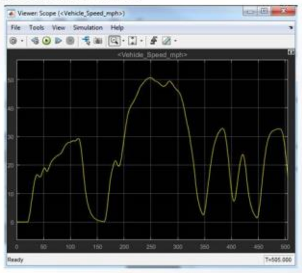
Solver options

Type: Variable-step Solver: ode23tb (stiff/TR-BDF2)

161

Results

- Run the simulation again
- Awesome! The vehicle is doing something!
- Note: in the Scope, go to Tools / Axes Scaling
- Select: Automatically Scale Axis Limits
- But How well?



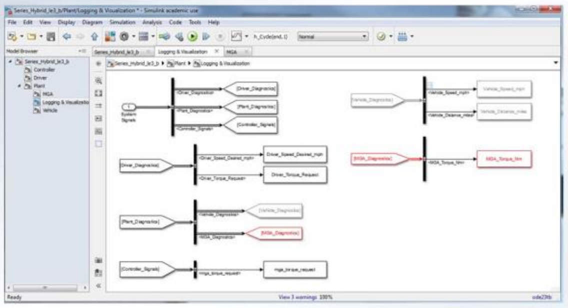
162

Logging & Visualization

- Go to the Logging & Visualization subsystem
- Delete the Terminator on the Driver_Diagnostics From block
- Use a Bus Selector to extract all Driver signals
- Drag in two To Workspace blocks to log the Driver Data
- Repeat for the Controller Signals

163

Logging & Visualization



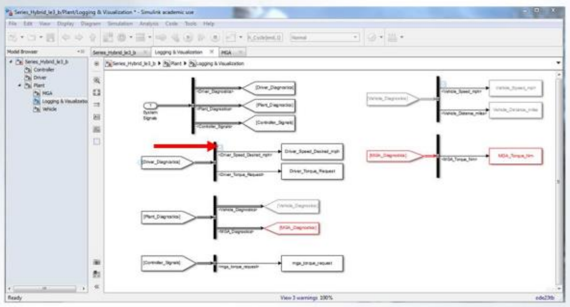
164

Logging & Visualization

- Right click on the Driver_Speed_Desired_mph signal
- Select
  - Connect to Existing Viewer
  - Scope
- Rerun the simulation and observe the scope

165

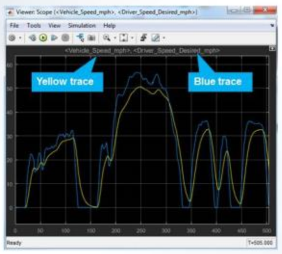
Logging & Visualization



166

Results

- Our vehicle does a poor job of following the drive cycle

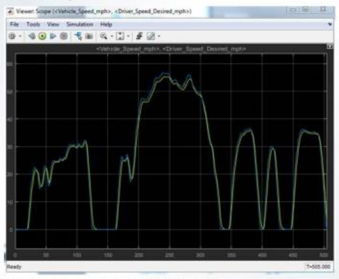


- Increase the MGA max torque in the init file until the traces are similar

167

Results

- Even with 1000 Nm, the vehicle still can not follow the trace exactly

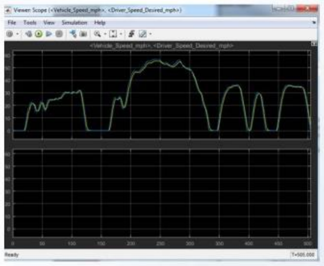


- Put the torque back to 200 Nm

168

Results

- Let's see what the driver is doing
- On the scope
- Select
  - View / Layout
- Choose a 2 × 1 layout
- Now we can add signals to both Axis 1 and Axis 2

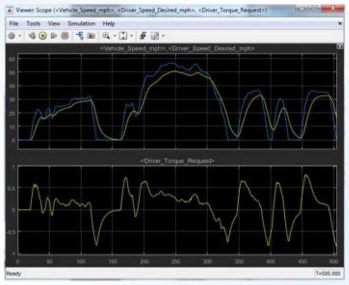

Viewer: Scope (<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>)
File Tools View Simulation Help
<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>
Ready

Results

- In the Logging & Visualization subsystem
  - Right click on the Driver_Torque_Request signal
  - Select
    - Connect to Viewer / Scope / Axes 2
- This will put the Driver torque request directly below the drive cycle
- Rerun the simulation

Results

- Wow – the driver torque request never goes above about 0.7


Viewer: Scope (<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>, <Driver_Torque_Request>)
File Tools View Simulation Help
<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>
<Driver_Torque_Request>
Ready

- Why?

Results

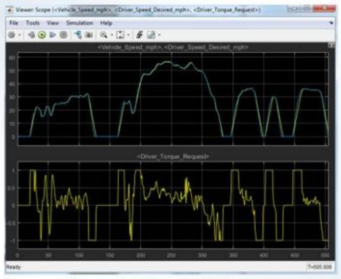
- Note that the vehicle is "chasing" a velocity and only occasionally off by more than 10 mph
- Change the error amplifier in the Driver subsystem to make the driver more aggressive
- Increasing the gain
  - Decreases the proportional error
  - Allows the system to respond faster
  - Can cause overshoot and oscillations

Results

- An error amplifier of 1 should work quite well

Results

- As we expected, error amplifier of 1 worked quite well for this drive cycle


Viewer: Scope (<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>, <Driver_Torque_Request>)
File Tools View Simulation Help
<Vehicle_Speed_mph>, <Driver_Speed_Desired_mph>
<Driver_Torque_Request>
Ready

- What about for a different drive cycle?

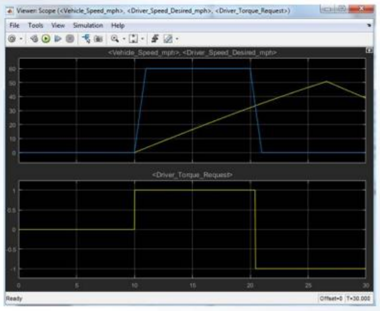
Results

- In the init file
  - Change the drive cycle to Schedule_Boston_Cab.mat
  - Increase the MGA torque to 1000 Nm
- Rerun the simulation

175

Results

- Definitely some overshoot



176

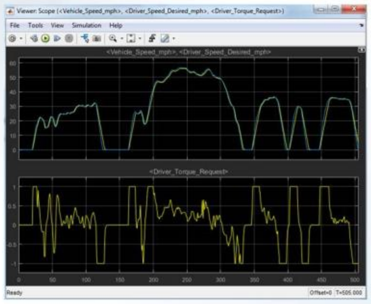
Results

- In the init file
  - Change the drive cycle back to the FU505
  - Change the MGA torque back to 200 Nm
- In the Driver subsystem
  - Change the error amplifier to 0.5
- Rerun the simulation

177

Results

- This provides a good balance of trace correlation and driver torque request

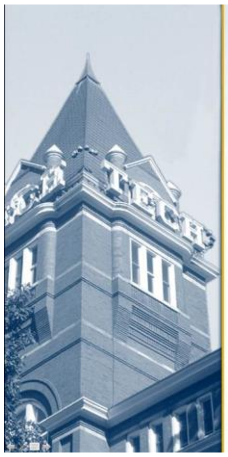


178

Initialization, Solver, and Driver

- Review
  - Created an initialization file to manage data
  - Changed to appropriate ODE solver
  - Created a logging and graphics subsystem
  - Developed driver proportional feedback loop
    - Velocity mismatch generates an error signal
    - Error signal converted to a driver torque request
    - Driver torque request goes to controller
    - Controller requests MGA torque
  - System response very sensitive to feedback gain value

179



MBSD Lecture 4

Improved MGA and Simple Battery

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Outline

- Make MGA an ideal electromechanical device
- Create a preposterous battery to power MGA
- Track the battery state of charge (SOC)

181

MGA

- Save your model as Series_Hybrid_le4_a.slx
- Go into the MGA subsystem
- MGA will still
  - Have a constant maximum torque
  - Have no rpm limits
  - Be modulated by the Controller
- By providing torque at a given RPM, power is transferred to the Vehicle

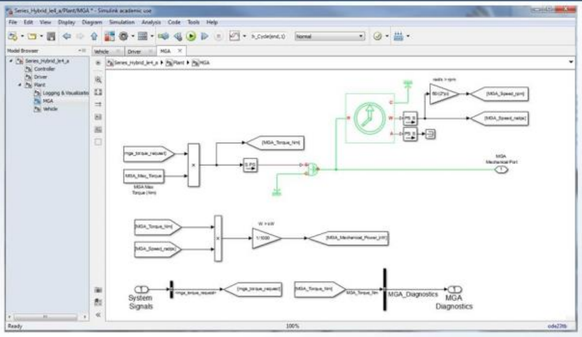
182

MGA

- From
  - Simscape / Foundation Library / Mechanical / Mechanical Sensors
- Drag in an Ideal Rotational Motion Sensor block
- From
  - Simscape / Foundation Library / Mechanical / Rotational Elements
- Drag in a Mechanical Rotational Reference block
- Also drag in three Gotos, two Froms, two Gains, two PS-Simulink Converters, One Terminator and a Product block
- Arrange and wire as shown on the next slide

183

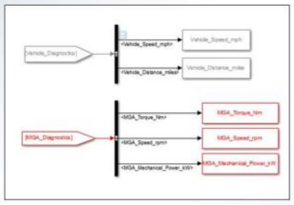
MGA

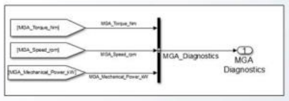


184

MGA

- Add the MGA_Speed_rpm and MGA_Mechanical_Power_kW to the MGA diagnostics bus
- In the Logging & Visualization subsystem, extract and log these signals





185

MGA & Battery

- To make the mechanical power, the motor will need to receive electrical power from the battery
- This power will be the battery voltage times the current drawn
- Let's now add an ideal battery to the Plant
- Drag in a Subsystem and rename it Battery
- Make the background orange

186

Battery

- The Battery will receive the system signals and broadcast its diagnostics
  - Delete the wire connecting the In1 and Out1 blocks and rename them System Signals and Battery Diagnostics
  - Terminate the System Signals
  - Drag in a Bus Creator, connect it to the Battery Diagnostics, and ground the inputs

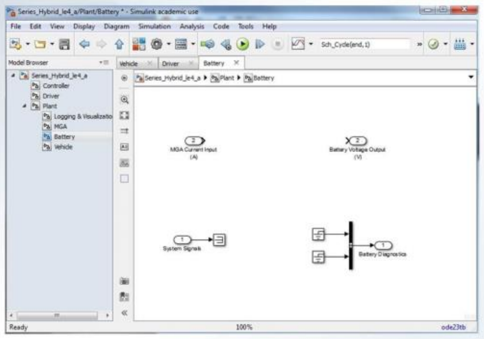
187

Battery

- Electrically, the battery will
  - Tell MGA its voltage
  - Receive a current demand from MGA
- From
  - Simulink / Commonly Used Blocks
- Drag in an In1 and an Out1 block
- Rename them
  - MGA Current Input (A)
  - Battery Voltage Output (V)

188

Battery


Series_Hybrid_le4_a/Plant/Battery * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
Model Browser
Series_Hybrid_le4_a
Controller
Driver
Plant
Logging & Visualizatio
MGA
Battery
Vehicle
Vehicle
Driver
Battery
Series_Hybrid_le4_a ▸ Plant ▸ Battery
MGA Current Input (A)
Battery Voltage Output (V)
System Signals
Battery Diagnostics
Ready
100%
ode23tb

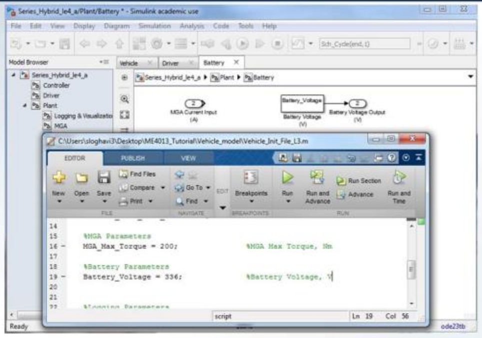
189

Battery

- For now, the battery voltage will be constant and defined in the init file
- Drag in a Constant block
  - Rename it Battery Voltage (V)
  - Give it a variable name of Battery_Voltage
  - Connect it to the Battery Voltage Output
- Update the init file for the battery voltage to be 336V

190

Battery



```
%MGA Parameters
MGA_Max_Torque = 200;            %MGA Max Torque, Nm

%Battery Parameters
Battery_Voltage = 336;           %Battery Voltage, V


%Logging Parameters
```

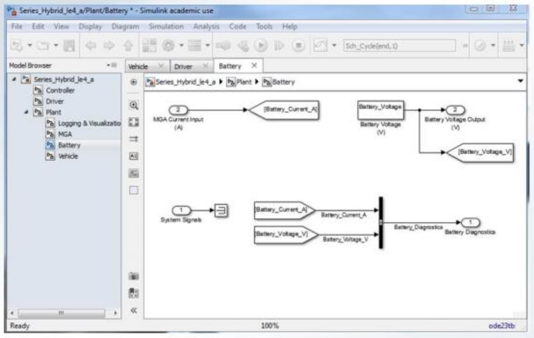
191

Battery

- Our magic battery can provide any amount of current required by the motor
- Put the battery current and the battery voltage on the Battery Diagnostics bus

192

Battery



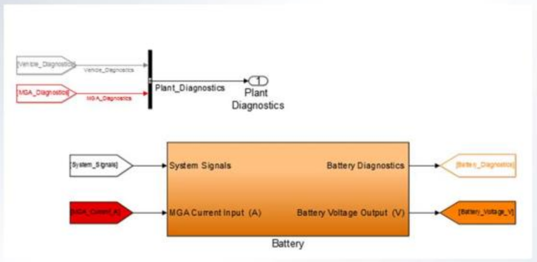
193

Battery

- Outstanding, we now have a constant voltage power source with unlimited amperage and capacity
- Return to the Plant level of the model
  - Make the Battery subsystem larger
  - Drag in three From and two Goto blocks
  - Wire and label as shown on the next slide
- Convention
  - Physical signals get colored background
  - CAN signals get colored foreground

194

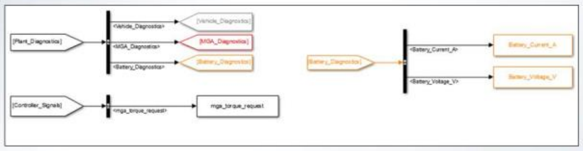
Battery



195

Battery

- Place the Battery Diagnostics onto the Plant CAN bus
- Extract the Battery signals in the Logging & Visualization subsystem



196

MGA

- Save the model as Series_Hybrid_le4_b.slx
- In the MGA subsystem, drag in an In1 and Out1 block
- Rename them
  - Battery Voltage Input (V)
  - MGA Current Output (A)
- For now, the electrical power from the battery will equal the mechanical power from the motor

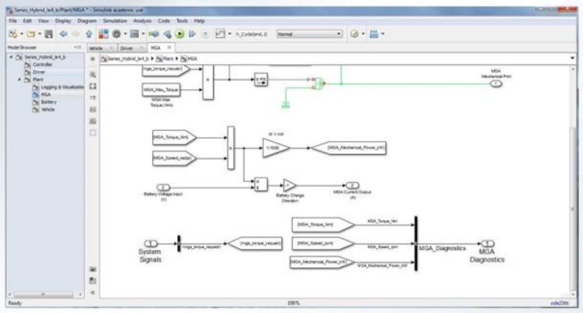
197

MGA

- Thus the current required will be the mechanical power divided by the battery voltage
- From
  - Simulink / Math Operations
- Drag in a Divide block
- When MGA is drawing current, it will come out of the battery
- Drag in a Gain block and give it a value of -1

198

MGA



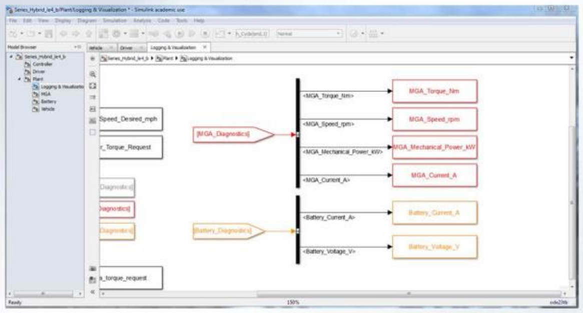
199

MGA

- Place the MGA Current signal onto the diagnostics bus
  - Label it MGA_Current_A
- Extract the signal in the Logging & Visualization subsystem

200

MGA / Logging & Visualization



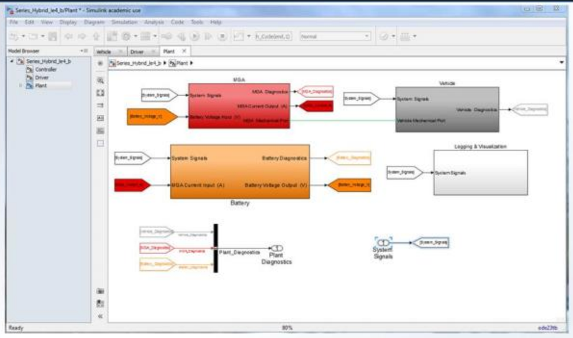
201

MGA

- Return to the Plant level of the model
- Make the MGA subsystem larger
- Use a Goto and a From block to connect MGA to the Battery
  - Color accordingly

202

MGA



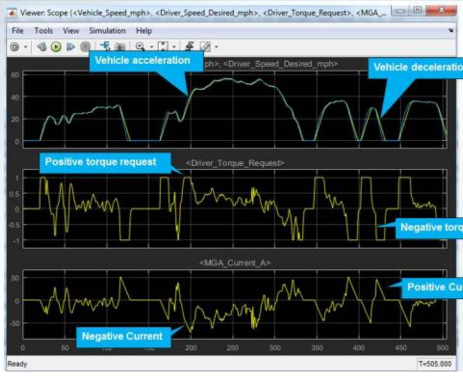
203

Logging & Visualization

- Let's now observe the MGA current as it follows the FU505 drive cycle
- Go to the Logging & Visualization subsystem
  - Add a third axis to the Scope
  - Place the MGA_Current_A signal on the third axis
  - Run the simulation

204

Results



205

Battery SOC

- Let's add battery charge to our model
- Battery Capacity is in units of Amp hours
  - The amount of charge transferred by a constant current of one amp for one hour
  - Thus we can integrate the battery current to find out how much charge has been removed
- This value can be non-dimensionalized by dividing by the maximum capacity of the battery to get the State of Charge (SOC)
  - 1 fully charged
  - 0 fully discharged

206

Battery SOC

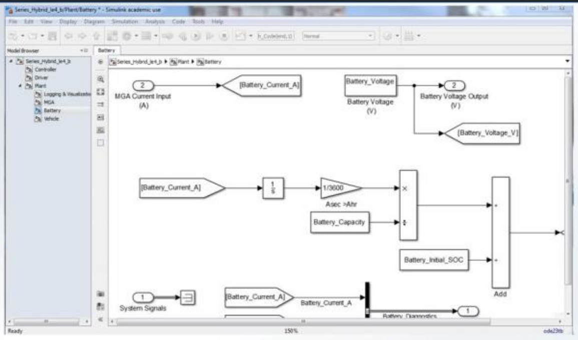
- In the Battery subsystem, drag in
  - An **Integrator** block
  - A **Gain** block
    - 1/3600
  - Two **Constant** blocks

Battery_Capacity	8.5 Ahrs
Battery_Initial_SOC	0.7

  - A **Divide** block
  - And a **Sum** block
- Arrange as shown on the next slide

207

Battery SOC



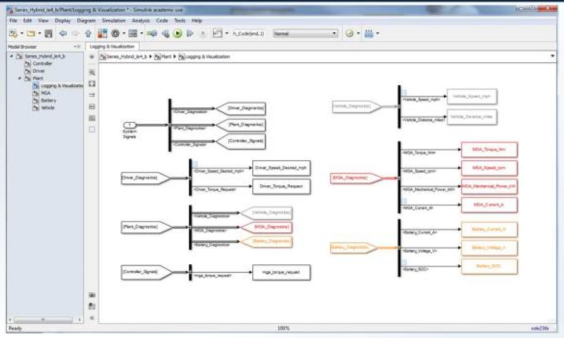
208

Battery SOC

- Add the Battery_SOC signal to the Battery Diagnostics bus
- Extract the signal in the Logging & Visualization subsystem
- Right click on the Driver_Torque_Request signal
  - Select Disconnect Viewer
  - Select Scope
  - Select Axes 2
- Add the SOC signal to Axes 2

209

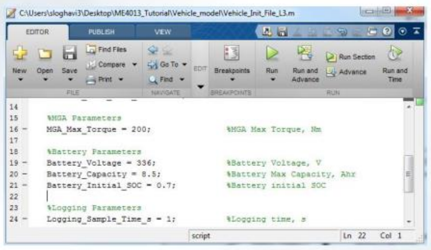
Logging & Visualization



210

Initialization

- The last step is to add the two new battery parameters to the init file

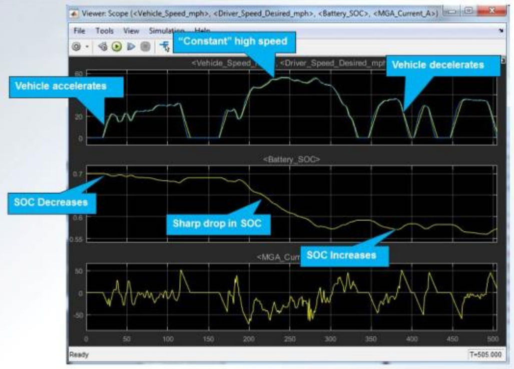


```
C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

%MGA Parameters
MGA_Max_Torque = 200;                   %MGA Max Torque, Nm

%Battery Parameters
Battery_Voltage = 336;                  %Battery Voltage, V
Battery_Capacity = 8.5;                 %Battery Max Capacity, Ahr
Battery_Initial_SOC = 0.7;              %Battery initial SOC

%Logging Parameters
Logging_Sample_Time_s = 1;              %Logging time, s
```

- Run the Simulation

211

Results

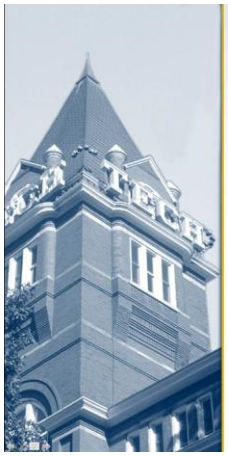


212

Review

- Developed an electromechanical model of MGA
  - 100% efficient
  - Constant torque capable
  - No rpm limits
- Developed an electrical battery model
  - 100% efficient
  - Constant voltage capable
  - No current limits
  - SOC ready

213



MBSD Lecture 5

Simple Charging logic

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Outline

- Simple Battery charging
- Simple GenSet charge logic

215

Battery SOC

- While having our vehicle follow a drive cycle, the Battery SOC decreased
- Eventually, the Battery SOC will get too low
  - Permanent damage
  - Stranded motorist
- This will require an on-board power generating unit
  - Diesel engine and MGB

216

Battery SOC / GenSet

- The net effect of the GenSet is to provide power to the battery
  - MGB current at Battery voltage
- Let's make a ridiculously simple MGB
  - Constant current output
- Rename the model Series_Hybrid_le5_a.slx
- In the plant level of the model, drag in a new Subsystem and rename it MGB
  - Make the background green

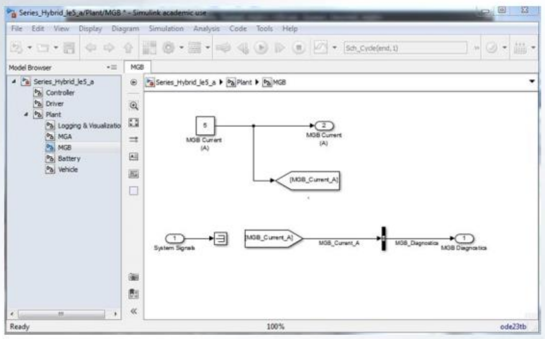
217

MGB

- In the MGB subsystem
  - Delete the connecting wire
  - Create the System Signals and MGB Diagnostics bus
  - Drag in a Constant block
    - Rename it MGB Current (A)
    - Give it a value of 5
  - Drag in an Out1 block
    - Rename it MGB Current Output (A)
  - Put the current on the Diagnostics Bus

218

MGB



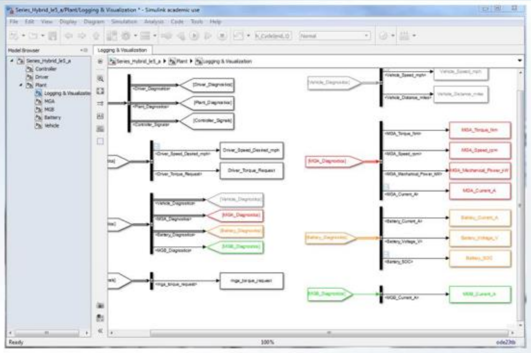
219

MGB

- Return to the Plant level and put the MGB Diagnostics on the Plant Diagnostics Bus
- Connect a System Signals From block
- Go to the Logging & Visualization subsystem and extract the MGB diagnostics
- Color accordingly!

220

MGB


Series_Hybrid_le5_a/Plant/Logging & Visualization * - Simulink academic use
Logging & Visualization
Series_Hybrid_le5_a ▸ Plant ▸ Logging & Visualization
Ready
100%
ode23tb

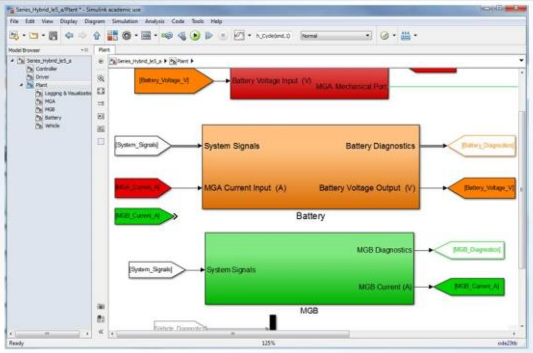
221

MGB / Battery

- Return to the Plant level of the model and use a Goto and a From block to make the electrical connection to the Battery

222

MGB / Battery



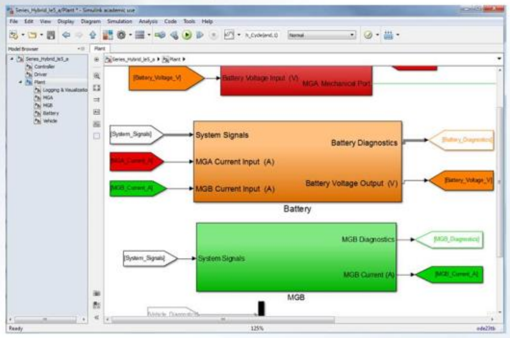
223

Battery

- Drag in an In1 block and rename it MGB Current Input (A)
- Drag in a Sum block and add the two Current Inputs together
- Connect the output to the Battery_Current Goto block
- Excellent – both MGs can remove/provide current from/to the Battery
- Return to the Plant level and connect the MGB_Current_A From block

224

Battery



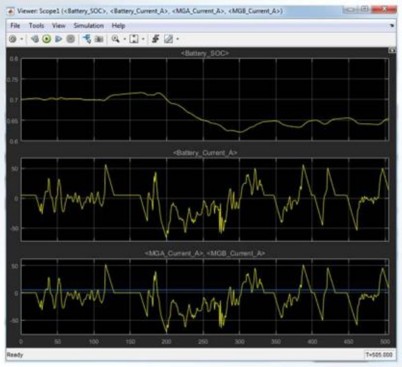
225

Battery

- In the Logging & Visualization subsystem
  - Right click on the Battery_SOC signal
  - Create a new scope
  - Give it three axis
    - Axis 1 – Battery SOC
    - Axis 2 – Battery Current
    - Axis 3 – MGA and MGB Current
- Run the simulation

226

Results



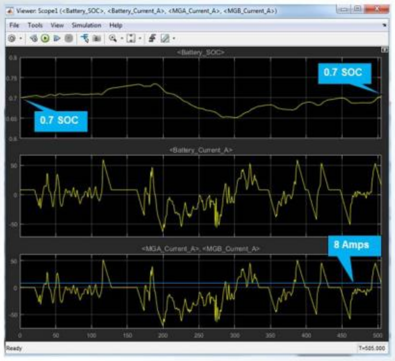
227

Results

- The "trickle charge" of a constant 5 amps kept the SOC up but the Battery was still *charge depleting*
- Vary the value of the MGB current until the Battery is relatively *charge sustaining*

228

Results



229

GenSet Controls

- Our series electric vehicle will use a genset which will be activated intermittently
  - SOC too low : start charging battery
  - SOC too high : stop charging battery
- Let's make the idealized genset do exactly that via MGB and a state machine
- Go to the Controller level of the model

230

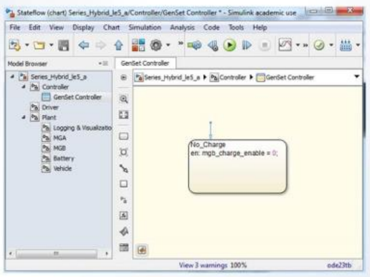
GenSet Controls

- From
  - Simulink / Stateflow
- Drag in a Chart block
  - Rename it GenSet Controller
  - Double click on it
- The first logical state will be the "no charge" state
  - The Battery SOC is above the lower limit

231

Stateflow

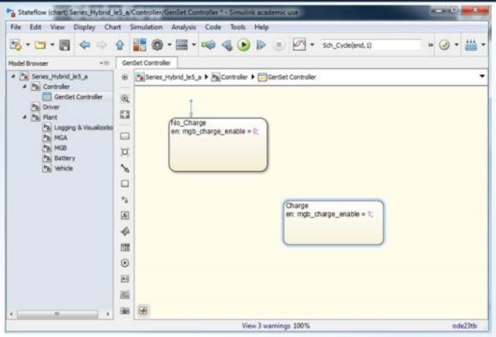
- Left click on the State icon
- Left click on the window
- Name the state
  - No_Charge
- On entry, set the MGB charge enable to 0
- Create a charge state



```
No_Charge
en: mgb_charge_enable = 0;
```

232

Stateflow



```
No_Charge
en: mgb_charge_enable = 0;

Charge
en: mgb_charge_enable = 1;
```

233

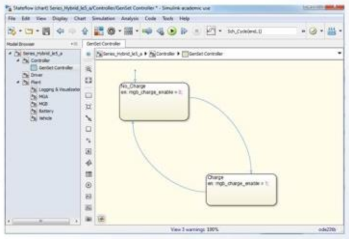
Stateflow

- To go from one state to another, a *transition* must occur
- These transitions are controlled by parameters called *guards*
- If, at the instance the guard is queried, the parameters are true then the transition occurs
- Let's now create the transitions followed by the guards

234

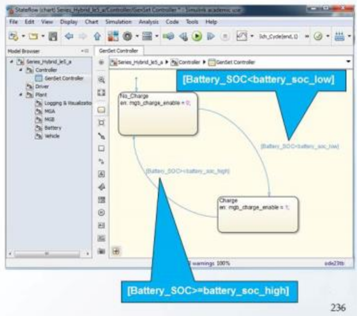
Stateflow

- Place the mouse over the No_Charge state
- Left Click
- Hold and drag to the Charge state
- Connect the Charge back to the No_Charge



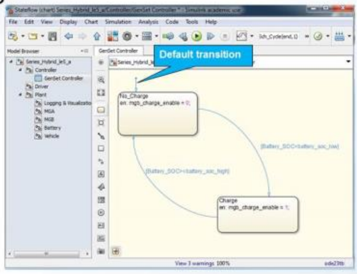
Stateflow

- Left click on the right transition
- Replace the question mark with the code at right
  - Enclosed in square brackets
- Repeat for the second transition


[Battery_SOC<battery_soc_low]
[Battery_SOC>=battery_soc_high]

Stateflow

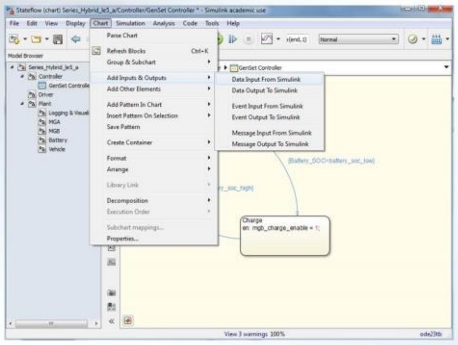
- Stateflow needs to know which state to "wake up" in
- The default transition was created when the first state was placed



Stateflow

- We need to add the Battery_SOC input from Simulink into Stateflow
  - Select
    - Chart / Add Inputs & Outputs / Data Input From Simulink
  - Change the Name to Battery_SOC
  - Click OK
- If you select
  - Tools / Model Explorer
- You can see that Battery_SOC has been added
- Close Model Explorer

Stateflow



Stateflow

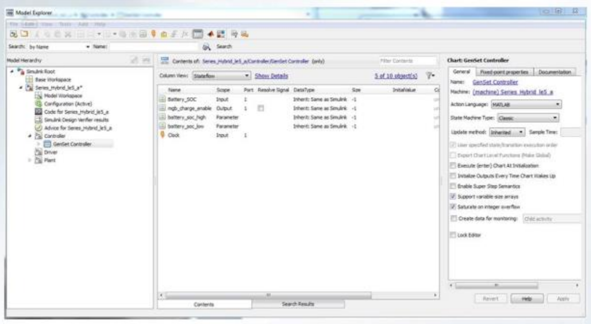
- Similarly we need to add the value of mgb_charge_enable to Simulink
  - Select
    - Chart / Add Inputs & Outputs / Data Output to Simulink
  - Change the Name to mgb_charge_enable
  - Click OK
- We also need to add the guards as parameters
  - Select
    - Chart / Add Other Elements / Parameter…
  - Change the Name to battery_soc_low
  - Click OK
- Repeat for battery_soc_high

Stateflow

- The final step is to set the frequency at which the guards are queried
  - Select
    - Chart / Add Inputs & Outputs / Event Input From Simulink
  - Change the Name to Clock
  - Click OK
- After this step, Model Explorer should be as shown in the next slide

241

Model Explorer



Name	Scope	Port	DataType
Battery_SOC	Input	1	Inherit: Same as Simulink
mgb_charge_enable	Output	1	Inherit: Same as Simulink
battery_soc_high	Parameter		Inherit: Same as Simulink
battery_soc_low	Parameter		Inherit: Same as Simulink
Clock	Input	1	

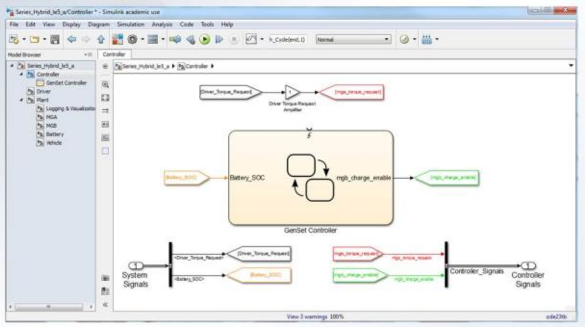
242

GenSet Controller

- Return to the Controller level of the model
- Make the GenSet Controller subsystem larger
- Extract the Battery_SOC from the System Signals bus and route it to the GenSet Controller
- Route the mga_charge_enable signal to the Control Signals bus and add it

243

GenSet Controller



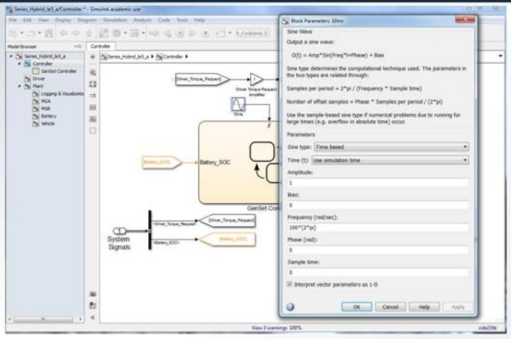
244

GenSet Controller

- From
  - Simulink / Sources
- Drag in a Sine Wave block
  - Rename it 10ms
- Connect it to the Clock Port
- Double Click on it
  - Change the Frequency to 100*(2*pi)
  - Click OK

245

GenSet Controller



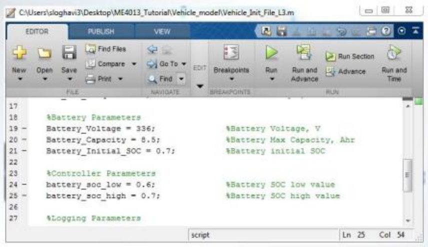
246

GenSet Controller

- Excellent, we have built a simple two state controller which, based on the Battery SOC, will determine whether or not it should be charged
- This check will be performed every 10 ms
- In the init file, we need to add guards
  - Low SOC : 0.6
  - High SOC : 0.7
- Add to the init file

247

GenSet Controller



C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

```
%Battery Parameters
Battery_Voltage = 336;              %Battery Voltage, V
Battery_Capacity = 8.5;             %Battery Max Capacity, Ahr
Battery_Initial_SOC = 0.7;          %Battery initial SOC

%Controller Parameters
battery_soc_low = 0.6;              %Battery SOC low value
battery_soc_high = 0.7;             %Battery SOC high value

%Logging Parameters
```

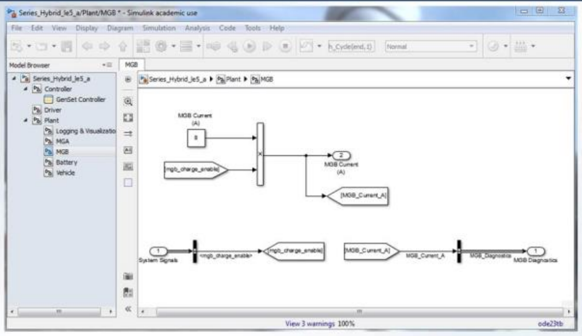
248

MGB

- Return to the MGB subsystem
- Extract the mgb_charge_enable signal from the System Signals
- Drag in a Product block and multiply the mgb_charge_enable signal with the MGB Current block
- Run the simulation

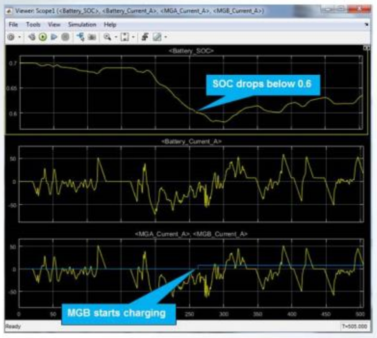
249

MGB



250

Results



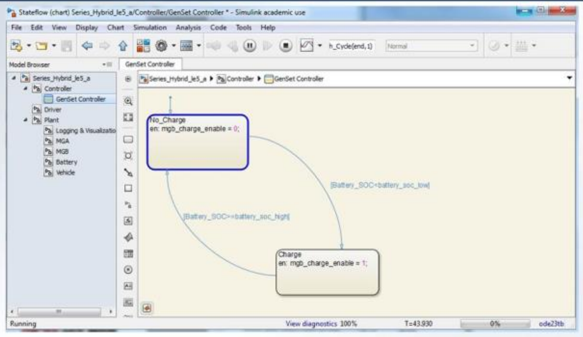
251

Results

- In the init file, change the drive cycle to
  - Schedule_FU505_Ten_Times.mat
- Increase the MGB current to 50 amps
- Run the simulation
  - MGB now cycles on and off
  - Watch the Stateflow chart
    - The current state is highlighted during the simulation

252

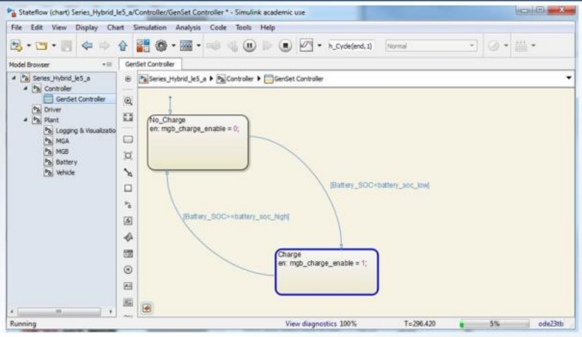
Results: MGB not charging


Stateflow (chart) Series_Hybrid_le5_a/Controller/GenSet Controller * - Simulink academic use

```
No_Charge
en: mgb_charge_enable = 0;

[Battery_SOC<battery_soc_low]

[Battery_SOC>=battery_soc_high]

Charge
en: mgb_charge_enable = 1;
```

253

Results: MGB charging


Stateflow (chart) Series_Hybrid_le5_a/Controller/GenSet Controller * - Simulink academic use

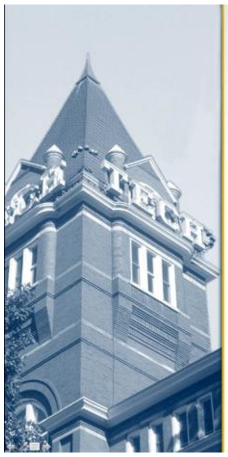
```
No_Charge
en: mgb_charge_enable = 0;

[Battery_SOC<battery_soc_low]

[Battery_SOC>=battery_soc_high]

Charge
en: mgb_charge_enable = 1;
```

254

GenSet and Stateflow

- Review
  - Created a simple GenSet which provides current to recharge the battery
  - Tweaked the trickle charge current for charge sustaining
  - Created a state controller to turn the charge on and off

255



MBSD Lecture 6

Improved GenSet and Controls

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Outline

- Improved MGB model
- Simple diesel engine model
- Charging logic

257

Improved GenSet

- Previously, the GenSet simply provided a constant current
- Improvements
  - Diesel Engine
    - Constant torque
    - Speed control
  - MGB
    - Recycle MGA model
- Rename your model
  - Series_Hybrid_le6_a.slx

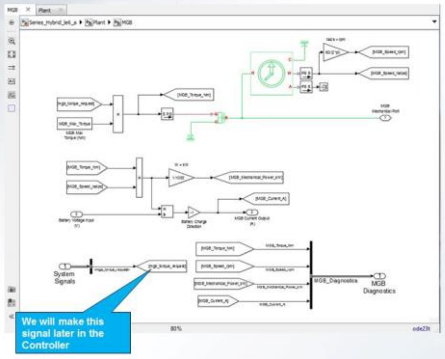
258

Improved MGB

- Delete the MGB model you just got done making
- Copy the MGA model
  - Make it green
  - Rename EVERYTHING from MGA to MGB
- Add the MGB max torque to the init file
  - 200 Nm
- Update the Logging & Visualization for MGB
  - Note: We will create the signal for mgb_torque_request later in the controller subsystem

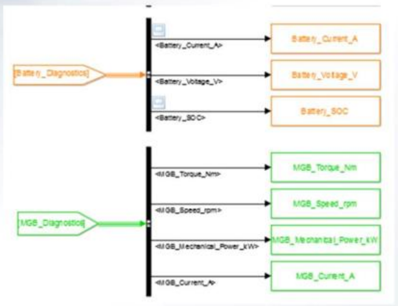
259

Improved MGB



260

Logging & Visualization



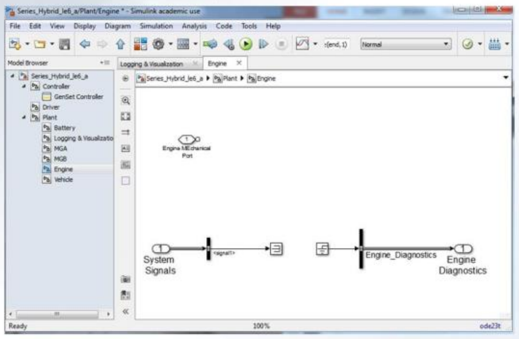
261

Engine

- Return to the Plant level, drag in a Subsystem, and rename it Engine
  - Make it yellow
- In the Engine subsystem, create the System Signals and Engine Diagnostics bus
- Drag in a Connection Port for the mechanical connection to MGB

262

Engine



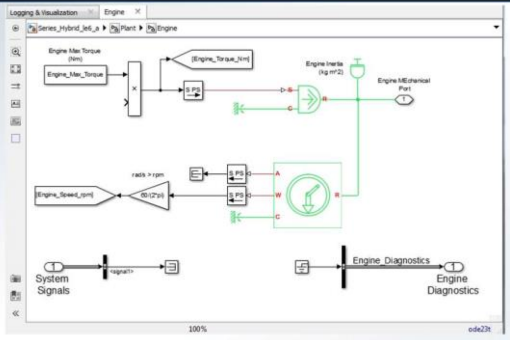
263

Engine

- Drag in
  - One Constant
  - One Simulink-PS Converter
  - One Ideal Torque Source
  - One Inertia
  - One Ideal Rotational Motion Sensor
  - One Gain
  - One Terminator
  - One Product
  - Two Goto
  - Two PS-Simulink Converters
  - Two Mechanical Rotational References
- Arrange and label as shown on the next slide

264

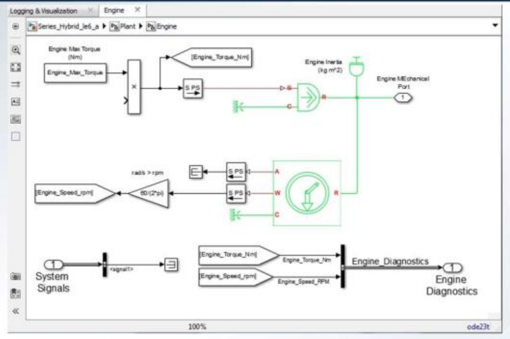
Engine



Engine

- Our engine torque will be modulated with a throttle request from the Controller
  - Range of 0 to 1
  - Analogous to a driver stepping on the accelerator pedal
- Place the engine torque and speed on the Diagnostic bus

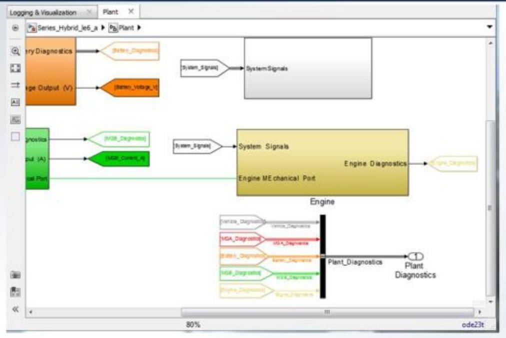
Engine



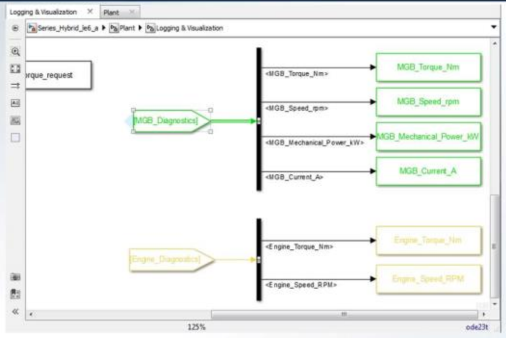
Engine

- Return to the Plant level of the model
- Connect a System Signals From block
- Place the Engine Diagnostics on the CAN Bus
- Connect the Engine mechanical port to the MGB mechanical port
- Go to the Logging & Visualization and extract the Engine diagnostics

Engine

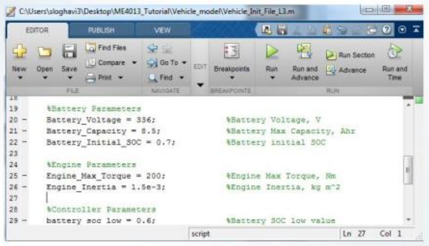


Logging & Visualization



Engine

- Now place the Engine parameters into the init file
  - Engine_Max_Torque = 200 Nm
  - Engine_Inertia = 1.5e-3 kg m^2

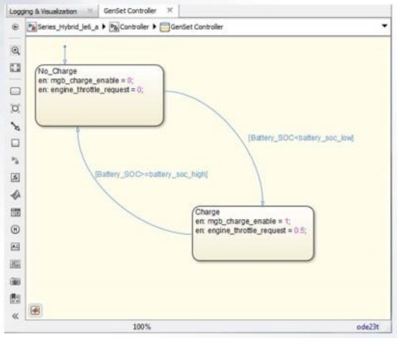

C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

```
%Battery Parameters
Battery_Voltage = 336;                %Battery Voltage, V
Battery_Capacity = 8.5;               %Battery Max Capacity, Ahr
Battery_Initial_SOC = 0.7;            %Battery initial SOC

%Engine Parameters
Engine_Max_Torque = 200;              %Engine Max Torque, Nm
Engine_Inertia = 1.5e-3;              %Engine Inertia, kg m^2

%Controller Parameters
battery soc low = 0.6;                %Battery SOC low value
```

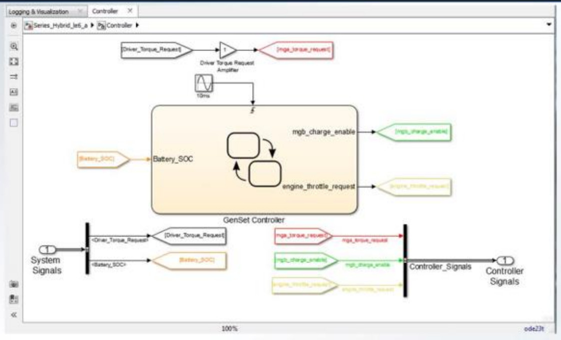
Engine / Controller

- To keep things simple
  - No Charge State
    - MGB charge enable = 0
    - Engine throttle request = 0
  - Charge State
    - MGB charge enable = 1
    - Engine throttle request = 0.5
- Add the new engine throttle data to the Stateflow Controller
- Place the engine throttle signal on the bus

Controller


Logging & Visualization
GenSet Controller
Series_Hybrid_le6_a ▸ Controller ▸ GenSet Controller
No_Charge
en: mgb_charge_enable = 0;
en: engine_throttle_request = 0;
[Battery_SOC<battery_soc_low]
[Battery_SOC>=battery_soc_high]
Charge
en: mgb_charge_enable = 1;
en: engine_throttle_request = 0.5;
100%
ode23t

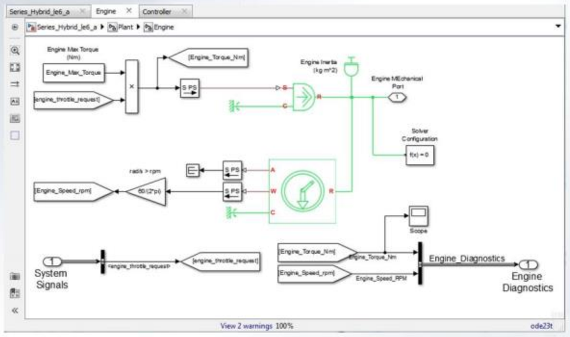
Controller



Engine / GenSet

- Return to the Engine subsystem and extract the throttle request signal
- Excellent, the first pass Engine and associated improved GenSet is complete
- Drag in a Solver Configuration block and connect the block as shown in the next slide

Engine

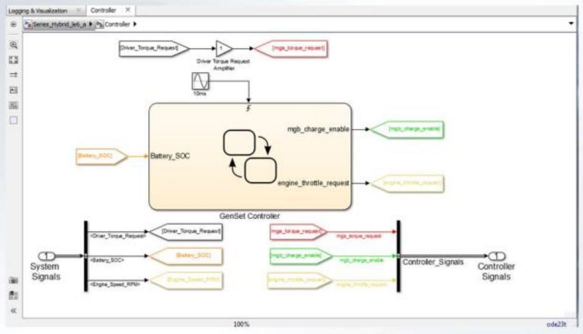


Controller

- The controller will need to send a torque request to MGB to hold the engine at a constant desired speed
  - Engine too fast – increase torque
  - Engine too slow – decrease torque
- Go to the Controller level of the model
- Extract the engine speed signal

277

Controller



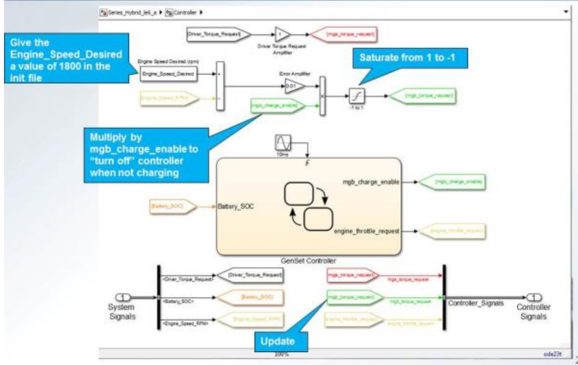
278

Controller

- As done with the Driver subsystem
  - The actual engine speed will be subtracted from the desired speed
  - This will create an error signal
  - The error signal will be amplified
  - The amplified signal will be saturated
  - The resulting torque request will be sent to MGB
- Implement this feedback controller
- Update the Control Signals bus

279

Controller


Give the Engine_Speed_Desired a value of 1800 in the init file

Saturate from 1 to -1

Multiply by mgb_charge_enable to "turn off" controller when not charging

Update

280

MGB and Visualization

- Go to the MGB subsystem and extract the mgb_torque_request signal
- Go to the Logging & Visualization subsystem
  - Right click on the Vehicle_Speed_mph signal
  - Open the Scope
  - Add the MGB_Current_A signal to the third axis
  - Create a fourth axis and put the Engine_Speed_RPM signal on it

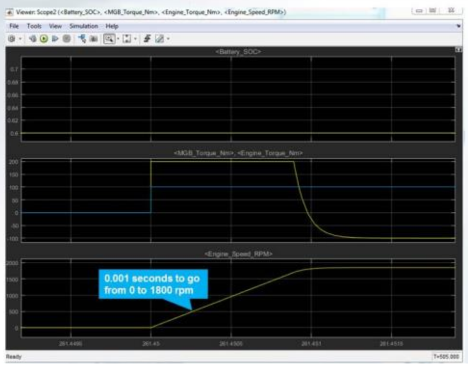
281

Visualization

- Create a new scope
  - Axis 1 : Battery SOC
  - Axis 2 : Engine Torque and MGB Torque
  - Axis 3 : Engine Speed
- Run the FU505 cycle to save time

282

Results



283

Improved GenSet & Controls

- Review
  - Made MGB a duplicate of MGA
  - Created a very simple diesel engine
  - Updated the controller to intermittently obtain current from the GenSet

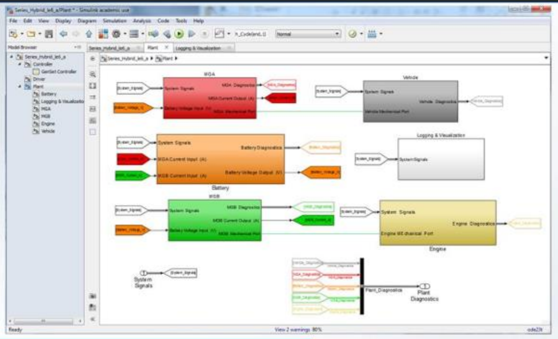
284

Lectures 0 - 6

- We built a model of a series electric vehicle
  - Plant
  - Driver
  - Controller
- All of the plant components are crude
  - System response could be intuited
  - Complex system was established
- Component refinement is next
  - Controller will need to be updated

285

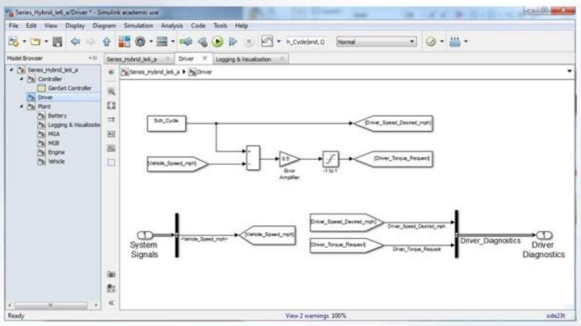
Lectures 0 - 6



The high level plant model will not change!

286

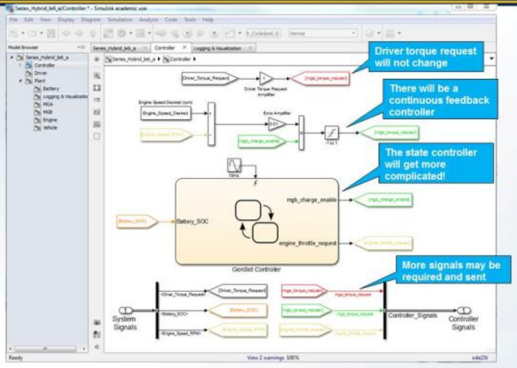
Lectures 0 - 6



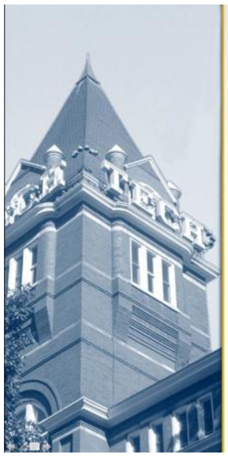
The Driver will not change!

287

Lectures 0 - 6


Driver torque request will not change
There will be a continuous feedback controller
The state controller will get more complicated!
More signals may be required and sent
GenSet Controller
Battery_SOC
mgb_charge_enable
engine_throttle_request
System Signals
Controller Signals

288



MBSD Lecture 7

Improved Engine Model and MPGGE SOC Fuel Efficiency

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Outline

- Improve Engine model
  - Experimental data for torque curves
  - Experimental data for fuel consumption
  - Controller refinements
- Fuel Efficiency
  - Calculate MPG, MPGGE, and MPGGE with SOC correction

290

Experimental Data

- Using the engine dynamometer at Rose-Hulman, a small diesel engine was tested at various engine speeds and throttle posititions
- Data for torque (Nm) and fuel consumption (grams/sec) were collected
- Let's add this to our model
- Rename your model
  - Series_Hybrid_le7_a.slx

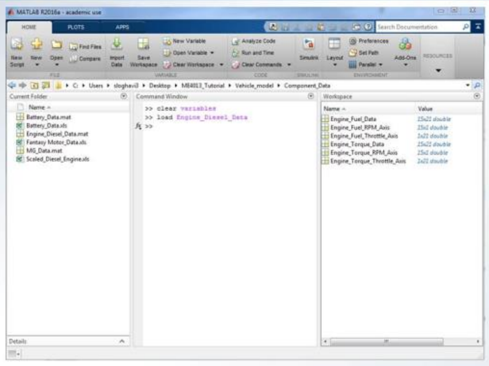
291

Experimental Data

- In the Matlab Window, go to
  - Current Directory window
  - Double click on the Component_Data folder
- At the command line, type
  - clear variables
  - load Engine_Diesel_Data
- The loaded variables will appear in the Workspace window

292

Experimental Data



```
>> clear variables
>> load Engine_Diesel_Data
>>
```

Name	Value
Engine_Fuel_Data	15x21 double
Engine_Fuel_RPM_Axis	15x1 double
Engine_Fuel_Throttle_Axis	1x21 double
Engine_Torque_Data	15x21 double
Engine_Torque_RPM_Axis	15x1 double
Engine_Torque_Throttle_Axis	1x21 double

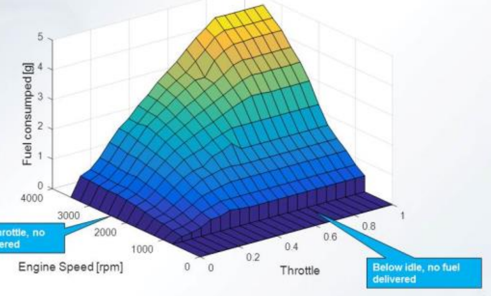
293

Experimental Data

- Note the size of the matrices
  - Engine_Fuel_Data : 15x21
  - Engine_Fuel_RPM_Axis: 15x1
  - Engine_Fuel_Throttle_Axis: 1x21
- Thus
  - The data rows are for given rpms
  - The data columns are for given throttle positions
- At the command line type
  - surf(Engine_Fuel_Throttle_Axis, Engine_Fuel_RPM_Axis, Engine_Fuel_Data)

294

Experimental Data



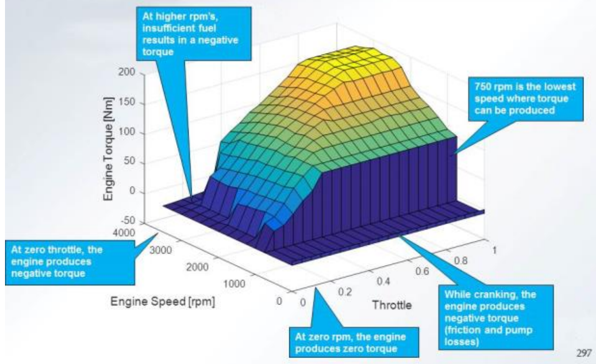
295

Experimental Data

- This engine does not have an idle circuit
  - 750 rpm is the lowest it can run
  - A throttle signal of zero will send no fuel
  - It will stall out
- This is great!
  - Our controller completely rules the throttle signal
- Modify the previous surf command to view the torque data

296

Experimental Data



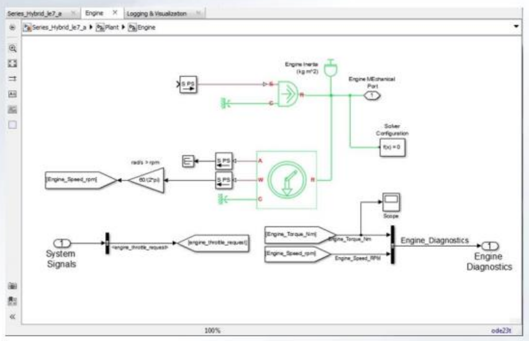
297

Engine

- We will need to read in this data with a 2-d lookup table
- Go to the Engine subsystem and delete the
  - Engine_Max_Torque Constant
  - engine_throttle_request From
  - Product block
  - Engine_Torque_Nm Goto

298

Engine



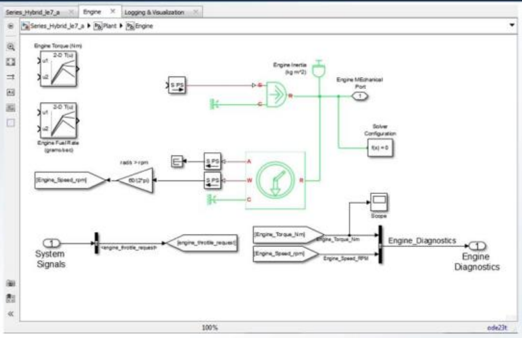
299

Engine

- From
  - Simulink / Lookup Tables
- drag in two
  - 2-D Lookup Table blocks
- Rename them
  - Engine Torque (Nm)
  - Engine Fuel Rate (grams/sec)

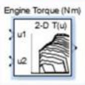
300

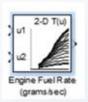
Engine



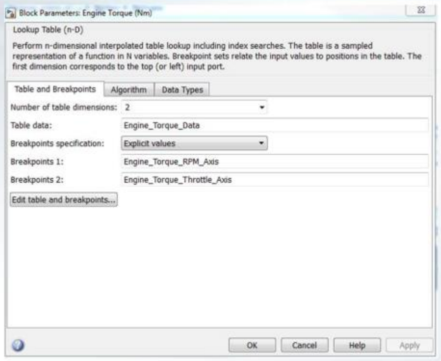
Engine

- Double click on the Engine Torque table
- In the dialogue box, change
  - Table data: Engine_Torque_Data
  - Breakpoints 1: Engine_Torque_RPM_Axis
  - Breakpoints 2: Engine_Torque_Throttle_Axis
- Click OK
- Note the shape of the table
  - It looks like the data!
- Repeat for the Fuel Rate table





Engine



Block Parameters: Engine Torque (Nm)

Lookup Table (n-D)

Perform n-dimensional interpolated table lookup including index searches. The table is a sampled representation of a function in N variables. Breakpoint sets relate the input values to positions in the table. The first dimension corresponds to the top (or left) input port.

Table and Breakpoints | Algorithm | Data Types

Number of table dimensions: 2

Table data: Engine_Torque_Data

Breakpoints specification: Explicit values

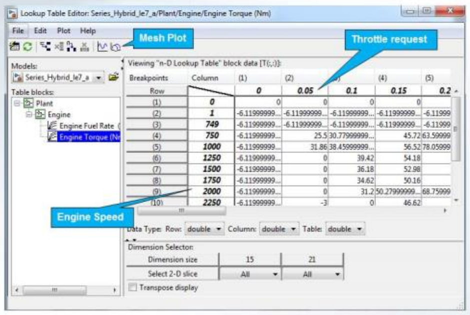
Breakpoints 1: Engine_Torque_RPM_Axis

Breakpoints 2: Engine_Torque_Throttle_Axis

Engine

- Double click on the Engine Torque table
- Click the Edit table and breakpoints… button
  - Note that all the data shows up
- Click on the Mesh Plot icon
- Verify that the shape is correct
  - Easy to make a mistake, especially with square matrix data!
- Close the mesh plot and the Editor
- Click Cancel on the dialogue box

Engine



Viewing "n-D Lookup Table" block data [T(:,:)]:

Breakpoints Row	Column	(1) 0	(2) 0.05	(3) 0.1	(4) 0.15	(5) 0.2
(1)	0	0	0	0	0	
(2)	1	-6.11999999...	-6.11999999...	-6.11999999...	-6.11999999...	-6.11999
(3)	749	-6.11999999...	-6.11999999...	-6.11999999...	-6.11999999...	-6.11999
(4)	750	-6.11999999...	25.5	30.77999999...	45.72	63.59999
(5)	1000	-6.11999999...	31.86	38.45999999...	56.52	78.05999
(6)	1250	-6.11999999...	0	39.42	54.18	
(7)	1500	-6.11999999...	0	36.18	52.98	
(8)	1750	-6.11999999...	0	34.62	50.16	
(9)	2000	-6.11999999...	0	31.2	50.27999999...	68.75999
(10)	2250	-6.11999999...	-3	0	46.62	

Data Type: Row: double Column: double Table: double

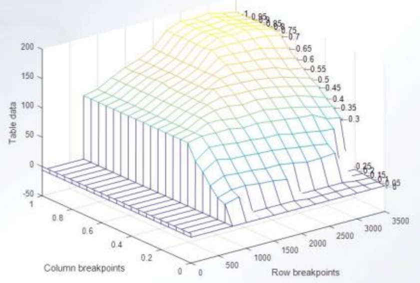
Dimension Selector:

Dimension size	15	21
Select 2-D slice	All	All

Transpose display

Engine

Table and breakpoints data for block: Series_Hybrid_le7_a/Plant/Engine/Engine Torque (Nm)

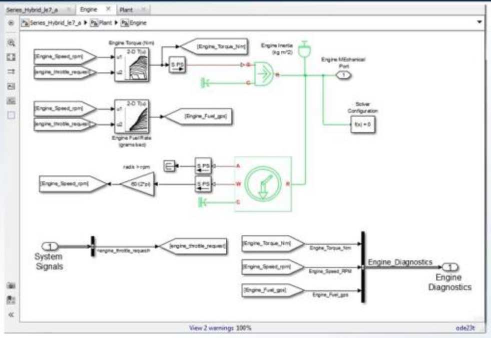


Engine & Logging

- In the Engine subsystem, route
  - An engine rpm signal to each table
    - Top input
  - A throttle request signal to each table
    - Bottom input
  - The torque output to the Simulink-PS Converter
  - The fuel rate output to the Engine Diagnostics bus
  - Put a Goto block on the Engine Torque signal: Engine_Torque_Nm
- In the Logging & Visualization subsystem
  - Extract the engine fuel rate signal

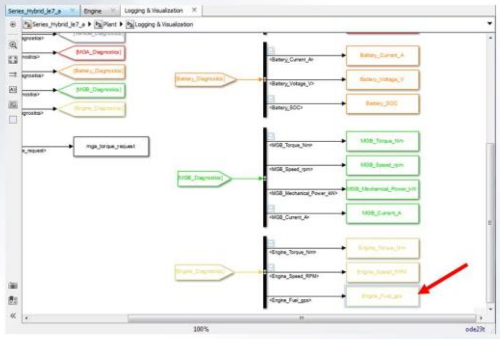
307

Engine & Logging



308

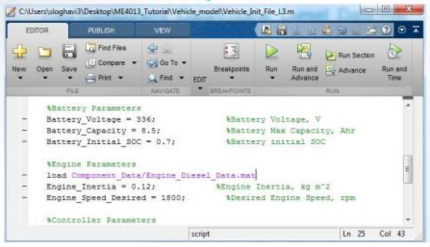
Engine & Logging



309

Engine & Initialization

- In the init file, add the line
  - load Component_Data/Engine_Diesel_Data.mat
- Update the engine inertia to 0.12
  - The crankshaft and flywheel



```
%Battery Parameters
Battery_Voltage = 336;              %Battery Voltage, V
Battery_Capacity = 8.5;             %Battery Max Capacity, Ahr
Battery_Initial_SOC = 0.7;          %Battery initial SOC

%Engine Parameters
load Component_Data/Engine_Diesel_Data.mat
Engine_Inertia = 0.12;            %Engine Inertia, kg m^2
Engine_Speed_Desired = 1800;        %Desired Engine Speed, rpm

%Controller Parameters
```

310

Directory

- Recall that we used the Matlab window to browse into the Component_Data directory
- Matlab MUST be in the same working directory as the Simulink model
- In the Matlab window
  - Click on the Go up one level icon
- Run the Simulink model

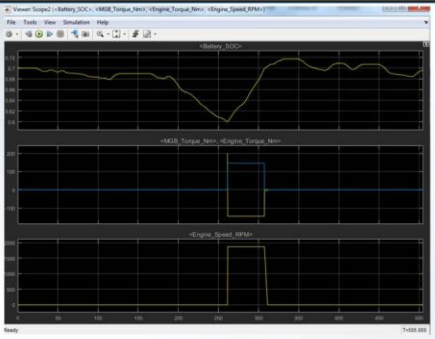
311

Results

- The results look very similar to those previous
  - The model still appears to be responding correctly
  - At 1800 rpm and 0.5 throttle, the engine torque should be about 150 Nm
  - Verify

312

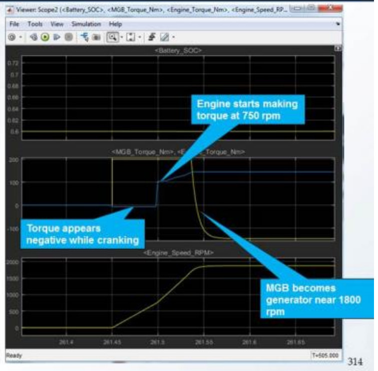
Results



313

Results

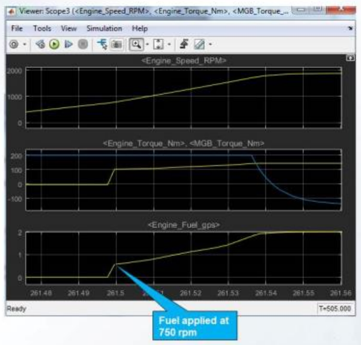
- Use the Zoom icon to zoom in on the time when the engine is starting
  - 261 seconds
- Lets's make a new scope for further investigation



314

Results

- For the new scope
  - Axis 1
    - Engine speed
  - Axis 2
    - Engine torque
    - MGB torque
  - Axis 3
    - Engine fuel rate
- Rerun the simulation



315

Results & Controller

- We don't want the engine to start making power until it is closer to the operating point
- An additional logic state will be needed
  - Set mgb_charge_enable to 1 to start engine
  - Set engine_throttle_request to 0 until "close" to operating rpm
- Then go into the Charge state

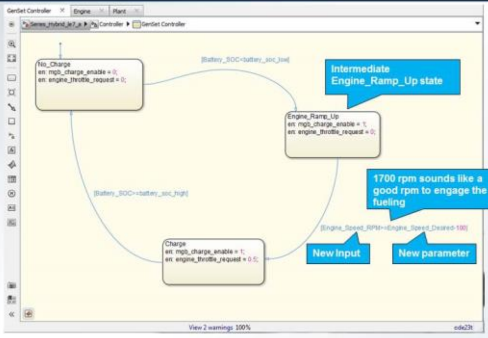
316

Controller

- Go to the Controller subsystem
  - Open the Stateflow GenSet Controller
  - Add this new state
    - Engine_Speed_RPM is a Simulink input
    - Engine_Speed_Desired is a parameter
  - What is a good value for "close" to 1800?

317

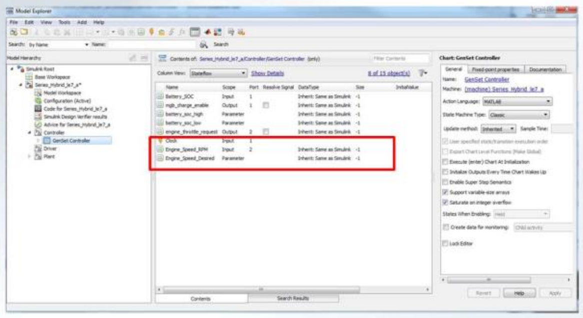
Controller



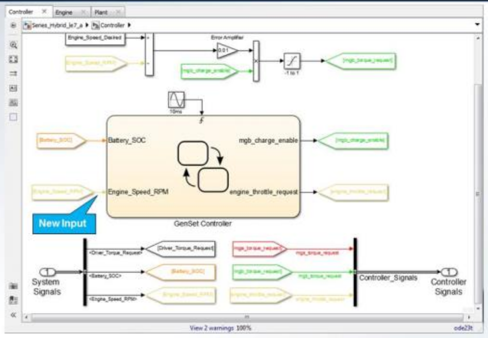
```
No_Charge
en: mgb_charge_enable = 0;
en: engine_throttle_request = 0;

[Battery_SOC<battery_soc_low]

Engine_Ramp_Up
en: mgb_charge_enable = 1;
en: engine_throttle_request = 0;

[Engine_Speed_RPM>=Engine_Speed_Desired-100]

Charge
en: mgb_charge_enable = 1;
en: engine_throttle_request = 0.5;

[Battery_SOC>=battery_soc_high]
```

318

Controller

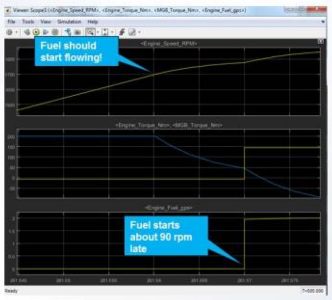


Controller



Results

- X-axis zoom in on the engine ramp up
- Y-axis zoom in on the engine speed
- The fuel rate is about 90 rpm late
  - Why?

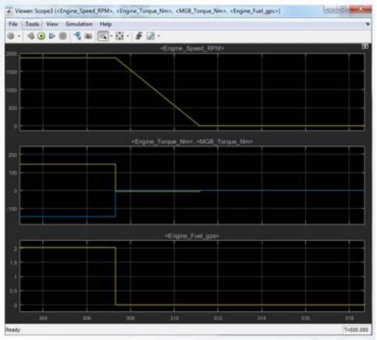


Results

- The ridiculous MGB starts the engine far too quickly
  - The transition guards are checked every 10 ms
  - In 10ms the engine speed goes from 1500 to almost 1800 rpm
  - The transition point was almost missed!
- Engine ramp-up rate will be addressed after we improve the MGB model

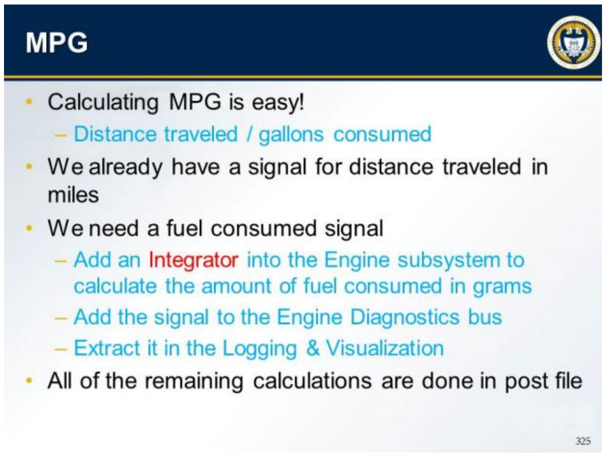
Results

- X-axis zoom in on the engine coast down
- Everything here looks fine
  - Fuel shuts off
  - Engine coasts down in about 4 seconds



MPG

- Let's now calculate the vehicle fuel efficiency
  - MPG
  - SOC correction
  - GE (gas equivalent)
- We will do this with a post-processing file
  - Simply for experience
  - Could be done in the Logging & Visualization subsystem

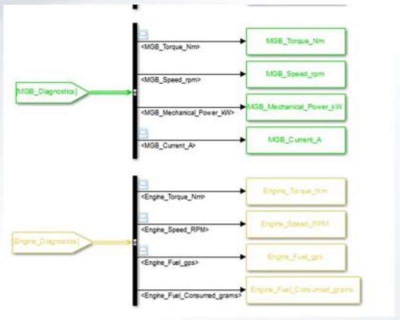


MPG

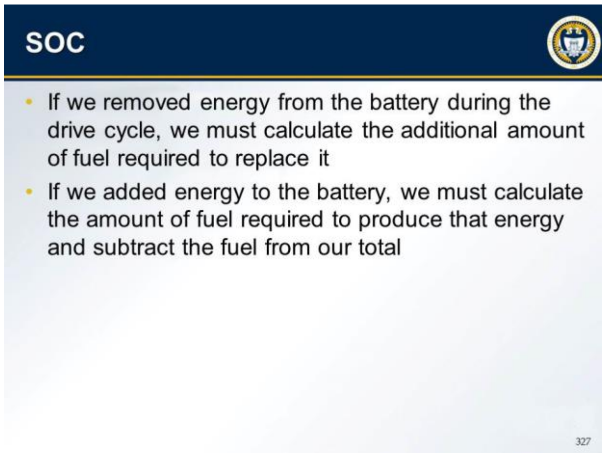
- Calculating MPG is easy!
  - Distance traveled / gallons consumed
- We already have a signal for distance traveled in miles
- We need a fuel consumed signal
  - Add an Integrator into the Engine subsystem to calculate the amount of fuel consumed in grams
  - Add the signal to the Engine Diagnostics bus
  - Extract it in the Logging & Visualization
- All of the remaining calculations are done in post file

325

Engine & Logging



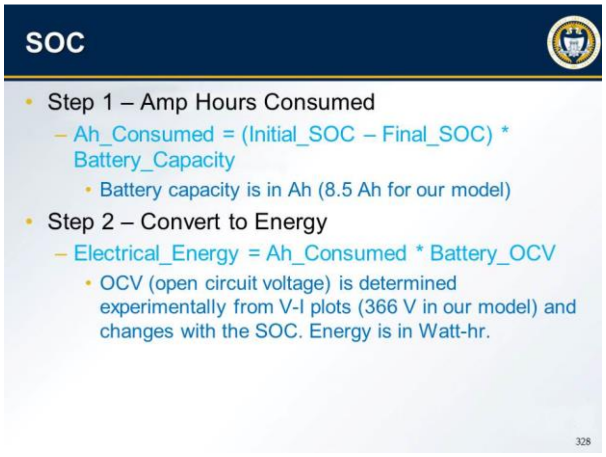
326



SOC

- If we removed energy from the battery during the drive cycle, we must calculate the additional amount of fuel required to replace it
- If we added energy to the battery, we must calculate the amount of fuel required to produce that energy and subtract the fuel from our total

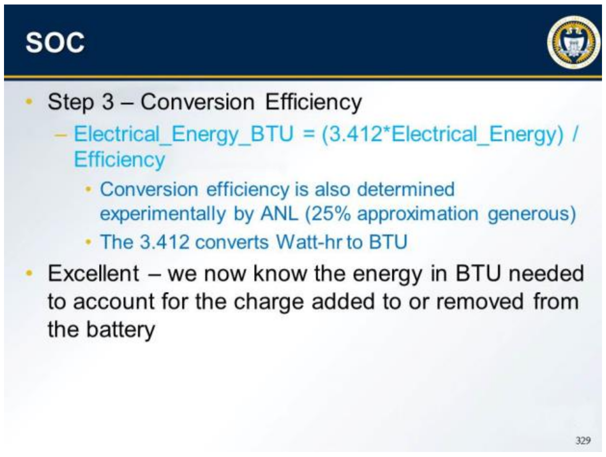
327



SOC

- Step 1 – Amp Hours Consumed
  - Ah_Consumed = (Initial_SOC – Final_SOC) * Battery_Capacity
    - Battery capacity is in Ah (8.5 Ah for our model)
- Step 2 – Convert to Energy
  - Electrical_Energy = Ah_Consumed * Battery_OCV
    - OCV (open circuit voltage) is determined experimentally from V-I plots (366 V in our model) and changes with the SOC. Energy is in Watt-hr.

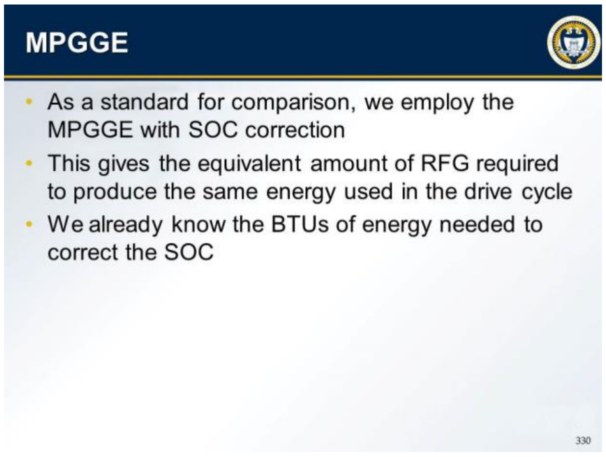
328



SOC

- Step 3 – Conversion Efficiency
  - Electrical_Energy_BTU = (3.412*Electrical_Energy) / Efficiency
    - Conversion efficiency is also determined experimentally by ANL (25% approximation generous)
    - The 3.412 converts Watt-hr to BTU
- Excellent – we now know the energy in BTU needed to account for the charge added to or removed from the battery

329



MPGGE

- As a standard for comparison, we employ the MPGGE with SOC correction
- This gives the equivalent amount of RFG required to produce the same energy used in the drive cycle
- We already know the BTUs of energy needed to correct the SOC

330

MPGGE

- Step 4 – Fuel Energy BTU
  - Fuel_Energy_BTU = Fuel_Consumed_Gallons* 133393.1102
- Step 5 – Total Energy BTU
  - Total_Energy_BTU = Electrical_Energy_BTU + Fuel_Energy_BTU
- 133393.1102 BTU/Gal is the fuel heating value of B-20
- Excellent – we now know the total energy consumed by our vehicle over the drive cycle

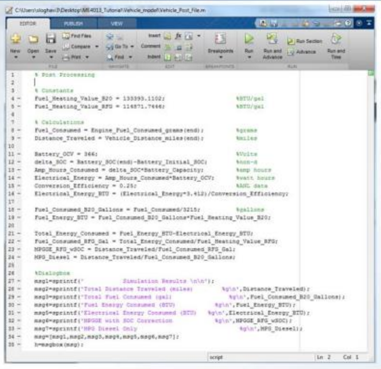
331

MPGGE

- Step 6 – RFG Required
  - Fuel_Consumed_RFG = Total_Energy_BTU / 114871.7446
    - This gives us the gallons of RFG required to produce the same energy
    - 114871.7446 BTU/gal is the fuel heating value of RFG
- Step 7 – MPGGE with SOC Correction
  - MPGGE = Distance_Travelled / Fuel_Consumed_RFG

332

Vehicle_Post_File.m File

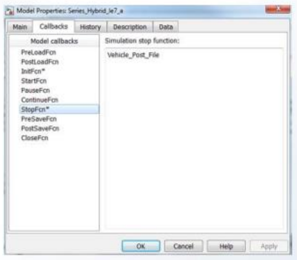


```
% Post Processing

% Constants
Fuel_Heating_Value_B20 = 133393.1102;                    %BTU/gal
Fuel_Heating_Value_RFG = 114871.7446;                    %BTU/gal

% Calculations
Fuel_Consumed = Engine_Fuel_Consumed_grams(end);         %grams
Distance_Traveled = Vehicle_Distance_miles(end);         %miles

Battery_OCV = 366;                                       %Volts
delta_SOC = Battery_SOC(end)-Battery_Initial_SOC;        %non-d
Amp_Hours_Consumed = delta_SOC*Battery_Capacity;         %amp hours
Electrical_Energy = Amp_Hours_Consumed*Battery_OCV;      %watt hours
Conversion_Efficiency = 0.25;                            %ANL data
Electrical_Energy_BTU = (Electrical_Energy*3.412)/Conversion_Efficiency;

Fuel_Consumed_B20_Gallons = Fuel_Consumed/3215;          %gallons
Fuel_Energy_BTU = Fuel_Consumed_B20_Gallons*Fuel_Heating_Value_B20;

Total_Energy_Consumed = Fuel_Energy_BTU-Electrical_Energy_BTU;
Fuel_Consumed_RFG_Gal = Total_Energy_Consumed/Fuel_Heating_Value_RFG;
MPGGE_RFG_wSOC = Distance_Traveled/Fuel_Consumed_RFG_Gal;
MPG_Diesel = Distance_Traveled/Fuel_Consumed_B20_Gallons;

%Dialogbox
msg1=sprintf('          Simulation Results \n\n');
msg2=sprintf('Total Distance Traveled (miles)        %g\n',Distance_Traveled);
msg3=sprintf('Total Fuel Consumed (gal)                %g\n',Fuel_Consumed_B20_Gallons);
msg4=sprintf('Fuel Energy Consumed (BTU)           %g\n',Fuel_Energy_BTU);
msg5=sprintf('Electrical Energy Consumed (BTU)   %g\n',Electrical_Energy_BTU);
msg6=sprintf('MPGGE with SOC Correction           %g\n',MPGGE_RFG_wSOC);
msg7=sprintf('MPG Diesel Only                          %g\n',MPG_Diesel);
msg=[msg1,msg2,msg3,msg4,msg5,msg6,msg7];
h=msgbox(msg);
```

333

Post Processing File

- Like the init file, the post file should run after every simulation
- Right click on the model and
  - Select Model Properties
  - Select Callbacks
  - Select StopFnc
  - Enter the name of the post processing file without the .m



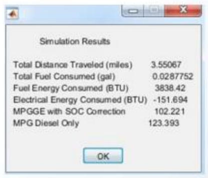
Simulation stop function:

Vehicle_Post_File

334

Results

- Rerun the simulation
- Not bad
  - 102 MPGGE w/SOC
  - 123 MPG straight
- Unrealistic?
  - 100% regen braking
  - 100% efficient battery
  - 100% efficient MGs
- We'll watch there numbers go down!



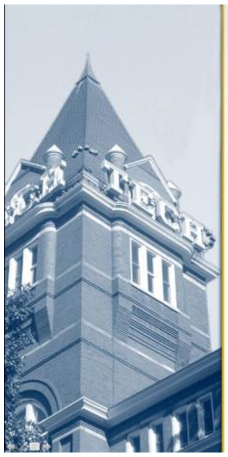
```
Simulation Results

Total Distance Traveled (miles)       3.55067
Total Fuel Consumed (gal)           0.0287752
Fuel Energy Consumed (BTU)            3838.42
Electrical Energy Consumed (BTU)     -151.694
MPGGE with SOC Correction             102.221
MPG Diesel Only                       123.393
```

335

Review

- Improve Engine model
  - Experimental data for torque curves
  - Experimental data for fuel consumption
  - Controller refinements
- Fuel Efficiency
  - Calculate MPG, MPGGE, and MPGGE with SOC correction

336



MBSD Lecture 8

Improved Battery Model

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

Outline

- Improve Battery model
  - First order math model
    - Variable battery voltage
  - Algebraic loops
  - Variable open circuit voltage
  - Variable charge/discharge resistance
  - Numeric error

338

Simple Battery

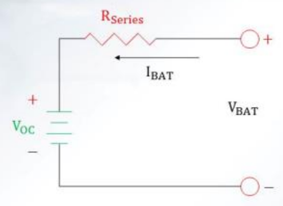
- Thus far, the battery model
  - Has had a constant voltage of 336V
  - Has had a constant OCV of 366V
  - Been able to provide infinite current
  - Has had no energy conversion losses
- Let's now improve the battery model by including the internal resistance

$$V_{BAT} = V_{OC} + I_{BAT} \times R_{Series}$$

  - The battery voltage is now a function of the current

339

Battery Terminal Voltage



- Current is defined as positive into the battery
- Positive current charges the battery and increases the battery SOC

340

Future Improvements

- For now, the open circuit voltage V_{OC} and the battery series resistance R_{Series} are a constant
- As our understanding of the model increases, we can make the battery model less ideal by:
  - Making V_{OC} a function of SOC and temperature
  - Having a different charge and discharge series resistances, both of which are a function of SOC and temperature
- These will all involve experimental (or manufacturer) data and 2-d lookup tables

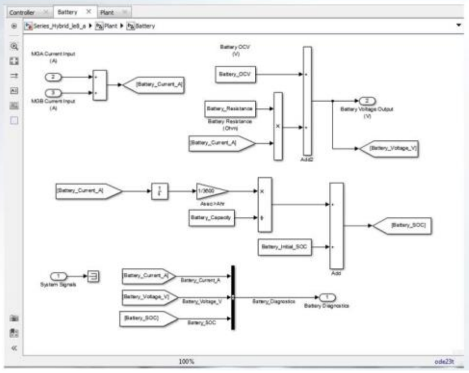
341

First Order Model

- The battery OCV and series resistance will be constants read from the init file
  - Battery_OCV = 366
  - Battery_Resistance = 0.4
- Add to the init file
- Remove the Battery_Voltage
- Remove the OCV from the post file
- Rename your model Series_Hybrid_le8_a.slx and implement the battery voltage equation

342

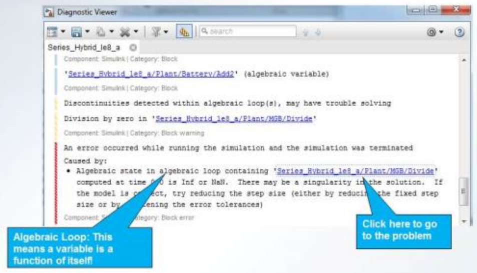
First Order Model



343

Results

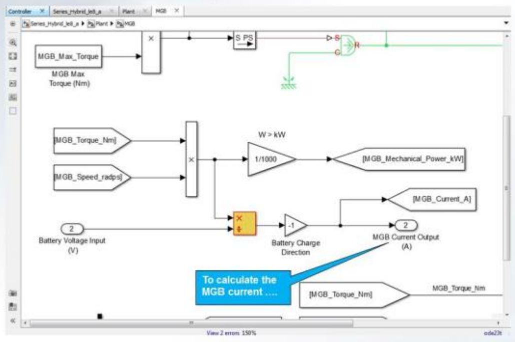
- Run the Simulation – Dooh!


Diagnostic Viewer
Series_Hybrid_le8_a
Algebraic Loop: This means a variable is a function of itself!
Click here to go to the problem

```
Component: Simulink | Category: Block
'Series_Hybrid_le8_a/Plant/Battery/Add2' (algebraic variable)
Component: Simulink | Category: Block
Discontinuities detected within algebraic loop(s), may have trouble solving
Division by zero in 'Series_Hybrid_le8_a/Plant/MGB/Divide'
Component: Simulink | Category: Block warning
An error occurred while running the simulation and the simulation was terminated
Caused by:
 • Algebraic state in algebraic loop containing 'Series_Hybrid_le8_a/Plant/MGB/Divide'
   computed at time 0.0 is Inf or NaN.  There may be a singularity in the solution.  If
   the model is correct, try reducing the step size (either by reducing the fixed step
   size or by tightening the error tolerances)
Component: Simulink | Category: Block error
```

344

Algebraic Loop



345

Algebraic Loop

- Aha! The loop involves the battery voltage which is a function of the battery current which is a function of the battery voltage …
- To break the algebraic loop a delay will be used to have either
  - The battery voltage signal lag the battery current calculation or
  - The battery current signal lag the battery voltage calculation
- Close the Simulations Diagnostics box

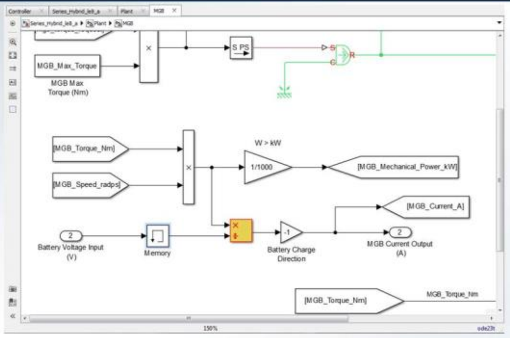
346

Algebraic Loop

- Go to the MGB subsystem where the algebraic loop was first identified
- From
  - Simulink / Discrete
- Drag in a Memory block
- Insert it just after the Voltage Input block
- Thinking ahead, MGA is also used to calculate the battery current
- Put a Memory block there too

347

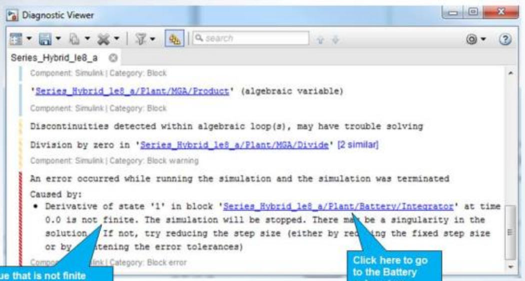
MGA/MGB Voltage Delay



348

Results

- Run the simulation and…



Diagnostic Viewer

Series_Hybrid_le8_a

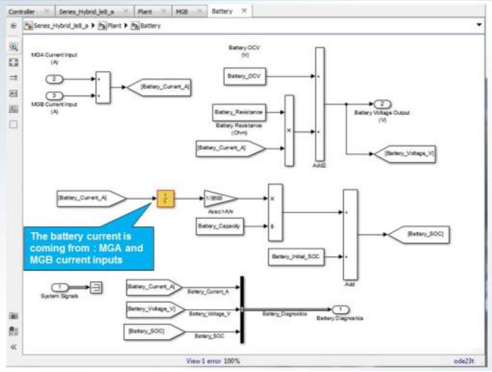
```
Component: Simulink | Category: Block
'Series_Hybrid_le8_a/Plant/MGA/Product' (algebraic variable)
Component: Simulink | Category: Block
Discontinuities detected within algebraic loop(s), may have trouble solving
Division by zero in 'Series_Hybrid_le8_a/Plant/MGA/Divide' [2 similar]
Component: Simulink | Category: Block warning
An error occurred while running the simulation and the simulation was terminated
Caused by:
 • Derivative of state '1' in block 'Series_Hybrid_le8_a/Plant/Battery/Integrator' at time
   0.0 is not finite. The simulation will be stopped. There may be a singularity in the
   solution. If not, try reducing the step size (either by reducing the fixed step size
   or by tightening the error tolerances)
Component: Simulink | Category: Block error
```

A value that is not finite means we are dividing by zero somewhere in the model … (Note that it occurs at time 0)

Click here to go to the Battery subsystem

349

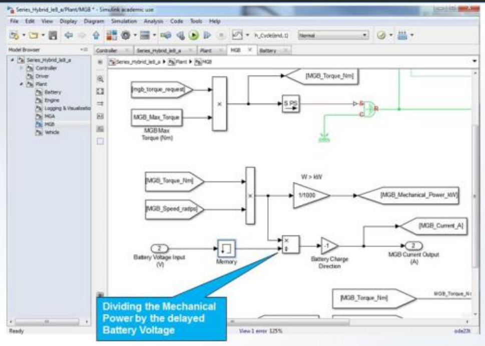
Not finite (infinity)



The battery current is coming from : MGA and MGB current inputs

350

Not finite (infinity)



Dividing the Mechanical Power by the delayed Battery Voltage

351

Not finite (infinity)

- At the start of the simulation (time zero), there is no delayed value of voltage
- What does the memory block do?
  - Double click on it
  - Aha! It uses zero as its first value…
  - which results in a divide by zero error at time zero
- The first strategy of delaying the Battery voltage signal was a bad one
- Undo the MGA/MGB improvements

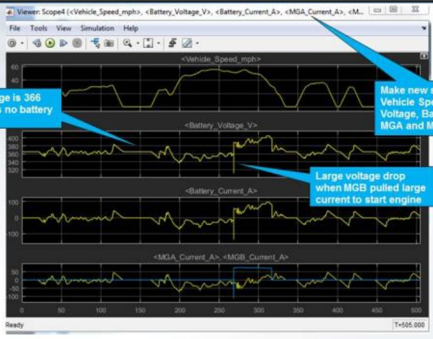
352

Algebraic Loop

- Let's try strategy two, delaying the Battery current in calculating the Battery Voltage
- Close the Simulation Diagnostics box
- Close the MPGGE dialogue boxes
- Go to the Battery subsystem
  - Insert the Memory block into the Battery voltage calculation
  - Verify that there are no downstream divides that could result in division by zero
- Run the simulation

353

Results



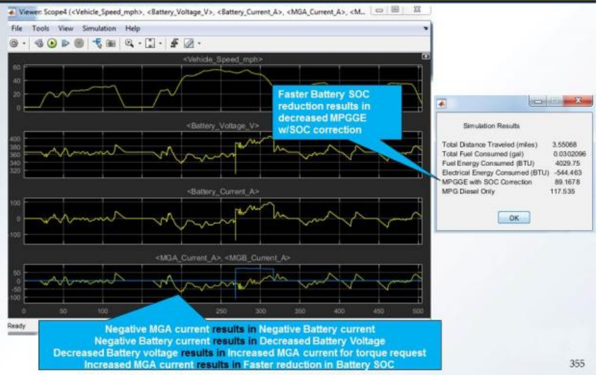
Battery Voltage is 366 when there is no battery current

Make new scope with Vehicle Speed, Battery Voltage, Battery current, MGA and MGB current

Large voltage drop when MGB pulled large current to start engine

354

Results


Viewer: Scope4 (<Vehicle_Speed_mph>, <Battery_Voltage_V>, <Battery_Current_A>, <MGA_Current_A>, <M...
<Vehicle_Speed_mph>
<Battery_Voltage_V>
<Battery_Current_A>
<MGA_Current_A>, <MGB_Current_A>
Faster Battery SOC reduction results in decreased MPGGE w/SOC correction
Negative MGA current results in Negative Battery current
Negative Battery current results in Decreased Battery Voltage
Decreased Battery voltage results in Increased MGA current for torque request
Increased MGA current results in Faster reduction in Battery SOC

```
Simulation Results

Total Distance Traveled (miles)      3.55068
Total Fuel Consumed (gal)            0.0302096
Fuel Energy Consumed (BTU)           4029.75
Electrical Energy Consumed (BTU)     -544.463
MPGGE with SOC Correction            89.1678
MPG Diesel Only                      117.535
```

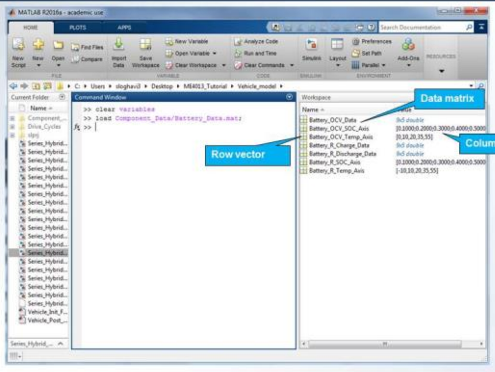
355

Results

- This simple improvement has substantially decreased our SOC corrected MPGGE
- Let's improve the model further and see how the fuel economy is affected
- First, we will add a variable OCV as a function of SOC and temperature
- At the Matlab command line type
  - clear variables
  - load Component_Data/Battery_Data.mat;
  - Click on the Workspace tab

356

Battery Data



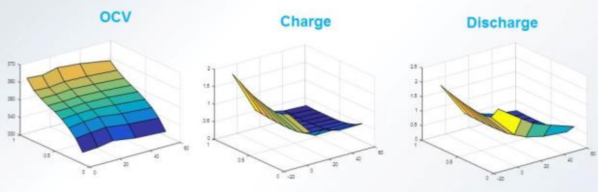
```
>> clear variables
>> load Component_Data/Battery_Data.mat;
>>
```

Name	Value
Battery_OCV_Data	9x5 double
Battery_OCV_SOC_Axis	[0.1000;0.2000;0.3000;0.4000;0.5000
Battery_OCV_Temp_Axis	[0,10,20,35,55]
Battery_R_Charge_Data	9x5 double
Battery_R_Discharge_Data	9x5 double
Battery_R_SOC_Axis	[0.1000;0.2000;0.3000;0.4000;0.5000
Battery_R_Temp_Axis	[-10,10,20,35,55]

357

Battery Data

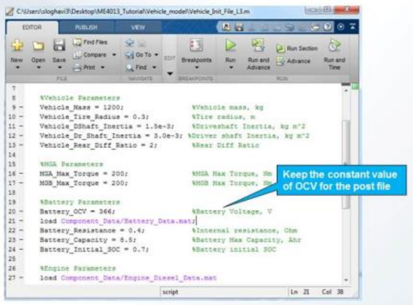
- At the command line, type
  - surf(Battery_OCV_Temp_Axis, Battery_OCV_SOC_Axis, Battery_OCV_Data)
- Repeat for the resistance data



358

Battery Data

- Add the loading of the battery data to the init file



```
%Vehicle Parameters
Vehicle_Mass = 1200;                     %Vehicle mass, kg
Vehicle_Tire_Radius = 0.3;               %Tire radius, m
Vehicle_DShaft_Inertia = 1.5e-3;         %Driveshaft Inertia, kg m^2
Vehicle_Dr_Shaft_Inertia = 3.0e-3;       %Driver shaft Inertia, kg m^2
Vehicle_Rear_Diff_Ratio = 2;             %Rear Diff Ratio

%MGA Parameters
MGA_Max_Torque = 200;                    %MGA Max Torque, Nm
MGB_Max_Torque = 200;                    %MGB Max Torque, Nm

%Battery Parameters
Battery_OCV = 366;                       %Battery Voltage, V
load Component_Data/Battery_Data.mat;
Battery_Resistance = 0.4;                %Internal resistance, Ohm
Battery_Capacity = 8.5;                  %Battery Max Capacity, Ahr
Battery_Initial_SOC = 0.7;               %Battery initial SOC

%Engine Parameters
load Component_Data/Engine_Diesel_Data.mat
```

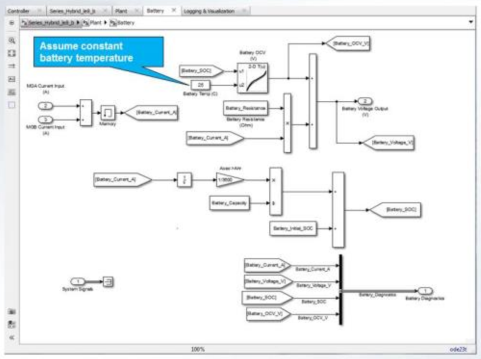
359

Variable OCV

- Save your model as
  - Series_Hybrid_le8_b.slx
- Go to the Battery subsystem
- Delete the Battery OCV Constant block
- Drag in a
  - 2-D Look up Table block
  - Constant block
- Add the OCV signal to the Battery Diagnostics Bus and extract it in the Logger

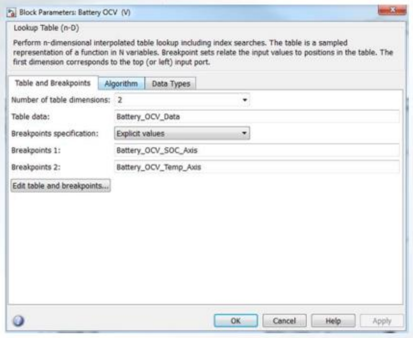
360

Variable OCV



361

Variable OCV



Block Parameters: Battery OCV (V)

Lookup Table (n-D)

Perform n-dimensional interpolated table lookup including index searches. The table is a sampled representation of a function in N variables. Breakpoint sets relate the input values to positions in the table. The first dimension corresponds to the top (or left) input port.

Table and Breakpoints | Algorithm | Data Types

Number of table dimensions: 2

Table data: Battery_OCV_Data

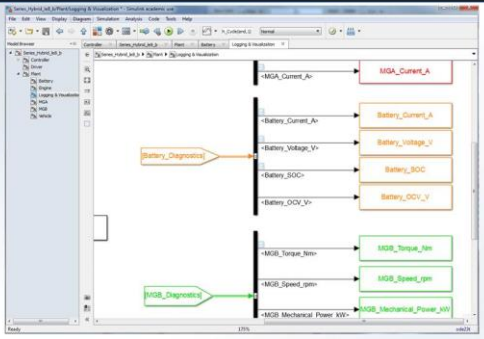
Breakpoints specification: Explicit values

Breakpoints 1: Battery_OCV_SOC_Axis

Breakpoints 2: Battery_OCV_Temp_Axis

362

Logging & Visualization



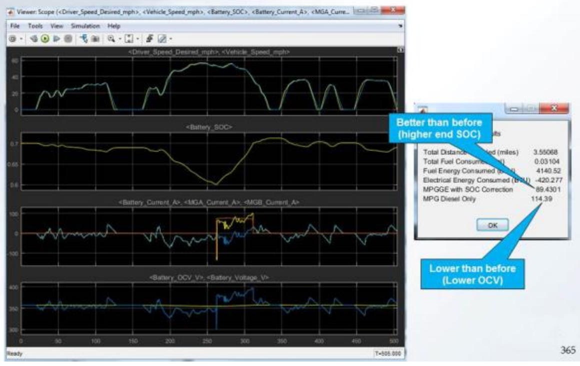
363

Variable OCV

- We've got a lot of scopes – delete them all
- Create a new one
  - Axis 1 – Vehicle Speed (Desired & Actual)
  - Axis 2 – Battery SOC
  - Axis 3 – Battery, MGA, & MGB current
  - Axis 4 – Battery OCV and Voltage
- Run the simulation

364

Variable OCV


Viewer: Scope (<Driver_Speed_Desired_mph>, <Vehicle_Speed_mph>, <Battery_SOC>, <Battery_Current_A>, <MGA_Curre...
<Driver_Speed_Desired_mph>, <Vehicle_Speed_mph>
<Battery_SOC>
<Battery_Current_A>, <MGA_Current_A>, <MGB_Current_A>
<Battery_OCV_V>, <Battery_Voltage_V>
Ready
T=505.000
Better than before (higher end SOC)
Lower than before (Lower OCV)
OK

Total Distance [illegible] (miles)	3.55068
Total Fuel Consum[illegible]	0.03104
Fuel Energy Consumed [illegible]	4140.52
Electrical Energy Consumed ([illegible])	-420.277
MPGGE with SOC Correction	89.4301
MPG Diesel Only	114.39

365

Variable Internal Resistance

- The internal resistance of the battery is also a function of the temperature and SOC
- It also varies for charging and discharging
- Drag in
  - A Subsystem block
- And rename it
  - Battery Internal Resistance (Ohms)

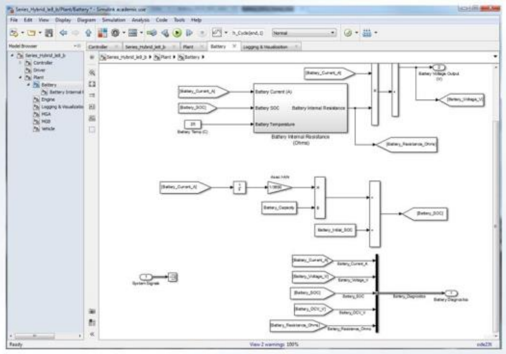
366

Variable Internal Resistance

- It will have three inputs
  - Battery Current
  - Battery SOC
  - Battery Temperature
- Add one output
  - Battery Internal Resistance
- Put the Battery Resistance on the Diagnostics Bus and extract it in the Logger

367

Variable Internal Resistance


Battery Current (A)
Battery SOC
Battery Temperature
Battery Internal Resistance
Battery Internal Resistance (Ohms)
Battery Voltage Output (V)
Battery Diagnostics

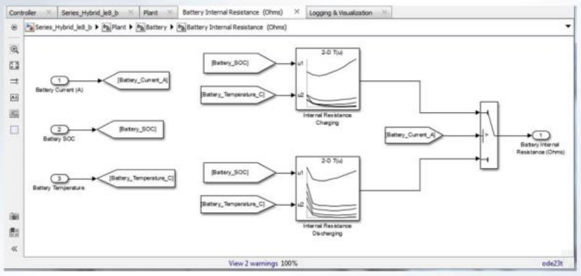
368

Variable Internal Resistance

- We will use the current to tell if we are charging or discharging
- From
  - Simulink / Commonly Used Blocks
- Drag a Switch block into the Battery Resistance subsystem
- Also drag in
  - two 2-D Lookup Table blocks
  - three Goto and five From blocks

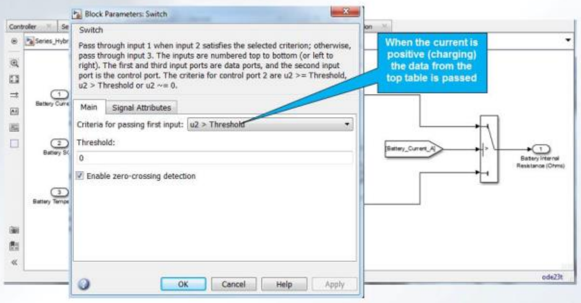
369

Variable Internal Resistance



370

Variable Internal Resistance


Block Parameters: Switch

Switch

Pass through input 1 when input 2 satisfies the selected criterion; otherwise, pass through input 3. The inputs are numbered top to bottom (or left to right). The first and third input ports are data ports, and the second input port is the control port. The criteria for control port 2 are u2 >= Threshold, u2 > Threshold or u2 ~= 0.

Main | Signal Attributes

Criteria for passing first input: u2 > Threshold

Threshold:

0

☑ Enable zero-crossing detection

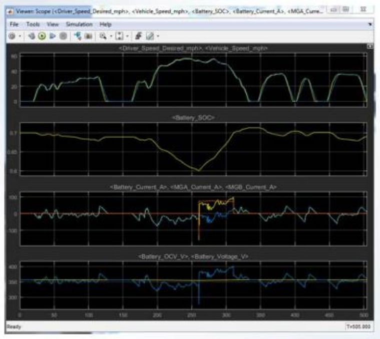
OK | Cancel | Help | Apply

When the current is positive (charging) the data from the top table is passed

371

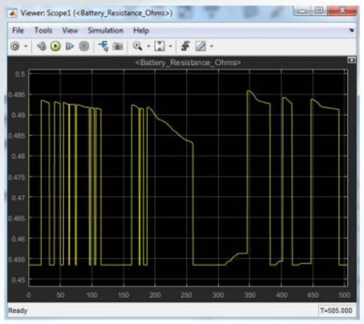
Variable Internal Resistance

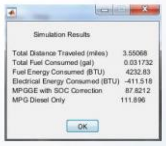
- Put the resistance on a new scope and run the simulation



372

Results




Simulation Results

```
Total Distance Traveled (miles)      3.55068
Total Fuel Consumed (gal)            0.031732
Fuel Energy Consumed (BTU)           4232.83
Electrical Energy Consumed (BTU)    -411.518
MPGGE with SOC Correction            87.8212
MPG Diesel Only                      111.896
```

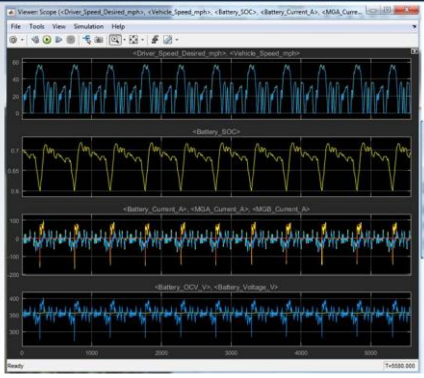
373

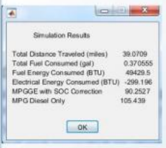
Results

- Higher internal resistance lowered the fuel economy
- Run the FU505_Ten_Times.mat drive cycle

374

Results




Simulation Results

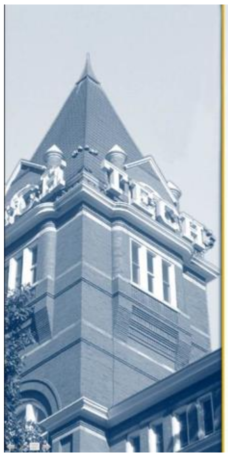
```
Total Distance Traveled (miles)      39.0709
Total Fuel Consumed (gal)            0.370555
Fuel Energy Consumed (BTU)           49429.5
Electrical Energy Consumed (BTU)    -299.196
MPGGE with SOC Correction            90.2527
MPG Diesel Only                      105.439
```

375

Review

- Improved the battery model
  - Used first order model battery model to incorporate internal resistance, making battery voltage a function of battery current
  - Discovered and corrected an algebraic loop with a memory block
  - Improved the model with open circuit voltage data based on SOC and battery temperature
  - Improved the model with variable charge and discharge internal resistances based on SOC and battery temperature

376



MBSD Lecture 9

Improved MG Model

Georgia Tech | The George W. Woodruff School of Mechanical Engineering

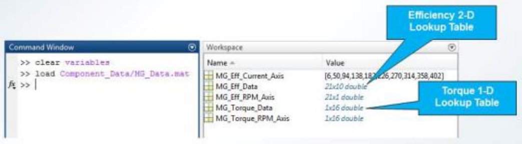
Outline

- Improved Motor/Generator model
  - Variable torque
  - Constant conversion efficiency
  - Variable conversion efficiency

378

MG Data

- At the Matlab command line type
  - clear variables
  - load Component_Data/MG_Data.mat



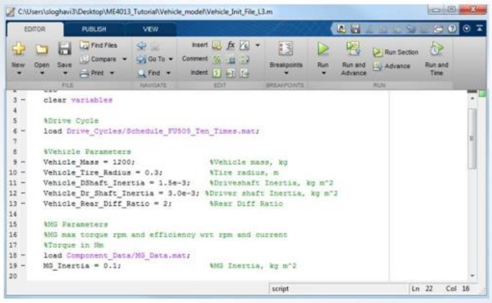
```
>> clear variables
>> load Component_Data/MG_Data.mat
>> |
```

Name	Value
MG_Eff_Current_Axis	[6,50,94,138,182,226,270,314,358,402]
MG_Eff_Data	21x10 double
MG_Eff_RPM_Axis	21x1 double
MG_Torque_Data	1x16 double
MG_Torque_RPM_Axis	1x16 double

379

Initialization

- Add the MG data loading to the init file
- Also add an MG inertia of 0.1 kg m^2



C:\Users\sloghavi3\Desktop\ME4013_Tutorial\Vehicle_model\Vehicle_Init_File_L3.m

```
clear variables

%Drive Cycle
load Drive_Cycles/Schedule_FU505_Ten_Times.mat;

%Vehicle Parameters
Vehicle_Mass = 1200;                    %Vehicle mass, kg
Vehicle_Tire_Radius = 0.3;              %Tire radius, m
Vehicle_DShaft_Inertia = 1.5e-3;        %Driveshaft Inertia, kg m^2
Vehicle_Dr_Shaft_Inertia = 3.0e-3; %Driver shaft Inertia, kg m^2
Vehicle_Rear_Diff_Ratio = 2;            %Rear Diff Ratio

%MG Parameters
%MG max torque rpm and efficiency wrt rpm and current
%Torque in Nm
load Component_Data/MG_Data.mat;
MG_Inertia = 0.1;                       %MG Inertia, kg m^2
```

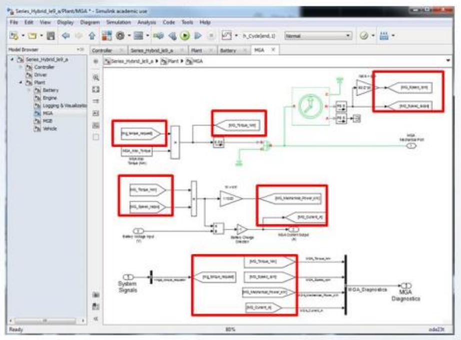
380

Variable Torque

- We will start with the MGA subsystem
  - Once we have it refined, we will copy it to MGB
- Rename your model
  - Series_Hybrid_le9_a.slx
- Go to the MGA subsystem
  - Change the name of all Goto and From blocks to MG

381

Variable Torque



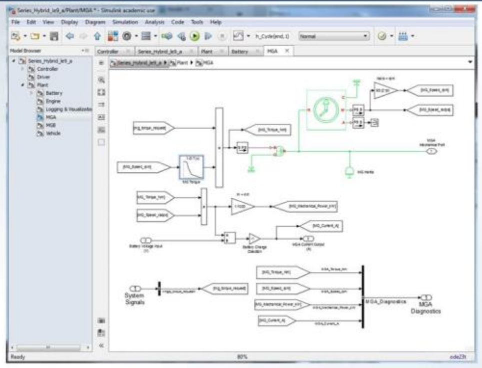
382

Variable Torque

- Drag in
  - An Inertia block
  - A 1-D Lookup Table block
- Delete the MGA_Max_Torque block
- Arrange as shown on the next slide
- Add the
  - MG_Torque_RPM_Axis
  - MG_Torque_Data
- to the Lookup Table

383

Variable Torque



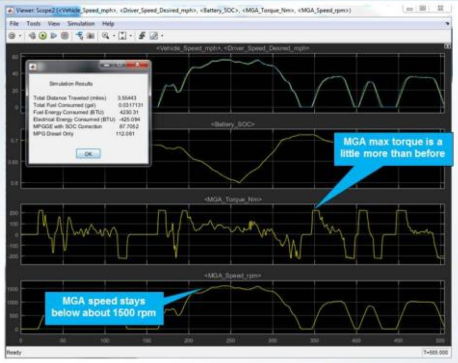
384

Results

- Let's create a new scope
  - Axis 1 – Vehicle speed (desired and actual)
  - Axis 2 – Battery SOC
  - Axis 3 – MGA Torque
  - Axis 4 – MGA Speed rpm
- Run the simulation for the FU505 drive cycle

385

Results


Simulation Results

Total Distance Traveled (miles)	3.55443
Total Fuel Consumed (gal)	0.0317131
Fuel Energy Consumed (BTU)	4230.31
Electrical Energy Consumed (BTU)	-425.094
MPGGE with SOC Correction	87.7062
MPG Diesel Only	112.081

386

Efficiency

- Electromechanical energy conversion can go two ways
  - Electrical to mechanical
    - Accelerating the vehicle
  - Mechanical to Electrical
    - Regenerative braking
- This is a little more challenging than the battery efficiency
- We will start by assuming a constant conversion efficiency of 0.85

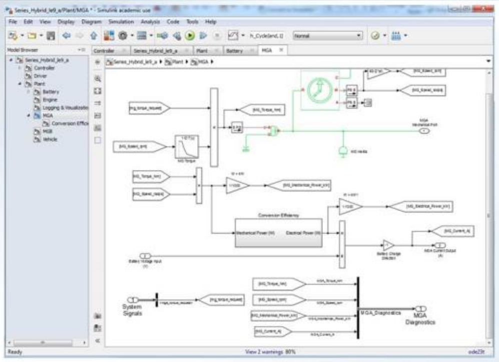
387

Efficiency

- Drag a Subsystem block into MGA and
- Rename
  - Subsystem to Conversion Efficiency
  - In1 to Mechanical Power (W)
  - Out1 to Electrical Power (W)
- Add a Gain and a Goto for the Electrical Power

388

Efficiency



389

Efficiency

- Go to the Conversion Efficiency subsystem
- Drag in
  - A Constant block
  - A Switch block
  - Two Product blocks
  - A Goto block
  - Two From blocks

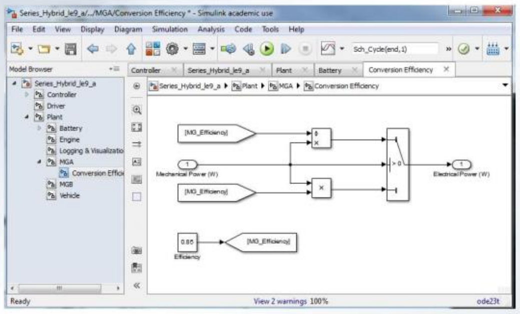
390

Efficiency

- If the mechanical power is positive
  - Top of switch
  - The motor is accelerating the vehicle
  - A higher electrical power is required
  - Therefore divide MP by the efficiency
- If the mechanical power is negative
  - Bottom of switch
  - The generator is making power
  - A lower electrical power is made
  - Therefore multiply MP by the efficiency
- Arrange as shown on the next slide

391

Efficiency



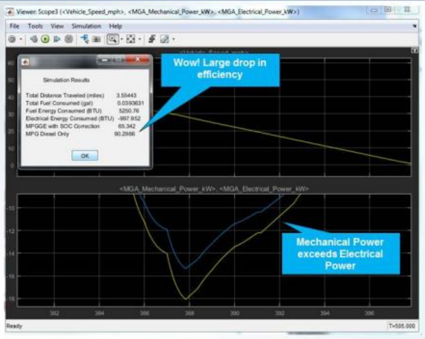
392

Efficiency

- Add the Electrical Power to the Diagnostics bus
- Extract it in the Logging & Visualization subsystem
- Create a scope
  - Axis 1 – Vehicle Speed (actual)
  - Axis 2 – Mechanical and Electrical Power

393

Results


Viewer: Scope3 (<Vehicle_Speed_mph>, <MGA_Mechanical_Power_kW>, <MGA_Electrical_Power_kW>)
File Tools View Simulation Help
Wow! Large drop in efficiency
<MGA_Mechanical_Power_kW>, <MGA_Electrical_Power_kW>
Mechanical Power exceeds Electrical Power
Ready
T=505.000

```
Simulation Results

Total Distance Traveled (miles)      3.55443
Total Fuel Consumed (gal)            0.0390631
Fuel Energy Consumed (BTU)           5250.76
Electrical Energy Consumed (BTU)     -997.952
MPGGE with SOC Correction            65.342
MPG Diesel Only                      90.2986
```

394

Variable Efficiency

- Conversion efficiency is a function of both MG speed and current
- Let's add that in
- Rename you model
  - Series_Hybrid_le9_b.slx
- In the Conversion Efficiency subsystem
  - Delete the Efficiency block and Goto/From blocks

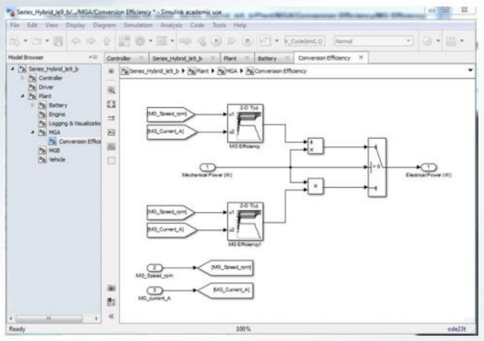
395

Variable Efficiency

- Drag in
  - Two 2-D Lookup Table blocks
  - Two In1 blocks
  - Two Goto blocks
  - Four From blocks
- Arrange and label as shown on the next slide

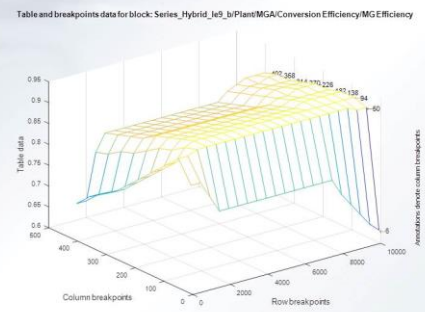
396

Variable Efficiency



397

Variable Efficiency



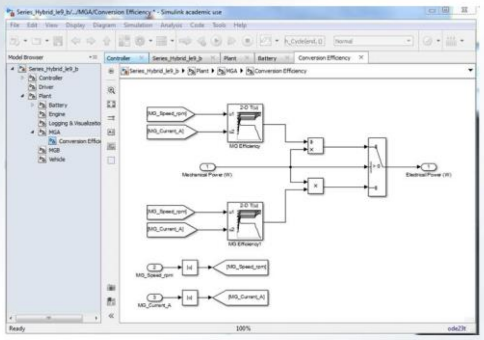
398

Variable Efficiency

- Hmmm, our efficiency data is for positive values of current and MG speed
  - Can either of these be negative?
- From
  - Simulink / Math Operations
- Drag in two Absolute Value blocks
- Return to the MGA subsystem
  - Connect the necessary From blocks
- Run the simulation

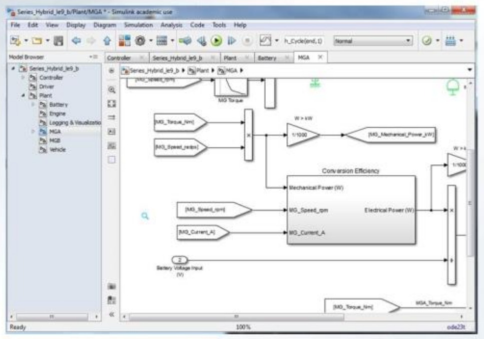
399

Variable Efficiency



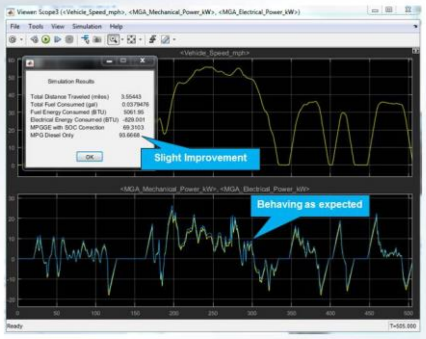
400

Variable Efficiency



401

Variable Efficiency


Viewer: Scope3 (<Vehicle_Speed_mph>, <MGA_Mechanical_Power_kW>, <MGA_Electrical_Power_kW>)
Slight Improvement
Behaving as expected

Simulation Results

```
Total Distance Traveled (miles)        3.55443
Total Fuel Consumed (gal)              0.0379476
Fuel Energy Consumed (BTU)             5061.95
Electrical Energy Consumed (BTU)       -829.001
MPGGE with SOC Correction              69.3103
MPG Diesel Only                        93.6668
```

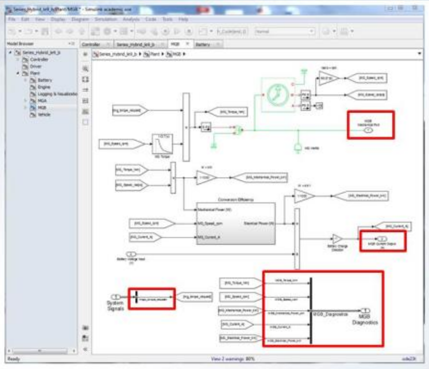
402

Variable Efficiency

- Outstanding, we now have a pretty good model of a motor/generator
- Go to the Plant level of the model
  - Delete MGB
  - Copy MGA
  - Update the names accordingly
- Extract the MGB electrical power in the Logging & Visualization subsystem

403

MGB



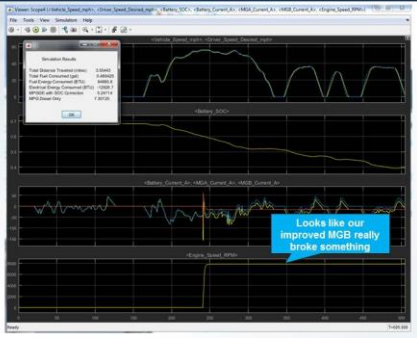
404

Results

- One last new scope
  - Axis 1 – Vehicle speed (actual and desired)
  - Axis 2 – SOC
  - Axis 3 – Battery, MGA, & MGB current
  - Axis 4 – Engine rpm
- Run the Simulation

405

Results



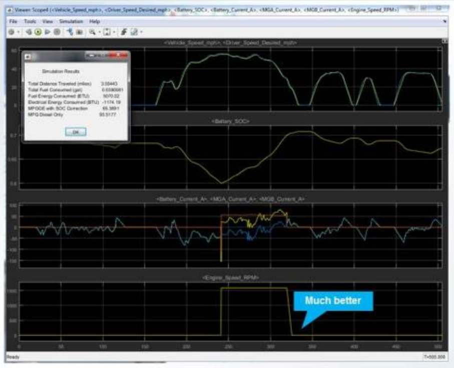
406

Results

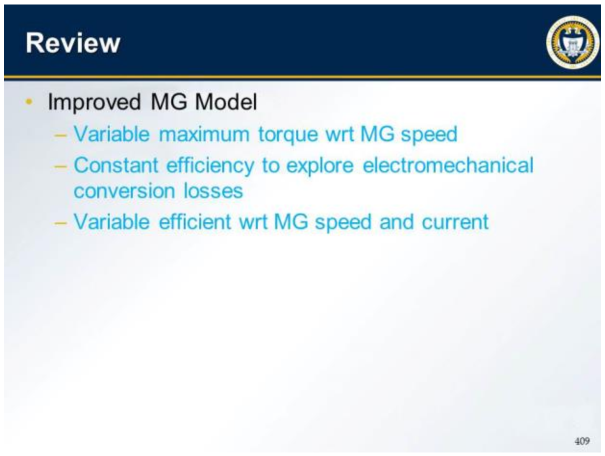
- Aha – our desired engine speed is 1800 rpm
  - Engine torque at 0.5 throttle 145 Nm
  - MGB max torque 140 Nm
- MGB can't control the engine!
- Decrease the desired engine speed to 1500 rpm

407

Results



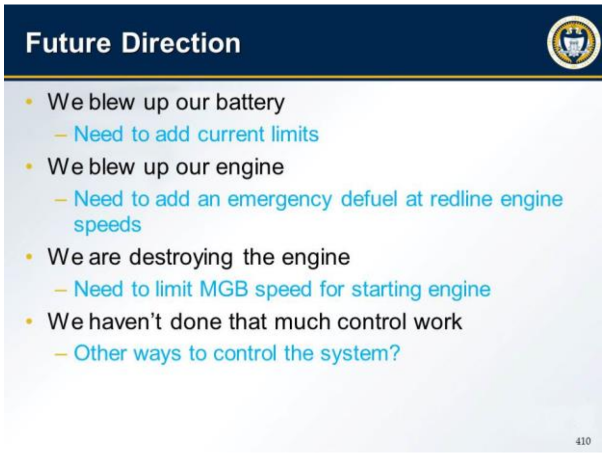
408



Review

- Improved MG Model
  - Variable maximum torque wrt MG speed
  - Constant efficiency to explore electromechanical conversion losses
  - Variable efficient wrt MG speed and current

409



Future Direction

- We blew up our battery
  - Need to add current limits
- We blew up our engine
  - Need to add an emergency defuel at redline engine speeds
- We are destroying the engine
  - Need to limit MGB speed for starting engine
- We haven’t done that much control work
  - Other ways to control the system?

410

2) Dynamic Programming Matlab code

```
% Dynamic Programming HEV script
%Written by Saeid Loghavi, August, 2015
clc
clear all
close all
tic
global max_engine_RPM
global radius;
global i_final;
global gear_ratios;
global EM_gear_ratio;

global Engine_Fuel_Data;
global Engine_Fuel_RPM_Axis;
global Engine_Fuel_Throttle_Axis;
global Engine_Torque_Data;
global Engine_Torque_RPM_Axis;
global Engine_Torque_Throttle_Axis;

% fuel consumption rate (next 3 lines taken from Rose-Hulman tutorial)
load Engine_Diesel_Data.mat;        % This is NOT BSFC, but instead grams/sec
as a function of throttle request and engine RPM
engine_max_torque = 200;            % Engine Max Torque, Nm
engine_inertia = 0.12;              % Engine Inertia, kg m^2
max_engine_RPM = max(Engine_Fuel_RPM_Axis);

% Vehicle data based on a 2011 VW Jetta
mass = 1389;    % mass in kg (2011 Prius)
g = 9.81;       % accel of gravity in m/s^2
Area = 2.642;   % frontal area in m^2 (guessed)
rho_air = 1.25; % density of air in kg/m^3
Cd = 0.32;      % drag coefficient (guessed)
delta_M = 1.2;  % mass factor
fr = 0.013;     % rolling resistance coefficient
radius = 0.25;% displaced tire radius, m
eta = 0.97;     % transmission efficiency
% http://www.bebo.com/new-cars/2011-volkswagen-jetta-sportwagen/specs/
i_final = 3.65; % Final drive ratio
gear_ratios = [3.78 2.12 1.36 1.03 0.84]; % Gear ratios
[nnn,NUM_GEARS] = size(gear_ratios);

% Additional Hybrid data (made up, and from http://www.eaa-
phev.org/wiki/Toyota_Prius_Battery_Specs)
battery_capacity = 1.5; % energy capacity in kW-h

SOC_low_range = 0.4;    % low desirable operating range for battery
SOC_high_range = 0.8;   % high desirable operating range for battery
SOC_initial=0.8;

SOC_final_low = 0.7;    % low SOC at final step (otherwise penalized)
SOC_final_high = 0.8;   % high SOC at final step (otherwise penalized)
```

```
SOC_penalty = 100000.0;   % proportional penalty parameter for operating
outside of SOC desirable (or final) range
NOT_ALLOW_PENALTY = 100000;

EM_max_torque = 600;    % Prius motor maximum torque is 400 Nm from Ehsani,
2nd ed for just one motor
EM_gear_ratio = 1.0;    % Gear ratio between EM and the final drive

regen_efficiency = 0.60; % The efficiency of regenerative braking, taken into
account only braking at front can occur
EM_efficiency=0.6;

%
% Drive cycle
%

%load Schedule_FU505_Ten_Times.mat
load Schedule_FU505.mat;
%load Schedule_Boston_Cab.mat;

[NUM_STEPS,n] = size(Sch_Cycle);
t = Sch_Cycle(:,1);
t_final = Sch_Cycle(NUM_STEPS,1);

% Convert vehicle speed from mph to m/s and compute acceleration
v = Sch_Cycle(:,2)*0.44704;
a = diff(v)./diff(t);

% Remove last data points for velocity and time since not present in
acceleration
v(end) = [];
t(end) = [];

[NUM_STEPS,n] = size(v);

% Calculate tractive force and power at each point in the drive cycle
force = delta_M*mass*a + fr*mass*g + 0.5*Cd*rho_air*Area*v.^2;
power_required = force.*v;



%
% Establish the state and control grid points
%

state_GRID_size = 400;
SOC_grid_size=state_GRID_size;
control_GRID_size = 400;
% engine_throttle_grid_size=length(Engine_Torque_Throttle_Axis);
ICE_thr_grid_size=control_GRID_size;
thr_vec = linspace(0,1,control_GRID_size);

SOC_vector=linspace(SOC_low_range,SOC_high_range,state_GRID_size);

% Evaluate the recursive cost starting at the last step in the drive cycle
% and working backwards.  The only state variable is SOC, so the cost
% function J_cost is size NUM_STEPS X SOC_grid_size.
% We also need to store the optimal control decisions at each time step.
% This is given as U_store and is size NUM_STEPS X SOC_grid_size X 3
% since we have three control variables (gear ratio, throttle request,
% EM torque)

J_cost = zeros(NUM_STEPS,SOC_grid_size);
U_store = zeros(NUM_STEPS,SOC_grid_size,3);

%
% First calculate J_cost(NUM_STEPS,:) - i.e., for all SOC in the grid at
% the last point on the drive cycle.  Penalize any SOC not between
% SOC_final_low and SOC_final_high.
%
for L = 1:SOC_grid_size;
    if SOC_vector(1,L)>SOC_final_low;
        J_cost(NUM_STEPS,L)=0;
    end
    if SOC_vector(1,L)<=SOC_final_low;
        J_cost(NUM_STEPS,L)=NOT_ALLOW_PENALTY;
    end
end;
%
% Next do the cost at all other points on the drive cycle using the
% recursive statement
%
[ig1_grid, thr_grid] = meshgrid (gear_ratios, thr_vec);
for STEP=(NUM_STEPS-1):-1:1
    STEP
    ICE_RPM_choices=v(STEP)./radius.*i_final.*ig1_grid./2./pi.*60;%RPM
choices for the time step (5 options based on gear)

ICE_torque_choices=interp2(Engine_Torque_Throttle_Axis,Engine_Torque_RPM_Axis
,Engine_Torque_Data,thr_grid,ICE_RPM_choices);
    ICE_torque_choices(isnan(ICE_torque_choices))=0;

ICE_fuel_consumption_choices=interp2(Engine_Fuel_Throttle_Axis,Engine_Fuel_RP
M_Axis,Engine_Fuel_Data,thr_grid,ICE_RPM_choices);
    Torque_tran=force(STEP)*radius/i_final;
    P_Wheel=power_required(STEP);
    Power_ICE=ICE_torque_choices.*ICE_RPM_choices*2*pi/60;

    for L = 1:SOC_grid_size;
        EM_Analysis; % Move up, and only compute a delta_SOC
        min_cost = inf;  % This keeps track of the minimum cost of using the
three controls

        phi_cost_choices=Not_Possible*NOT_ALLOW_PENALTY;
        if STEP==1 & L~=SOC_grid_size;
            phi_cost_choices=phi_cost_choices+NOT_ALLOW_PENALTY;
        end

        J_cost_next=zeros(control_GRID_size,NUM_GEARS);
        for s=1:NUM_GEARS
```

```
J_cost_next_geari=interp1(SOC_vector,J_cost(STEP+1,:),SOC_next_possible(:,s))
;
            J_cost_next(:,s)=J_cost_next_geari;
            check=isnan(J_cost_next);
            for ch=1:5;
                for ch2=1:100;
                    if check(ch2,ch)==1
                        J_cost_next(ch2,ch)=NOT_ALLOW_PENALTY;
                    end
                end
            end
        end


Cost_possible=Possible_options.*ICE_fuel_consumption_choices+phi_cost_choices
+J_cost_next;
        % For each SOC on the grid, seek the controls gear ratio, engine
throttle, and battery current at this time
        % step which minimize the cost from this point in the drive cycle
forward
        [M,min_cost_indicies_row]=min(Cost_possible);
        SOC_location=zeros(1,5);
        for count=1:5;

SOC_location(1,count)=SOC_next_possible(min_cost_indicies_row(count),count);
        end
        [min_cost,min_cost_indicies_column]=min(M);
        minimum_row_index = min_cost_indicies_row(min_cost_indicies_column);
        optimal_next_state(STEP,L) = SOC_next_possible(minimum_row_index,
min_cost_indicies_column);
        SOC_next_possible_indicies=interp1(SOC_vector, 1:state_GRID_size ,
SOC_next_possible,'nearest', 'extrap');

optimal_next_state_index(STEP,L)=SOC_next_possible_indicies(minimum_row_index
,min_cost_indicies_column);
        for S=1:NUM_GEARS;
            if S==1;
                if M(S)==min_cost;
                    Store_Gear=S;
                    Store_thr=min_cost_indicies_row(S);
                end
            end
            if S==2;
                if M(S)==min_cost & M(S-1)~=min_cost;
                    Store_Gear=S;
                    Store_thr=min_cost_indicies_row(S);
                end
            end
            if S==3;
                if M(S)==min_cost & M(S-1)~=min_cost & M(S-2)~=min_cost;
                    Store_Gear=S;
                    Store_thr=min_cost_indicies_row(S);
                end
            end
            if S==4;
```

```
                if M(S)==min_cost & M(S-1)~=min_cost & M(S-2)~=min_cost &
M(S-3)~=min_cost;
                    Store_Gear=S;
                    Store_thr=min_cost_indicies_row(S);
                end
            end
            if S==5;
                if M(S)==min_cost & M(S-1)~=min_cost & M(S-2)~=min_cost &
M(S-3)~=min_cost & M(S-4)~=min_cost;
                    Store_Gear=S;
                    Store_thr=min_cost_indicies_row(S);
                end
            end
        end
        U_store_1(STEP,L)=Store_Gear;
        U_store_2(STEP,L)=Store_thr;
        U_Store_3(STEP,L)=SOC_location(Store_Gear);
        J_cost(STEP,L) = min_cost;

    end;
%     U_store(STEP,L,3)=SOC_vector(L);

end;
toc
%%
close all
[global_min global_min_indices]=min(J_cost(1,:));
SOC_indices = global_min_indices;
for count3=1:NUM_STEPS-1
    O_SOC(count3)=SOC_vector(SOC_indices);
    O_Throttle(count3)=U_store_2(count3,SOC_indices);
    O_gear(count3)=U_store_1(count3,SOC_indices);
    if O_Throttle(count3)==1
        O_gear(count3)=0;
    end
    SOC_indices = optimal_next_state_index(count3,SOC_indices);
end
O_SOC(end)=SOC_vector(SOC_indices);
```

3) Co-simulation components

Configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<CoupledMultiphysicsSimulation xmlns:cse="http://www.simulia.com/CSESchema">
   <header>
      <SchemaVersion>1.1</SchemaVersion>
   </header>
   <components>
      <component name="Abaqus/Standard">
         <bottomUpImplementation>
            <codeName>Abaqus/Standard</codeName>
            <modelDescription>
               <UnitDefinitions>
                  <Unit name="kcal/kg">
                     <BaseUnit factor="4186.8" m="2" s="-2"></BaseUnit>
                  </Unit>
                  <Unit name="degC">
                     <BaseUnit K="1" factor="1.0" offset="273.15"></BaseUnit>
                  </Unit>
               </UnitDefinitions>
               <ModelVariables>
                  <ScalarVariable name="BODYFLUX">
                     <Real unit="kcal/kg"></Real>
                  </ScalarVariable>
                  <ScalarVariable name="TEMP">
                     <Real unit="degC"></Real>
                  </ScalarVariable>
               </ModelVariables>
            </modelDescription>
         </bottomUpImplementation>
      </component>
      <component name="CFD">
         <bottomUpImplementation>
            <codeName>Abaqus/Cfd</codeName>
         </bottomUpImplementation>
      </component>
   <component name="BatteryCoSim_System">
      <topDownImplementation>
         <identifier>BatteryCoSim_System</identifier>
      </topDownImplementation>
   </component>
</components>
<componentInstances>
   <componentInstance name="Fluidz">
      <component>CFD</component>
      <timeIncrementation>
         <lockstep>false</lockstep>
      </timeIncrementation>
      <initialConditions>
         <sendBeforeReceive>false</sendBeforeReceive>
      </initialConditions>
   </componentInstance>
   <componentInstance name="Module1">
      <component>Abaqus/Standard</component>
      <timeIncrementation>
         <lockstep>false</lockstep>
      </timeIncrementation>
   </componentInstance>
   <componentInstance name="BatteryCoSim_System">
      <component>BatteryCoSim_System</component>
   </componentInstance>
</componentInstances>
<connectors>
   <connector name="CFD_to_STD_INPUT">
      <componentInstance>Module1</componentInstance>
      <variables>
         <input>
            <variable>heat_flux</variable>
            <variable>heat_capacitance</variable>
         </input>
      </variables>
   </connector>
   <connector name="CFD_to_STD_OUTPUT">
      <componentInstance>Fluidz</componentInstance>
   <variables>
      <output>
         <variable>heat_flux</variable>
         <variable>heat_capacitance</variable>
      </output>
   </variables>
</connector>
<connector name="STD_to_CFD_INPUT">
   <componentInstance>Fluidz</componentInstance>
   <variables>
      <input>
         <variable>temperature</variable>
      </input>
   </variables>
</connector>
<connector name="STD_to_CFD_OUTPUT">
   <componentInstance>Module1</componentInstance>
   <variables>
      <output>
         <variable>temperature</variable>
      </output>
   </variables>
</connector>
<connector name="STD_from_FMI">
   <componentInstance>Module1</componentInstance>
   <variables>
      <output>
         <variable>TEMP</variable>
      </output>
      <input>
         <variable>BODYFLUX</variable>
      </input>
   </variables>
</connector>
<connector name="FMI_from_STD">
   <componentInstance>BatteryCoSim_System</componentInstance>
   <variables>
      <input>
```

```
            <variable>temp</variable>
         </input>
         <output>
            <variable>bodyflux</variable>
         </output>
      </variables>
   </connector>
</connectors>
<connectionSets>
   <connectionSet name="CFD_to_STD" type="FIELD">
      <connection>
         <connector>CFD_to_STD_INPUT</connector>
         <connector>CFD_to_STD_OUTPUT</connector>
      </connection>
   </connectionSet>
   <connectionSet name="STD_to_CFD" type="FIELD">
      <connection>
         <connector>STD_to_CFD_INPUT</connector>
         <connector>STD_to_CFD_OUTPUT</connector>
      </connection>
   </connectionSet>
   <connectionSet name="STD_and_FMI" type="SIGNAL">
      <connection>
         <connector>STD_from_FMI</connector>
         <connector>FMI_from_STD</connector>
      </connection>
   </connectionSet>
</connectionSets>
<execution>
   <compositeActors>
      <compositeActor name="twoCodeContinuousTime">
         <actors>
            <atomicActor>Module1</atomicActor>
            <atomicActor>Fluidz</atomicActor>
            <atomicActor>BatteryCoSim_System</atomicActor>
         </actors>
         <modelOfComputation>
            <continuousTime>
```

```xml
                  <algorithm>GAUSS-SEIDEL</algorithm>
                  <negotiationMethod>MIN</negotiationMethod>
               </continuousTime>
            </modelOfComputation>
         </compositeActor>
      </compositeActors>
      <connectionGroups>
         <connectionCategory name="InitialConditions">
            <connectionSet>CFD_to_STD</connectionSet>
            <connectionSet>STD_to_CFD</connectionSet>
            <connectionSet>STD_and_FMI</connectionSet>
         </connectionCategory>
         <connectionCategory name="CouplingStep">
            <connectionSet>CFD_to_STD</connectionSet>
            <connectionSet>STD_to_CFD</connectionSet>
            <connectionSet>STD_and_FMI</connectionSet>
         </connectionCategory>
      </connectionGroups>
      <scenario>
         <duration>300.</duration>
      </scenario>
   </execution>
</CoupledMultiphysicsSimulation>
```

Standard Model:

```
*Heading
** Job name: Module1 Model name: Model-1
** Generated by: Abaqus/CAE 2016
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Module
*Node
      1,   7.94627619,   10.1000004,   64.5999985
…
```

```
25426, -0.0205669049,  -10.0534954,  -1.76602697
*Element, type=DC3D4
   1, 14320, 14464, 14449, 14450
…
72829, 14137, 24455, 24456, 24487
*Surface, type=ELEMENT, name=s_Surf-26
_s_Surf-26_S3, S3
_s_Surf-26_S1, S1
_s_Surf-26_S4, S4
_s_Surf-26_S2, S2
*End Assembly
*Amplitude, name=BODYFLUX, definition=ACTUATOR
**
** MATERIALS
**
*Material, name=Aluminum
*Conductivity
167.,
*Density
2700.,
*Specific Heat
902.,
*Material, name=Copper
*Conductivity
400.,
*Density
8960.,
*Specific Heat
385.,
*Material, name=Plastic
*Conductivity
 0.2,
*Density
2000.,
*Specific Heat
1800.,
*Material, name=Silicon
*Conductivity
```

```
149.,
*Density
2329.,
*Specific Heat
700.,
**
** INTERACTION PROPERTIES
**
*Surface Interaction, name=Conduction
1.,
*Gap Conductance
20000.,0.
    0.,2.
**
** PREDEFINED FIELDS
**
** Name: Predefined Field-1   Type: Temperature
*Initial Conditions, type=TEMPERATURE
Set-5, 30.
**
** INTERACTIONS
**
** Interaction: Int-1
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-1, m_Surf-1
** Interaction: Int-2
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
m_Surf-3, s_Surf-3
** Interaction: Int-3
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
m_Surf-5, s_Surf-5
** Interaction: Int-4
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
m_Surf-9, s_Surf-9
** Interaction: Int-5
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
m_Surf-11, s_Surf-11
** Interaction: Int-6
```

```
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-15, m_Surf-13
** Interaction: Int-7
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-16, m_Surf-16
** Interaction: Int-8
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-18, m_Surf-18
** Interaction: Int-9
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
m_Surf-20, s_Surf-20
** Interaction: Int-10
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-22, m_Surf-22
** Interaction: Int-11
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-24, m_Surf-24
** Interaction: Int-12
*Contact Pair, interaction=Conduction, type=SURFACE TO SURFACE
s_Surf-26, m_Surf-26
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1, nlgeom=NO, inc=1000
*Heat Transfer, end=PERIOD, deltmx=4.
0.1, 300., 0.003, 250.,
**
** LOADS
**
** Name: Load-1   Type: Body heat flux
*Dflux, amplitude=BODYFLUX
Set-6, BF, 500000.
** Interaction: Int-1
*Co-simulation, name=Int-1, controls=Int-1_Ctrls, program=MULTIPHYSICS
*Co-simulation Region, import, type=SURFACE
interface, CFL
interface, LUMPEDHEATCAPACITANCE
```

```
*Co-simulation Region, export, type=SURFACE

interface, NT

*Co-simulation Controls, name=Int-1_Ctrls, coupling scheme=gauss-seidel, time
incrementation=subcycle, time marks=yes,

step size=min, scheme modifier=lead

**

** OUTPUT REQUESTS

**

*Restart, write, frequency=0

**

** FIELD OUTPUT: F-Output-1

**

*Output, field, variable=PRESELECT

*Output, history, frequency=0

*End Step
```

Fluid Model:

```
*Heading

** Job name: Fluidz Model name: Model-1

** Generated by: Abaqus/CAE 2016

*Preprint, echo=NO, model=NO, history=NO, contact=NO

**

** PARTS

**

*Part, name=Shell2-1

*Node

      1,   16.4462757,   40.3134003,  -11.5000019

…

…

      1155,   15.4851465,   31.3644524,  -14.9610233

*Element, type=FC3D4

   1,  578,  995,  576,   35

3412, 1002,  998,  508,  507

….

*Surface, type=ELEMENT, name=INTERFACE

_INTERFACE_S3, S3
```

```
_INTERFACE_S4, S4
_INTERFACE_S2, S2
_INTERFACE_S1, S1
*Elset, elset=_OUTLET_S3, internal, instance=Shell2-1-1
   21,  214,  479,  480, 2183
*Elset, elset=_OUTLET_S1, internal, instance=Shell2-1-1
  139, 1432
*Elset, elset=_OUTLET_S4, internal, instance=Shell2-1-1
  477,  879, 2252
*Surface, type=ELEMENT, name=OUTLET
_OUTLET_S3, S3
_OUTLET_S1, S1
_OUTLET_S4, S4
*Elset, elset=_PDF_WholeModel, internal, instance=Shell2-1-1, generate
    1,  3412,     1
*End Assembly
**
** MATERIALS
**
*Material, name=COOLANT
*Conductivity
 0.43,
*Density
1056.,
*Specific Heat, type=CONSTANTPRESSURE
50.,
*Viscosity
 0.008495,
**
** PREDEFINED FIELDS
**
** Name: Predefined Field-1   Type: Fluid thermal energy
*Initial Conditions, type=TEMPERATURE, Element Average
_PDF_WholeModel, 15.
** Name: Predefined Field-2   Type: Fluid velocity
*Initial Conditions, type=VELOCITY, Element Average
_PDF_WholeModel, 1, 0.
_PDF_WholeModel, 2, 0.
```

```
_PDF_WholeModel, 3, 0.
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1
*CFD, incompressible navier stokes, energy equation=TEMPERATURE
0.01, 300., 0.025, 0.45, 1
1e-10, 0.5,  , 0.5, 0.5
*Momentum Equation Solver
250, 2, 1e-05
*Pressure Equation Solver
250, 2, 1e-05
ICC, 1, 1, CG
*Transport Equation Solver
250, 2, 1e-05
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Fluid wall condition
*Fluid Boundary, type=PHYSICAL, wall, surface=INTERFACE
** Name: BC-2 Type: Fluid inlet/outlet
*Fluid Boundary, type=PHYSICAL, velocity inlet, surface=INLET
VELX, 0.
VELY, -0.07536
VELZ, 0.
TEMP, 15.
** Name: BC-3 Type: Fluid inlet/outlet
*Fluid Boundary, type=PHYSICAL, pressure outlet, surface=OUTLET
P, 0.
passiveoutflow, 0.0
TEMP, 18.
** Interaction: Int-1
*Co-simulation, name=Int-1, controls=Int-1_Ctrls, program=MULTIPHYSICS
*Co-simulation Region, import, type=SURFACE
interface, TEMP
*Co-simulation Region, export, type=SURFACE
interface, HFL
```

```
interface, LUMPEDHEATCAPACITANCE

*Co-simulation Controls, name=Int-1_Ctrls, coupling scheme=gauss-seidel, time
incrementation=lockstep, time marks=yes,

step size=min, scheme modifier=lag

**

** OUTPUT REQUESTS

**

*Restart, write, frequency=0

**

** FIELD OUTPUT: F-Output-1

**

*Output, field, variable=PRESELECT

*Output, history, frequency=0

*End Step
```

References

[1] M. Ehsani, Y. Gao, S. Gay and A. Ahmadi, Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design, CRC Press, 2004.

[2] Y. Ho, J. Park, J. T. Lee, J. Seo and S. Park, "Estimation of CO2 reduction by parallel hard-type power hybridizationfor gasoline and diesel vehicles," *Science of the Total Environment*, vol. 595, pp. 2-12, 2017.

[3] G. Wu and Z. Dong, "Design, analysis and modeling of a novel hybrid powertrain system based on hybridized automated manual transmission," *Mechanical Systems and Signal Processing*, no. 93, pp. 688-705, 2017.

[4] D. Pei and M. Leamy, "Dynamic Programming-Informed Equivalent Cost Minimization Control Strategies for Hybrid-Electric Vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, 2013.

[5] Y. Zhao, M. Kuang, B. Nefcy, D. Colvin, S. Ford and Z. Liu, "Regenerative Braking Control Development for P2 Parallel Hybrid Electric Vehicles," *SAE*, 11 January 2017.

[6] L. Serrao, S. Onori and G. Rizzoni, "A Comparative Analysis of Energy Management Strategies for Hybrid Electric Vehicles," *Journal of Dynamic Systems, Measurement, and Control*, 25 March 2011.

[7] D. Kum, H. Peng and N. Bucknor, "Supervisory Control of Parallel Hybrid Electric Vehicles for Fuel and Emission Reduction," *Journal of Dynamic Systems, Measurement, and Control*, 11 November 2011.

[8] N. Kim, S. Cha and H. Peng, "Optimal Control of Hybrid Electric Vehicles Based on Pontryagin's Minimum Principle," *IEEE Transactions on Control Systems Technology*, pp. 1279-1287, 5 September 2011.

[9] J. Wilbanks, F. Favaretto, F. Cimatti and M. Leamy, "High-Performance Plug-In Hybrid Electric Vehicle Design Studies and Considerations," *SAE Technical Papers*, 21 April 2015.

[10] J. Arata, M. Leamy and K. Cunefare, "Power-split HEV control strategy development with refined engine transients," *SAE Technical Papers*, pp. 119-133, 24 April 2012.

[11] B. Scrosati and J. Garche, "Lithium batteries: Status, prospects and future," *Journal of Power Sources*, vol. 195, no. 9, pp. 2419-2430, 2010.

[12] K.-Y. Oh, N. Samad, Y. Kim, J. Siegel, A. Stefanopoulou and B. Epureanu, "A novel phenomenological multi-physics model of Li-ion battery cells," *Journal of Power Sources*, vol. 326, pp. 447-458, 2016.

[13] M. Doyle, T. Fuller and J. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell," *Journal of Electrochemical Society*, vol. 140, no. 6, pp. 1526-1533, 1993.

[14] L. Xiwen, M. Yan and Y. Zhenhua, "Research of SOC Estimation for Lithium-ion Battery of Electric Vehicle Based on AMEsim-Simulink Co-Simulation," in *32nd Chinese Control Conference*, Xi'an,China, 2013.

[15] K. Murashko, J. Pyrhönen and L. Laurila, "Optimization of the passive thermal control system of a lithium-ion battery with heat pipes embedded in an aluminum plate," in *15th European Conference on Power Electronics and Applications, EPE*, Lille, France, 2013.

[16] M. R. Khan and S. K. Kaer, "Three Dimensional Thermal Modeling of Li-Ion Battery Pack Based on Multiphysics and Calorimetric Measurement," in *2016 IEEE Vehicle Power and Propulsion Conference, VPPC*, Hangzhou, China, 2016.

[17] S. Peck, T. Olszanski, S. Zanardelli and M. Pierce, "Validation of a Thermal-Electric Li-Ion Battery Model," *SAE International Journal of Passenger Cars*, vol. 121, no. 7, pp. 154-163, 2012.

[18] G.-H. Kim, K. Smith, K.-J. Lee, S. Santhanagopalan and A. Pesaran, "Multi-Domain Modeling of Lithium-Ion Batteries Encompassing Multi-Physics in Varied Length Scales," *Journal of Electrochemical Society*, vol. 158, no. 8, pp. 955-969, 2011.

[19] Y. Xie, J. Li and C. Yuamn, "Multiphysics modeling of lithium ion battery capacity fading process with solid-electrolyte interphase growth by elementary reaction kinetics," *Journal of Power Sources*, vol. 248, pp. 172-179, 2014.

[20] S. U. Kim, P. Albertus, D. Cook, C. Monroe and J. Christensen, "Thermoelectrochemical simulations of performance and abuse in 50-Ah automotive cells," *Journal of Power Sources*, vol. 268, pp. 625-633, 2014.

[21] R. Melville, N. Clauvelin and J. Milios, "A High-performance Model Solver for "in-the-Loop" Battery Simulations," in *American Control Conference (ACC)*, Boston, MA, 2016.

[22] L. Jiang, S. Yuan, H. Wu, C. Yin and W. Miao, "Electro-Thermal Modeling and Experimental Verification for 18650 Li-Ion Cell," in *Vehicle Power and Propulsion*

Conference (VPPC), Hangzhou, China, 2016.

[23] M. Reiner and D. Zimmer, "Object-oriented modelling of wind turbines and its application for control design based on nonlinear dynamic inversion," *Mathematical and Computer Modelling of Dynamical Systems,* vol. 23, pp. 319-240, 2017.

[24] R. Montañés, J. Windahl, J. Pålsson and M. Thern, "Dynamic Modeling of a Parabolic Trough Solar Thermal Power Plant with Thermal Storage Using Modelica," *Heat Transfer Engineering*, vol. 0, pp. 1-16, 2017.

[25] T. Schwan, R. Unger and J. Pipiorke, "Aspects of FMI in Building Simulation," in *Proceedings of the 12th International Modelica Conference*, Prague, Czech Republic, 2017.

[26] M. Association, "FMI standard," Modelica Association, [Online]. Available: http://www.fmi-standard.org.

[27] M. Einhorn, F. V. Conte, C. Kral, C. Niklas, H. Popp and J. Fleig, "A Modelica Library for Simulation of Electric Energy Storages," in *Proceedings of the 8th International Modelica Conference*, Dresden, Germany, 2011.

[28] Dassault Systèmes, "Abaqus 2016 CAE User's Guide," Dassault Systèmes, 2015.

[29] Dassault Systèmes, "Co-Simulation Engine user's guide," Dassault Systèmes, 2016.

[30] Dassault Systèmes, "API reference guide," Dassault Systèmes, 2016.

[31] Dynasim AB, "Dymola User's Manual," Dynasim AB, Lund, Sweden, 2016.

[32] T. Bergman, A. Lavine, F. Incropera and D. DeWitt, Fundamentals of Heat and Mass Transfer, Hoboken,NJ: John Wiley & Sons,Inc., 2011.

[33] J. de Hoog, J.-M. Timmermans, D. Ioan-Stroe, M. Swierczynski, J. Jaguemont, S. Goutam, N. Omar, J. Van Mierlo and P. Van Den Bossche, "Combined cycling and calendar capacity fade modeling of a Nickel-Manganese-Cobalt Oxide Cell with real-life profile validation," *Applied Energy,* vol. 200, pp. 47-61, 2017.