MODULAR MODELING AND CONTROL OF A HYBRID UNMANNED AERIAL VEHICLE'S POWERTRAIN

BY

CHRISTOPHER THOMAS AKSLAND

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Andrew Alleyne

Abstract

The emerging trend of vehicle electrification is revolutionizing the transportation industry by replacing traditional mechanical and hydraulic components with higher performing, more reliable, and more efficient electrical components. However, the introduction of a complex electrical network onboard mobile systems poses significant challenges for control design engineers. The most notable challenge is the coordination of multi-domain and multi-timescale system dynamics. This thesis seeks to address the challenge of coordination between the slow battery state of charge dynamic and faster electro-mechanical dynamics for a hybrid unmanned aerial vehicle.

The graph-based modeling framework for multi-domain systems is leveraged to capture the interactions between relevant energy domains. Additionally, the modularity and scalability of this modeling approach is used to develop a dynamic model for a hybrid unmanned aerial vehicle. The system model facilities the design and development of three control architectures of varying complexity. A baseline controller is developed for sake of comparison. A battery state of charge bounding algorithm in integrated into a centralized model predictive controller to provide system coordination across timescales. Lastly, an alternative model predictive hierarchical controller is designed to provide real-time planning of the slow battery state of charge dynamics.

The proposed models and controllers are experimentally validated on a novel hybrid electric UAV powertrain testbed. The controllers are evaluated on three core figures of merit: performance, reliability, and efficiency. Both simulation and experimental results show that the advanced controllers outperform the baseline in all figures of merit with a $9 - 12.5\%$ reduction in fuel usage.

To my friends, family, and teachers.

Acknowledgments

I would like to first thank my adviser, Professor Andrew Alleyne, for his instruction and commitment to my development as both a researcher and individual. Your dedication to students is unparalleled and I'm extremely fortunate to have such an outstanding adviser. Although I was originally hesitant about attending graduate school at UIUC, you have made the entire experience worth it. Thank you for fostering such an open, exciting, and friendly lab environment.

Next, I would like to thank the 18 outstanding colleges within the Alleyne Research Group (ARG) that I've had the opportunity to collaborate with while at UIUC. First, I must give a special thanks to Justin Koeln for introducing me to undergraduate research, being my first mentor, and helping me shape my career. I'm grateful to have worked so closely with senior lab members Matt Williams, Herschel Pangborn, and Donald Docimo. I'm appreciative of our friendships and I look forward to what we can all accomplish in the future. Although our time together was brief, thank you to Bryan Keating and Spencer Kieffer for openly welcoming me into the lab. Thank you Malia Kawamura, Sarah Garrow, and Sunny Sharma for the engaging lunch conversations and organizing exciting events outside of work. To my current labmates Pamela Tannous, Ashley Armstrong, Oyuna Angatkina, Nate Weir, Spencer Igram, Cary Laird, and Mindy Wagenmaker, thank you for your continued support and friendship. You are all incredible individuals that make coming to work everyday fun and exciting. Although we only shared one semester together, I need to thank Erik Salazar for his friendship and great ARG seminar. Lastly, I would like to thank an alumni, Tim Deppen, for providing great guidance and helping convince me to stay at UIUC for graduate school.

To my grandparents: Margaret, Donald, Beverly, and Rudy; parents, Ann and Jeff; broth-

ers, Erik and Alex; aunt and uncle, Stephanie and Dave; and my remaining extended family, thank you for instilling within me a passion for learning and independence. These are two invaluable qualities that I feel have led to my successes thus far and I appreciate your continued support and interest in my work.

To my friends from home, we've know each other for so long you all are essentially family at this point. I treasure our friendship and I'm always looking forward to the next reunion. To my college friends, thank you for being so kind, generous, and supportive. It has been incredible to watch you all become so successful in such a short period of time. To all of you, thanks for providing fun distractions from work. I'm looking forward to next time I can beat you all at Mario Kart.

Next, I would like to thank the faculty and staff of the Center for Power Optimization of Electro-Thermal Systems. Special thanks go to Jodi Gritten and Cyril Rybicki. Jodi, your greeting every morning makes everyday better. The experimental results in this thesis were made possible with Cy's expertise in data acquisition and software development. I would also like to thank Dr. Karen Bender, Sonya Chambers, Dr. Jessica Perez, Joe Muskin, and John Wierschem for providing a multitude of opportunities for us students and fostering a pleasant work environment. Thank you to Dr. David Carroll and Dr. Joe Zimmerman for their engagement and providing my first research project. Thank you to my collaborators at the Air Force Research Lab. Your continued interest and feedback is appreciated.

Lastly, I would like to acknowledge the support of the National Science Foundation Engineering Research Center for Power Optimization of Electro-Thermal Systems (POETS) under cooperative agreement EEC-1449548. This support has provided me the flexibility to pursue transformative research that will benefit both my field and society.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BMS	Battery Management System
BLDC	Brushless DC
CDKF	Central-Difference Kalman Filter
CV	Constant Current
CC	Constant Voltage
CAN	Controller Area Network
DAE	Differential Algebraic Equation
dq or dq0	Direct-Quadrature
DP	Dynamic Programming
ESC	Electronic Speed Controller
ESS	Energy Storage System
EKF	Extended Kalman Filter
LKF	Linear Kalman Filter
MIQP	Mixed-Integer Quadratic Program
MPC	Model Predictive Control
OCV	Open circuit voltage
PCKA	PC Krause and Associates
POETS	Power Optimization of Electro-Thermal Systems
RK4	Runge-Kutta 4
SVM	Space Vector Modulation

SFC	Specific Fuel Consumption
SOC	State of Charge
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol

List of Symbols

Common Variables

A	Area
b	Modified power flow coefficients, viscous friction coefficient, or slack variable
c	Power flow coefficients or static friction coefficient
C	Capacitance
C_D	Drag coefficient
C_e	Electrical capacitance
C_T	Thermal capacitance or thrust coefficient
C_{tau}	Torque coefficient
$\mathcal{C}$	Set of component graphs
D	Source flow matrix or diameter
e	Graph edge or error signal
E	Edge property map
$\mathcal{E}$	Set of graph edges
g	Gravity
$\mathcal{G}$	Directed graph
I	Current
J	Moment of inertia or cost function
k, k_τ	Linear or angular spring constant
K_i	Integral gain

K_r	Regular Park Transformation
k_v	Motor constant
K_p	Power-invariant Park Transformation or proportional gain
L	Inductance
λ	Flux linkages
Λ	Flux linkages, the set of Type 1 interconnections, or cost function weights
m	Mass or modulation signal
M	Incidence matrix
N_e,	Number of edges in a graph
N_g	Number of graphs
N_s	Number of sources in a graph
N_t,	Number of sinks in a graph
N_v	Number of vertices in a graph
Ξ	Set of component graph edges
p	Pole pairs
P	Power flow
$\mathcal{P}^v, \mathcal{P}^e$	Vertex and edge properties
q	State of charge
Q	Charge or capacity
θ	Angle
r	Reference signal
R	Resistance
ρ	Density or binary state
sfc	Specific fuel consumption
s, σ	Switch, binary state, or slack variable
Σ	Set of Type 2 interconnections
t	Time

T	Temperature or Threshold
τ	Torque or time constant
$\mathcal{T}_v, \mathcal{T}_e$	Vertex and edge type
u	Input or duty cycle
v	Graph vertex or velocity
V	Voltage or the vertex property map
$\mathcal{V}$	Set of graph vertices
ω	Angular speed
ω_n	Cutoff frequency
x	State variable
χ	Set of component graph vertices
ζ	Damping ratio

Superscripts and Subscripts

a	Algebraic variable
abc	Rotating abc reference frame variable
bat	Battery property
d	Dynamic variable
$data$	Experimental data
dc	DC value
$dq0$	Stationary abc reference frame variable
gen	Genset property
h	Head state (as subscript)
$head$	Head vertex
i	Inverter property
m	Motor property
ref	Reference signal

s	Source edge or system property
σ	Switched property
t	Sink vertex (as superscript) or tail state (as subscript)
$tail$	Tail vertex
‡	Modified graph property

Notation

$[1 : N] \coloneqq \{1, 2, \ldots, N\}$

$\mathrm{sat}(x)_b^a \coloneqq \min(a, \max(b, x))$

Chapter 1

Introduction

1.1 Motivation

The emerging trend of vehicle electrification is revolutionizing the transportation industry by impacting the design of various types of ground, air, and water vehicles. As seen by the trend in Figure 1.1, vehicle electrification provides an optimistic approach towards the development of high power level vehicles [1]. This shift towards electrification has provided various improvements on classic mechanical and hydraulic powertrains. First, electrified vehicles are more environmentally friendly because they can better leverage renewable energy sources. Furthermore, maintenance costs tend to be lower for electrified systems since complex mechanical and hydraulic systems are replaced by their electrical counterparts [2]. The modularity of electrical systems also provides an increasingly large design space that better facilities mission-specific design for more capable vehicles [3]. Additionally, electrified vehicles may have reduced sound levels, resulting in decreased noise pollution in urban areas such as airports.

In spite of the these benefits, there are still many challenges facing electrification. In comparison to fuel, batteries are significantly less energy dense [5]. This issue is compounded when considering vehicle mass because batteries do not decrease in weight over the course of a mission. In fact, weight is a significant market challenge because heavier systems have more significant operating costs that deter widespread adoption of the technology. For electrified aircraft, corona discharge (arcing) is more prevalent at high altitudes [6], and therefore additional safety systems and weight must be added to the aircraft. The most significant challenge facing electrified systems is the necessity for an onboard thermal management system [5]. In traditional vehicles, engines and gas turbines that operate at high

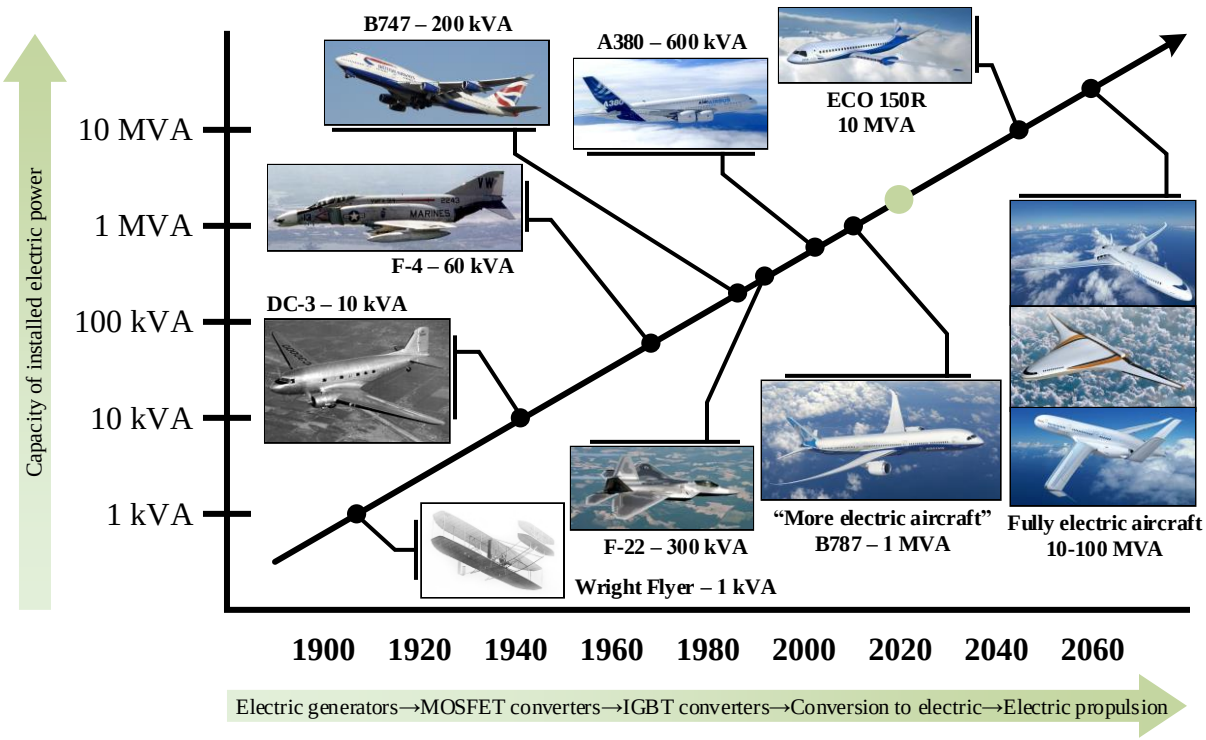
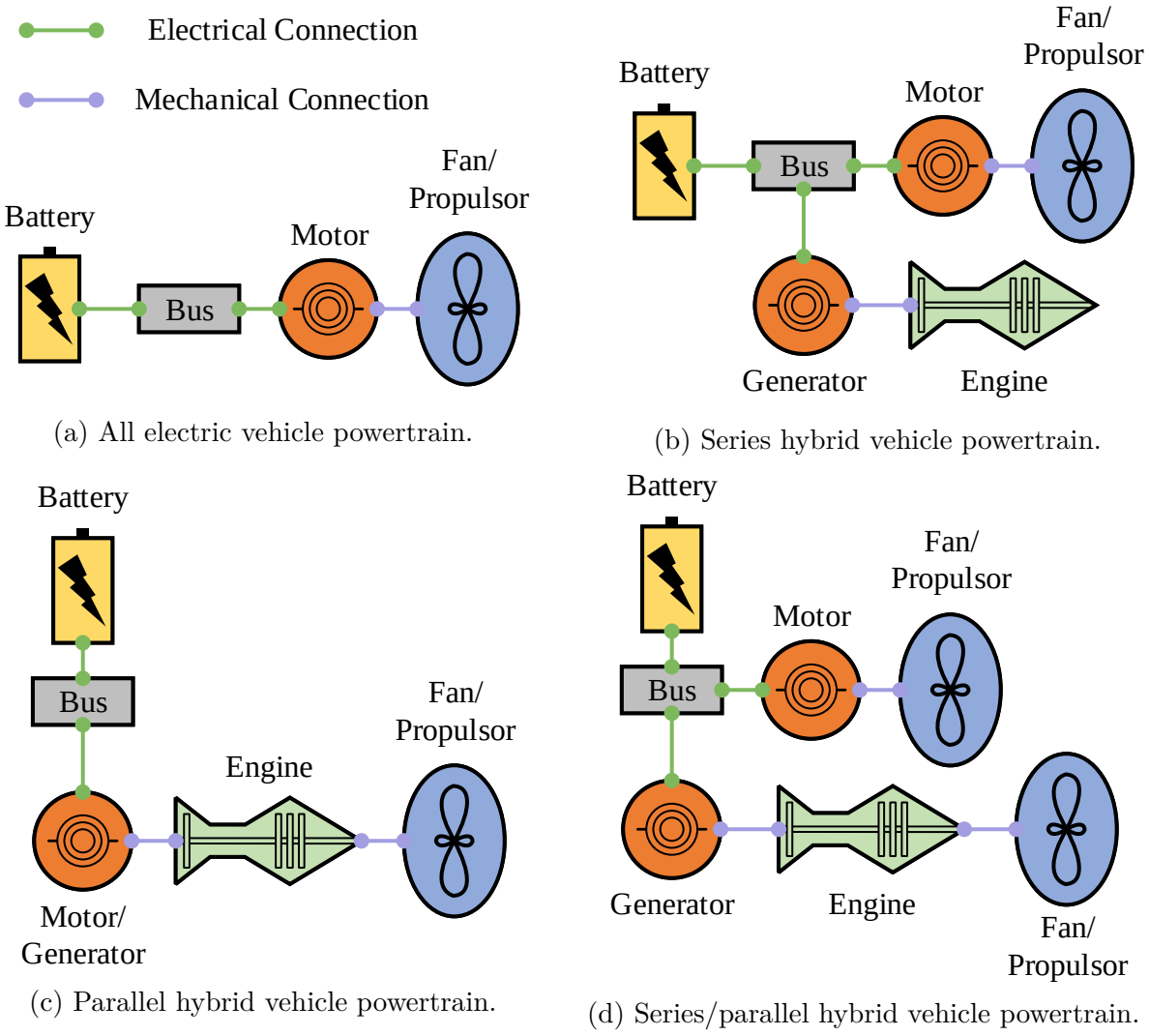


Figure 1.1: Historical and predicted trends for electrification of aircraft. Modified from [4].

temperatures are able to reject heat to ambient air. In an electrified vehicle, inefficiencies in the electrical components generate heat. As the components' temperatures change, so do their electrical characteristics [7, 8]. To maintain safe operation, heat is rejected to a thermal management system because the electronics, located inside the vehicle body, cannot directly reject heat to ambient air [5]. The ability to understand this coupling between electrical and thermal energy domains is key to improving not just vehicle reliability, but also the vehicle performance and efficiency.

An aspect of the trend of electrification is hybridization. In contrast to an all electric powertrain (Figure 1.2a), a hybrid powertrain commonly utilizes both an engine and a battery pack to provide power for the vehicle. The system can leverage the energy dense fuel for slower transient loads while the battery is more adept at servicing the faster electrical loads. A hybrid electric aircraft's drivetrain has 3 architecture options: series, parallel, and series/parallel (Figure 1.2). In a series drivetrain (Figure 1.2b), the engine is connected to a generator to generate electrical energy and thrust is produced by an electric motor. In a parallel configuration (Figure 1.2c), the engine, an electric motor, and main driveshaft are

mechanically connected. In this case, the engine can provide thrust or generate electrical energy via the motor. The electric motor can also provide thrust. A series/parallel configuration (Figure 1.2d) is a combination of both series and parallel architectures. Similar to the parallel configuration, the engine directly provides thrust, but also generates electrical power via a generator. Similar to the series architecture, additional electric motors are used to generate additional thrust. The series/parallel configuration for aircraft is similar to the power split architecture for ground vehicles [10]. Hybridization yields an extra degree-



(a) All electric vehicle powertrain.

(b) Series hybrid vehicle powertrain.

(c) Parallel hybrid vehicle powertrain.

(d) Series/parallel hybrid vehicle powertrain.

Figure 1.2: Representations of (a) all electric, (b) series hybrid, (c) parallel hybrid, and (d) series/parallel hybrid vehicle configurations. Electrical and mechanical connections are listed in green and purple respectively. Note that these schematics neglect any other loads, energy storage, or power generation connected to the bus. Modified from [9].

of-freedom for improving the performance or efficiency of the vehicle. However, the extra degree-of-freedom adds significant complexity to the controller that must choose when and how to operate the engine to achieve operational gains.

Lastly, electrified vehicles are a highly complex interconnection of multi-domain systems-of-systems. This thesis will specifically focuses on the analysis of the powertrain of a hybrid Unmanned Aerial Vehicle (UAV). In practice, UAVs have mission-specific designs, a characteristic that highlights the modularity of this class of systems; facilitating the development of various architectures from the same set of core components. It is important to utilize a modular modeling framework to reflect the physical system because model-based control strategies are useful for these classes of systems. Furthermore, it is important to adhere to a similar modular framework for controller design, such that if a new architecture was designed, there would be seamless integration with a new controller. It is also key that the modeling and control practices maximize computational efficiency. Reducing the necessary demand for computation power onboard the aircraft may result is cheaper manufacturing costs, decreased vehicle weight, and/or less heat generation.

1.2 Background

1.2.1 Current Practices

There has been a significant effort towards dynamic modeling of energy and power systems. Through the development of dynamic system models, systems and control engineers can rapidly evaluate various system and control architectures prior to physical implementation. In the thermal domain, simulation tools such as Thermosys and ATTMO [11, 12] have been used to design and evaluate refrigeration cycles for buildings and mobile systems. The MATLAB Simscape toolbox [13] has been used for the design of dynamic electrical systems. For target aircraft applications, the PowerFlow toolbox [14] couples multiple energy domains by offering various electrical, thermal, hydraulic, and mechanical components models. The modular tip-to-tail scope of these toolboxes makes them particularly useful for system design and simulation. However, capturing dynamics that span a wide range of timescales in a uni-

fied, scalable and computationally efficient framework has not been addressed. For example, it would be difficult to simulate a very fast millisecond time scale electrical model built in Simscape with a slow 10-100 second time scale thermal management system designed in Thermosys. Furthermore, extracting models from proprietary software for implementation in a model-based controller can be very challenging if the user cannot access the underlying model. Therefore, current modeling practices need to be improved through the development of a multi-domain, modular, scalable, and computationally efficient modeling framework from which a set of dynamic equations can be easily extracted.

Electrical powertrain control considers the actuation of individual components as well as higher level coordination of the power sharing between the battery and engine/generator. Since there exist significant losses in both the electrical and mechanical systems, system-wide efficiency gains are achieved through coordination between energy domains. Initial control strategies for the battery-engine/generator power share were rule-based techniques such as thermostatic [15] and power following [16] control. Later, additional heuristics were layered on to these baseline approaches to improve robustness and efficiency [17]. In these rule-based state-machine-like strategies, different operating modes typically depend on distinct thresholds. However, these control designs were sensitive so fuzzy logic rule-based strategies were developed to improve robustness to disturbances [16, 17]. For more optimal solutions implementable in real-time, dynamic programming (DP) has been used to determine the optimal power share offline [18, 17]. However, the DP complexity grows exponentially with the number of system states [19].

Model predictive control (MPC) [20] is another suitable optimal real-time control tool for hybrid systems. In MPC, a model is used to predict system behavior over some predefined time horizon and optimization tools determine an optimal set of inputs that are then applied to the system. In addition to hybrid vehicles [21], MPC has been applied to building HVAC systems [22], chemical plants [23], grid power economics [24], and aircraft fuel thermal management systems [25] (to name a few). Since optimization problems take time to solve, application of MPC to a system with fast dynamics is challenging. This issue is compounded when considering a centralized control approach for a system with many states and a wide range of timescales, making it challenging to predict system behavior far into the future.

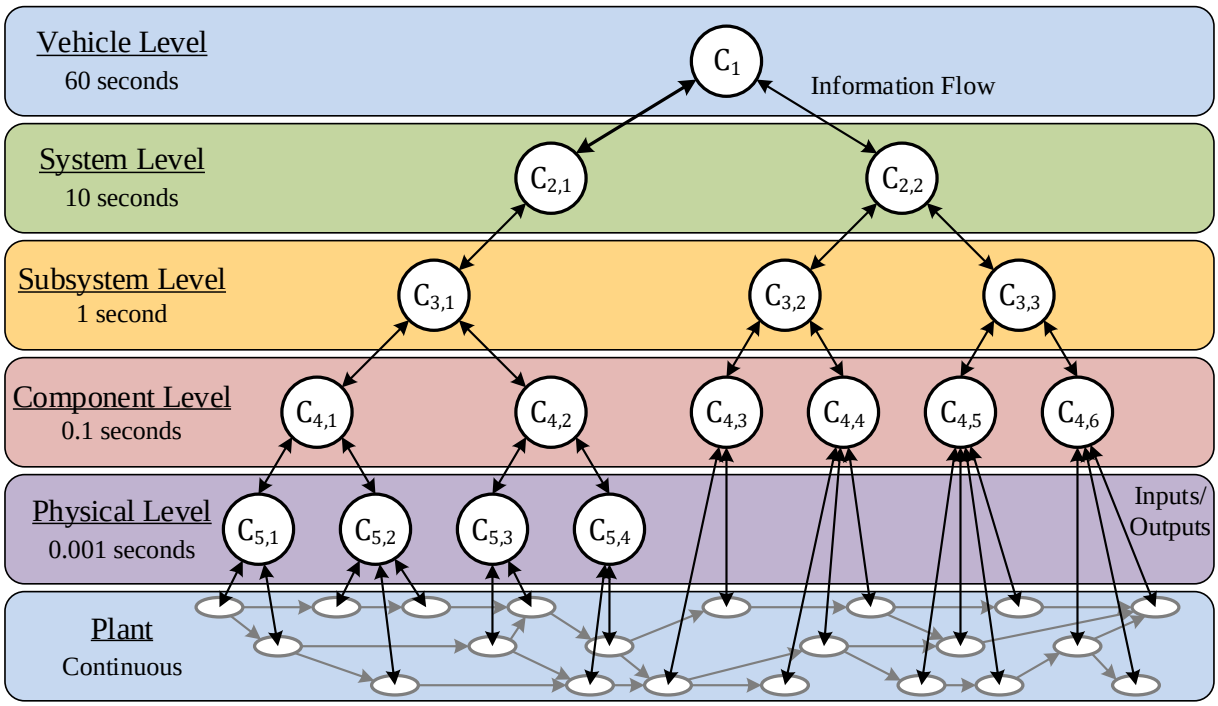


Figure 1.3: Example of a hierarchical control framework where each node at each level of the hierarchy represents a controller. Modified from [26].

Therefore, hierarchical MPC [4, 26, 27] approaches have been developed to address the issues related with computation and long-term mission planning. At the top of a hierarchy (Figure 1.3), a supervisor manages and optimizes vehicle performance with respect to slower system dynamics and objectives. The supervisor passes objectives to various predictive or regulatory controllers tailored to lower level subsystems. Each of the predictive controllers at the lower levels only model dynamics relevant to that subsystem, thus enabling the application of MPC to systems with both fast and slow dynamics. Two-level hierarchical MPC has been applied to building thermal systems [28] while more complex multi-level hierarchical MPC has been utilized in the control of an aircraft thermal and electrical system [25].

1.2.2 Scope of Thesis

The thesis considers issues related to the modeling and control of a hybrid UAVs powertrain. Specifically, this thesis addresses the multi-domain, modular, scalable, and computation issues associated with current modeling techniques. Addressing these issues facilities

a generalizable approach to the modeling and control of a variety of energy-based systems. Since UAVs are commonly designed for specific missions, the modular and scalable attribute is particularly relevant here. This thesis also offers two novel control techniques to address the concern of applying MPC algorithms to multi-timescale dynamic electro-mechanical systems. These contributions combine short-term reference tracking and disturbance rejection with long-term mission planning. Each controller will be evaluated on the system performance, reliability, and efficiency. Lastly, all models and controllers will be experimentally validated on flight-ready hardware to demonstrate how simulation results translate to a physical system.

1.3 Thesis Organization

The remainder of this thesis is as follows. Chapter 2 will introduce the generic formulation for the dynamic graph-based modeling framework for multi-domain systems. Graph-based models for components in a hybrid UAV powertrain are introduced, and then a novel system composition method is described and used to develop a system model. Chapter 3 describes the baseline, centralized MPC, and hierarchical MPC controller formulation. Chapter 4 establishes experimental validation techniques for the components and system model introduced in Chapter 2. Chapter 5 provides simulation and experimental results for each control design. Chapter 6 summarizes key contributions and outlines areas for future research.

Chapter 2

Modeling

2.1 Background

A mathematical modeling framework must be chosen to represent the systems onboard a hybrid UAV. To facilitate control design, the modeling framework used in this thesis should address the following requirements:

- *Energy domain agnostic* - Dynamic coupling between relevant energy domains across a large range of timescales must be captured in a unified modeling framework. By understanding the coupling between energy domains, a controller can make better decisions to satisfy mission objectives while maintaining safe operation.
- *Modular* - A full system model should be composed of various interconnected components and systems. Since UAVs have mission-specific designs, modularity provides flexibility in the types of systems that can by represented.
- *Scalable* - The modeling framework should be able to represent systems of multiple scales. It is important to be able to represent various sizes of UAVs as well as other types of mobile systems.
- *Computationally efficient* - System models must be computationally efficient for applications in real-time optimal model-based controllers.
- *Variable fidelity* - Depending on objectives and computational resources, the accuracy of a model may have a large variance. The modeling framework should be able to capture system dynamics at varying levels of complexity.

Various modeling frameworks have been used to capture the dynamics of mobile systems. Some of these efforts have led to the development of useful toolboxes such as Thermosys [11], ATTMO [12], Simscape [13], PowerFlow [14], etc. While some toolboxes meet the modularity and scalability requirement, they mostly lack the ability to capture the coupling between energy domains in a computationally efficient manner. Furthermore, it may be difficult to abstract a model useful for control design from proprietary software. Finite element approaches are typically quite accurate but lack the computational efficiency required for application in a real-time controller [29]. Bond graphs have been used to represent the complex dynamics of multi-domain systems in a modular and causal manner [30]. However, when modeling a large-scale system, the bond graph representation can become increasingly complex [31]. Additionally, bond graphs lack many analysis tools (e.g. model order reduction) that may facilitate advanced control design [27].

Therefore, this work chooses to employ a graph-based modeling framework to represent the powertrain of a hybrid UAV. Based in conversation laws, the graph-based modeling framework has been used to capture dynamics of hydraulic, thermal, electrical, and mechanical systems [3, 25, 32]. Furthermore, the structure of a graph model is inherently modular and scalable, which enables the representation of various classes of mobile systems. Initial efforts have highlighted the computational efficiency of the framework [3, 26]. Validation efforts have shown that graph models can adequately capture system dynamics [33, 34].

The remainder of this chapter is organized as follows. Section 2.2 outlines the basic and multi-domain graph model formulations. Section 2.3 develops modular component models used in the development of a hybrid UAV system model. Section 2.4 details an algorithmic method for the development of a system scale graph model composed of various component graph models. The UAV system graph model is described in Section 2.5. Lastly, Section 2.6 summarizes the contributions of this chapter.

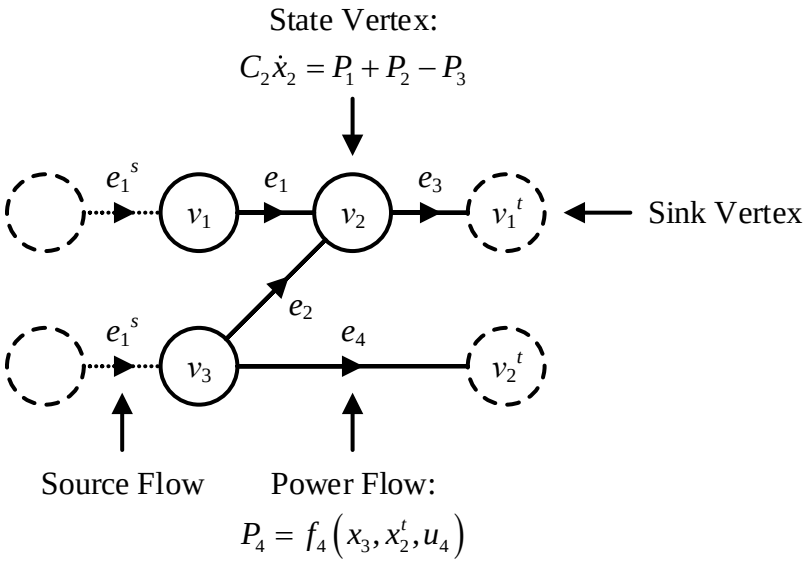


Figure 2.1: Notional graph used to highlight elements of the graph-based modeling framework [27].

2.2 Graph-Based Modeling

2.2.1 Graph-Based Modeling Fundamentals

System interconnections are captured by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of *order* N_v (number of vertices) and *size* N_e (number of edges) [35]. The graph $\mathcal{G}$ consists of vertices $v_i \in \mathcal{V} : i \in [1 : N_v]$ and directed edges $e_j \in \mathcal{E} : j \in [1 : N_e]$. Each directed edge e_j connects a *tail* vertex v_j^{tail} to a *head* vertex v_j^{head}. The set of edges entering and leaving vertex v_i is denoted by $\mathcal{E}_i^{head} = \{e_j : v_j^{head} = v_i\}$ and $\mathcal{E}_i^{tail} = \{e_j : v_j^{tail} = v_i\}$ respectively. A notional graph is provided in Figure 2.1.

Using conservation laws, the graph $\mathcal{G}$ is used to develop a dynamic system model $\mathcal{S}$. In the graph-based modeling framework, each vertex v_i represents a storage element/state variable x_i and each edge e_j represents the rate of transfer P_j between neighboring vertices. The transfer rates P_j will be called "power flows" because an energy-based analysis will be provided in the subsequent sections of this thesis. These vertices and edges are drawn as solid circles and directed edges as seen in Figure 2.1. The edge orientation defines positive power flow from v_j^{tail} to v_j^{head}, but note that the flow of power is bi-directional. Using conservation

laws, the state dynamics of of a single vertex v_i of $\mathcal{S}$ are given by

$$C_i \dot{x}_i = \sum_{\{j:e_j \in \mathcal{E}_i^{head}\}} P_j - \sum_{\{j:e_j \in \mathcal{E}_i^{tail}\}} P_j, \tag{2.1}$$

where $C_i \geq 0$ is the capacitance of the state x_i. Each power flow P_j in the system $\mathcal{S}$ can be any nonlinear function f_j of the two adjacent vertex states and an input u_j

$$P_j = f_j(x_j^{tail}, x_j^{head}, u_j). \tag{2.2}$$

The graph-based modeling framework also captures interactions with external systems such as the environment. These external interactions are represented by sink vertices or source edges. Sink vertices are treated as disturbances created by neighboring states external to the system and are represented as dashed circles as seen in Figure 2.1. Sink vertices are denoted by $v_i^t \in \mathcal{V} : i \in [1 : N_t]$ where N_t is the number of sink vertices in the graph. The sink state x_i^t is associated with sink vertex v_i^t, but note that sink states are excluded from the system's state vector x. Graph vertices can be partitioned by state vertices $v_i \in \bar{\mathcal{V}} : i \in [1 : N_v - N_t]$ and sink vertices $v_i^t \in \underline{\mathcal{V}} : i \in [1 : N_t]$ where $\bar{\mathcal{V}} \cup \underline{\mathcal{V}} = \mathcal{V}$.

Similarly, source flows are treated as disturbances created by neighboring systems. Source flows are represented by dashed edges (Figure 2.1) $e_j^s : j \in [1 : N_s]$ where N_s is the total number or source flows in the graph. Here, a power flow P_j^s is associated with each source flow e_j^s. Note that source flows are excluded from the set of graph edges $e^s \notin \mathcal{E}$. In this thesis, source flows are not utilized when formulating graph models.

The incidence matrix $M = [m_{ij}] \in \mathbb{R}^{N_v \times N_e}$ is used to describe the structure of vertex and edge connections of a graph and is defined as

$$m_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the tail of } e_j, \\ -1 & \text{if } v_i \text{ is the head of } e_j, \\ 0 & \text{else.} \end{cases} \tag{2.3}$$

The incidence matrix can be partitioned as

$$M = \begin{bmatrix} \bar{M} \\ \underline{M} \end{bmatrix} \text{ with } \bar{M} \in \mathbb{R}^{(N_v - N_t) \times N_e}, \tag{2.4}$$

where $\bar{M}$ maps power flows P to states x and and $\underline{M}$ maps power flows to sink states x^t.

Similarly, source flows P^s are mapped to states x using $D = [d_{ij}] \in \mathbb{R}^{(N_v - N_t) \times N_s}$ and is defined as

$$d_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the head of } e_j^s, \\ 0 & \text{else.} \end{cases} \tag{2.5}$$

Using the conservation equation (2.1), the full system dynamics are given by

$$C\dot{x} = -\bar{M}P + DP^s, \tag{2.6}$$

where C is a diagonal matrix of the capacitances associated with the system states. Using equation (2.2), the power flow vector P is described by

$$P = F(x, x^t, u). \tag{2.7}$$

The M and D matrices for the example graph in Figure 2.1 are provided below. In the graph, $N_v = 5$, $N_e = 4$, $N_t = 2$, and $N_s = 2$.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.8}$$

2.2.2 Multi-Domain Graph Models

The generic graph formulation has been applied to various physical domains. Initially, the graph-based modeling framework was used to model and validate hydraulic and thermal dynamics of various systems such as an aircraft fuel-thermal management system [27], turbomachinery [4], and a power inverter [36]. Graphs have recently been developed for electrical and mechanical systems in order to capture the coupling between energy domains in a modular and scalable modeling framework. Recently, the graph model formulation has been extended to model the powertrain of both aerial [3] and ground vehicles [32].

This chapter will expand upon the work of [3] and introduce graph models for components and subsystems of a hybrid UAV. System dynamics will be derived by applying energy conservation laws to the storage elements in the electrical, mechanical, and thermal energy domains. Hydraulic dynamics will not be considered. Furthermore, the following formulations will draw from parallels between electrical and mechanical systems. For example, the RLC circuit is the electrical counterpart to the mechanical mass-spring-damper system. The remainder of this subsection will first introduce relevant energy storage elements and their associated dynamics. Second, methods of power transfer between elements will be discussed. Lastly, rules for the construction of a multi-vertex and multi-domain graph model and model simplification steps are introduced.

The electrical and mechanical system dynamics are governed by compliance (capacitors and springs) and inertance (inductors and masses) elements, whereas thermal system dynamics are associated with thermal masses. These elements will be represented as vertices in a graph model. The graph capacitance C and state dynamic $\dot{x}$ for each each these elements are outlined in Table 2.1. The elements capacitance's are electrical capacitance C_e, inductance L, moment of inertia J, spring constants k and k_τ, mass m, and thermal capacitance C_T. The element states are voltage V, current I, torque τ, angular speed ω, force F, linear velocity v, and temperature T. In some instances, the graph capacitance and state of each element may differ from what is shown. For example, charge could be used in place of voltage (Figure 2.4). The classification of each vertex will be referred to as the vertex type $\mathcal{T}_v$.

Table 2.1: Graph model storage elements by energy domain for linear elements.

Energy Domain	Element	Capacitance	State	Vertex Type
Thermal	Thermal Mass	C_T	T	1
Electrical	Capacitor	C_eV	V	2
	Inductor	LI	I	3
Angular Mechanical	Rotating Mass	$J\omega$	ω	4
	Torsional Spring	$\frac{1}{k_\tau}\tau$	τ	5
Linear Mechanical	Translating Mass	mv	v	6
	Compression/ Tension Spring	$\frac{1}{k}F$	F	7

As mentioned previously, edges of the graph represent an exchange of energy between energy storage elements. Between the relevant energy domains there are various physical representations for these power flows P such as electro-magnetics, electrical resistance or friction loss, convection, etc. Specific means of power transfer for physical systems is described in Section 2.3. Although power flows can represent any nonlinear function, it is useful, for analysis, to extract and aggregate common linear or nonlinear terms. The results of this aggregation are shown below where any power flow P_j will be represented by the following generic formulation

$$\begin{aligned} P_j = g_j(x_t, x_h, u_j)(c_1 x_t \quad &+ c_2 x_h \quad + c_3 x_h x_t \quad + c_4 x_t^2 \quad + c_5 x_h x_t^2 \\ &+ c_6 x_t u_j + c_7 x_h u_j + c_8 x_h x_t u_j + c_9 x_t^2 u_j + c_{10} x_h x_t^2 u_j \\ &+ c_{11} x_t^3), \end{aligned} \tag{2.9}$$

where $c_k : k \in [1:11]$ are constant coefficients for the aggregated terms and g is any nonlinear function of the tail state $x_t = x_j^{tail}$, head state $x_h = x_j^{head}$, and edge input u_j. Equation (2.9) can still capture any nonlinearities through g. Note that g is commonly used to represent look-up tables that cannot be modeled as smooth functions. In this thesis, each term in (2.9) will be referred to as an edge type $\mathcal{T}_e$. Terms can be removed from the power flow calculation for edge e_i by choosing the associated constant coefficient to be zero, $c_k = 0$.

Next, when generating a multi-vertex and multi-domain graph model, it is important to

adhere to the following structural rules. When modeling within the electrical or mechanical energy domains, there should be alternating compliance and inertance vertices [32]. For example, in an electrical graph model a capacitor type vertex can only be connected to inductor type vertices and vice versa. When coupling between electrical and mechanical domains, there should be adjacent compliance or inertance vertices. For example, an inductor vertex could be connected to a rotating mass type vertex.

Observe the capacitance column for the electrical and mechanical dynamics in Table 2.1. Most of the graph capacitances are nonlinear because they are dependent on the state variable. Although the energy based analysis is useful for constructing a multi-domain graph, it becomes apparent that the state dynamics may not be represented in simplest form. For example, the voltage dynamics V for a capacitor C_e in parallel with a resistor R can be given as $C_e V\dot{V} = V^2/R$. As the capacitor voltage approaches 0, the graph capacitance approaches zero. For simulation, it is preferred to model the state dynamics in a simplified form $C_e\dot{V} = V/R$. To compensate for this issue, the generic graph formulation is altered, which results in the *modified graph formulation*. Note that this vocabulary is defined in, and specific to, this application. Modified graph properties, denoted by the superscript ‡, parallel the properties of the generic graph. The dynamics for a modified graph are given by

$$C^{\ddagger}\dot{x} = -\bar{M}^{\ddagger}P^{\ddagger} + DP^{s}, \quad \text{where} \tag{2.10a}$$

$$C^{\ddagger}_{ii} = \begin{cases} C_{ii} & \text{if } \mathcal{T}_{v,i} = 1, \\ C_{ii}/x_i & \text{else.} \end{cases} \tag{2.10b}$$

$$\bar{M}^{\ddagger} = [\bar{M}^{\ddagger}_k] = \begin{bmatrix} \bar{M}^{\ddagger}_1 & \bar{M}^{\ddagger}_2 & \bar{M}^{\ddagger}_3 & \bar{M}^{\ddagger}_4 & \bar{M}^{\ddagger}_5 & \bar{M}^{\ddagger}_6 & \bar{M}^{\ddagger}_7 \end{bmatrix}, \tag{2.10c}$$

$$P^{\ddagger} = [P^{\ddagger}_k] = \begin{bmatrix} P^{\ddagger}_1 & P^{\ddagger}_2 & P^{\ddagger}_3 & P^{\ddagger}_4 & P^{\ddagger}_5 & P^{\ddagger}_6 & P^{\ddagger}_7 \end{bmatrix}^T, \tag{2.10d}$$

where each partition $\bar{M}_k^\ddagger = [m_{ij}]$ of $M^\ddagger$ is defined as

$$m_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the tail of } e_j \text{ and } \mathcal{T}_{v,i} = k, \\ -1 & \text{if } v_i \text{ is the head of } e_j \text{ and } \mathcal{T}_{v,i} = k, \\ 0 & \text{else.} \end{cases} \tag{2.11}$$

Similarly, each partition $P_k^\ddagger = F(x, x^t, u)$ of $P^\ddagger$ is defined as

$$P_{k,j}^\ddagger = F_j(x_t, x_h, u_j) = \begin{cases} F_j(x_t, x_h, u_j) & \text{if } k = 1, \\ F_j(x_t, \cdot, u_j)/x_t & \text{if } v_j^{tail} \in \{\mathcal{V} : \mathcal{T}_{v,i} = k \neq 1\}, \\ F_j(\cdot, x_h, u_j)/x_h & \text{if } v_j^{head} \in \{\mathcal{V} : \mathcal{T}_{v,i} = k \neq 1\}, \\ 0 & \text{else.} \end{cases} \tag{2.12}$$

for each $j \in [1 : N_e]$ and i is the vertex index of v_j^{tail} or v_j^{head}. It can be seen that, for example, the partition $P_3^\ddagger$ represents all the power flows that are a function of a current state divided by that same current state (e.g. $P_i = IV \rightarrow P_{3,i}^\ddagger = V$). Similarly, $\bar{M}_3^\ddagger$ maps the flows $P_3^\ddagger$ to system's current states. In a modified graph, the "power" flows $P_k^\ddagger$ are now thermal energy, current, voltage, torque, angular velocity, force, and linear velocity flows.

Now, any power flow $P_{k,j}^\ddagger$ will be represented by the following generic formulation

$$\begin{aligned} P_{k,j}^\ddagger = f_j(x_t, x_h, u_j)(b_1 x_t \quad &+ b_2 x_h \quad + b_3 x_t^2 \quad + b_4 \quad + b_5 x_h x_t \\ &+ b_6 x_t u_j + b_7 x_h u_j + b_8 x_t^2 u_j + b_9 u_j + b_{10} x_h x_t u_j \\ &+ b_{11} x_t^3), \end{aligned} \tag{2.13}$$

where $b_n : n \in [1 : 11]$ are constants relating to c_k (from (2.9)) through (2.12). An example detailing the construction of a multi-domain graph is provided in Appendix A.2.

2.2.3 System Linearization

Linear models are particularly useful to estimate local system behavior and efficiently solve complex optimization problems. The following section will outline the linearization procedure for the modified graph formulation given by (2.10a) and (2.13). Note, that with a few simplifications, the following process still holds for a full graph.

System nonlinearities are aggregated in the modified power flow vector $P^{\ddagger} = F(x, x^t, u)$. Therefore, (2.13) can be approximated by a first-order Taylor expansion

$$P^{\ddagger}_{k,j} \approx P^{\ddagger}_{k,j}(\bar{x}_t, \bar{x}_h, \bar{u}_j) + \lambda_{t,j}(x_t - \bar{x}_t) + \lambda_{h,j}(x_h - \bar{x}_h) + \lambda_{u,j}(u_j - \bar{u}_j), \tag{2.14}$$

about some trajectory $(\bar{x}_t, \bar{x}_h, \bar{u}_j)$ where $\lambda_{t,j}$, $\lambda_{h,j}$, and $\lambda_{u,j}$ are linearization coefficients for flow j. The linearization coefficients are the same size as the power flow vector and are given by

$$\begin{aligned}\lambda_{t,j} = {} & (b_1 + b_6\bar{u}_j)(\bar{f} + \bar{f}_{x_t}\bar{x}_t) + (b_2 + b_7\bar{u}_j)(\bar{f}_{x_t}\bar{x}_h) + (b_3 + b_8\bar{u}_j)(2\bar{f}\bar{x}_t + \bar{f}_{x_t}\bar{x}_t^2) \\ & + (b_4 + b_9\bar{u}_j)(\bar{f}_{x_t}) + (b_5 + b_{10}\bar{u}_j)(\bar{f} + \bar{f}_{x_t}\bar{x}_t)\bar{x}_h + (b_{11})\,(3\bar{f}\bar{x}_t^2 + \bar{f}_{x_t}\bar{x}_t^3),\end{aligned} \tag{2.15a}$$

$$\begin{aligned}\lambda_{h,j} = {} & (b_1 + b_6\bar{u}_j)(\bar{f}_{x_h}\bar{x}_t) + (b_2 + b_7\bar{u}_j)(\bar{f} + \bar{f}_{x_h}\bar{x}_h) + (b_3 + b_8\bar{u}_j)(\bar{f}_{x_h}\bar{x}_t^2) \\ & + (b_4 + b_9\bar{u}_j)(\bar{f}_{x_h}) + (b_5 + b_{10}\bar{u}_j)(\bar{f} + \bar{f}_{x_h}\bar{x}_h)\bar{x}_t + (b_{11})\,(\bar{f}_{x_h}\bar{x}_t^3),\end{aligned} \tag{2.15b}$$

$$\begin{aligned}\lambda_{u,j} = {} & (b_1\bar{x}_t + b_2\bar{x}_h + b_3\bar{x}_t^2 + b_4 + b_5\bar{x}_t\bar{x}_h + b_{11}\bar{x}_t^3)\bar{f}_u \\ & + (b_6\bar{x}_t + b_7\bar{x}_h + b_8\bar{x}_t^2 + b_9 + b_{10}\bar{x}_t\bar{x}_h)(\bar{f} + \bar{f}_u\bar{u}_j),\end{aligned} \tag{2.15c}$$

where $\bar{f} = f_j\,(\bar{x}_t, \bar{x}_h, \bar{u}_j)$, and $\bar{f}_{x_t}$, $\bar{f}_{x_h}$, and $\bar{f}_{x_u}$ are the derivatives of f_j with respect to the tail state, head state, and input evaluated at $(\bar{x}_t, \bar{x}_h, \bar{u}_j)$. Using (2.14), (2.10a) can be rewritten as

$$C^{\ddagger}\dot{x} \approx -\bar{M}^{\ddagger}\left(P^{\ddagger}(\bar{x}_t, \bar{x}_h, \bar{u}) + \lambda_t(x_t - \bar{x}_t) + \lambda_h(x_h - \bar{x}_h) + \lambda_u(u - \bar{u})\right) + DP^s. \tag{2.16}$$

Simplifying (2.16) will yield

$$C^{\ddagger}\dot{x} \approx A_1 x + A_2 x^t + B_1 u + B_2 + DP^s, \quad \text{where} \tag{2.17a}$$

$$A_1 = -\bar{M}^{\ddagger}(\lambda_t \bar{M}_{tails} + \lambda_h \bar{M}_{heads}) \in \mathbb{R}^{N_v - N_t \times N_v - N_t}, \tag{2.17b}$$

$$A_2 = -\bar{M}^{\ddagger}(\lambda_t \underline{M}_{tails} + \lambda_h \underline{M}_{heads}) \in \mathbb{R}^{N_v - N_t \times N_t}, \tag{2.17c}$$

$$B_1 = -\bar{M}^{\ddagger}\lambda_u B_u \in \mathbb{R}^{N_v - N_t \times N_u}, \tag{2.17d}$$

$$B_2 = -(\bar{M}^{\ddagger}P_{sp}(\bar{x}_t, \bar{x}_h, \bar{u}) + A_1\bar{x} + A_2\bar{x}^t + B_1\bar{u}) \in \mathbb{R}^{N_v - N_t}, \tag{2.17e}$$

where M_{tails} and M_{heads} are mappings from states to power flows and B_u is a mapping from inputs to power flows. Note that $x_t = \bar{M}_{tails}x$ and $x_h = \bar{M}_{heads}x$. These quantities are defined as

$$M_{tails} = repmat([m_{ij}], 1, 7)^T, \text{ where } [m_{ij}] = \begin{cases} 1 & \text{if } v_i \text{ is the tail of } e_j, \\ 0 & \text{else.} \end{cases}, \tag{2.18a}$$

$$M_{heads} = repmat([m_{ij}], 1, 7)^T, \text{ where } [m_{ij}] = \begin{cases} 1 & \text{if } v_i \text{ is the head of } e_j, \\ 0 & \text{else.} \end{cases}, \tag{2.18b}$$

$$B_u = repmat([b_{ij}], 7, 1), \text{ where } [b_{ij}] = \begin{cases} 1 & \text{if } u_j \text{ is an input to } e_i, \\ 0 & \text{else.} \end{cases}. \tag{2.18c}$$

In (2.18), the $repmat(A, r, c)$ function duplicates matrix A by r rows and c columns. Furthermore, the 7 is indicative of the total number of relevant vertex types $\mathcal{T}_v$ in the the graph. For example, if one was interested in modeling only electro-thermal behaviors, the 7 becomes a 3 (temperature, voltage, and current states).

It would be reasonable to invert the C matrix to arrive at a linear state space model. However, there may exist algebraic states in the graph ($C_{ii} = 0$) and therefore C may be singular. In this case, (2.17) can be rewritten as a continuous time linear differential algebraic equation (DAE):

$$\begin{bmatrix} \dot{x}_d \\ 0 \end{bmatrix} = A_1' \begin{bmatrix} x_d \\ x_a \end{bmatrix} + A_2' x^t + B_1' u + B_2' + D' P^s, \quad \text{where} \tag{2.19a}$$

$$A_1' = \begin{bmatrix} \bar{A}_{1,d}' & \bar{A}_{1,a}' \\ \underline{A}_{1,d}' & \underline{A}_{1,a}' \end{bmatrix} = \begin{bmatrix} [C_d^\ddagger]^{-1} \bar{A}_{1,d} & [C_d^\ddagger]^{-1} \bar{A}_{1,a} \\ \underline{A}_{1,d} & \underline{A}_{1,a} \end{bmatrix}, \tag{2.19b}$$

$$A_2' = \begin{bmatrix} \bar{A}_2' \\ \underline{A}_2' \end{bmatrix} = \begin{bmatrix} [C_d^\ddagger]^{-1} \bar{A}_2 \\ \underline{A}_2 \end{bmatrix}, \tag{2.19c}$$

$$B_1' = \begin{bmatrix} \bar{B}_1' \\ \underline{B}_1' \end{bmatrix} = \begin{bmatrix} [C_d^\ddagger]^{-1} \bar{B}_1 \\ \underline{B}_1 \end{bmatrix}, \tag{2.19d}$$

$$B_2' = \begin{bmatrix} \bar{B}_2' \\ \underline{B}_2' \end{bmatrix} = \begin{bmatrix} [C_d^\ddagger]^{-1} \bar{B}_2 \\ \underline{B}_2 \end{bmatrix}, \tag{2.19e}$$

$$D' = \begin{bmatrix} \bar{D}' \\ \underline{D}' \end{bmatrix} = \begin{bmatrix} [C_d^\ddagger]^{-1} \bar{D} \\ \underline{D} \end{bmatrix}. \tag{2.19f}$$

The overbars and underbars signify the rows of the matrix that map to dynamic states and algebraic states respectively. Similarly, the subscripts d and a signify the columns of A_1 that are multiplied by dynamic and algebraic states respectively. Note that there exists a unique solution to the DAE (2.19) when $\underline{A}_{1,a}$ is non-singular, which is assumed for the graph models introduced in the remainder of this thesis.

Next, the system (2.19) is discretized using the Forward Euler method

$$\dot{x} = \frac{x_{k+1} - x_k}{\Delta t} \tag{2.20}$$

where x_k is the the state vector at step k, x_{k+1} is the the state vector at step $k+1$, and Δt is the step size. The linear-discrete time DAE system is given as

$$\begin{bmatrix} x_{(k+1),d} \\ 0 \end{bmatrix} = A_{z1} \begin{bmatrix} x_{k,d} \\ x_{k,a} \end{bmatrix} + A_{z2} x_k^t + B_{z1} u_k + B_{z2} + D_z P_k^s \quad \text{where,} \tag{2.21a}$$

$$A_{z1} = \begin{bmatrix} \Delta t \bar{A}'_{1,d} + I & \Delta t \bar{A}'_{1,a} \\ \underline{A}'_{1,d} & \underline{A}'_{1,a} \end{bmatrix}, \tag{2.21b}$$

$$A_{z2} = \begin{bmatrix} \Delta t \bar{A}'_{2} \\ \underline{A}'_{2} \end{bmatrix}, \tag{2.21c}$$

$$B_{z1} = \begin{bmatrix} \Delta t \bar{B}'_{1} \\ \underline{B}'_{1} \end{bmatrix}, \tag{2.21d}$$

$$B_{z2} = \begin{bmatrix} \Delta t \bar{B}'_{2} \\ \underline{B}'_{2} \end{bmatrix}, \tag{2.21e}$$

$$D_{z} = \begin{bmatrix} \Delta t \bar{D}' \\ \underline{D}' \end{bmatrix}, \tag{2.21f}$$

where $I \in \mathbb{R}^{N_d \times N_d}$ and N_d is the total number of dynamic states.

In summary, we have presented the following graph model formulations: full non-linear model (2.6) (2.9), modified non-linear model (2.10a) (2.13), continuous time linear DAE model (2.19), and discrete time linear DAE model (2.21).

2.3 Component Modeling

The goal of the modeling efforts is to develop a graph-based model for the hybrid UAV system architecture as described in Figure 2.2. However, it is first important to understand the dynamics of the individual components and subsystems. This candidate powertrain consists of a battery pack, motor, power electronics, genset, electrical bus, and auxiliary load.

At the core of the electrical system is the main electrical bus, which appropriately distributes electrical power to four subsystems. The first subsystem is the drivetrain that consists of power electronics, a single drive motor, and propeller. The power electronics condition DC electrical power from the main bus to AC electrical power required to drive the motor. A propeller fixed to the main shaft of the motor generates thrust to propel the vehicle. Next, the auxiliary subsystem consists of power electronics and an auxiliary load.

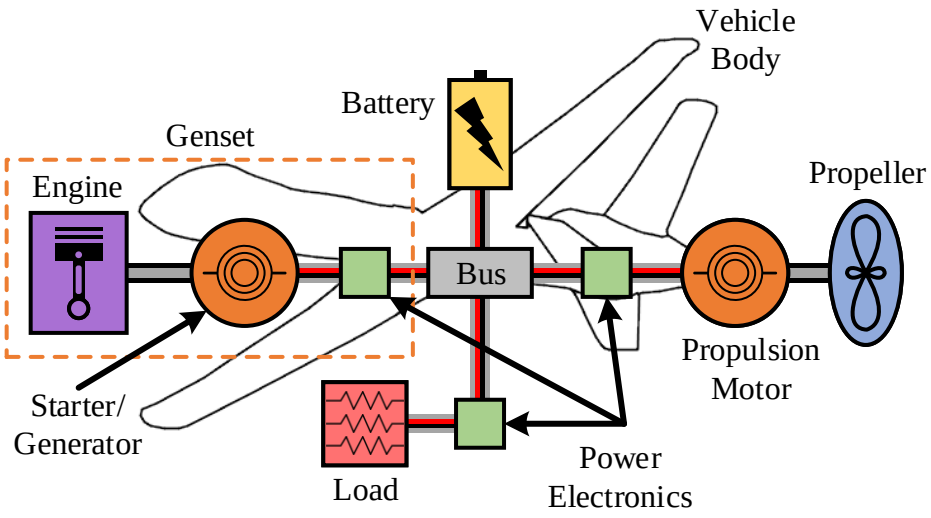
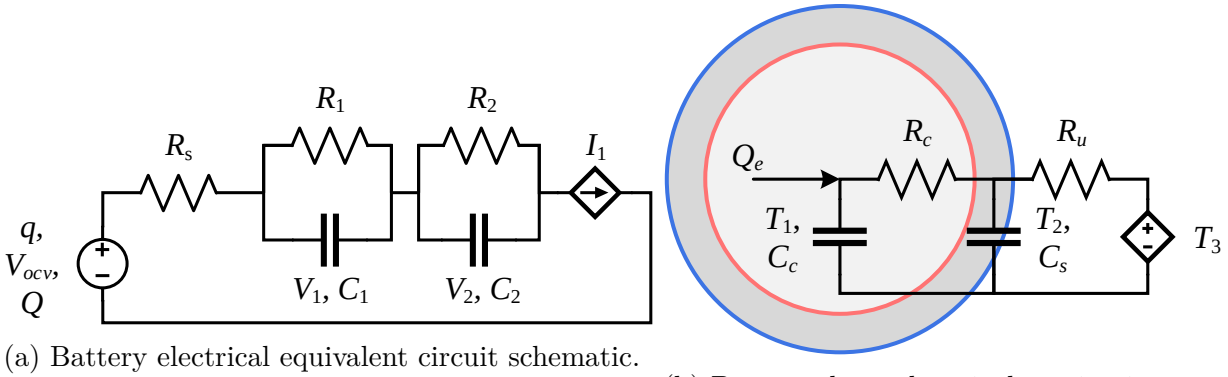


Figure 2.2: A series hybrid UAV architecture.

Similar to the drivetrain, the power electronics regulate the voltage applied to the auxiliary load that represents background electrical loads, avionics, and sensors onboard the aircraft. Third, the genset is composed of an engine, starter/generator, and power electronics. This subsystem converts chemical potential energy of fuel to electrical energy that is utilizable by the rest of the electrical network. Lastly, the battery is the main electrical energy storage device and can charge and discharge to meet the power demands of the vehicle. The remainder of this section will derive graph-based models for each of the aforementioned components as well as a simplified dynamic vehicle model. These models will serve as the modular building blocks to model the full vehicle in Section 2.5. For visualization, the vertex color in the following graphs models is as follows: thermal mass (red), capacitor (green), inductor (yellow), rotating mass (blue), and translating mass (orange).

2.3.1 Battery

Battery packs are a common choice for energy storage onboard a hybrid vehicle. In comparison to turbomachinery, batteries have greater power ramp rates to meet the desired power demand. However, batteries have strict state of charge (SOC), charge/discharge rates, and thermal constraints to maintain safe operation. Constraint violations may cause permanent damage to the pack or thermal runaway may ensue [37]. For control purposes, batteries are



(a) Battery electrical equivalent circuit schematic.

(b) Battery thermal equivalent circuit schematic. Note that $Q_e = R_s I_1^2 + \frac{V_1^2}{R_1} + \frac{V_2^2}{R_2}$.

Figure 2.3: The (a) electrical and (b) thermal circuit schematics used to model battery dynamics.

typically modeled as equivalent RC circuits [8]. Here, a battery is modeled as a 3$^{\text{rd}}$-order electrical circuit (Figure 2.3a) and 2$^{\text{nd}}$-order thermal circuit (Figure 2.3b).

The energy storage elements of each circuit are analyzed to develop the graph model. In the electrical domain, the voltage dynamics of the RC pairs and voltage source are given by

$$C_1 V_1 \dot{V}_1 = -\frac{V_1^2}{R_1} + V_1 I_1, \tag{2.22a}$$

$$C_2 V_2 \dot{V}_2 = -\frac{V_2^2}{R_2} + V_2 I_1, \tag{2.22b}$$

$$Q V_{ocv} \dot{q} = -V_{ocv} I_1, \tag{2.22c}$$

where C_1 and C_2 are the electrical capacitance of the capacitors, V_1 and V_2 are the voltages across the capacitors, R_1 and R_2 are the resistances of the resistors in the RC pairs, Q is the battery capacity, q is the battery SOC, $V_{ocv} = f(q)$ is the open circuit voltage (OCV), and I_1 is the pack's current demand.

In the thermal domain, the cell temperature dynamics are given by

$$C_c \dot{T}_1 = R_s I_1^2 + \frac{V_1^2}{R_1} + \frac{V_2^2}{R_2} - \frac{1}{R_c}(T_1 - T_2), \tag{2.23a}$$

$$C_s \dot{T}_2 = \frac{1}{R_c}(T_1 - T_2) - \frac{1}{R_u}(T_2 - T_3), \tag{2.23b}$$

where R_s is the series internal resistance of the battery, C_c is the heat capacity of the battery core, C_s is the heat capacity of the battery shell, T_1 is the core temperature, T_2 is the surface temperature, T_3 is a surrounding fluid temperature, R_c is the internal thermal conduction resistance, and R_u is the thermal convection resistance.

This model captures 3 types of power flows: electrical power transfer VI, power loss through a resistor V^2/R and RI^2, and thermal conduction/convection $\Delta T/R$. The battery graph model is provided in Figure 2.4. The graph state vector, capacitance vector, power flow coefficients c, and property look-up coefficients f are provided below.

$$x = \begin{bmatrix} q & V_1 & V_2 & T_1 & T_2 \end{bmatrix}^T, \tag{2.24a}$$

$$C = \begin{bmatrix} QV_{ocv} & C_1V_1 & C_2V_2 & C_c & C_s \end{bmatrix}^T, \tag{2.24b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_s & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{R_c} & -\frac{1}{R_c} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{R_u} & -\frac{1}{R_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.24c}$$

$$f = \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} V_{ocv}(q) & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{2.24d}$$

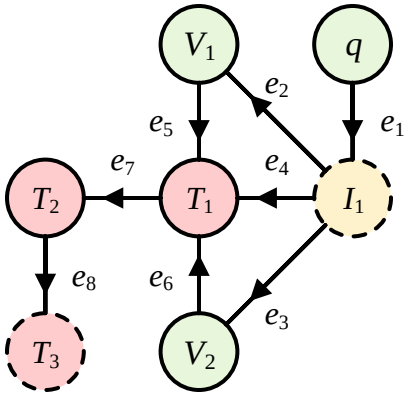


Figure 2.4: Graph model for the battery.

2.3.2 Electric Motor

Electric motors are used for electric propulsion in an aircraft's powertrain. There are various types of electrical motor architectures such as DC, brushless DC (BLDC) and induction type motors [38]. Motors operate by using a varying electric current to generate a magnetic field. The induced magnetic field interacts with a second magnetic field generated by a permanent magnet (or another induced field) to generate torque that spins a rotor. A DC motor has a permanent magnet in the stator (stationary part) and a field winding in the rotor. Mechanical brushes are used to conduct DC current from the motor terminals and invert it to an AC current in the field winding in order to generate a magnetic field. A BLDC motor uses similar principles except the windings are located on the stator and the rotor has a permanent magnet. In place of mechanical brushes, an electric inverter is used to generate the AC current that induces the magnetic field. An induction motor has both a stator winding and field winding. The stator winding, powered by an electric inverter, induces an AC current in the field winding which induces its own magnetic field.

Brushless DC motors have become a popular choice for many mobile applications. In comparison to the DC motor, BLDC motors have low maintenance requirements because the do not use mechanical brushes [39]. In comparison to induction machines, BLDC motors tend to be higher performing and more efficient [39]. In this section, a DC motor model is introduced and then adapted to represent the BLDC motor.

2.3.2.1 DC Motor

A DC motor is electrically modeled as a single armature circuit, mechanically modeled as a rotating shaft, and thermally modeled as a single thermal mass (Figure 2.5). The electrical, mechanical, and thermal energy storage elements of Figure 2.5 are analyzed to develop the graph model. The dynamics of each element are described by

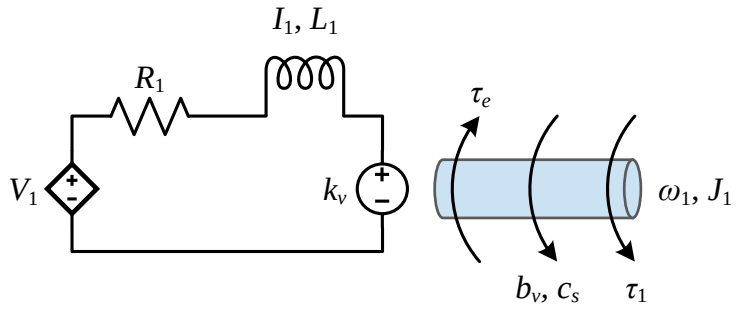
$$L_1 I_1 \dot{I}_1 = V_1 I_1 - R_1 I_1^2 - k_v I_1 \omega_1, \tag{2.25a}$$

$$J_1 \omega_1 \dot{\omega}_1 = k_v I_1 \omega_1 - \tau_1 \omega_1 - b_v \omega_1^2 - c_s \omega_1 sig(\omega_1), \tag{2.25b}$$

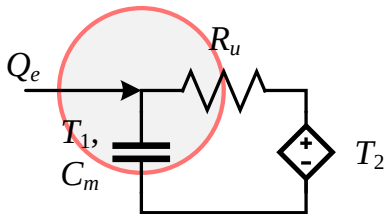
$$C_m \dot{T}_1 = R_1 I_1^2 + b_v \omega_1^2 + c_s \omega_1 sig(\omega_1) - \frac{1}{R_u}(T_1 - T_2), \tag{2.25c}$$

where L_1 is the coil inductance, I_1 is the motor current, V_1 is the terminal voltage, R_1 is the coil resistance, k_v is the motor constant, J_1 is the shaft moment of inertia, ω_1 is the shaft speed, $\tau_1 = f(\omega_1, v_1)$ is the load torque, b_v is the viscous friction coefficient, c_s is the static friction coefficient, C_m is the motor heat capacity, T_1 is the motor temperature, T_2 is a surrounding fluid temperature, and R_u is the thermal convection resistance. Static friction is typically modeled using the sign function $sgn(\omega_1)$. However, the sign function can create numerical issues near $\omega_1 = 0$. Instead, the sign function is approximated using a sigmoid function

$$sig(\omega_1) = \frac{2}{1 + e^{-a\omega_1}} - 1 \tag{2.26}$$



(a) Motor electrical equivalent circuit and mechanical schematic.



(b) Motor thermal equivalent circuit schematic. Note that $Q_e = R_1 I_1^2 + b_v \omega_1^2 + c_s \omega_1 sig(\omega_1)$.

Figure 2.5: The (a) electro-mechanical and (b) thermal schematics used to model motor dynamics.

where $a > 0$ is a smoothing coefficient.

This model captures 5 types of power flows: electrical and mechanical power transfer VI and $\omega\tau$, power loss due to a resistor or friction RI^2 and $b\omega^2$, static friction loss $c\omega sig(\omega)$, electro-mechanical power conversion $k_v I\omega$ and thermal convection $\Delta T/R$. The DC motor graph model is provided in Figure 2.6. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} I_1 & \omega_1 & T_1 \end{bmatrix}^T, \tag{2.27a}$$

$$C = \begin{bmatrix} L_1 I_1 & J_1 \omega_1 & C_m \end{bmatrix}^T, \tag{2.27b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_v & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{R_u} & -\frac{1}{R_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.27c}$$

$$f = \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & \tau_1 & 1 & (b_v + c_s sig(\omega_1)) & 1 \end{bmatrix} \end{array}. \tag{2.27d}$$

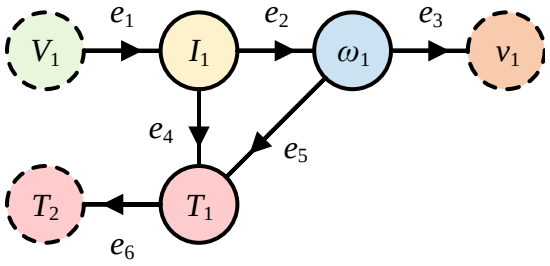


Figure 2.6: Graph model for the motor.

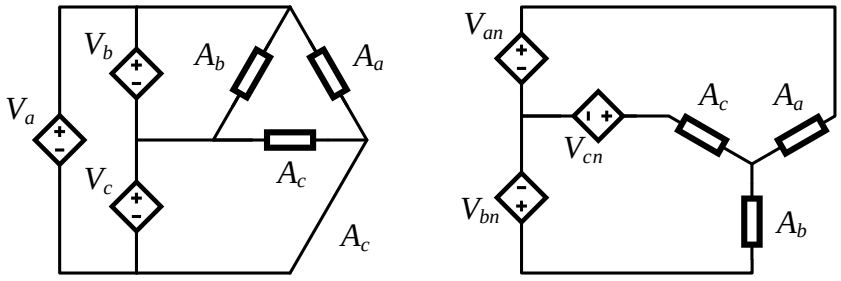


Figure 2.7: Examples of a Δ (left) and Y (right) connected loads A_a, A_b, and A_c.

2.3.2.2 Brushless DC Motor

Brushless DC motors typically have 3 armatures connected in a delta (Δ) or wye (Y) configuration (Figure 2.7). A detailed derivation for a direct-quadrature (dq0) BLDC motor model using the regular Park Transformation K_r (2.28a) is found in [40]. However, that model cannot be directly modified to fit within the graph-based modeling framework because it does not follow conservation of energy. The following analysis parallels the derivation in [40], but instead utilizes the power-invariant Park Transform K_p (2.28b) [41] such that the resulting dynamics can be represented by a graph.

$$K_r = \sqrt{\frac{2}{3}} \begin{bmatrix} \sin(\theta_e) & \sin(\theta_e - \frac{2\pi}{3}) & \sin(\theta_e + \frac{2\pi}{3}) \\ \cos(\theta_e) & \cos(\theta_e - \frac{2\pi}{3}) & \cos(\theta_e + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \tag{2.28a}$$

$$K_p = \frac{2}{3} \begin{bmatrix} \sin(\theta_e) & \sin(\theta_e - \frac{2\pi}{3}) & \sin(\theta_e + \frac{2\pi}{3}) \\ \cos(\theta_e) & \cos(\theta_e - \frac{2\pi}{3}) & \cos(\theta_e + \frac{2\pi}{3}) \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix}. \tag{2.28b}$$

Here $K_r, K_p : \begin{bmatrix} a & b & c \end{bmatrix} \rightarrow \begin{bmatrix} d & q & 0 \end{bmatrix}$ map quantities in the abc reference frame to the $dq0$ reference frame and θ_e is the electrical angle of the machine.

First, it is known that the total instantaneous electrical power in abc variables P_{abc} is given by

$$P_{abc} = V_{abc} I_{abc} = V_a I_a + V_b I_b + V_c I_c, \tag{2.29}$$

where $V_{abc} = \begin{bmatrix} V_a & V_b & V_c \end{bmatrix}$ and $I_{abc} = \begin{bmatrix} I_a & I_b & I_c \end{bmatrix}$ are voltages and currents expressed in abc variables. Next define, $V_{dq0} = \begin{bmatrix} V_d & V_q & V_0 \end{bmatrix}$ and $I_{dq0} = \begin{bmatrix} I_d & I_q & I_0 \end{bmatrix}$ as the voltages and currents expressed in dq0 variables. Using the inverse of the power-invariant Park Transform, the abc variables can be expressed in terms of the $dq0$ variables.

$$V_{abc} = K_p^{-1} V_{dq0}, \tag{2.30a}$$

$$I_{abc} = K_p^{-1} I_{dq0}. \tag{2.30b}$$

Through algebraic manipulation and using (2.29), it is derived that the total instantaneous electrical power in $dq0$ variables P_{dq0} is given by

$$P_{dq0} = V_{dq0} I_{dq0} = V_d I_d + V_q I_q + V_0 I_0 = V_{abc} I_{abc} = P_{abc}. \tag{2.31}$$

For a balanced three-phase system, $V_0 = I_0 = 0$.

Next, the dynamics of each motor armature, in abc variables, parallels that given by (2.25a)

$$V_{abc} = R I_{abc} + \dot{\lambda}_{abc}, \tag{2.32a}$$

$$\lambda_{abc} = L_s I_{abc} + \Lambda_{abc}, \tag{2.32b}$$

$$\Lambda_{abc} = \Lambda \begin{bmatrix} \sin(\theta_e) \\ \sin(\theta_e - \frac{2\pi}{3}) \\ \sin(\theta_e + \frac{2\pi}{3}) \end{bmatrix}, \tag{2.32c}$$

$$L_s = \begin{bmatrix} L_0 + L_a + L_b \cos(2\theta_e) & -\frac{1}{2} L_a + L_b \cos(2\theta_e - \frac{2\pi}{3}) & -\frac{1}{2} L_a + L_b \cos(2\theta_e + \frac{2\pi}{3}) \\ -\frac{1}{2} L_a + L_b \cos(2\theta_e - \frac{2\pi}{3}) & L_0 + L_a + L_b \cos(2\theta_e - \frac{2\pi}{3}) & -\frac{1}{2} L_a + L_b \cos(2\theta_e + 2\pi) \\ -\frac{1}{2} L_a + L_b \cos(2\theta_e + \frac{2\pi}{3}) & -\frac{1}{2} L_a + L_b \cos(2\theta_e + 2\pi) & L_0 + L_a + L_b \cos(2\theta_e + \frac{2\pi}{3}) \end{bmatrix}, \tag{2.32d}$$

where L_s is the stator inductance matrix, λ_{abc} are the armature flux linkages, Λ is the flux linkage, L_0 is self inductance, and L_a and L_b are mutual inductances. Because the magnetic field is rotating, V_{abc}, λ_{abc}, and L_s are all sinusoidal varying parameters, which is not ideal for

application in a controller that may be trying to predict system behavior seconds or minutes into the future. Therefore, the rotating reference frame abc is converted to a stationary reference frame $dq0$ by applying the power-invariant Park Transform to (2.32a).

$$K_p V_{abc} = K_p R I_{abc} + K_p \dot{\lambda}_{abc} \tag{2.33a}$$

$$V_{dq0} = R I_{dq0} + K_p \frac{d}{dt}\left(K_p^{-1} \lambda_{dq0}\right) \tag{2.33b}$$

$$V_{dq0} = R I_{dq0} + K_p K_p^{-1} \dot{\lambda}_{dq0} + K_p \frac{d}{dt}\left(K_p^{-1}\right) \lambda_{dq0} \tag{2.33c}$$

$$V_{dq0} = R I_{dq0} + \dot{\lambda}_{dq0} + \omega_m \begin{bmatrix} -\lambda_q \\ \lambda_d \\ 0 \end{bmatrix} \tag{2.33d}$$

where ω_e is the electrical frequency. The step to (2.33b) is required to convert the flux linkage to $dq0$ variables. Equation (2.33c) applies product rule since both K_p and λ_{dq0} are functions of time. The result is simplified in (2.33d).

Next, applying the power-invariant Park Transform to (2.32b)-(2.32d) yields

$$K_p L_s K_p^{-1} = L_{dq0} = \begin{bmatrix} L_d & 0 & 0 \\ 0 & L_q & 0 \\ 0 & 0 & L_0 \end{bmatrix} \tag{2.34a}$$

$$K_p \Lambda_{abc} = \Lambda_{dq0} = \Lambda \begin{bmatrix} \frac{\sqrt{6}}{2} \\ 0 \\ 0 \end{bmatrix} \tag{2.34b}$$

$$K_p \lambda_{abc} = \lambda_{dq0} = \begin{bmatrix} L_d I_d + \Lambda \frac{\sqrt{6}}{2} \\ L_q I_q \\ L_0 I_0 \end{bmatrix} \tag{2.34c}$$

where L_d and L_q are the d and q variable inductances. Combining (2.33d) and (2.34) yields

$$V_d = RI_d - \frac{p}{2}L\omega_m I_q + L\dot{I}_d, \tag{2.35a}$$

$$V_q = RI_q + \frac{p}{2}L\omega_m I_d + L\dot{I}_q + \frac{p}{2}\frac{\sqrt{6}}{2}\Lambda\omega_m, \tag{2.35b}$$

$$V_0 = RI_0 + L_0\dot{I}_0, \tag{2.35c}$$

$$\tau_m = \frac{p}{2}\frac{\sqrt{6}}{2}\Lambda I_q, \tag{2.35d}$$

where $\omega_m = \frac{2}{p}\omega_e$ is rotor angular velocity, p is the number of pole pairs, and τ_m is the electro-magnetic torque. The electro-magnetic torque is derived using the electrical dynamics and (2.31). Note the simplification that $L = L_d = L_q$ for a non-salient round rotor motor design [40]. For a balanced 3-phase system, the 0-axis dynamics in the analysis of the BLDC motor can be neglected. In the current formulation, the dq0 motor model cannot be formulated as a graph because the $\frac{p}{2}L\omega_m I_d I_q$ terms are a function of 3 states. However, it is a common control objective to drive the d-axis current to zero during operation. Assuming the control objective is met, the BLDC motor graph model is identical to the DC motor graph model (defined by (2.25), (2.27), and Figure (2.6)) with the following definition changes: $L_1 = L$, $V_1 = V_q$, $I_1 = I_q$, $\omega_1 = \omega_m$, and $k_v = \frac{p}{2}\frac{\sqrt{6}}{2}\Lambda$ (e.g. L_1 for the DC model is equivalent to L in the BLDC model).

If the reader is interested in more information, they are directed to directed to Appendix C.2 in [38] and Chapter 3 (sections 1-4) and Chapter 4 (sections 1-5) of [40].

2.3.3 Power Electronics

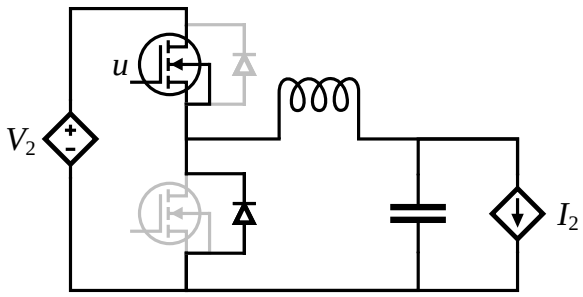
Power electronics are devices used to control electrical power flows within an electrified network. These components utilize high frequency switching semi-conducting devices to permit or block the flow of electrical power. This thesis will consider converters and inverters that handle DC-DC and DC-AC power conversion respectively. Converters are typically classified into buck and/or boost categories. Buck converters will output power at a lower voltage level than was input, whereas boost converters output power at a higher voltage level.

There are various inverter designs and control algorithms that impact the device efficiency [42, 43]. Typically, an inverter uses a switching algorithm to approximate an AC waveform as a series of square wave pulses of varying width [44]. In this section, graph models for both a bi-directional buck-boost converter and inverter are introduced. Because 3-phase circuits are analyzed, we will also consider the conversion between Y and Δ connections.

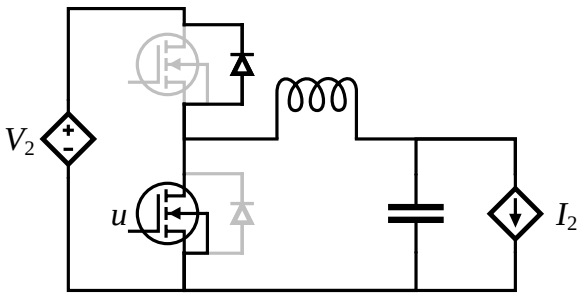
2.3.3.1 Buck-Boost Converter

A bi-directional buck-boost converter electrical circuit with loss is shown in Figure 2.8. The converter will be treated as a single lumped thermal mass (Figure 2.8c). Figure 2.9 shows the 4 circuit model variations depending on whether the semi-conductors are conducting or blocking and if the circuit is in buck or boost mode. For each variation, the energy storage elements are analyzed

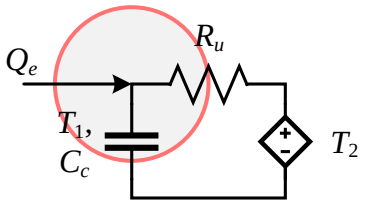
$$C_1 V_1 \dot{V}_1 = V_1 I_1 - V_1 I_2, \tag{2.36a}$$



(a) The conducting elements of the bi-directional buck-boost converter in buck mode.

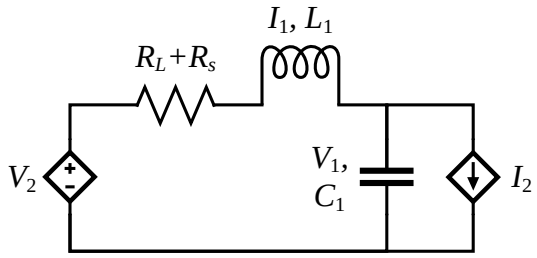


(b) The conducting elements of the bi-directional buck-boost converter in boost mode.

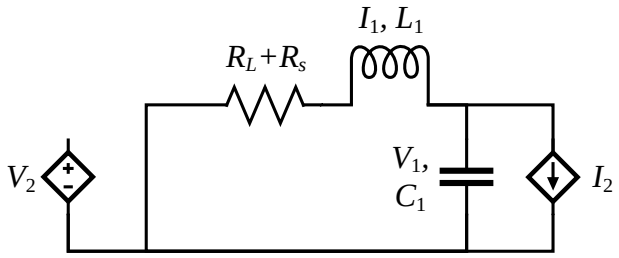


(c) Thermal circuit used to model the buck-boost converter's thermal dynamics. Note that $Q_e = R_L I_1^2 + u R_s I_1^2 + (1-u) R_D I_1^2$.

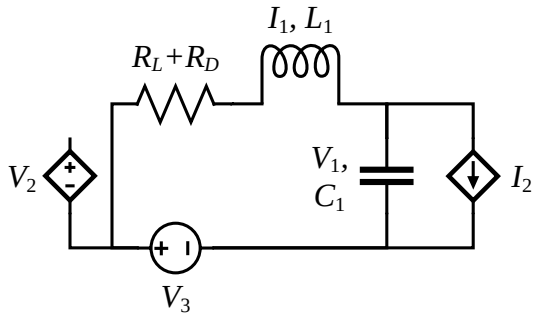
Figure 2.8: A combined bi-directional buck-boost converter circuit topology.



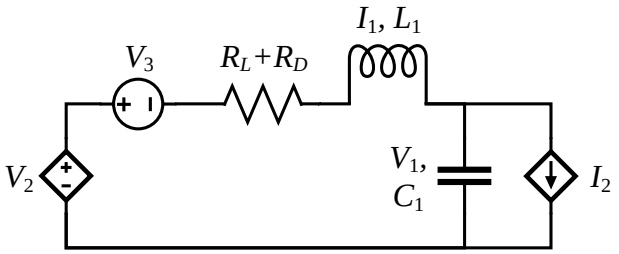
(a) The conducting elements of the bi-directional buck-boost converter in buck mode when the duty cycle $u = 1$.



(b) The conducting elements of the bi-directional buck-boost converter in boost mode when the duty cycle $u = 1$.



(c) The conducting elements of the bi-directional buck-boost converter in buck mode when the duty cycle $u = 0$.



(d) The conducting elements of the bi-directional buck-boost converter in boost mode when the duty cycle $u = 0$.

Figure 2.9: The bi-directional buck-boost converter partitioned depending on the 4 operating modes: (a) buck conducting, (b) boost conducting, (c) buck blocking, and (d) boost blocking.

Buck $(s = 0)$:

$$\begin{aligned} \textit{Conducting:} \quad & L_1 I_1 \dot{I}_1 = V_2 I_1 - R_s I_1^2 - R_L I_1^2 - V_1 I_1, \\ \textit{Blocking:} \quad & L_1 I_1 \dot{I}_1 = -R_D I_1^2 - R_L I_1^2 - V_1 I_1 - V_3 I_1, \end{aligned} \tag{2.36b}$$

Boost $(s = 1)$:

$$\begin{aligned} \textit{Conducting:} \quad & L_1 I_1 \dot{I}_1 = -R_s I_1^2 - R_L I_1^2 - V_1 I_1, \\ \textit{Blocking:} \quad & L_1 I_1 \dot{I}_1 = V_2 I_1 - R_D I_1^2 - R_L I_1^2 - V_1 I_1 + V_3 I_1, \end{aligned} \tag{2.36c}$$

$$C_c \dot{T}_1 = R_L I_1^2 + u R_s I_1^2 + (1-u) R_D I_1^2 - \frac{1}{R_u}(T_1 - T_2), \tag{2.36d}$$

where V_1, V_2, and V_3 are the capacitor, applied, and diode forward voltages respectively, I_1 and I_2 are inductor and demanded current respectively, C_1 is the capacitor's electrical

capacitance, L_1 is the inductor's inductance, R_s is the switch resistance, R_D is the diode resistance, R_L is the coil resistance, T_1 is converter temperature, T_2 is a surrounding fluid temperature, C_c is the converter heat capacity, R_u is the thermal convection resistance, $s = \{0, 1\}$ denotes buck or boost mode, and $u \in [0, 1]$ is the switch duty cycle. The system conducts for time uT_s and blocks for time $(1-u)T_s$ where T_s is the switching period. Therefore, (2.36) can be combined into

$$\begin{aligned} \textit{Buck}: \quad L_1 I_1 \dot{I}_1 &= uV_2 I_1 - \left(uR_s + R_L + (1-u)\, R_D\right) I_1^2 \\ &\quad - (1-u)V_3 I_1 - V_1 I_1, \end{aligned} \tag{2.37a}$$

$$\begin{aligned} \textit{Boost}: \quad L_1 I_1 \dot{I}_1 &= (1-u)V_2 I_1 - \left(uR_s + R_L + (1-u)\, R_D\right) I_1^2 \\ &\quad + (1-u)V_3 I_1 - V_1 I_1. \end{aligned} \tag{2.37b}$$

Assuming the losses are independent of buck or boost mode operation, (2.37) can be combined further since $s = \{0, 1\}$.

$$\begin{aligned} L_1 I_1 \dot{I}_1 &= (s + u - 2su)V_2 I_1 - \left(uR_s + R_L + (1-u)\, R_D\right) I_1^2 \\ &\quad + (-1 + 2s + u - 2su)V_3 I_L - V_1 I_1. \end{aligned} \tag{2.38}$$

The sign of the current flow will determine whether the system is operated in buck or boost mode. Therefore, s can be described as a function of I_1:

$$s = g(I_1) = \begin{cases} 0 & \text{for } I_1 \geq 0, \\ 1 & \text{for } I_1 < 0. \end{cases} \tag{2.39}$$

The power flows, as mentioned in previous sections, are representative of electrical power transfer VI, resistive loss RI^2, and thermal convection $\Delta T/R$. The bi-directional buck-boost converter graph is provided in Figure 2.10. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} I_1 & V_1 & T_1 \end{bmatrix}^T, \tag{2.40a}$$

$$C = \begin{bmatrix} L_1 I_1 & C_1 V_1 & C_c \end{bmatrix}^T, \tag{2.40b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_L + R_D & 0 & 0 & 0 & 0 & R_s - R_D & 0 & 0 \\ \frac{1}{R_u} & -\frac{1}{R_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.40c}$$

$$f = \begin{array}{c} \begin{array}{cccccc} e_1 & & e_2 & e_3 & e_4 & & e_5 & e_6 \end{array} \\ \begin{bmatrix} (g(I_1) + u - 2g(I_1)u) & 1 & 1 & (-1 + 2g(I_1) + u - 2g(I_1)u) & 1 & 1 \end{bmatrix} \end{array}. \tag{2.40d}$$

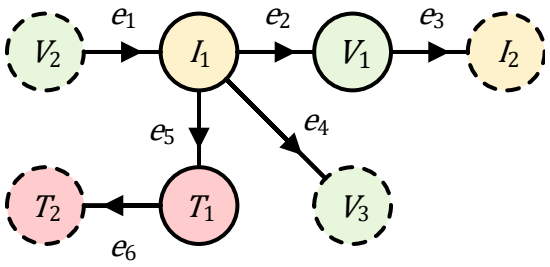


Figure 2.10: Graph model for the bi-directional buck-boost converter.

The function $g(I_1)$ is discontinuous at $I_1 = 0$. This may result in numerically instability. As mentioned in Section 2.3.2, the discontinuity can be smoothed by a sigmoid function.

It is important to highlight the ease of variable fidelity modeling of the converter. If one was interested in the analysis of the oscillatory dynamics of the converter, the electrical capacitance and inductance can be be given a physical non-zero value. If only steady-state information is required for an analysis, the electrical capacitance and inductance values can be defined as 0 (this results in a DAE model). This simplification may be useful depending on the target application. For example, modeling the oscillatory behavior may be necessary for a model-based controller with a fast update rate that seeks to regulate output voltage while mitigating oscillations. To the contrary, a steady-state model maybe be sufficient in

a model-based controller seeking to optimize the behavior of the converter over the course of seconds or minutes. The variable fidelity characteristic of the graph-based modeling framework will be useful in the design of advanced control systems.

2.3.3.2 Inverter

The circuit topology for an electrical inverter is shown in Figure 2.11. An inverter's switching algorithm is used to approximate a sine wave that has a q-axis voltage amplitude V_q bounded by the inverter DC link voltage V_{dc} (ie. $V_q < \sqrt{\frac{3}{2}}V_{dc}$) [40]. This behavior is similar to the buck converter operation where the converter's output voltage is less than the applied voltage. By this observation, the inverter will be modeled with a similar structure to the buck converter. The inverter q-axis voltage and current states V_1 and I_1 and temperature state T_1 is given by

$$C_1 V_1 \dot{V}_1 = V_1 I_1 - V_1 I_2, \tag{2.41a}$$

$$L_1 I_1 \dot{I}_1 = u\sqrt{\frac{3}{2}} V_2 I_1 - V_1 I_1 - u R_i I_1^2, \tag{2.41b}$$

$$C_i \dot{T}_1 = u R_i I_1^2 - \frac{1}{R_u}(T_1 - T_2), \tag{2.41c}$$

where I_2 is the q-axis current demand, V_2 is the applied DC voltage, C_1 and L_1 are virtual capacitances and inductances and are identically zero (see Section 2.3.8), u is the modulation

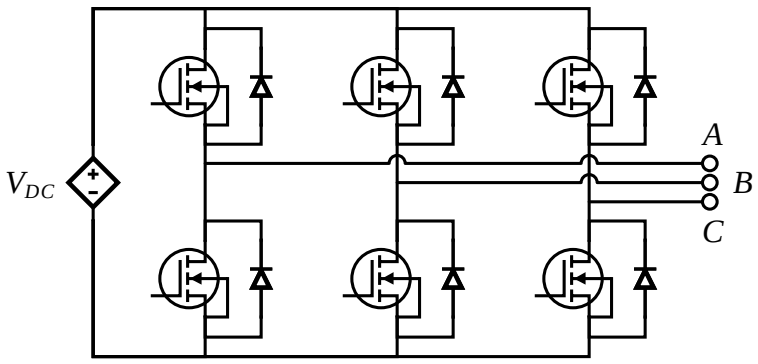


Figure 2.11: A three-phase bridge converter circuit used to model the inverter dynamics.

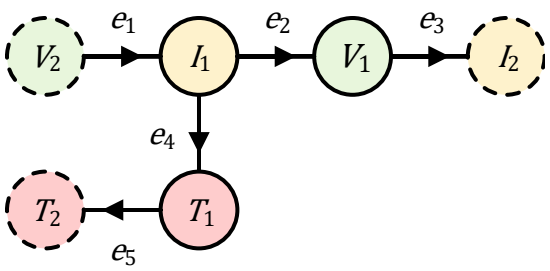


Figure 2.12: Graph model for the electrical inverter.

signal for the inverter switches, R_i is the bulk inverter loss, C_i is the thermal capacitance of the inverter, T_1 is the temperature of the inverter, T_2 is a cooling fluid temperature, and R_u is the convection resistance. The thermal dynamics are modeled after the structure defined by Figure 2.8c where $Q_e = uR_iI_2^2$. Similar to the BLDC motor model, this model is derived using the power-invariant Park Transform and assumes that the d-axis currents are zero.

The inverter graph is provided in Figure 2.12. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} I_1 & V_1 & T_1 \end{bmatrix}^T, \tag{2.42a}$$

$$C = \begin{bmatrix} L_1 I_1 & C_1 V_1 & C_c \end{bmatrix}^T, \tag{2.42b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{\frac{3}{2}} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_i & 0 & 0 \\ \frac{1}{R_u} & -\frac{1}{R_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.42c}$$

$$f = \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{2.42d}$$

2.3.3.3 Y and Δ Connections

Three-phase systems can be connected in Y or Δ configurations as seen by Figure 2.7. The operation and analysis of these systems are similar. In electric motor applications, a wye wound motor typically produces more torque and operates at a lower speed than its delta wound counterpart. This is known since the applied line voltages span two phases in a wye machine, whereas the same voltage spans one phase in a delta machine. Since the induced motor voltage is proportional to speed ($k_v\omega$), the delta wound machine can reach greater speeds. Similar reasoning can be applied to line currents and generated torque.

Although the analysis of these systems remains the same regardless of winding configuration, there is some necessary scaling to convert between wye and delta connections. As seen in [38], (2.43) can be used to help convert between balanced three-phase wye and delta connections.

$$V_{ab} = \sqrt{3}V_a\angle 30^\circ, \tag{2.43a}$$

$$I_a = \sqrt{3}I_{ab}\angle -30^\circ, \tag{2.43b}$$

where V_{ab} is the line to line voltage, V_a is the line to neutral voltage of a wye connection, I_{ab} is a phase current in a delta connection, and I_a is a line current. Note that (2.43a) applies to wye connections and (2.43b) applies to delta connections. Furthermore, the analysis in this thesis ignores phases angles because the AC system dynamics were converted to a stationary reference frame. Therefore, the conversions described in (2.43) apply a scalar gain of $\sqrt{3}$.

For the experimental system that will be described in Chapter 4, the inverter is wye connected and the electric machine is delta wound. The following equations are used to convert between the wye and delta connected configuration

$$L_Y I_Y \dot{I}_Y = V_Y I_Y - \sqrt{\frac{1}{3}}V_\Delta I_Y, \tag{2.44a}$$

$$C_\Delta V_\Delta \dot{V}_\Delta = \sqrt{\frac{1}{3}}V_\Delta I_Y - V_\Delta I_\Delta, \tag{2.44b}$$

where $C_\Delta = L_Y =$ are virtual capacitances and inductances (Section 2.3.8), V and I are

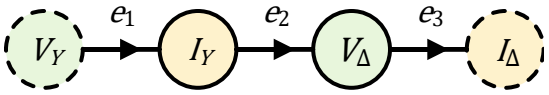


Figure 2.13: Graph model for the wye to delta connection conversion.

voltages and currents respectively, and the subscripts Δ and Y denote delta and wye states respectively. The dynamics of (2.44) can be formulated as a graph (Figure 2.13). The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} I_Y & V_\Delta \end{bmatrix}^T, \tag{2.45a}$$

$$C = \begin{bmatrix} L_Y I_Y & C_\Delta V_\Delta \end{bmatrix}^T = 0, \tag{2.45b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{1}{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.45c}$$

$$f = \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{2.45d}$$

2.3.4 Genset

A genset is a system composed of an engine, generator, and control electronics. In a hybrid system, an engine is used to convert the chemical potential energy of fuel to rotating mechanical energy of a spinning shaft. When connected to a generator, the mechanical energy of the shaft is converted to electrical energy that can be used to power vehicle loads or recharge the battery pack. The amount of power produced by the engine can be varied by adjusting the throttle or air-fuel ratio to change both engine speed and torque production. These actuators are typically controlled by a governor designed by the engine manufacturer. The generator can passively or actively rectify the AC current waveform induced by the rotating

shaft. An active rectifier receives a control signal that actuates switching semiconductor devices whereas a passive rectifier has no control input (typically achieved via mechanical brushes). Active rectifiers can be used as an extra degree of freedom to optimize the power production of the subsystem.

The goal of this section is to construct a model that can adequately capture the dynamics of the genset subsystem on the testbed described in Chapter 4. The engine, as currently configured, operates at a single speed setpoint that is tracked by the engine governor. The generator current is actively rectified to track a current setpoint passed to the generator controller. Although a motor model was presented in Section 2.3.2, it would prove difficult to capture the engine and rectifier dynamics without understanding the underlying control architecture and system inputs. Therefore, a model for the entire subsystem is abstracted based on empirical data. Consequently, this system will not be modeled in the graph-based modeling framework.

The genset is modeled as first-order linear dynamic system

$$\tau \dot{I} = -I + Ku, \tag{2.46}$$

where I is the generated DC current, K is an input gain, $u \in [0, 1]$, and τ is the time constant of the subsystem.

An objective of the control systems developed in Chapter 3 is to minimize fuel consumption. Fuel consumption is commonly evaluated using the engine specific fuel consumption (SFC) which is commonly characterized as complex functions of engine speed ω and torque τ [45]. For control, a few simplifications are made. Assuming the engine operates at a constant speed, the SFC can be re-characterized as a function of the engine power P (since $P = \omega\tau$ and ω is constant). Based on the analysis in Sections 2.3.2 and 2.3.3.2, the engine power can be directly related to the inverter DC power assuming negligible generator and inverter losses. Although this may be a strong assumption, the bulk inefficiency in the genset subsystem is a result of the fuel combustion. Therefore, the engine SFC sfc is described as

a function of the genset DC voltage V and current

$$sfc = \begin{cases} a_1 I^2 + a_2 V^2 + a_3 + a_4 V I + a_5 I + a_6 V & \text{if } I > 0, \\ 0 & \text{if } I = 0, \end{cases} \tag{2.47}$$

where the coefficients a_i are chosen such that the resulting surface is convex (see Figure 4.31). A convex surface is required for integration into a quadratic cost function introduced in Chapter 3. Validation of the current dynamic will be presented in Chapter 4. Full characterization and validation of the engine torque production and fuel consumption is outside the scope of this thesis.

2.3.5 Electrical Bus

Parallel connections between components and subsystems that operate at similar power levels are made by electrical busbars. In practice, busbars receive and distribute electrical power in an electrified network. In addition to facilitating parallel connections, the electrical bus model presented in this work is also a DC-link with capacitance to mitigate current ripple in the electrical network. This DC-link capacitance may be representative of capacitors located elsewhere in the system.

The circuit diagram for the electrical bus model (Figure 2.14) has a couple of key features. First, there are a variable number of components that can be connected to the bus. Second, switches in the circuit permit components and subsystems to be disconnected from the rest of the system. Furthermore, the inductors on the left-hand side of the schematic are necessary such that a voltage source component can be connected directly to the bus. These inductive elements are virtual (see Section 2.3.8) and have zero or near-zero inductance. There is a bleed resistor in parallel with each inductor in order to dissipate stored energy when the associated subsystem is disconnected. Also, there are multiple grounds in Figure 2.14. This was done to make the figure compact. In reality, each branch of the bus can share the same ground. To develop the graph model, each energy storage element is analyzed

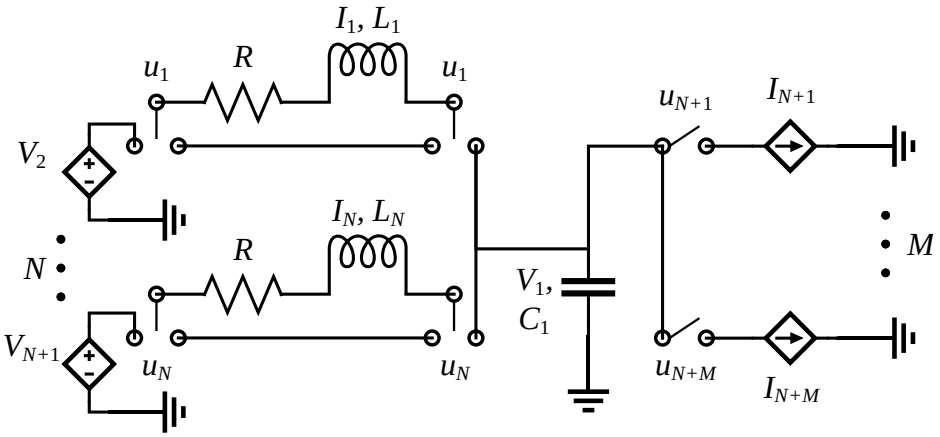


Figure 2.14: Electrical circuit schematic used to derive the electrical bus model dynamics.

$$L_n I_n \dot{I}_n = u_n V_{n+1} I_n - R I_n^2 - u_n V_1 I_n \quad \forall n \in [1:N], \tag{2.48a}$$

$$C_1 V_1 \dot{V}_1 = \sum_{n=1}^{N} u_n V_1 I_n - \sum_{m=1}^{M} u_{N+m} V_1 I_{N+m}. \tag{2.48b}$$

Here L_n is an inductance value, I_n is the current in inductor n, $u \in \{0, 1\}$ is a switch input, V_n is the voltage applied to inductor n, R is the bleed resistance, C_1 is the capacitance of the DC-link, and V_1 is the bus voltage. The total number voltage sources and current sinks on the bus is denoted by N and M respectively. The thermal state of the bus is not modeled because the bleed resistors are assumed to be small. The actual value should be chosen such that the power flow along edges e_{3n} is less than $0.1 - 1\%$ of the nominal power flow along edges e_{3n-2} and e_{3n-2} (for $n \in [1:N]$). If modeling the thermal dynamics is of interest, it is readily observed that the heat produced is RI_n^2.

The relevant modes of power transfer are electrical power transfer VI and resistive loss RI^2. The electrical bus graph is provided in Figure 2.15. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} V_1 & I_1 & \cdots & I_N \end{bmatrix}^T, \tag{2.49a}$$

$$C = \begin{bmatrix} C_1 V_1 & L_1 I_1 & \cdots & L_N I_N \end{bmatrix}^T, \tag{2.49b}$$

$$
c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{3N-2} \\ e_{3N-1} \\ e_{3N} \\ e_{3N+1} \\ \vdots \\ e_{3N+M} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & R & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & R & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\end{array}, \tag{2.49c}
$$

$$
f_i = 1 \quad \forall i \in [1 : 3N + M]. \tag{2.49d}
$$

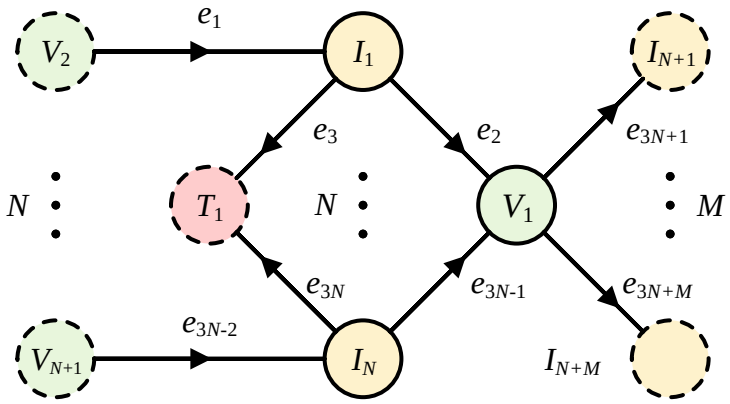


Figure 2.15: Graph model for the electrical bus.

2.3.6 Vehicle Body Dynamics

It is necessary to understand the dynamics of an airframe to maintain stable flight. From a power-based analysis, it is required to know how much power is necessary to generate lift and overcome drag, gravity, and other flight disturbances. Airframes have been analyzed using

a variety of methods. There has been some work that analyzes the 3-dimensional forces and moments on an aircraft using Newton's second law [46, 47]. Simpler approaches utilize force balances on a point mass [48]. For this work, a 2D point mass analysis will suffice since the purpose of this model is to emulate a load on the propulsion motor.

A free body diagram for a propeller driven aircraft (Figure 2.16) is used to derive the velocity dynamics.

$$m_1 v_1 \dot{v}_1 = \rho \frac{D^4}{4\pi^2} C_T \omega_1^2 v_1 - \frac{1}{2} \rho A C_D v_1^3 - m_1 g v_1 \sin\theta \tag{2.50}$$

where m_1 is the vehicle mass, v_1 is the vehicle velocity along its trajectory θ, ρ is the air density, D is the prop diameter, C_T is the thrust coefficient, ω_1 is the angular velocity of the propeller, A is the effective frontal area of the aircraft, C_D is the drag coefficient, and g is gravity. In (2.50), the first term is the thrust power and the second term is power loss to drag. Inefficiency in the propeller yields extra loss in the system since not all angular mechanical power is converted to linear mechanical power. The power loss P_{loss} is given by

$$P_{loss} = (1 - \eta) \rho \frac{D^5}{4\pi^2} C_\tau \omega_1^3 \tag{2.51}$$

where η is the propeller efficiency and C_τ is the torque coefficient. In a typical analysis of a propeller, the propeller efficiency and thrust and torque coefficients are a function of the advance ratio $J = 2\pi v_1/(\omega_1 D)$. For the sake of simplicity, these parameters are held as constants and act as tuning variables of the model. The vehicle body graph is illustrated

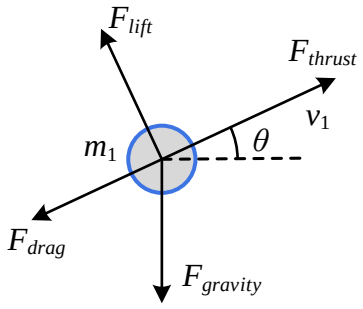


Figure 2.16: Free body diagram used to derive the vehicle velocity dynamics.

in Figure 2.17. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

$$x = \begin{bmatrix} v_1 \end{bmatrix}^T, \tag{2.52a}$$

$$C = \begin{bmatrix} m_1 v_1 \end{bmatrix}^T, \tag{2.52b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & \rho\frac{D^4}{4\pi^2}C_T & 0 & 0 & 0 & 0 & 0 & 0 \\ m_1 g \sin\theta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\rho A C_D \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (1-\eta)\rho\frac{D^5}{4\pi^2}C_\tau \end{bmatrix} \end{array}, \tag{2.52c}$$

$$f = \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{2.52d}$$

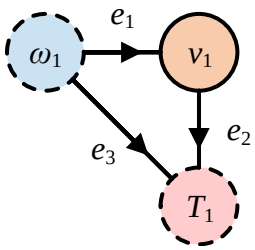


Figure 2.17: Graph model for the vehicle body.

2.3.7 Processor Load

As mentioned earlier, the system model described in Section 2.5 will be experimentally validated in Chapter 4. That experimental platform has multiple electronic speed controllers (ESCs) that actuate the switching of the power electronic devices. The processing chip inside the ESCs require power. For simplicity, the processing load P is treated as a static function of the ESC bus voltage V_1

$$P = a_1 V_1^2 + a_2 V_1, \tag{2.53}$$

where the coefficients a_1 and a_2 are constants. The processor load graph is illustrated in Figure 2.18. The graph state vector, capacitance vector, power flow coefficients, and property look-up coefficients are provided below.

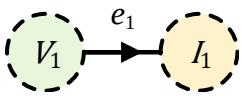


Figure 2.18: Graph model for the processor load.

$$x = \begin{bmatrix} - \end{bmatrix}^T, \tag{2.54a}$$

$$C = \begin{bmatrix} - \end{bmatrix}^T, \tag{2.54b}$$

$$c = \begin{array}{c} \\ e_1 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} a_2 & 0 & 0 & a_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{2.54c}$$

$$f = \begin{array}{c} e_1 \\ \begin{bmatrix} 1 \end{bmatrix} \end{array}. \tag{2.54d}$$

2.3.8 Virtual Elements

Virtual elements do not have a physical representation within a system and are used to facilitate the construction, simulation, and analysis of a multi-domain graph model. They typically have zero or near-zero capacitance values. If near-zero, the capacitance value should be treated as a tuning parameter chosen sufficiently small such that the other system dynamics are not affected. As mentioned in Section 2.2.2, there must be alternating inertance and compliance elements [32]. If there were a desire to connect two compliance or inertance elements, a virtual element added in between would facilitate that connection. For example, virtual inductor elements were used to develop the electrical bus model in Section 2.3.5. If it were desired to connect the battery directly to the DC-link capacitance on the bus, there would exist a connection between two voltage type compliance elements, which violates

the rules for constructing a multi-domain graph. Instead, a virtual current type inertance element is used to enable that connection.

2.3.9 Comments on Thermal System Dynamics

In the previous sections, the thermal dynamics of the components were simple and limited to first and second-order lumped mass models. However, the thermal dynamics can be modeled by other more complex configurations. For example, a significantly more complex thermal circuit model with 9 temperature states for a power inverter can be found in [36]. The ability to adapt the thermal model highlights the flexibility of the graph-based modeling framework.

While this thesis mostly considers the electrical and mechanical system dynamics, it will be important to couple the electro-mechanical system to a thermal system. The temperature sink states of the components outlined in this chapter provide these *hooks* into more complex thermal systems. The sink states can connect to latent heat storage modules that contain phase change material [49], novel cooling and heat spreading devices [50], or directly to a fluid-thermal system [25]. The modularity of the graph-based modeling framework is particularly useful for facilitating theses interconnections of systems across energy domains.

2.4 System Composition Methods

Similar to how physical systems can be represented as an interconnection between subsystems and components, a system graph model can be represented as an interconnection between subsystem and component levels graphs. This scalability feature of the graph-based modeling framework motivates its application to modeling energy or other conservation-based systems. The graph framework has recently been used in design optimization problems to size components in a pure electric automobile [51], and evaluate cooling topologies for a fuel thermal management system [52]. In past applications, the system level graph models were manually constructed using heuristics based on prior experience working with graphs. Although effective, the manual process is error prone and time intensive. A manual pro-

cess is not convenient for design optimization problems where it may be required to rapidly generate and evaluate numerous system architectures.

This section elaborates upon the work in [3] to present methods to develop system graph models as an interconnection of component graph models. First, background for system development methods are introduced. Second, a novel algorithm is presented that facilities the automatic generation of a system model.

2.4.1 Background for System Development

A set of of component graphs is described by $\mathcal{C} = \{\mathcal{G}_i : i \in N_g\}$ where N_g is the total number of component graphs and $\mathcal{G}$ is as described in Section 2.2. In this section, $v_{i,j} \in \mathcal{V}_j : i \in [1 : N_{v,j}], j \in [1 : N_g]$ is vertex i of component j and $e_{i,j} \in \mathcal{E}_j : i \in [1 : N_{e,j}], j \in [1 : N_g]$ is edge i of component j. Graph interconnections can be described by two interconnection types. The first interconnection type is defined by a vertex equivalency $v_{i,m} = v_{j,n}$ and will be referred to as a *Type 1* interconnection. The second interconnection type is defined by an edge equivalency $e_{i,m} = e_{j,n}$ and will be referred to as a *Type 2* interconnection (this vocabulary is defined in and specific to this thesis). Figure 2.19 provides a visual representation of the interconnections types. Because an edge connects two vertices, a Type 2 interconnection consists of 1 edge equivalency and 2 vertex equivalencies. Second, when the graphs represent a dynamical system, it is necessary that each vertex equivalency includes at most one state vertex. Otherwise, that interconnection would violate causality rules.

When interconnecting graph-based models, a Type 1 interconnection is utilized when multiple components have the same interaction with a state or sink state. For example, consider connecting two batteries in series as described in Figure 2.20. In electrical circuits, components in series share the same current. This property is reflected in Figure 2.20 where both batteries share the same current sink state $v_{6,1} = v_{6,2} = v_{13,s}$.

A Type 2 interconnection is utilized when all the power along an edge leaving a state vertex of one component is incident to an edge of a state vertex of a second component. For example, consider connecting a converter in series with a motor. In this case, all the electrical power leaving the converter along $e_{3,1}$ should be equivalent to the electrical power

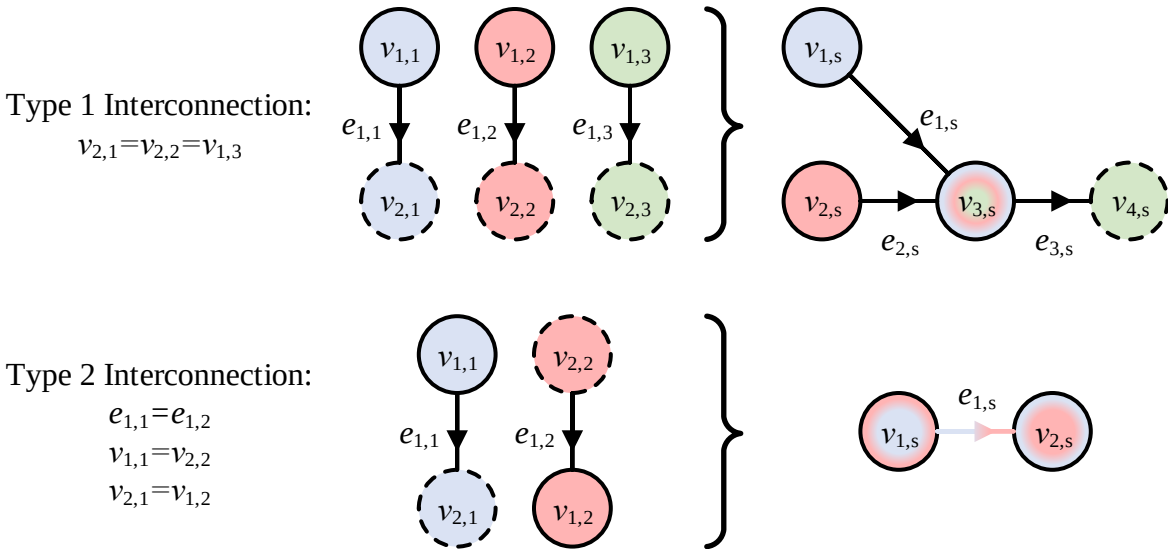


Figure 2.19: Visualization of Type 1 and Type 2 interconnection types. Note the coloration used to denote which vertex and edge equivalencies. The subscript s denotes the system graph.

entering the motor along $e_{1,2}$ in Figure 2.21. In addition to the equivalency between edges, note the equivalency between vertices. The current demanded by the converter ($v_{5,1}$) is a state of the motor ($v_{1,2}$), and the terminal voltage of the motor ($v_{4,2}$) is a state of the converter ($v_{2,1}$).

The two interconnections types are the main heuristics used when developing a system graph model. Large systems will consist of multiple Type 1 and Type 2 interconnections.

When constructing a system model by hand it is important to record which vertices and edges of the component graphs correspond to vertices and edges of the system graph because, as seen in Section 2.3, there are various specific properties associated with each edge and vertex. If these properties are not mapped properly, the resulting system dynamics will not match the desired behavior. The algorithm presented in the next section seeks to resolve this issue of mapping component graph to system graph properties.

2.4.2 Graph-Model Interconnection Algorithm

Various sets will be useful in the understanding the following algorithm. First, the set of all component vertices is defined as $\chi \coloneqq \{\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_i : i \in [1 : N_g]\}$. Similarly, the set of

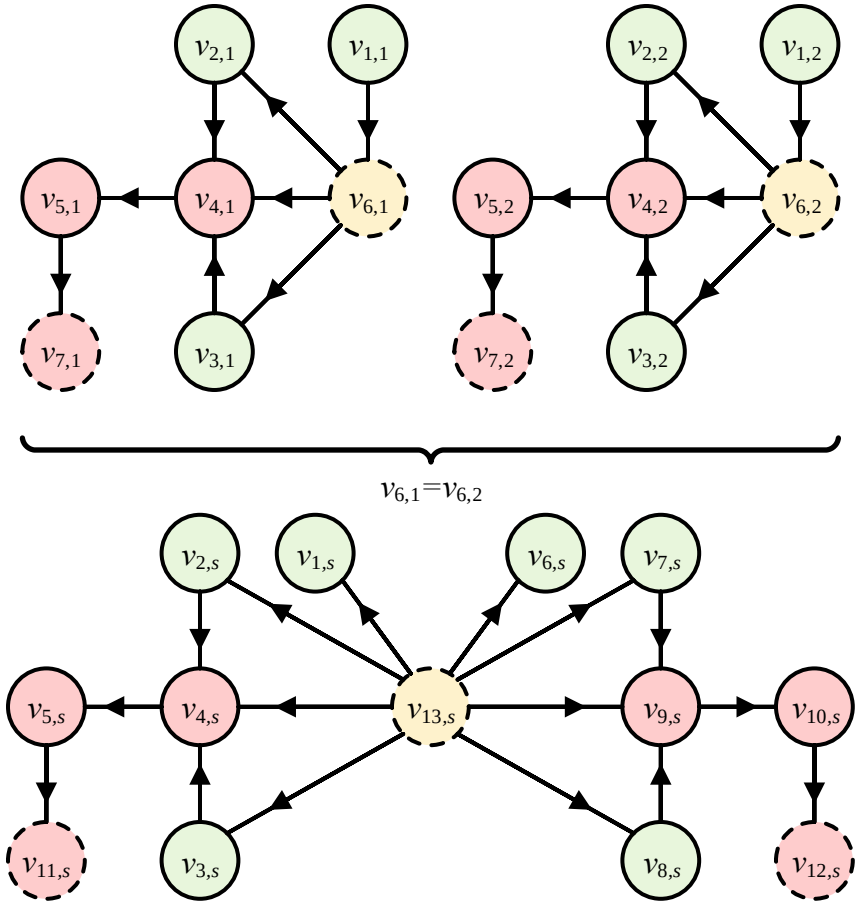


Figure 2.20: Example of a Type 1 interconnection where two batteries (represented as graphs) are connected in series.The subscript s denotes the system graph.

all component edges is defined as $\Xi \coloneqq \{\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_i : i \in [1 : N_g]\}$. Recall from Section 2.2 that χ includes sink vertices and Ξ does not include source edges. Therefore, χ can be partitioned such that $\bar{\chi} \coloneqq \{\bar{\mathcal{V}}_1 \cup \cdots \cup \bar{\mathcal{V}}_i : i \in [1 : N_g]\}$ is the set of all component state vertices and $\underline{\chi} = \chi \setminus \bar{\chi}$ is the set of all component sink vertices.

Next, the set of all Type 1 interconnections Λ and Type 2 interconnections Σ are defined as

$$\Lambda \coloneqq \{\{v_{i,m}, \cdots, v_{j,n}\} : v_{i,m} =, \cdots, = v_{j,n}, \; i \in [1 : N_{v,m}],\ j \in [1 : N_{v,n}],\ m \in [1 : N_g],\ n \in [1 : N_g]\} \tag{2.55a}$$

$$\Sigma \coloneqq \{\{e_{i,m}, e_{j,n}\} : e_{i,m}, = e_{j,n}, \; i \in [1 : N_{e,m}],\ j \in [1 : N_{e,n}],\ m \in [1 : N_g],\ n \in [1 : N_g]\} \tag{2.55b}$$

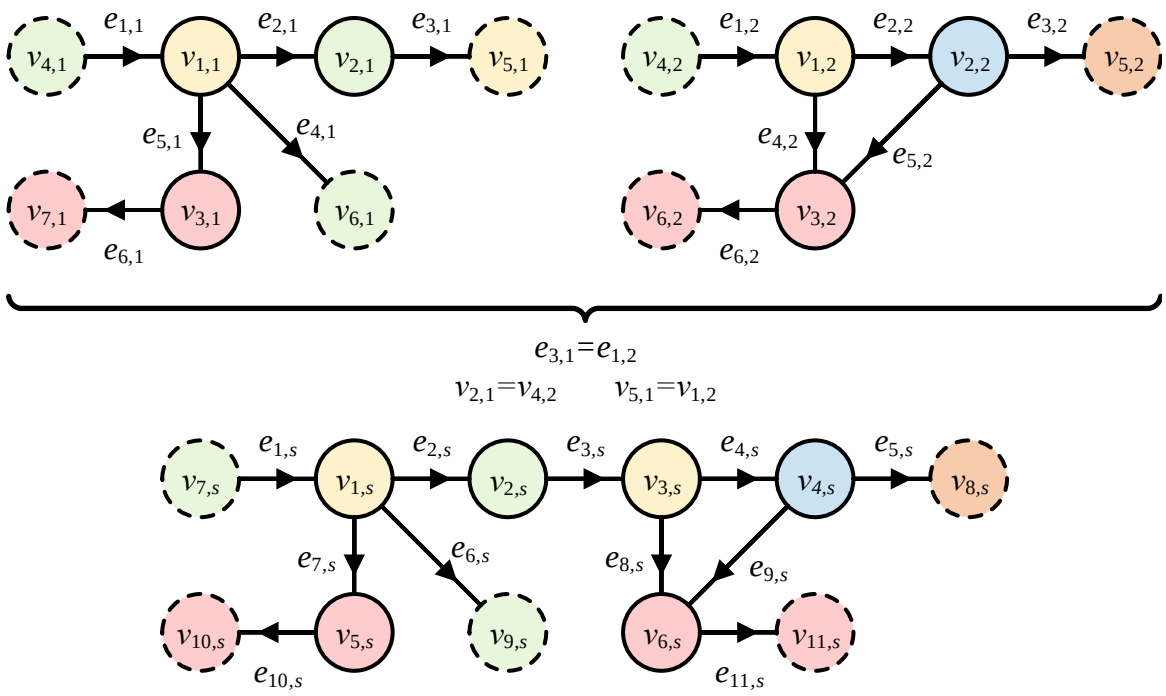


Figure 2.21: Example of a Type 2 interconnection where a converter and motor (represented as graphs) are connected in series. The subscript s denotes the system graph.

These are user-defined sets that are inputs to the algorithm. Recall that at most one vertex in a Type 1 interconnection may be a state vertex. Next, the set Λ is partitioned into $\bar{\Lambda} \coloneqq \left\{\Lambda_i : \Lambda_i \cap \bar{\mathcal{V}} \neq \{\emptyset\}, i \in [1 : N_{T1}]\right\}$ as the set of all Type 1 interconnections that include 1 state vertex, with $\underline{\Lambda} = \Lambda \setminus \bar{\Lambda}$ as the set of all Type 1 interconnections that do not include a state vertex. Here, N_{T1} is the number of Type 1 interconnections. Lastly, define $\hat{\chi} \coloneqq \{\Lambda_i \cap \chi : i \in [1 : N_{T1}]\}$ as the set of all vertices included in an interconnection. Similarly, $\hat{\Xi} \coloneqq \{\Sigma_i \cap \Xi : i \in [1 : N_{T2}]\}$ is the set of all edges included in an interconnection where N_{T2} is the number of Type 2 interconnections.

Property mapping matrices can be developed using the various sets designed in the previous two paragraphs. The vertex property map V maps component vertex properties $\mathcal{P}_i^v : i \in [1 : N_g]$ to system vertex properties $\mathcal{P}_s^v$ as defined by the connection set Λ. Similarly, the edge property map E maps component edge properties $\mathcal{P}_i^e : i \in [1 : N_g]$ to system edge properties $\mathcal{P}_s^e$ as defined by the connection set Σ. Relevant vertex and edge properties are listed in Table 2.2. These property maps are be utilized by

Table 2.2: Graph model vertex and edge properties.

Vertex Properties	Edge Properties	Both
Capacitance	Power Flow	
Initial Condition	Edge Type	Incidence Matrix
Vertex Type	Input	

$$\mathcal{P}_s^v = V^T \begin{bmatrix} \mathcal{P}_1^v \\ \vdots \\ \mathcal{P}_i^v \end{bmatrix}, \tag{2.56a}$$

$$\mathcal{P}_s^e = E \begin{bmatrix} \mathcal{P}_1^e \\ \vdots \\ \mathcal{P}_i^e \end{bmatrix}. \tag{2.56b}$$

The incidence matrix, which relates edges to vertices, stores information about both edges and vertices. Therefore, the system incidence matrix M_s can be determined by

$$M_s = V^T \begin{bmatrix} M_1 & & 0 \\ & \ddots & \\ 0 & & M_i \end{bmatrix} E \tag{2.57}$$

where $M_i = \begin{bmatrix} \bar{M}_i \\ \underline{M}_i \end{bmatrix}$ are component incidence matrices and $i \in [1 : N_g]$. The two property maps V and E are constructed according to the following.

$$V = \left[\; V^{\bar{t}\bar{c}} \;\middle|\; V^{\bar{t}c} \;\middle|\; V^{tc} \;\middle|\; V^{t\bar{c}} \; \right], \quad \text{where} \tag{2.58a}$$

$$V^{\bar{t}\bar{c}} = [v_{ij}] = \begin{cases} 1 & \chi_i = \{\bar{\chi} \setminus \hat{\chi}\}_j, \\ 0 & else, \end{cases} \tag{2.58b}$$

$$V^{\bar{t}c} = [v_{ij}] = \begin{cases} 1 & \chi_i \in \bar{\Lambda}_j, \\ 0 & else, \end{cases} \tag{2.58c}$$

$$V^{tc} = [v_{ij}] = \begin{cases} 1 & \chi_i \in \underline{\Lambda}_j, \\ 0 & else, \end{cases} \tag{2.58d}$$

$$V^{t\bar{c}} = [v_{ij}] = \begin{cases} 1 & \chi_i = \left\{ \underline{\chi} \setminus \hat{\chi} \right\}_j, \\ 0 & else. \end{cases} \tag{2.58e}$$

$$E = \left[\begin{array}{c|c} E^{\bar{c}} & E^{c} \end{array} \right], \quad \text{where} \tag{2.59a}$$

$$E^{\bar{c}} = [e_{ij}] = \begin{cases} 1 & \Xi_i = \left\{ \Xi \setminus \hat{\Xi} \right\}_j, \\ 0 & else, \end{cases} \tag{2.59b}$$

$$E^{c} = [e_{ij}] = \begin{cases} 1/2 & \Xi_i \in \Sigma_j, \\ 0 & else. \end{cases} \tag{2.59c}$$

The application of this algorithm is conducted with a sample system in Appendix A.1.

2.5 Hybrid UAV Graph-Based Model

The hybrid UAV architecture described by Figure 2.2 is modeled as a graph using the component graph models described in Section 2.3 and the system composition methods described in Section 2.4. The system graph-based model is shown in Figure 2.22. Table 2.3 outlines which vertices in the system graph are associated with individual components. The parameters used in the system model are outlined in Chapter 4. In Figure 2.22, individual components are outlined in blue to highlight the modularity of the graph-based modeling framework.

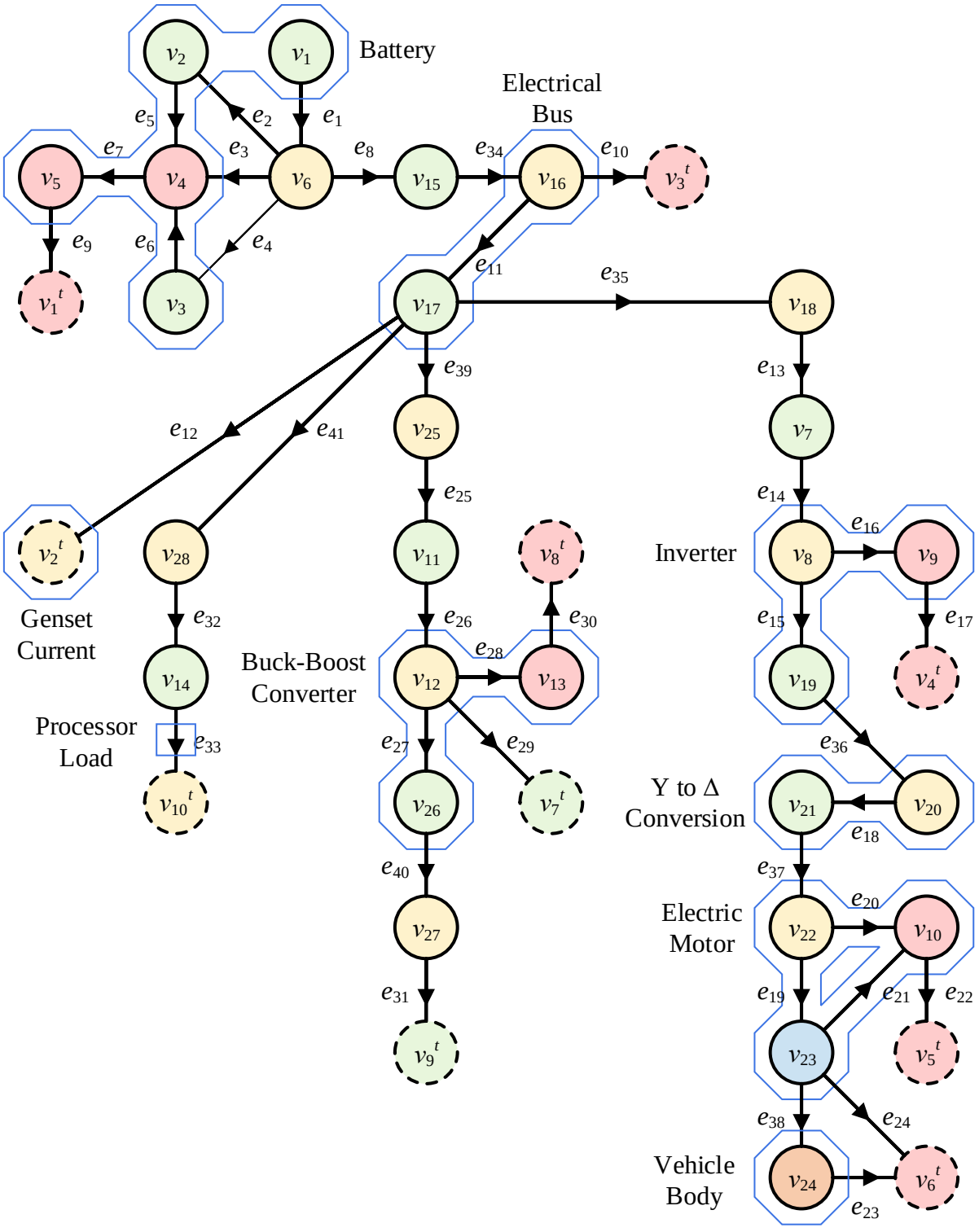


Figure 2.22: Full system graph model used to represent the dynamics of the hybrid UAV architecture described by Figure 2.2. Recall the coloration of each vertex type: thermal mass (red), capacitor (green), inductor (yellow), rotating mass (blue), and translating mass (orange).

Table 2.3: Component graph model state indices in the system level graph model.

Component Model	Associated State Vertices
Battery	1, 2, 3, 4, 5
Electrical Bus	16, 17
Genset	Sink 2
Buck-Boost Converter	12, 13, 26
Inverter	8, 9, 19
Y to Δ Conversion	20, 21
Electric Motor	10, 22, 23
Vehicle Body	24
Virtual Elements	6, 7, 11, 14, 15, 18, 25, 27, 28

2.6 Conclusion

In this chapter the graph-based modeling framework for representing multi-domain energy-based systems was introduced. A generic graph formulation, modified graph formulation, and linearized system was presented in Section 2.2. Adhering to the graph framework, electrical and mechanical component and subsystem models were formulated in terms of an energy balance. Lastly, a novel system composition algorithm was described to facilitate the development of a hybrid UAV system model.

Recall the 5 modeling framework requirements introduced at the start of this chapter: energy domain agnostic, modular, scalable, computationally efficient, and variable fidelity. Thus far, 4 of these 5 criteria have been discussed in this thesis. The graph-based modeling framework is inherently energy domain agnostic because it is based in conservation laws. The modular and scalable characteristic is best illustrated by the description of the modular component graphs and their composition into the larger scale UAV system model. As illustrated by the buck-boost converter (but generalizable to all components), the introduction of virtual elements provides a model with variable fidelity. The variable fidelity modeling will facilitate the development of hierarchical controllers. Computational efficiency metrics are not provided in this chapter. However, it will be shown that the graph model is com-

putationally efficient in Chapter 5 when demonstrating a model-based optimal controller operating in real-time on experimental hardware.

Chapter 3

Controller Design

3.1 Background

The control of mobile hybrid electric systems has gained popularity with the increasing trend of vehicle electrification. Since electrified systems have strong thermal limitations, significant research effort has been placed in optimizing the performance of on-board cooling systems [7, 25, 53, 54, 55, 56]. To optimize system-wide performance and efficiency, it is also important to consider the control of the electro-mechanical vehicle powertrain. The main losses within the electrical system result from resistances and switching losses. The engine, which operates most efficiently in a small operating range, yields the greatest inefficiency in the mechanical system. Deciding how and when to operate the vehicle's engine has been a challenge for control engineers.

As introduced in Chapter 1.1, thermostatic, rule-based, and heuristically formulated controllers were initially designed where the operating mode of the engine was discretely chosen by state thresholds [15, 16]. However, these controllers lacked robustness and provided sub-optimal behaviors. Although still sub-optimal, fuzzy-logic rule-based controllers were introduced to make the classical rule-based controllers more robust [16, 17]. More optimal control strategies were implemented by dynamic programs and model-predictive control strategies. Optimal system inputs can be solved a priori using a dynamic program (DP) [17, 18], however, the complexity of the DP grows exponentially with the number of states [19]. Additionally, dynamic programs are not robust to disturbances. Model predictive control strategies offer the benefit of being more robust to disturbances while finding optimal inputs in real-time [25]. However, there is a large computation requirement when using MPC since large optimization problems must be solved in real-time. The computation re-

quirement of MPC increases as the timescale of the system dynamics decreases because the optimization program has less time to solve.

Here, we focus on the development and comparison of a baseline rule-based controller and two model-based optimal controllers for a hybrid electric UAV powertrain. In Section 3.2, the embedded controllers are introduced. Section 3.3 describes baseline vehicle speed regulator and power share controller formulations. A more optimal centralized MPC approach is presented in Section 3.4. A hierarchical controller is formulated to manage both long and short-term mission objectives in Section 3.5. Lastly, in advance of the experimental controller validation in Chapter 5, the state estimation algorithm is briefly discussed in Section 3.6. Section 3.7 summarizes the contributions of this chapter.

3.2 Embedded Controllers

We interact with the experimental platform described in Chapter 4 through two proprietary embedded controllers: the motor speed regulator and avionic load current regulator. Therefore we introduce augmented plant dynamics that are composed of the embedded controller dynamics and the plant dynamics as illustrated in Figure 3.1. The signals passed to the

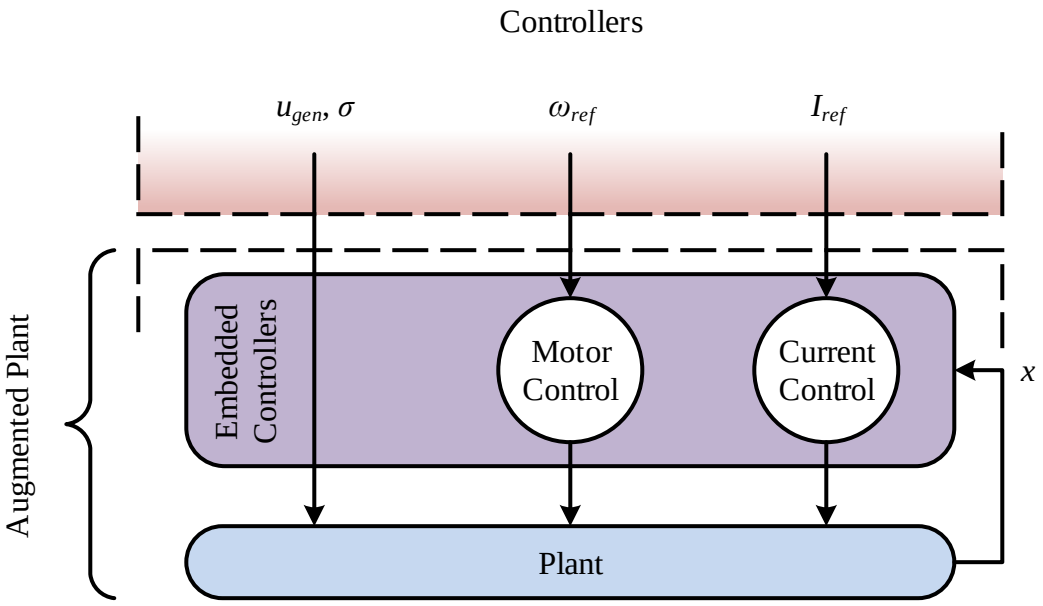


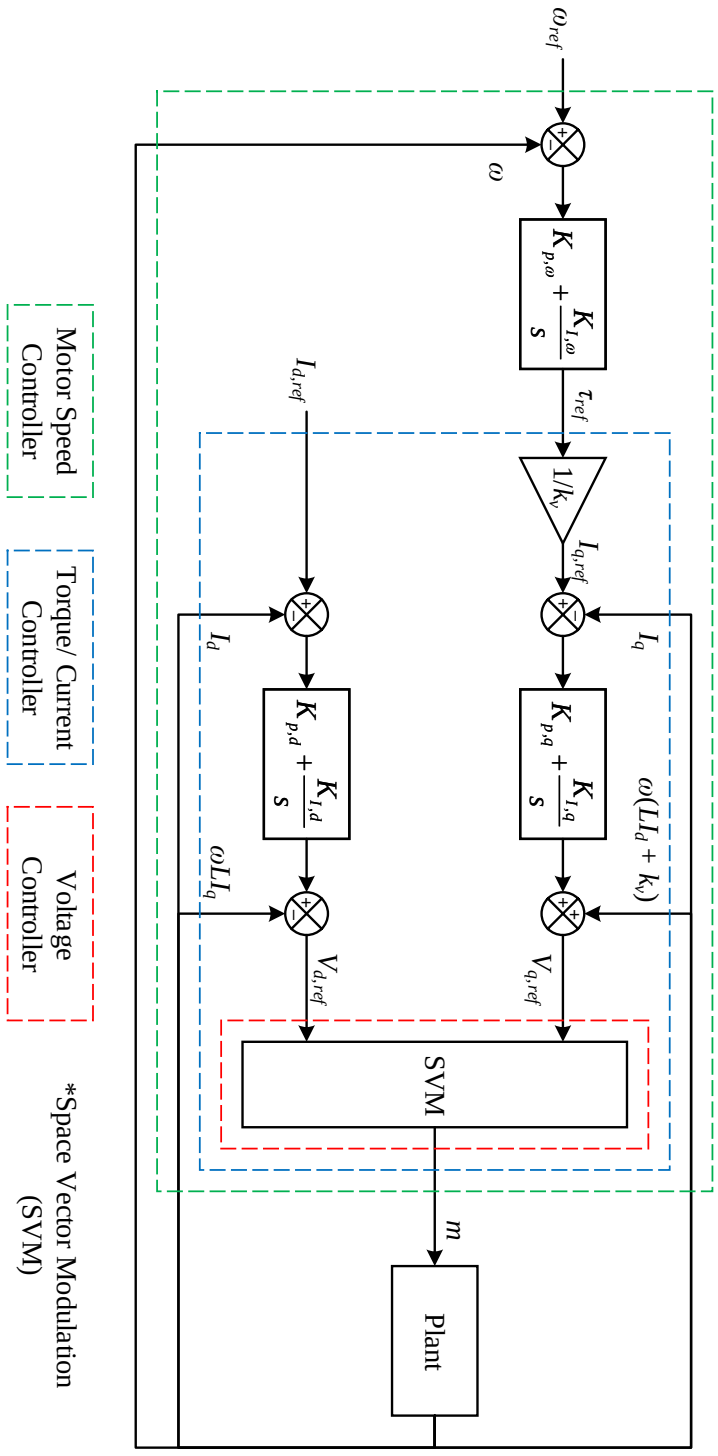
Figure 3.1: Signal flow diagram for the augmented plant.

augmented plant are the genset input u_{gen}, genset switch σ, commanded prop speed ω_{ref}, and commanded avionic load current I_{ref}. To experimentally validate the augmented plant, it is necessary to assume a structure of the embedded controller dynamics. The following two sections define the control architecture for the two embedded controllers.

3.2.1 Motor Speed Regulator

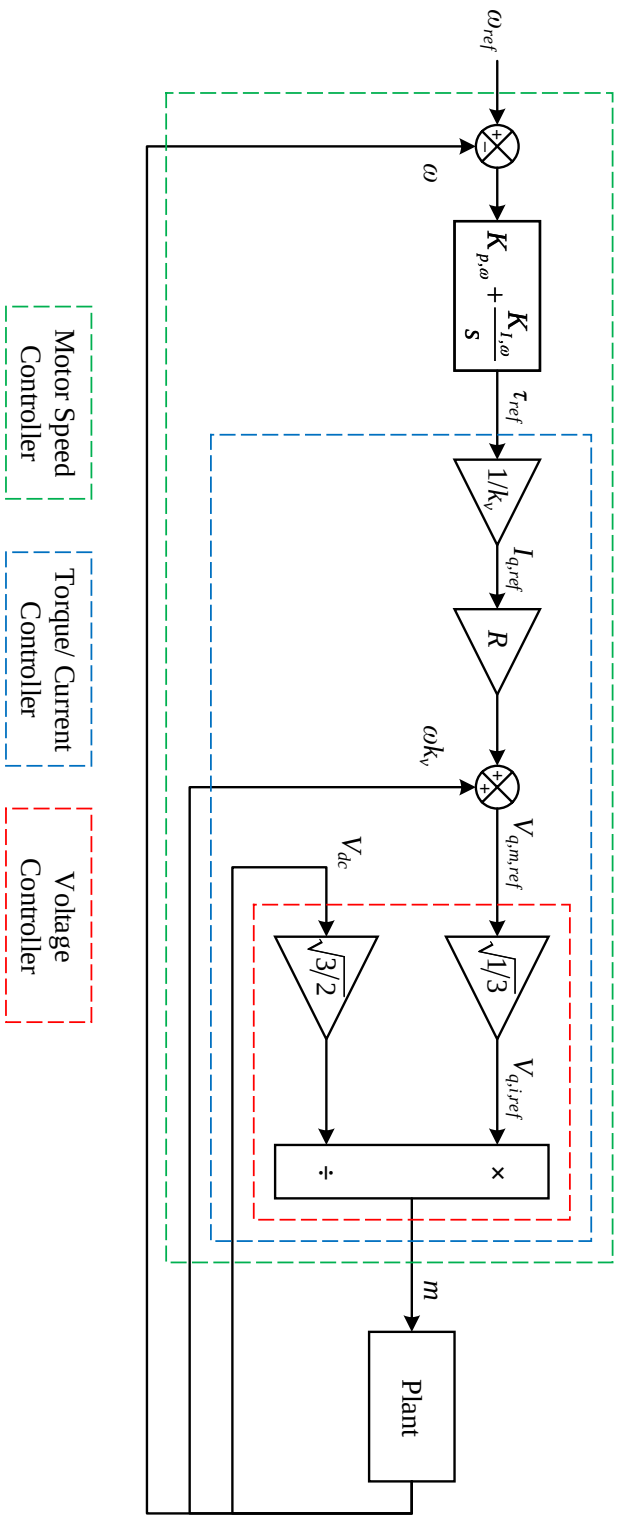
An initial topology of the motor speed regulator is proposed in Figure 3.2a. The control topology is based on the motor speed control formulation presented in [40]. By observation, the controller consists of 3 nested proportional-integral (PI) loops and 1 static rule-based switching controller (labeled SVM). First, the outermost control loop (motor speed controller) maps the motor speed tracking error to a motor torque reference. The middle control loop is the torque/current controller. Each branch of the middle loop maps q and d-axis current tracking error to q and d-axis voltage references respectively. Lastly, the innermost loop is termed the voltage control loop and consists of the space vector modulation (SVM) algorithm [40, 57]. The SVM algorithm determines a switch modulation sequence from the d and q-axis voltage references output by the torque/current control loop.

To validate the control dynamics, two simplifications are made to the initial control topology because there are physical limitations in the experimental hardware. First, a common control objective for a non-salient BLDC motor is to command the d-axis current to zero. Similar to Section 2.3.2, we assume that this control objective is achieved. Therefore, the lower PI loop in the torque/current controller can be removed. Second, the electrical system control dynamics are significantly faster than the mechanical system dynamics and communication rate of the experimental hardware. Therefore, the torque/current controller can be treated as static. Applying these simplifications to the control architecture described in Figure 3.2a yields a simplified motor speed controller composed of a single PI loop (Figure 3.2b). The following discussion analyzes the simplified control architecture.



(a) Generic motor speed controller.

Figure 3.2: Generic (a) and simplified (b) motor speed controller block diagrams.



(b) Simplified motor speed controller.

Figure 3.2 (cont.): The (a) generic and (b) simplified motor speed controller block diagrams.

Because the d-axis currents are assumed to be identically 0, the modulation signal m of the SVM algorithm, as described in [40], is given by

$$m = \frac{V_{q,i,ref}}{\sqrt{\frac{3}{2}}V_{dc}}, \tag{3.1}$$

where $V_{q,i,ref}$ is the q-axis voltage reference of the inverter and V_{dc} is the inverter DC-link voltage. The $\sqrt{\frac{3}{2}}$ term is a result of applying the power-invariant Park Transform. Next, the fast electrical system control dynamics are assumed static to simplify the torque/current controller. Treating the electrical motor dynamics (2.25) as static, the torque/current controller is given as

$$V_{q,m,ref} = RI_{q,m,ref} + k_v\omega, \tag{3.2}$$

where $V_{q,m}$ and $I_{q,m}$ are the q-axis motor voltages and currents respectively, R is the motor coil resistance, k_v is the motor constant, ω is the motor speed, and the subscript ref defines a reference signal. In Figure 3.2b, the $\sqrt{\frac{1}{3}}$ gain between the torque/current and voltage controllers converts the motor delta voltage reference to the inverter wye voltage reference.

The subsequent analysis of the PI loop shows that there will be zero steady state tracking error. Neglecting nonlinear losses, static friction, and the shaft load, the shaft speed dynamics and controller output are given by

$$J\dot{\omega} = \tau - b\omega, \tag{3.3a}$$

$$\tau_{ref} = \left(K_{p,\omega} + \frac{1}{s}K_{I,\omega}\right)(\omega_{ref} - \omega), \tag{3.3b}$$

where τ is the motor torque, J is the shaft inertia, $K_{p,\omega}$ and $K_{I,\omega}$ are the proportional and integral gains for the motor speed controller respectively, and b is the viscous friction constant. The fast electrical dynamics permit the assumption that the commanded motor torque is always achieved at each update of the speed controller. Therefore, the transfer

function from desired motor speed to plant motor speed is given by

$$\frac{\omega}{\omega_{ref}} = \frac{\frac{K_{p,\omega}}{J}s + \frac{K_{I,\omega}}{J}}{s^2 + \frac{b + K_{p,\omega}}{J}s + \frac{K_{I,\omega}}{J}}. \tag{3.4}$$

There are two free parameters, so the poles of the second-order system described by (3.4) can be placed arbitrarily. The placement of the poles will be experimentally validated in Chapter 4.

3.2.2 Current Regulator

The buck-boost converter requires a control loop to regulate the output voltage or current. Although only current regulation is considered in this thesis, similar methods can be applied to voltage regulation operation. The switching duty cycle for current regulation is determined via a single PI loop as shown in Figure 3.3.

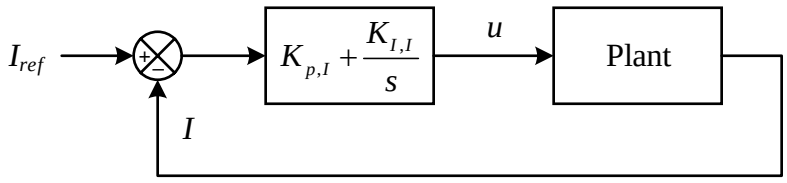


Figure 3.3: Current regulator block diagram.

3.3 Baseline Control Formulation

The baseline controller is composed of a vehicle speed and power share controller. The vehicle speed controller generates a speed reference that is passed to the embedded motor speed regulator. The power share controller generates an input and switch that is passed directly to the plant. The interactions between the baseline controller and augmented plant are illustrated in Figure 3.4.

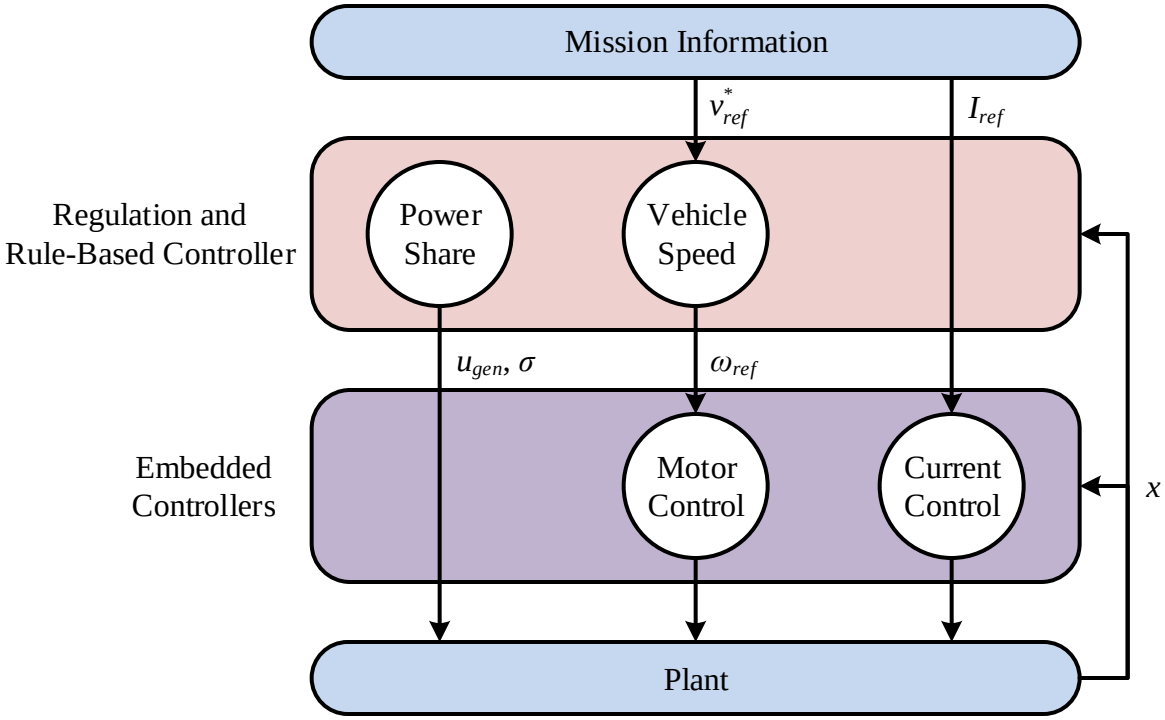


Figure 3.4: Signal flow diagram for the baseline controller design. Embedded controller descriptions in Section 3.2. Power Share controller description in Section 3.3.2. Vehicle speed controller description in Section 3.3.1.

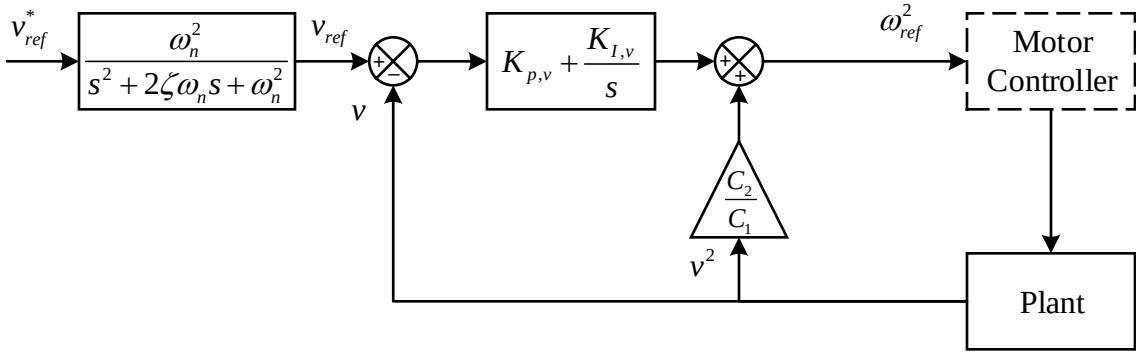


Figure 3.5: Block diagram for the vehicle speed regulator.

3.3.1 Vehicle Speed Regulator

The vehicle speed regulator is a PI loop that generates a motor speed reference based on the vehicle speed tracking error (Figure 3.5). Neglecting constant loss due to gravity, the vehicle speed dynamics and feedback law are given by

$$m\dot{v} = C_1\omega^2 - C_2v^2, \tag{3.5a}$$

$$\omega_{ref}^2 = \frac{C_2}{C_1}v^2 + \left(K_{p,v} + \frac{1}{s}K_{I,v}\right)(v_{ref} - v), \tag{3.5b}$$

where $C_1 = \rho\frac{D^4}{4\pi^2}C_T$ and $C_2 = \frac{1}{2}\rho AC_D$ are as described in Section 2.3.6, v is the vehicle speed, m is the vehicle mass, and $K_{p,v}$ and $K_{I,v}$ are the proportional and integral gains for the vehicle speed controller. Notice the feedforward term $\frac{C_2}{C_1}v^2$ used to linearize the system dynamics. To show that there will be zero steady-state tracking error, notice that the motor angular speed and controller dynamics are fast in comparison to the vehicle linear velocity dynamics. Therefore, assume that the commanded angular speed is achieved, $\omega = \omega_{ref}$. The transfer function from desired to plant vehicle speed is given by

$$\frac{v}{v_{ref}} = \frac{\frac{C_1K_{p,v}}{m}s + \frac{C_1K_{I,v}}{m}}{s^2 + \frac{C_1K_{p,v}}{m}s + \frac{C_1K_{I,v}}{m}}. \tag{3.6}$$

Similar to the motor speed controller, there are two free parameters so the poles of the second-order system described by (3.6) can be arbitrarily placed. Because the linear velocity dynamics are considerably slower that the motor speed dynamics, the poles of the vehicle speed controller should be slower than the poles of the motor speed controller (10-100x slower).

Lastly, a rapid change in the propeller speed reference can yield high currents in the propulsion subsystem. To adhere to physical system limitations, a 2nd-order low-pass filter with cutoff frequency ω_c and damping ratio ζ is applied to the the reference signal passed to the vehicle speed controller. This low-pass filter maps v_{ref}^* to v_{ref}.

3.3.2 Power Share Controller

The *power share* in a hybrid system is the ratio of engine/generator power to the total load power. In this work, the power share controller determines how much power is required

from the genset. Simple power share controllers are typically formatted as state machines where the operating mode is dependent on the battery state of charge and the system's total electrical load. The state machine thresholds are designed using rules and heuristics.

Control objectives for power share controllers are commonly categorized into charge depleting or charge sustaining strategies [58]. A charge depleting strategy will prioritize discharging the battery pack while relying on the engine to provide the remaining power necessary to meet vehicle objectives. If a charge depleting strategy is implemented, the engine should be appropriately sized to be able to provide peak load power. In a charge sustaining strategy it is more common for the engine to provide most of the demanded system power such that the battery state of charge does not deplete significantly over a mission. Because the engine for the experimental hybrid system described in Chapter 4 cannot independently provide peak power, a charge sustaining power share controller is developed.

Charge sustaining mechanisms are typically categorized into 4 approaches: state changing, threshold changing, power changing, and emergency handling [17]. A state changing approach switches between operating rules as a direct function of the battery state of charge. A threshold changing approach utilizes state of charge dependent thresholds that then govern the operating state of the state machine. A power changing approach characterizes the demanded genset power as a function of battery state of charge. Emergency handling rules help ensure that system constraints are not violated. The state machine in this thesis implements each of the charge sustaining mechanisms so examples will be provided in the following discussion.

The power share controller state machine is described in Figure 3.6. The load power is the sum of the propulsive load, avionic load, and processor load represented by the power flows along edges 35, 39, and 41 (respectively) of the system graph described in Figure 2.22. Similarly, the battery and genset power is the value of edges 11 and 12 (respectively) of the system graph. Because each of the power flows operate at the same voltage level (vertex 17), the power based analysis can be simplified to a current based analysis. Therefore, the horizontal axis of the state machine is labeled with total load current I_{load} instead of total load power. The vertical axis is labeled with the battery state of charge q. The following discussion outlines the operating modes of the state machine.

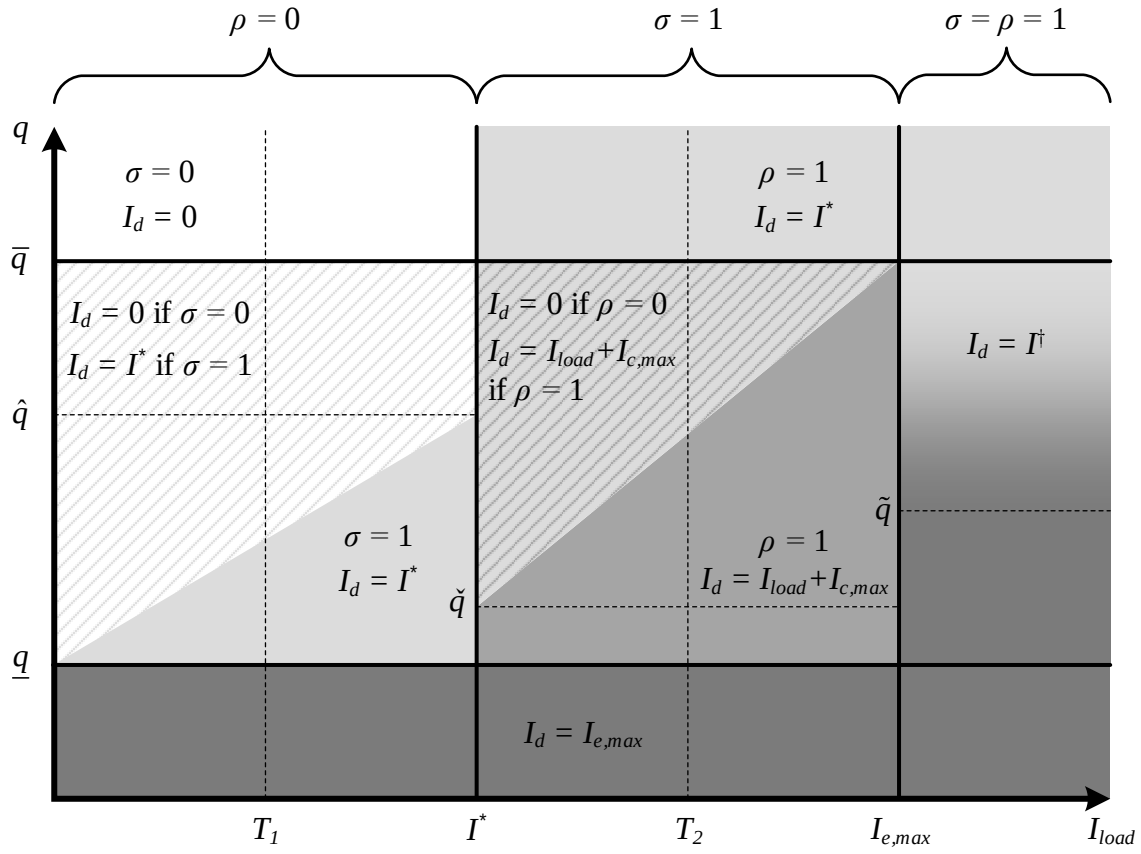


Figure 3.6: State machine representation of the power share controller. $I^\dagger$ is defined by the second case in (3.9c). The shading indicates a single state of the state machine. For example, the darkest grey sections all indicate that $I_d = I_{c,max}$. The striping indicates that the presence of multiple states in that region. Figure inspired by [17].

The relevant power share parameters are battery state of charge q, the battery state of charge upper and lower bounds $\bar{q}$ and $\underline{q}$, the commanded genset current I, the desired genset current I_d, the total system load I_{load}, the optimal genset current I^*, the max genset current $I_{e,max}$, and max battery charge current I_{ch}. The genset's optimal state is given by the state that minimizes engine specific fuel consumption. These parameters are physical states or constraints of the system. Note that the genset input u_{gen} is given by $u_{gen} = I/K$ where K is defined in 2.46.

For loads less than the sum of the optimal genset current and max battery charge current

($I_{load} < I^*$), the control strategy is

$$\sigma(t) = \begin{cases} 1 & \text{if } T_1(t) \leq I_{load}(t) \text{ and } q(t) \in [\underline{q}, \bar{q}], \\ 0 & \text{if } q(t) \geq \bar{q}, \\ \sigma(t_{-1}) & \text{else,} \end{cases}, \tag{3.7a}$$

$$\rho(t) = 0, \tag{3.7b}$$

$$I_d(t) = \begin{cases} 0 & \text{if } \sigma(t) = 0, \\ I^* & \text{if } \sigma(t) = 1, \end{cases}, \tag{3.7c}$$

$$T_1(t) = (I^*)\,\text{sat}\left(\frac{q(t) - \underline{q}}{\hat{q} - \underline{q}}\right)_0^1, \tag{3.7d}$$

where σ is a binary input variable representing whether the engine is 'on' (1) or 'off' (0), ρ is a binary variable governing an operating mode, T_1 is a changing threshold defined by (3.7d), and $\hat{q} \in (\underline{q}, \bar{q}]$ is a parameter that biases the slope of threshold T_1. Here, the genset is operated at its most efficient state whenever it is turned on. The variable threshold T_1 is implemented to bias the point at which the engine is turned 'on'. As seen in Figure 3.6, if the load is large, the engine will turn 'on' at a higher SOC to sustain battery charge. Together, (3.7a) and (3.7c) are examples of a state changing approach since the demanded current changes as a function of the state σ. Furthermore, (3.7d) is an example of a threshold changing mechanism because of the dependence on the pack SOC.

For loads where $I_{load} \in [I^*, I_{e,max})$, the control strategy is

$$\sigma(t) = 1, \tag{3.8a}$$

$$\rho(t) = \begin{cases} 1 & \text{if } T_2(t) \leq I_{load} \text{ and } q(t) \in [\underline{q}, \bar{q}], \\ 0 & \text{if } q(t) \geq \bar{q}, \\ \rho(t_{-1}) & \text{else,} \end{cases}, \tag{3.8b}$$

$$I_d(t) = \begin{cases} I^* & \text{if } \rho(t) = 0, \\ I_{load}(t) + I_{ch} & \text{if } \rho(t) = 1, \end{cases}, \tag{3.8c}$$

$$T_2(t) = (I^*) + (I_{e,max} - I^*)\,\text{sat}\left(\frac{q(t) - \check{q}}{\bar{q} - \check{q}}\right)_0^1, \tag{3.8d}$$

where T_2 is a variable SOC dependent threshold, and $\check{q} \in [\underline{q}, \bar{q})$ is a parameter used to bias the slope of threshold T_2. For this section of the state machine, the goal is to operate the engine at its most efficient state or recharge the battery. To maintain a higher pack SOC, the threshold T_2 biases the engine to recharge the pack at greater SOC when the system load is large. Similar to (3.7), (3.8b) and (3.8c) are an example of a state changing mechanism and (3.8d) is an example of a threshold changing rule.

Lastly, when $I_{load} \geq I_{e,max}$, the control strategy is

$$\sigma(t) = 1, \tag{3.9a}$$

$$\rho(t) = 1, \tag{3.9b}$$

$$I_d(t) = \begin{cases} I^* & \text{if }, q(t) > \bar{q} \\ I^* + (I_{e,max} - I^*)\,\text{sat}\left(\dfrac{\bar{q} - q(t)}{\bar{q} - \tilde{q}}\right)_0^1, & \text{if } q(t) \in [\underline{q}, \bar{q}], \\ I_{e,max} & \text{if } q(t) < \underline{q}, \end{cases} \tag{3.9c}$$

where $\tilde{q} \in [\underline{q}, \bar{q})$ biases the commanded current production. The objective of this mode is to operate efficiently only at high pack SOC and at max current otherwise. Here, the second case of (3.9c) is an example of a power changing mechanism.

Three emergency handling methods are necessary to maintain safe operation

$$I(t) = \text{sat}\,(I_d(t))_0^p \quad \text{where } p = \min\{I_{e,max}, I_{load} + I_{ch}\}, \tag{3.10a}$$

$$I_d(t) = I_{e,max} \quad \text{if } q(t) < \underline{q}, \tag{3.10b}$$

$$I_d(t) = I_{e,max} \quad \text{if } I_{load} > I_{b,max} + I^*. \tag{3.10c}$$

Equation (3.10a) saturates the command current such that the pack is not charged at excessive rates. This saturation is commonly active at low loads. Equation (3.10b) demands max current if the pack state of charge decreases below its lower limit. Equation (3.10c) demands max current if the load exceeds the sum of the battery max discharge rate $I_{b,max}$ and opti-

mal genset current. Note that (3.10b) and (3.10c) should be unlikely operating modes. To prevent high frequency switching between operating states, a dwell time on operating mode is implemented (Figure 3.7), where T_{dw} is the dwell time before a new mode can be selected, $\tilde{x}_k$ is the chosen state machine operating mode, and x_k is the actual operating state machine operating mode.

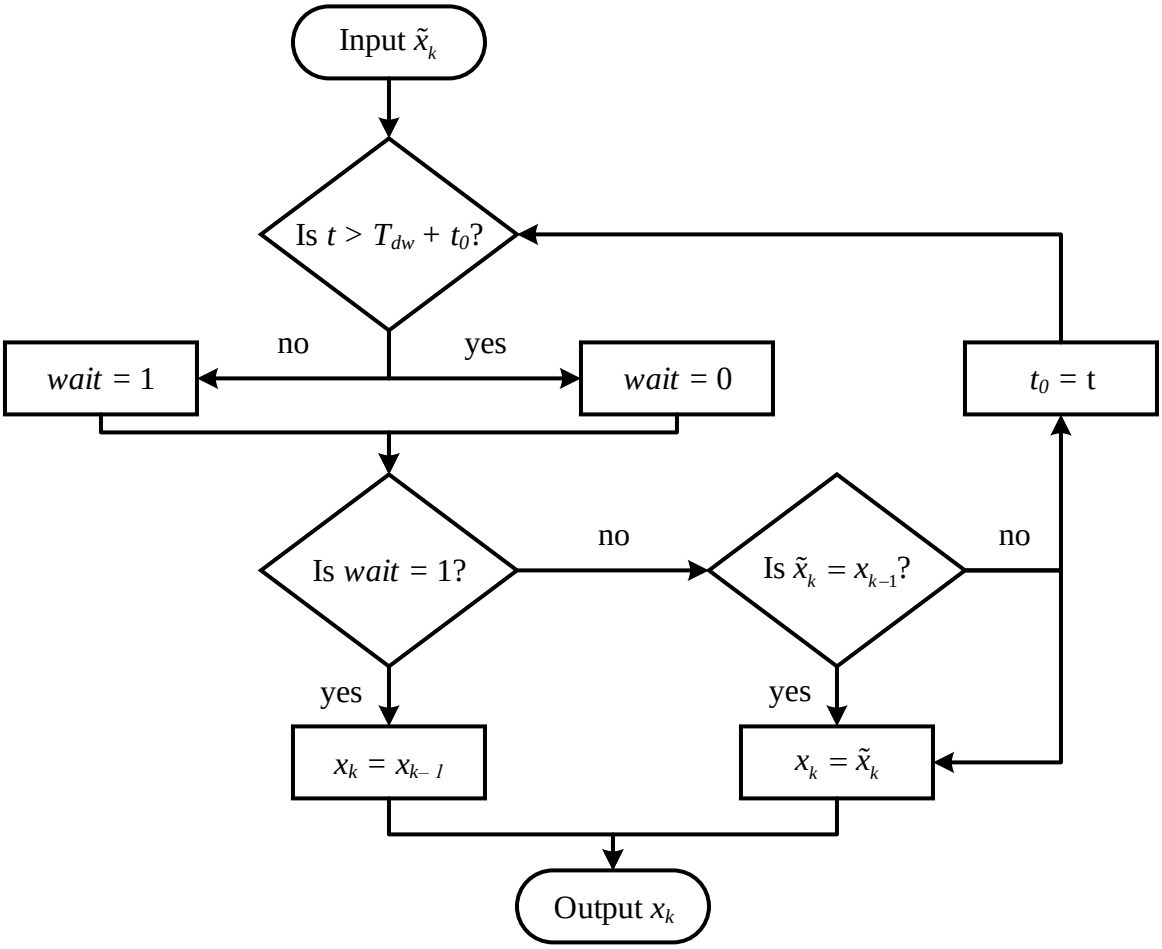


Figure 3.7: Decision tree for the dwell time algorithm.

This power share control formulation has 4 tuning variables: $\hat{q}$, $\check{q}$, $\tilde{q}$, and T_{dw}. The variables $\hat{q}$ and $\check{q}$ bias the threshold at which the engine should be turned 'on' to charge the pack. If $\hat{q}$ and $\check{q}$ are large, the battery will be charged to a higher SOC. However, there is a trade-off that the engine may be used more while operating less efficiently. A similar trade-off occurs when tuning $\tilde{q}$. The dwell time parameter T_{dw} is used to mitigate high frequency mode switching and should be tuned based on operating conditions and component specifications.

3.4 Centralized Model Predictive Controller

3.4.1 Model Predictive Control Description

A model predictive controller uses information about system dynamics and objectives to forecast a sequence of inputs that optimize system behavior across a time horizon [20]. A system model and optimization program are central to the formulation of a model predictive controller. The model is used to predict system state trajectories as a response to an input sequence calculated by an optimization program. Assuming full-state feedback at each controller call, the initial system states and objectives over the horizon are input to the optimization program that solves for the optimal set of inputs at each step in the horizon. The inputs at the first step in the horizon are applied to the plant. The controller's horizon N and time step Δt govern the time horizon of the predicted state trajectory and are typically treated as tuning variables. In practice Δt typically has an upper limit associated with the timescale of the fastest system dynamic. Additionally, the choice of N is dependent on the available computational power, model accuracy, required disturbance preview.

3.4.2 Controller Formulation

The centralized model predictive controller is responsible for coordination between both embedded controllers and the operating mode of the engine. The proposed controller topology is shown in Figure 3.8. The predictive controller is passed the current measured or estimated system state and mission information (mission objectives, known disturbances, etc.). Similar to the baseline controller, the mission objectives include a desired vehicle velocity and avionic load current. Note that the predictive controller receives future mission information as well, whereas the baseline controller does not. Known disturbances are represented by sink states of the graph model. In this work, the only time-varying sink state is the genset current (sink vertex 2) with dynamics modeled outside the graph framework. The centralized controller outputs are the genset mode ('on' or 'off'), the genset input sequence, motor speed state trajectory, and avionic load current trajectory. As mentioned in Section 3.4.1, only the first index of the input sequence or state trajectory is passed to the augmented plant. Lastly, the

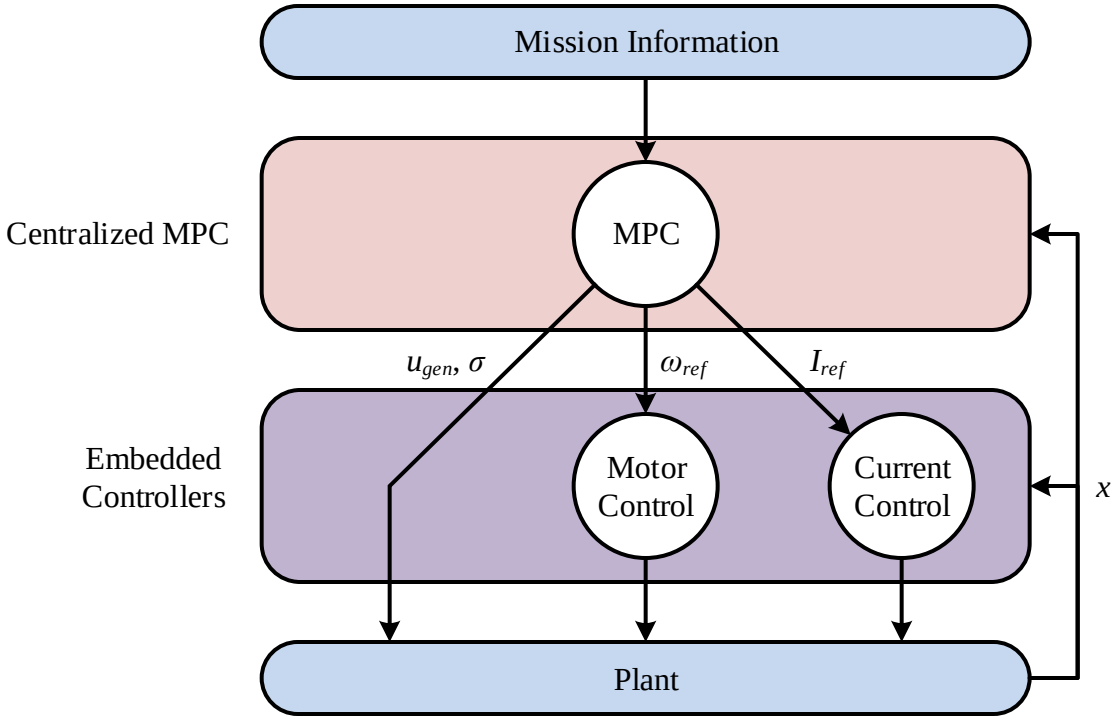


Figure 3.8: Signal flow diagram for the centralized controller design. Embedded controller descriptions in Section 3.2. Centralized MPC controller description in Section 3.4.2.

signal flow diagram (Figure 3.8) looks like a two-level hierarchical controller. However, recall that the embedded controllers and plant are combined into the augmented plant. Therefore, this controller is centralized with respect to the augmented plant.

A challenge of the application of MPC to hybrid electric systems is the ability to plan for both fast and slow mission objectives in a computationally efficient and accurate manner [17]. Because an electro-mechanical system has many fast dynamics, it is typical to choose a fast update rate for the controller. However, a fast update rate inherently limits the controller's capability for real-time long-term mission planing. To compensate for this limitation, an SOC planning algorithm is developed to provide a time-varying lower bound to the model predictive controller. The lower SOC bound is designed to represent a conservative estimate of how much battery charge is required to complete mission objectives for H seconds into the future.

The lower SOC bound is developed with a priori knowledge of the mission and a static analysis of part of the graph model. The drivetrain and avionic load are the main power

consuming subsystems in the vehicle model (assuming the processor load is negligible). Assuming that the demanded velocity v_{ref} and avionic load I_{ref} profiles are statically tracked, the propulsion power P_{prop} and avionic load power P_{LP} are given by

$$P_{prop} = \frac{1}{2}\rho A C_D v_{ref}^3 - mgv_{ref}\sin(\theta), \tag{3.11a}$$

$$P_{LP} = V_{LP} I_{ref}, \tag{3.11b}$$

where the parameters in (3.11a) are defined in Section 2.3.6 and V_{LP} is the operating voltage of the avionic load (treated as a disturbance in the graph model). Assuming some minimum efficiency for the propulsion η_{prop} and avionic load subsystems η_{LP}, the total system load is $P_{load} = \frac{P_{prop}}{\eta_{prop}} + \frac{P_{LP}}{\eta_{LP}}$. P_{load} represents the sum of the power flows along edges 35 and 39 (respectively) of the system graph (Figure 2.22). Next, assume the battery voltage dynamics are negligible with an upper bound on the battery series resistance R_s and a lower bound on the battery open-circuit voltage V_{ocv}. A system of equations can be developed by applying conservation laws to the bus voltage state V_{bus} (vertex 17) and battery current state I_{bat} (vertex 6)

$$V_{bus} I_{bat} = P_{load} - \bar{I}_{gen} V_{bus}, \tag{3.12a}$$

$$V_{bus} I_{bat} = V_{ocv} I_{bat} - R_s I_{bat}^2, \tag{3.12b}$$

where $\bar{I}_{gen} = b_1 V_{bus} + b_2$ is the max genset current as a linear function of the bus voltage. The max genset current is used to study a "worst-case" operating condition. The system of equations can be solved by substitution and has two solutions. Only one solution yields plausible values for V_{bus} and I_{bat}. Using the valid solution, the time-varying lower SOC bound $\underline{q}(t)$ is given by

$$\underline{q}(t) = \underline{q} + \frac{\int_t^{t+H} \text{sat}(I_{bat})_0^{\infty} d\tau}{Q} \tag{3.13}$$

where $\underline{q}$ is the static minimum SOC and Q is the battery capacity. The preview horizon is treated as a tuning parameter for the algorithm. A long preview horizon may yield a more conservative controller whereas a short preview horizon may yield a more aggressive

controller. Lastly, recall the mention of minimum or maximum values for the open-circuit voltage, battery series resistance, and subsystem efficiency parameters. Choosing these parameters in this manner yields a max-lower SOC bound for the static system under the aforementioned assumptions.

Next, the non-linear plant dynamics described by (2.10) are not appropriate for real-time optimization methods because non-linear optimization can be too computationally expensive. Therefore, a discrete linear system model (2.21) is used in the optimization formulation. The non-linear system is re-linearized at every time step to improve the controller performance over a large state space. Because the genset operating mode is treated as a binary input, a discrete switched linear model with two modes is implemented. Assuming the non-linear graph model does not have source flow edges, the augmented discrete switched linear model dynamics are given by

$$\begin{bmatrix} x_{(k+1),d} \\ 0 \end{bmatrix} = A^{\sigma}_{z1} \begin{bmatrix} x_{k,d} \\ x_{k,a} \end{bmatrix} + A^{\sigma}_{z2} x^{t}_{k} + B^{\sigma}_{z1} u_k + B^{\sigma}_{z2} \tag{3.14}$$

where the superscript σ denotes a switched analogue to the matrices presented in (2.21). Note that the augmented model combines the discrete linear dynamics of the graph model and the genset model.

The optimization problem is formulated using the following mixed-integer quadratic program (MIQP):

$$J_k = \Lambda_x \left\| r_{k+1} - x_{k+1} \right\|_2^2 + \Lambda_s \left\| s_k \right\|_2^2 + \Lambda_b \left\| b_k \right\|_2^2 + \Lambda_{\text{sfc}} \text{sfc}_k + \tag{3.15a}$$

$$+ \Lambda_{dx} \left\| x_{k+1} - x_k \right\|_2^2 + \Lambda_{du} \left\| u_k - u_{k-1} \right\|_2^2 + \Lambda_{\sigma} \left\| \sigma_k - \sigma_{k-1} \right\|_2^2, \tag{3.15b}$$

$$\underset{u,s,b,\sigma}{\text{minimize}} \quad J = \sum_{k=1}^{N} J_k, \tag{3.15c}$$

subject to

$\forall k \in [1:N]$

$$\begin{bmatrix} x_{(k+1),d} \\ 0 \end{bmatrix} = A^{\sigma}_{z1} x_k + A^{\sigma}_{z2} x^t_k + B^{\sigma}_{z1} u_k + B^{\sigma}_{z2}, \tag{3.15d}$$

$$\underline{x} - s_{k+1} \leq x_{k+1} \leq \bar{x} + s_{k+1}, \tag{3.15e}$$

$$s_{k+1} \geq 0, \tag{3.15f}$$

$$\underline{x}_{k+1} - b_{k+1} \leq x_{k+1}, \tag{3.15g}$$

$$b_{k+1} \geq 0, \tag{3.15h}$$

$$\underline{u} \leq u_k \leq \bar{u}, \tag{3.15i}$$

$$u_0 - \delta_u \leq u_k \leq u_0 + \delta_u, \tag{3.15j}$$

$$\sigma_k = \{0, 1\}. \tag{3.15k}$$

Note that the index at $k = 0$ denotes then current system state at the controller call. The decision variables of the optimization problem are the system input sequence $\mathbf{u} = [u_1, \cdots, u_N]$, slack variables $\mathbf{s} = [s_1, \cdots, s_{N+1}]$ and $\mathbf{b} = [b_1, \cdots, b_{N+1}]$, and switch state sequence $\sigma = [\sigma_1, \cdots, \sigma_{N+1}]$. Note that the switch state is binary (3.15k) and allowed to change across the horizon. Equations (3.15a) and (3.15b) are a quadratic cost function composed of seven separate costs (the sfc term is quadratic 2.47). First, there is state tracking cost $\Lambda_x \geq 0$ associated with states x tracking reference r. Second and third, there are slack penalties $\Lambda_s \geq 0$ and $\Lambda_b \geq 0$. Fourth, there is a fuel cost $\Lambda_{\text{sfc}} \geq 0$ associated with the genset. Fifth and sixth, there is cost associated with the rate of change of states $\Lambda_{dx} \geq 0$ and inputs $\Lambda_{du} \geq 0$. Lastly, there is a switching cost Λ_σ that penalizes switching the engine 'on' or 'off'. Note that Λ_x, Λ_s, Λ_b, Λ_{dx}, and Λ_{du} are diagonal matrices of appropriate size.

Equation (3.15d) is the discrete switched linear state dynamics. Equations (3.15e) and (3.15g) enforce constant and time varying softened state constraints with lower bounds $\underline{x}$ and upper bound $\bar{x}$. The inclusion of positive slack variables (3.15f) and (3.15h) ensure feasibility of the optimization problem by softening state constraints. Equation (3.15i) provides hard constraints on system inputs with lower and upper bound $\underline{u}$ and $\bar{u}$. Lastly, equation (3.15j)

bounds the deviation from the previously applied input u_0 by δ_u. Bounding the input biases the state dynamics to stay near the linearization point and therefore improves the controller prediction.

3.5 Hierarchical Model Predictive Controller

3.5.1 Hierarchical Controller Description

A notional example of a hierarchical controller is shown in Figure 3.9 [26, 59]. In this framework, a supervisor at the vehicle level predicts and optimizes system behavior with respect to slower system dynamics and objectives. This information is passed down in the hierarchy to various controllers that are tuned to regulate the behavior of specific systems and subsystems. These lower level controllers update faster than the upper level controllers. The combined long preview of the upper level controllers and short-preview of the lower level controllers enables coordination between both fast and slow systems dynamics. This inherent ability to effectively coordinate system behavior across timescales is a key merit of the hierarchical control framework.

3.5.2 Controller Formulation

As mentioned previously, a challenge with the application of MPC to fast dynamic electro-mechanical systems is its ineffectiveness to achieve long-term real-time mission planning while regulating fast system dynamics. In the previous section, an algorithm was introduced to plan a lower-bound for a fast updating centralized MPC. Recall that the algorithm is designed to run before the mission start. The motivation for the hierarchical controller is to provide an optimal alternative to the SOC bounding algorithm that can run in real-time.

The structure of the hierarchical controller is shown in Figure 3.10. The predictive controllers at each level of the hierarchy have knowledge of the full plant dynamics. Here, mission information is passed to an MPC algorithm designed with a long mission preview that outputs a desired vehicle speed, avionic load current, and lower bound for the battery

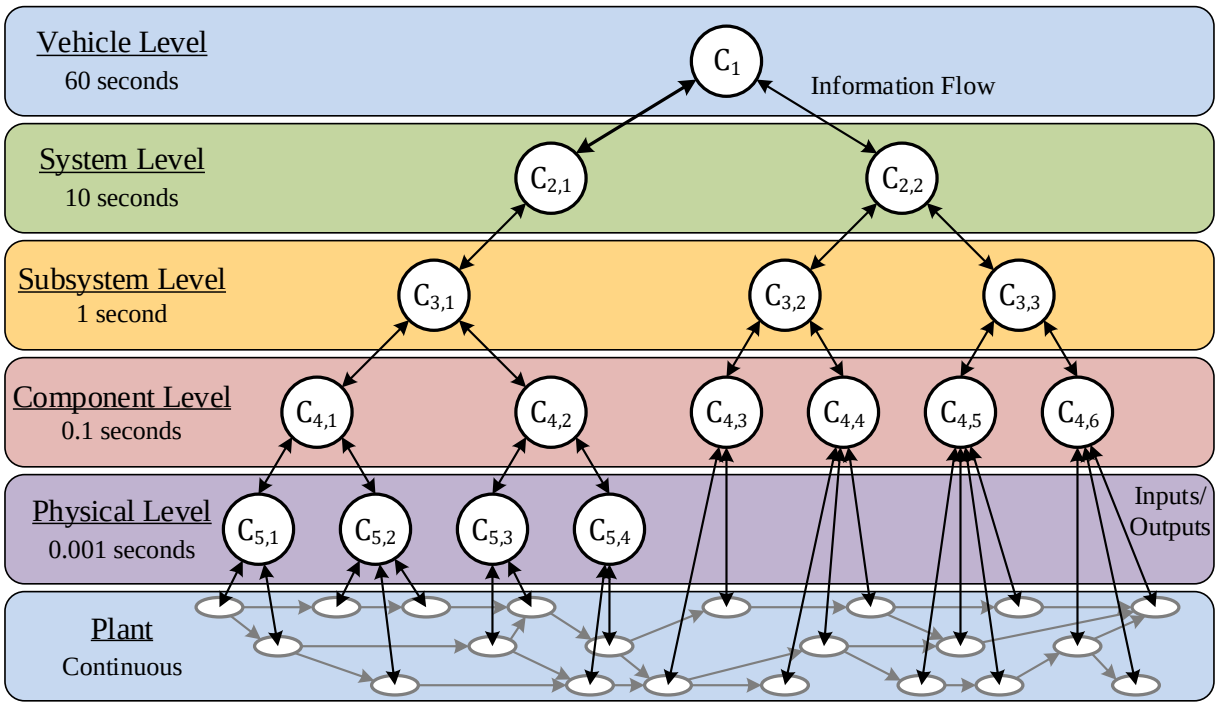


Figure 3.9: Example of a hierarchical control framework where each node at each level of the hierarchy represents a controller. Modified from [27].

SOC. These outputs are passed to the lower level MPC which calculates a desired propeller speed, avionic load current, and genset input. The lower level MPC plans over a shorter mission preview in comparison to the upper level controller to better regulate the fast system dynamics. The outputs of the lower level MPC (propeller speed, avionic load current, genset command) are passed to the embedded controllers.

Various methods can be used to pass information from upper to lower level controllers. The simplest method is to downsample the output sequence $\mathbf{o} = [o_1, \cdots, o_N]$ from the upper level controller at even intervals of the lower level controller's update rate. The downsampled sequence can then be passed as mission information to the lower level controller. A second option is to augment the upper level controller's output sequence with the current system state x_k to yield $\mathbf{o}' = [x_k, o'_1, \cdots, o'_N]$. Next, interpolate between consecutive points in $\mathbf{o}'$, downsample the interpolated set at the update rate of the lower level controller, and then pass the resulting sequence as mission information into the lower level controller. More complex methods based on these principles can be developed. In this work, the upper level MPC passes information to the lower level MPC via the first method and the lower level

MPC passes information to the embedded controllers via the second method.

The precise mathematical control formulation for the upper and lower level controllers is identical to that of the centralized controller described by (3.15). For completeness, the control formulation is repeated below and the reader is directed to Section 3.4 for variable descriptions.

$$J_k = \Lambda_x \left\| r_{k+1} - x_{k+1} \right\|_2^2 + \Lambda_s \left\| s_k \right\|_2^2 + \Lambda_b \left\| b_k \right\|_2^2 + \Lambda_{\text{sfc}} \text{sfc}_k + \tag{3.16a}$$

$$+ \Lambda_{dx} \left\| x_{k+1} - x_k \right\|_2^2 + \Lambda_{du} \left\| u_k - u_{k-1} \right\|_2^2 + \Lambda_\sigma \left\| \sigma_k - \sigma_{k-1} \right\|_2^2, \tag{3.16b}$$

$$\underset{u,s,b,\sigma}{\text{minimize}} \quad J = \sum_{k=1}^{N} J_k, \tag{3.16c}$$

subject to

$\forall k \in [1:N]$

$$\begin{bmatrix} x_{(k+1),d} \\ 0 \end{bmatrix} = A_{z1}^\sigma x_k + A_{z2}^\sigma x_k^t + B_{z1}^\sigma u_k + B_{z2}^\sigma, \tag{3.16d}$$

$$\underline{x} - s_{k+1} \leq x_{k+1} \leq \bar{x} + s_{k+1}, \tag{3.16e}$$

$$s_{k+1} \geq 0, \tag{3.16f}$$

$$\underline{x}_{k+1} - b_{k+1} \leq x_{k+1}, \tag{3.16g}$$

$$b_{k+1} \geq 0, \tag{3.16h}$$

$$\underline{u} \leq u_k \leq \bar{u}, \tag{3.16i}$$

$$u_0 - \delta_u \leq u_k \leq u_0 + \delta_u, \tag{3.16j}$$

$$\sigma_k = \{0, 1\}. \tag{3.16k}$$

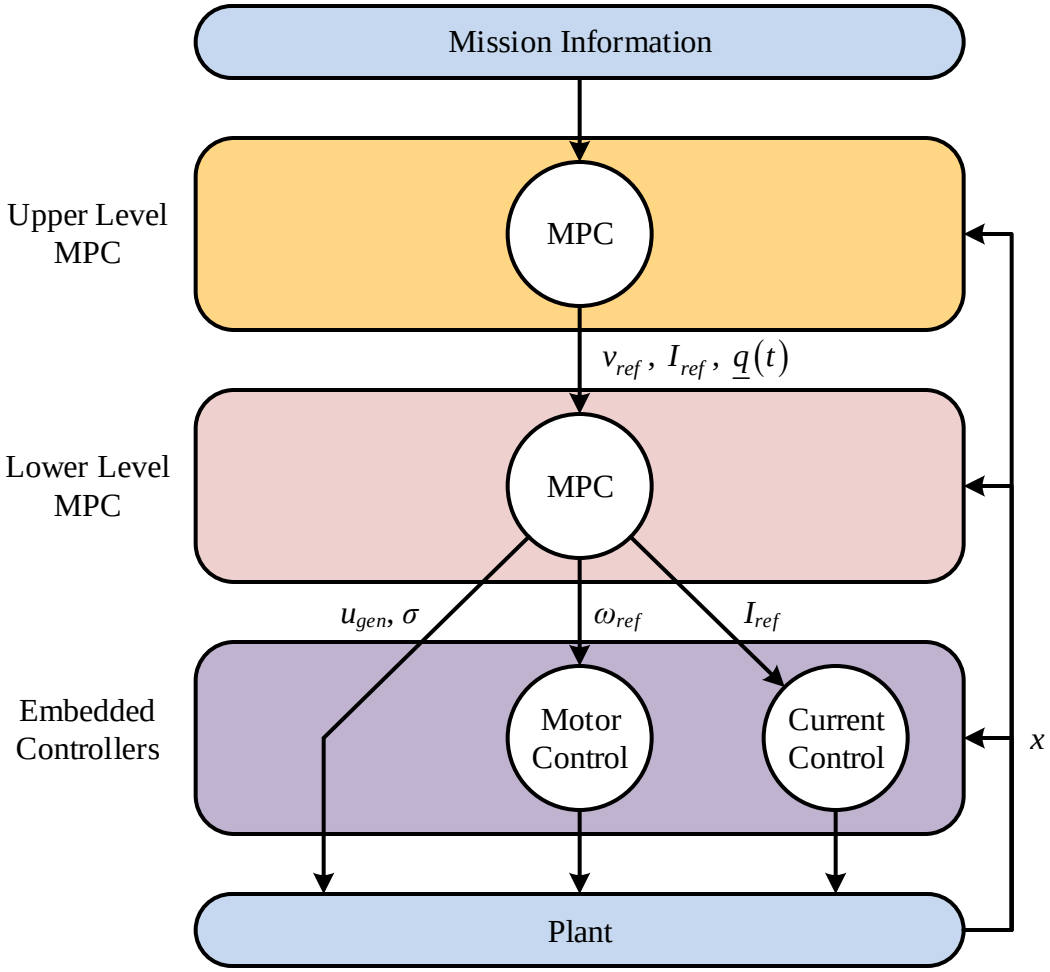


Figure 3.10: Signal flow diagram for the hierarchical control design. Embedded controller descriptions in Section 3.2. Centralized MPC controller description in Section 3.5.2.

3.6 State Estimation

The controllers of the previous sections require full-state feedback for hardware implementations. Full-state knowledge is typically obtained using an observer which estimates unknown states using sensor measurements. Figure 3.11 describes the decentralized observer developed in this work. The battery observer estimates the 5 battery states $\hat{x}_{bat}$ and is formulated as a Central Difference Kalman Filter (CDKF) [60, 61]. The CDKF is a version of sigma-point Kalman Filtering that maintains the non-linear form of the model to improve estimation accuracy. The system observer estimates the remaining system states $\hat{x}_{sys}$ and is formulated as an Extended Kalman Filter (EKF) [61]. The EKF uses a linearized version of the system model to estimate states. The CDKF and EKF algorithms are provided in Appendix C.

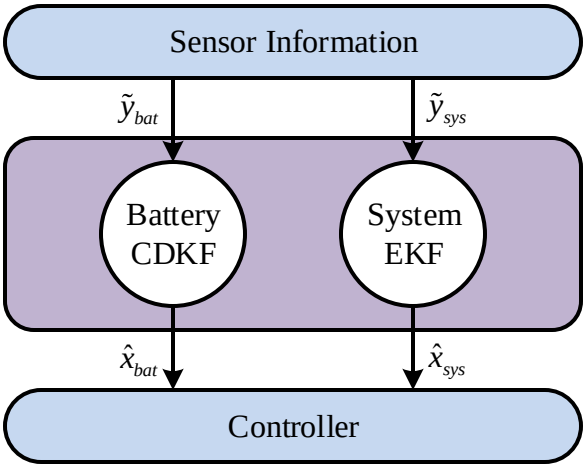


Figure 3.11: Signal flow diagram for the observer. The signals $\tilde{y}$ represent sensor measurements.

3.7 Conclusion

In review, this chapter introduced 2 controller models, 3 control designs, and an observer. The embedded controllers described in Section 3.2 are modeled as PI controllers and will be experimentally validated in Chapter 4. The first control design is a decentralized controller (Section 3.3) consisting of a vehicle speed regulator and power share controller. The power share controller was formulated to sustain the battery pack state of charge. Then a centralized model predictive controller was described in Section 3.4. Long term mission planning was integrated in the centralized control design through a time-vary bound on the battery state of charge. As an alternative, a hierarchical controller was developed (Section 3.5) to conduct real-time planning for the battery state of charge. The advanced control designs that provide state of charge bounds are unique contributions of this work. Lastly, the state estimation structure was introduced.

Chapter 4

Model Validation

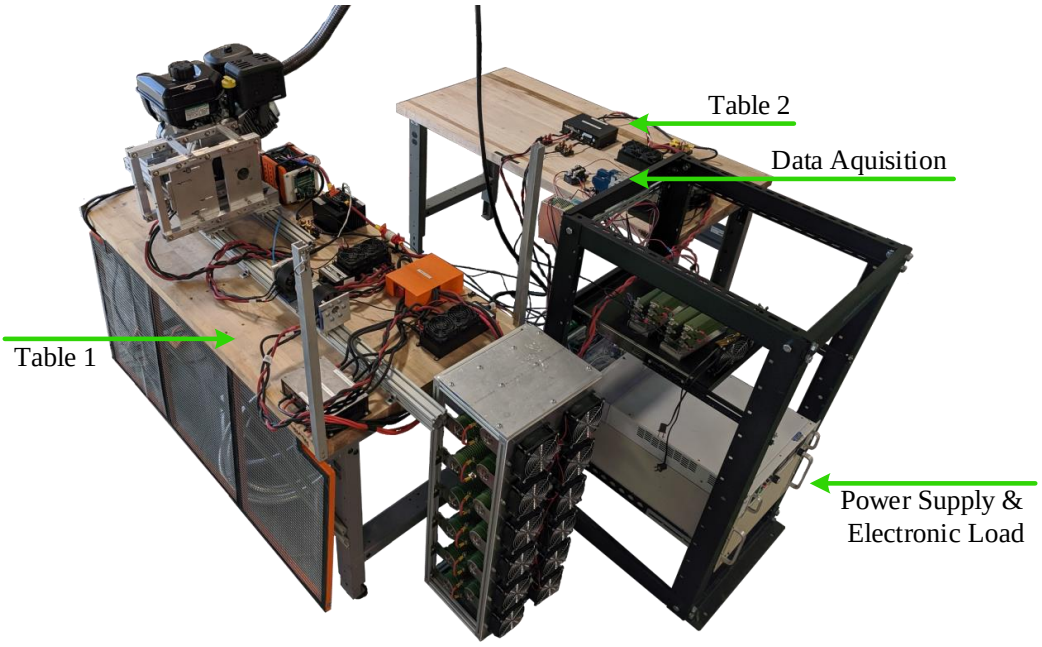
4.1 Background

The baseline and model-based control algorithms described in Chapter 3 will be experimentally validated and evaluated to highlight improvements in both performance and efficiency in Chapter 5. To evaluate the advanced model-based controllers on experimental hardware, it is necessary to have a validated system model. Previous validation efforts have been done to individual components such as batteries, motors, etc. [8, 62, 63, 64]. This chapter will aggregate efforts developed by other researchers as well as introduce new validation methods. The goal is that this thesis can provide comprehensive procedures to validate the electrical and mechanical dynamics of a hybrid electric unmanned aerial vehicle.

Section 4.2 will provide a description of the experimental hardware. The battery model validation procedure and analysis will be introduced in Section 4.3. The motor parameter identification process will be described in Section 4.4. The inverter and its control dynamics are validated in Section 4.5. The DC-DC conversion power electronics losses are identified in Section 4.6. Genset characterization is accomplished in Section 4.7. The processor load model is validated in Section 4.8. System wide open-loop validation results are described in Section 4.9. The chapter contributions are summarized in Section 4.10.

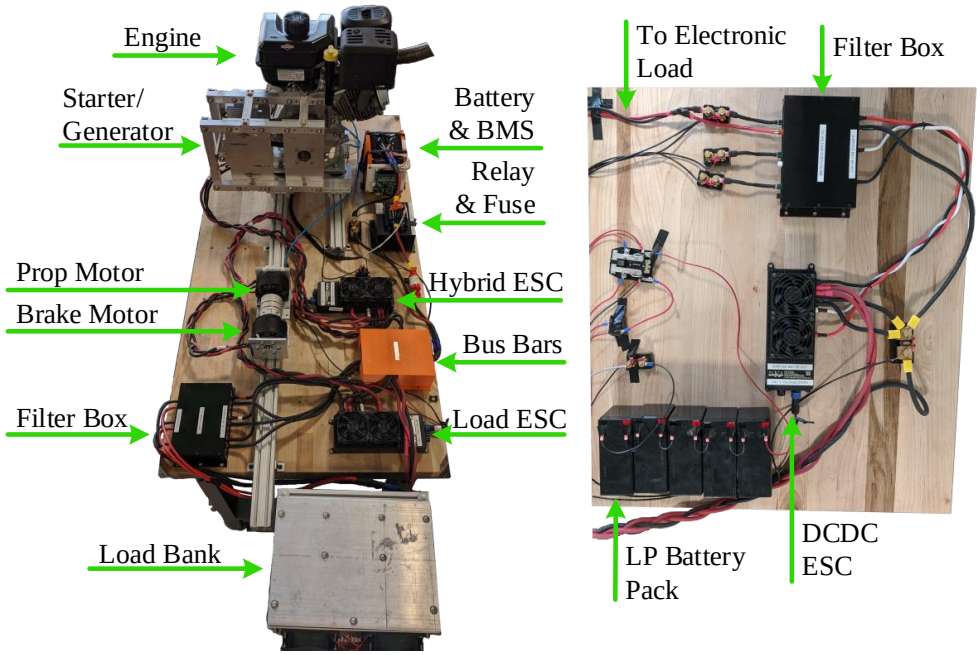
4.2 Testbed Description

The novel POETS hybrid electric UAV powertrain testbed (Figure 4.1) is used to facilitate the model validation process. A schematic is provided in Figure 4.2. The system model described in Figure 2.2 was modeled after the architecture of the experimental platform.



(a) Setup for the hybrid electric UAV powertrain testbed.

(i) (ii)



(b) Detailed labeling for the hybrid electric UAV powertrain testbed components. (i) Labels the components on Table 1 and (ii) labels the components on Table 2.

Figure 4.1: An (a) overhead perspective of the full testbed and a birds-eye view of (b.i) table 1 and (b.ii) table 2.

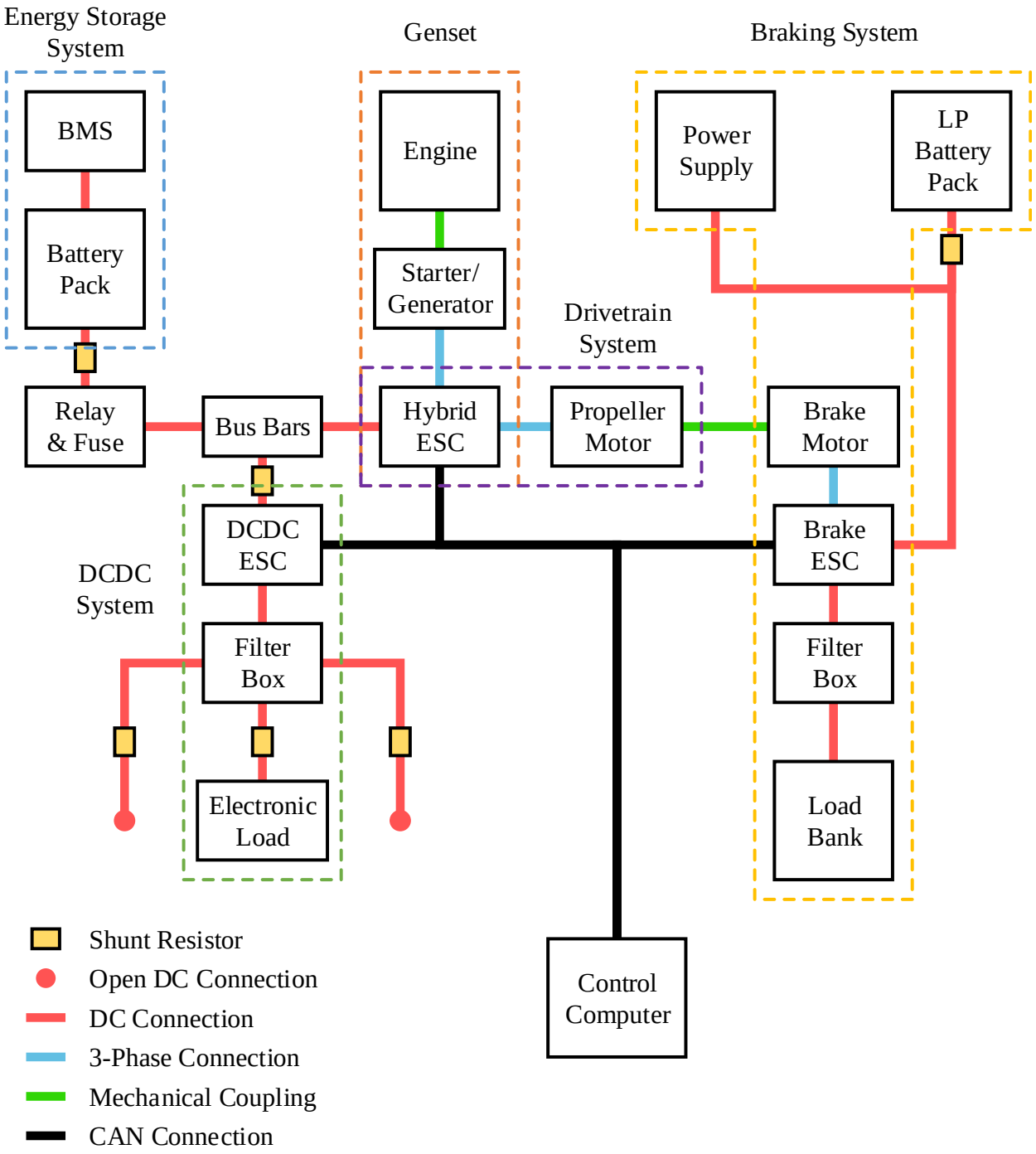


Figure 4.2: Layout schematic of the experimental testbed shown in Figure 4.1.

Before describing the testbed subsystems, it is first important to understand the operation and purpose of the 3 electronic speed controllers (ESCs) located on the testbed. Although each ESC has the same circuit topology, they all have different operating modes and control strategies. Each speed controller consists of a single DC-link capacitor and two three-phase bridge converter circuits (Figure 4.3). The DC-link capacitor helps regulate the bus voltage. The three-phase bridge converter is versatile and can represent a 3-phase DC-AC inverter, a 3-phase AC-DC active rectifier, or 3 independent DC-DC bi-directional buck-boost converters. When operating as a 3-phase device connected to an electric machine, the control objectives are to either regulate shaft speed or torque. A sensorless position algorithm is used to estimate motor position and a space vector modulation algorithm is used to command the switching devices. When operating as a buck-boost converter, the control objectives are to regulate output voltage or current (similar to the operation of a programmable power supply). Each DC channel can operate at up to 60A at 45V. If operating as a buck-boost converter, additional filtering is required at the ESC output (see Figure 2.8). Filters are located in 2 filter boxes on the testbed (circuit diagram in Figure 4.4). The speed controllers and filter boxes were designed by PC Krause and Associates (PCKA). Note that although the ESCs are referred to as "speed controllers", they have multiple operating modes and applications beyond speed regulation. The naming and operation of each ESC on the testbed is highlighted in Table. 4.1.

The testbed is composed of 5 distinct subsystems. The Energy Storage System (ESS) consists of a battery pack and battery management system (BMS). The battery is a 16S7P pack composed of Samsung 18650 cells. Battery pack information and operating limits are summarized in Table 4.2. The BMS provides continuous monitoring of the cells during operation, measures pack voltage, and can passively balance the cells through resistors. A

Table 4.1: The electronic speed controller names, subsystem placement (Figure 4.2), and control objective(s).

ESC Name	Hybrid		Brake	DCDC
Subsystem	Drivetrain	Genset	Braking	DCDC
Regulated State	Speed	Speed or Torque	Torque	Current

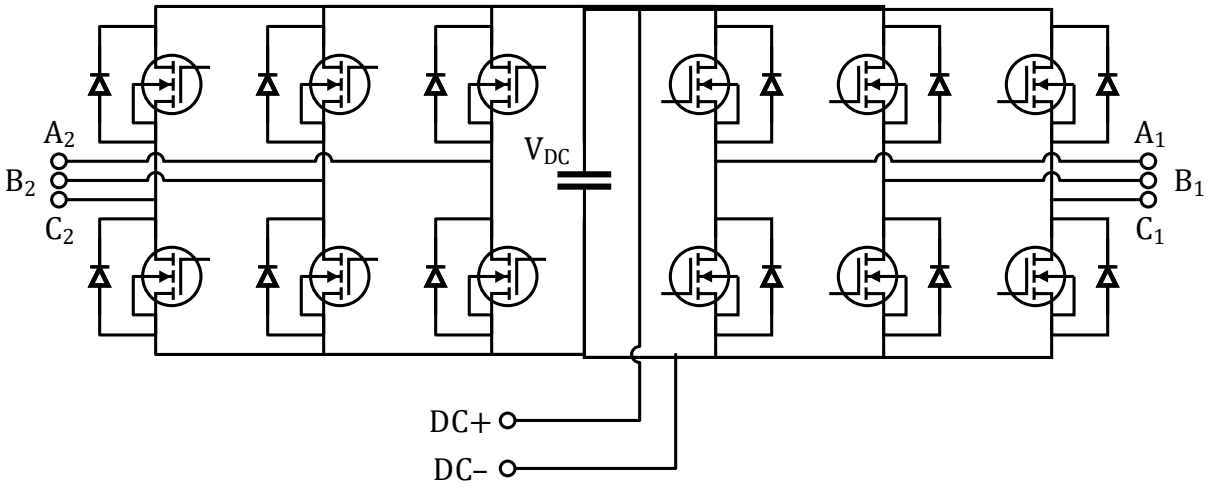


Figure 4.3: Dual three-phase bridge converter circuit topology internal to the testbed ESCs.

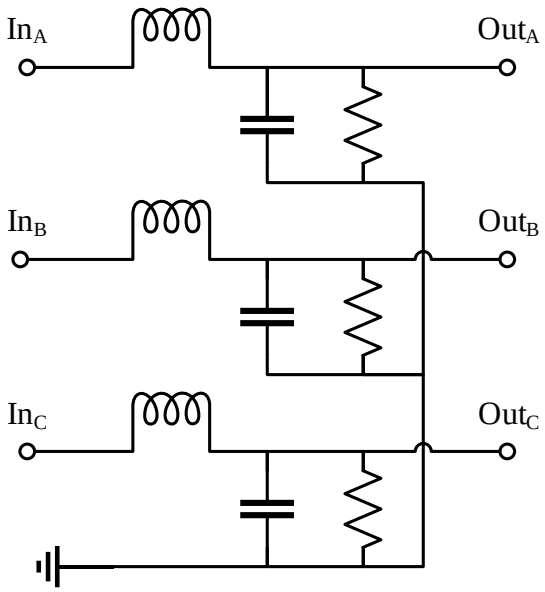


Figure 4.4: Three parallel low-pass RLC filter circuits internal to the testbed filter boxes.

relay and fuse are integrated to protect the battery pack and a shunt resistor is used to measure the pack current.

As mentioned in Chapter 2, the genset consists of an internal combustion engine, starter/generator (S/G), and the hybrid speed controller. The internal combustion engine is a Briggs and Stratton 19N1 engine rated at ∼7kW. An internal governor regulates a constant engine speed determined by a lever that cannot be adjusted during a mission at present. The

Table 4.2: Battery pack information and operating limits.

Chemistry	Lithium Nickel Cobalt Aluminum Oxide
Cell Type	Samsung INR18650-30Q
Pack Size	16S7P
Voltage Limits (Min-Max)	40-67.2V
Max Continuous Discharge Current	105A
Standard Charge Current	10.5A
Max Charge Current	28A
Rated Capacity	21Ah
Measured Pack Capacity	Discharge: 21.33Ah Charge: 22.38Ah

governor is proprietary to the manufacturer and is not described in this thesis. The engine is connected to the S/G by a belt-pulley system. The S/G is a 16 pole outrunner brushless DC motor (BLDC) developed by Neu Motors. The S/G is controlled as a starter in speed control mode when the engine is "off". When the engine is "on", the S/G is controlled as a generator in torque control model. Since the governor regulates the shaft speed, it is necessary that the S/G switches to a torque control mode so that the two controllers do not fight to regulate the same state. One branch of the hybrid ESC controls the S/G. Depending on the engine operating state, the three-phase bridge converter will operate as an inverter or active rectifier.

The drivetrain consists of a propulsion (prop) motor and the hybrid ESC. The propulsion motor is a 16 pole outrunner brushless DC motor developed by Neu Motors. The second branch of the hybrid ESC operates as an inverter in speed control mode to regulate the prop's shaft speed.

The braking subsystem consists of a dynamometer (dyno or brake), brake speed controller, a filter box, high power load bank, low power battery pack, and power supply. The dyno is a 16 pole outrunner brushless DC motor (BLDC) developed by Neu Motors and is coupled to the prop motor by a shaft coupling. The dynamometer is controlled by one branch of the load ESC operating as an active rectifier in torque control mode. Together, the dynamometer

and ESC are used to emulate the propeller load of an actual aircraft. The second branch of the load ESC is a buck-boost converter. The control objective of the buck-boost converter is to draw as much DC current as the dyno is producing. In other words, the DC-link bus DC current should be identically zero. The high power load bank is composed of twelve 1Ω power resistors configured in a 2S6P pattern to generate a $\frac{1}{3}\Omega$ equivalent resistance. The low power battery pack and power supply provide DC power to the ESC. The low power battery, a 5S1P pack of lead-acid batteries, increases the capacitance of the DC-link inside the ESC to prevent voltage fluctuations as a result of large current spikes. The power supply will provide power if the battery voltage starts to decrease. A protection diode blocks current flow into the power supply and a shunt resistor measures the battery current.

The voltage steps (or DCDC) subsystem is composed of a load, DCDC speed controller, and filter box. Together, the voltage step ESC and filter box create three independent bi-directional buck-boost converters. In this thesis, a programmable electronic load is connected to one of the converter outputs to emulate aircraft avionics. Shunt resistors are used to measure the DC current draw of the DCDC speed controller and the current output of each buck-boost converter.

The testbed is directly controlled by LabVIEW running on a desktop PC with a 4.2GHz Intel i7 Processor and 16GB of RAM. Each ESC provides measurements or estimates of internal states (e.g. voltages, currents, speeds, etc.). External shunt measurements are recorded using a National Instruments CompactDAQ (cDAQ) with NI-9205 and NI-9403 cards installed. At a rate of 10Hz, LabVIEW communicates with the ESCs via a Controller Area Network (CAN) bus, the BMS via RS-232, the cDAQ via USB, and MATLAB via the User Datagram Protocol (UDP). An example of the testbed control GUI is shown in Figure 4.5. A complete parts list and computer specifications are provided in Appendix D.

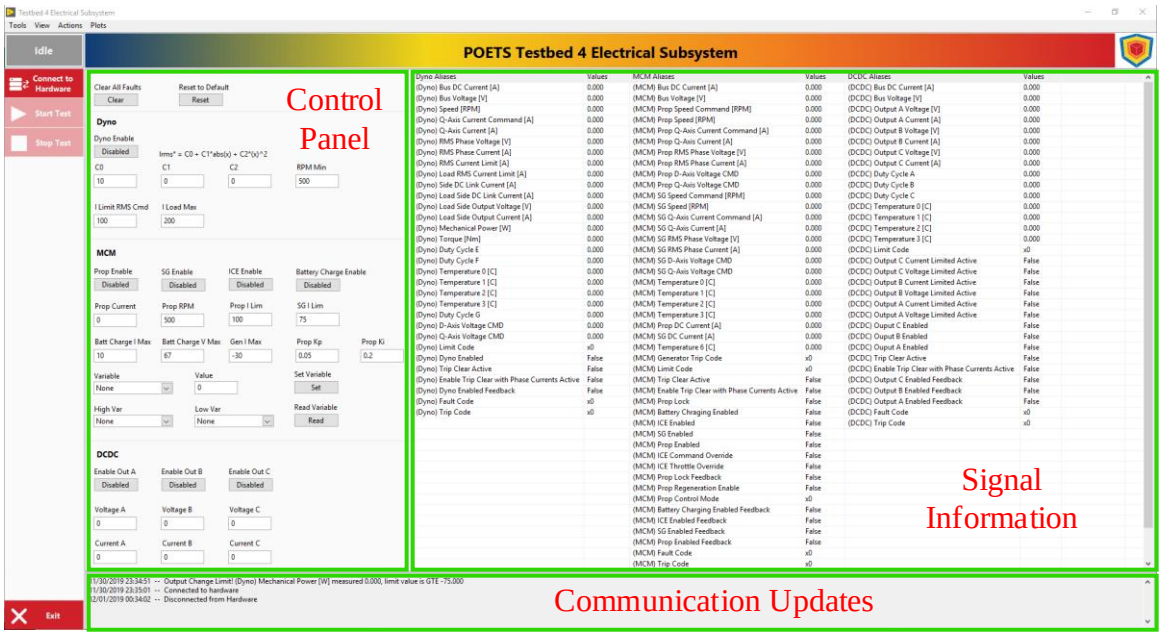


Figure 4.5: Example of the LabVIEW GUI used to control the testbed.

4.3 Battery Pack Validation

4.3.1 Experimental Setup

This thesis identifies the electrical battery parameters using a current pulse test [8]. The concept of this test is to excite the battery dynamics using a current pulse and then analyze the resulting relaxation of the voltage dynamics once the pulse is removed. A schematic of the experimental setup is outlined in Figure 4.6 and the test equipment is shown in Table 4.3. The BMS measures pack voltage, but redundant voltage measurements are also reported by the power supply and electronic load via voltage sense pins connected to the battery terminals. System actuation and data acquisition is handled by LabVIEW, which samples the system at 1Hz.

Conditioning cycles are used to relax the battery pack voltage dynamics before the start of the current pulse test. The following steps were used to condition the battery pack. Note that charge rates are indicated in terms of *C-rate*. The C-rate is the charge/discharge rate defined by the battery current divided by the current required to fully charge/discharge the battery in one hour. For example, if a battery has a 5Ah capacity and is being charged at

1A, its C-rate is C/5 and it would take 5 hours to charge from empty to full.

Conditioning Cycle Procedure:

1. Start the pack at full charge.
2. Connect the electronic load and start a constant current-constant voltage (CC-CV) discharge cycle at a C/21 rate (1A) and 40V.
3. Stop the CV portion of the discharge cycle by disconnecting the load when the pack current is less than 100mA and let the system rest for 3 hours.
4. Connect the power supply and start a CC-CV charge cycle at a C/21 rate and 67.2V.
5. Stop the CV portion of the charge cycle by disconnecting the power supply when the pack current is less than 100mA and let the system rest for 3 hours.
6. Repeat steps (2)-(5)

The CV voltage limits correspond the voltage limits of the battery pack and the 100mA cut-off current is recommended by the manufacturer. The current pulse test should directly follow the conditioning cycles so that the battery dynamics are not re-excited. The current pulse waveform described in the test procedure below has magnitude 7A, period 189min, and duty cycle $\sim$0.048.

Current Pulse Test Procedure:

1. Start the pack at full charge, begin data acquisition, and let the pack rest for 30 minutes.
2. Connect the electronic load and discharge the pack at C/3 for 9 minutes.
3. Disconnect the electronic load and let the pack dynamics relax for 180 minutes.
4. Repeat steps (2)-(3) until the pack voltage reaches 40V.
5. Start a 40V CV discharge cycle.

6. Stop the CV portion of the discharge cycle by disconnecting the load when the pack current is less than 100mA and let the system rest for 240 minutes.

7. Repeat steps (2)-(6) except using the power supply to charge the pack. Step (5) should read "Start a 67.2V CV charge cycle".

Note that the sign of the current changes depending on whether the battery is being charged or discharged. The physical test setup is shown in Figure 4.7. Because a test chamber was not available, the battery was tested in a 3ft pit for safety. A thermal camera provided temperature feedback to check safe thermal operation, but was used only for safety precautions. No thermal validation efforts are presented in this thesis.

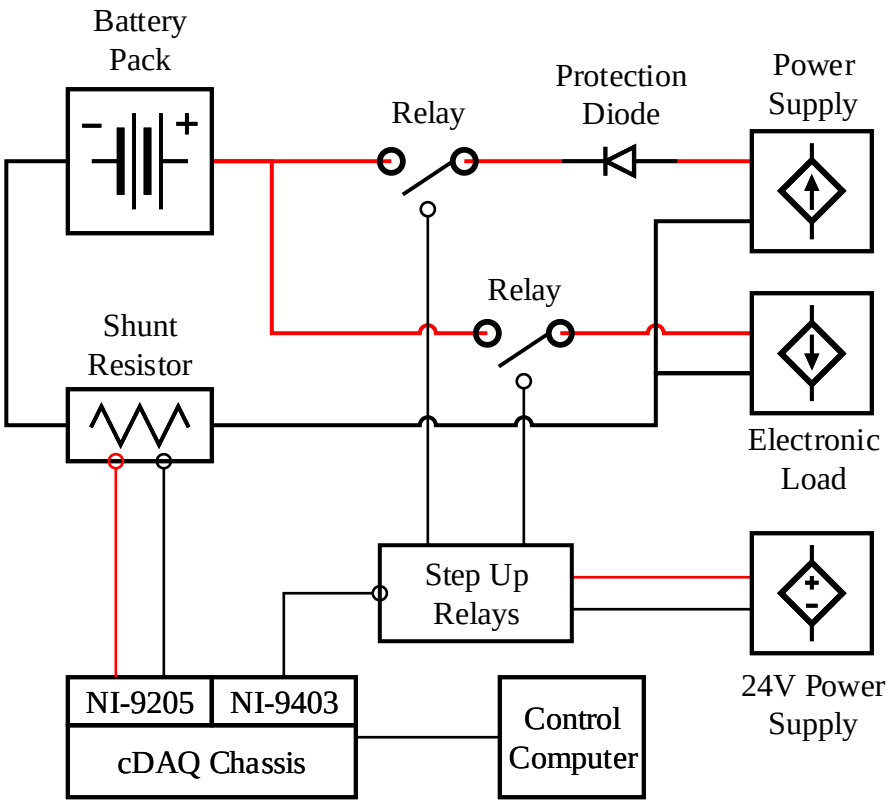


Figure 4.6: Battery pulse test circuit schematic.

Table 4.3: Current pulse test equipment.

Item	Part Used	Purpose
Battery	16S7P Battery Pack	This is the device under test.
Shunt Resistor	Rideon RSA-20-100	Measures pack current.
Power Supply	Magna-Power XR400-10.0	Controls the battery pack charge cycle.
Electronic Load	Hewlett Packard 6050A	Controls the battery pack discharge cycle.
24V Power Supply	BK Precision 9129B	Powers the step-up relays.
Relay (x2)	Potter and Brumfield PRD-3DJ0-24	Enables switching between charge a discharge cycles.
Protection Diode	Vishay VS-T40HF-10	Prevents current flow into the power supply.
Step Up Relays (x2)	Winford Relay Board RLY204	Steps up voltage for the relays.
DAQ	cDAQ Chassis with 9205 and 9403 Cards	Data acquisition for the shunt resistor and controls the step up relays.
BMS	TI Evaluation Board bq76PL455A-Q1	Measures pack voltage and monitors cell health.

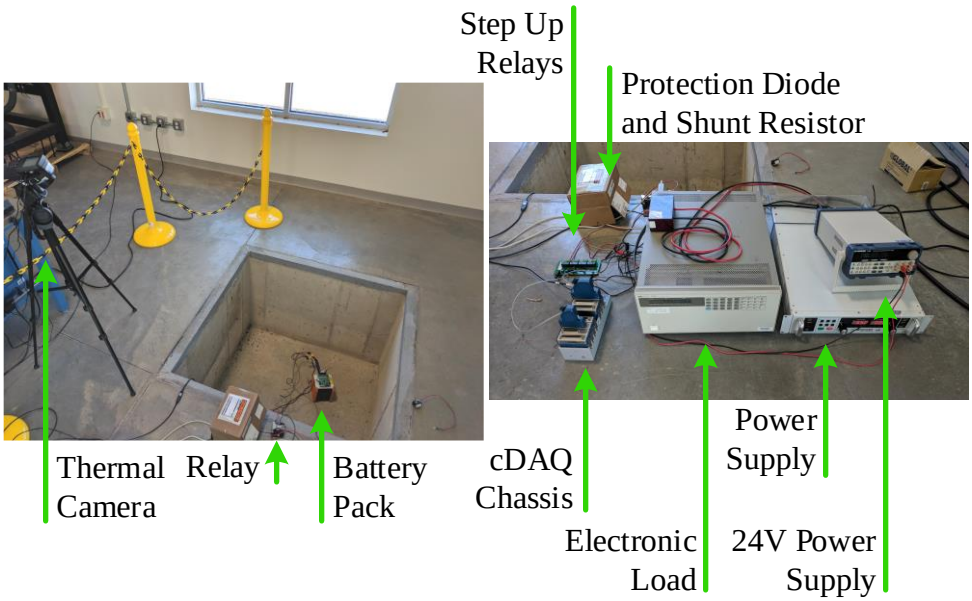


Figure 4.7: Battery pulse test physical setup of Figure 4.6.

4.3.2 Data Analysis Methods

As defined in Section 2.3.1, the battery has 7 electrical parameters to identify: V_{ocv}, Q, R_s, R_1, R_2, C_1, and C_2. The total capacity is taken as the integral of the pack current

$$Q = \int_0^T I_1 dt, \tag{4.1}$$

where I_1 is the pack current and T is the total test time. The battery OCV curve is obtained as the steady-state voltage value of each relaxation period. In practice, approximately the last 60 data points of each relaxation period were averaged to determine the open circuit voltage at each SOC. The series resistance R_s was determined using Ohm's Law

$$R_s = \Delta V / I_1. \tag{4.2}$$

Here I_1 is the magnitude of the current pulse and ΔV is the instantaneous pack voltage change that results from the current step change. Figure 4.8 shows a single pulse with the OCV and instantaneous voltage change labeled.

The RC pair values are obtained using optimization methods. Assuming the voltage

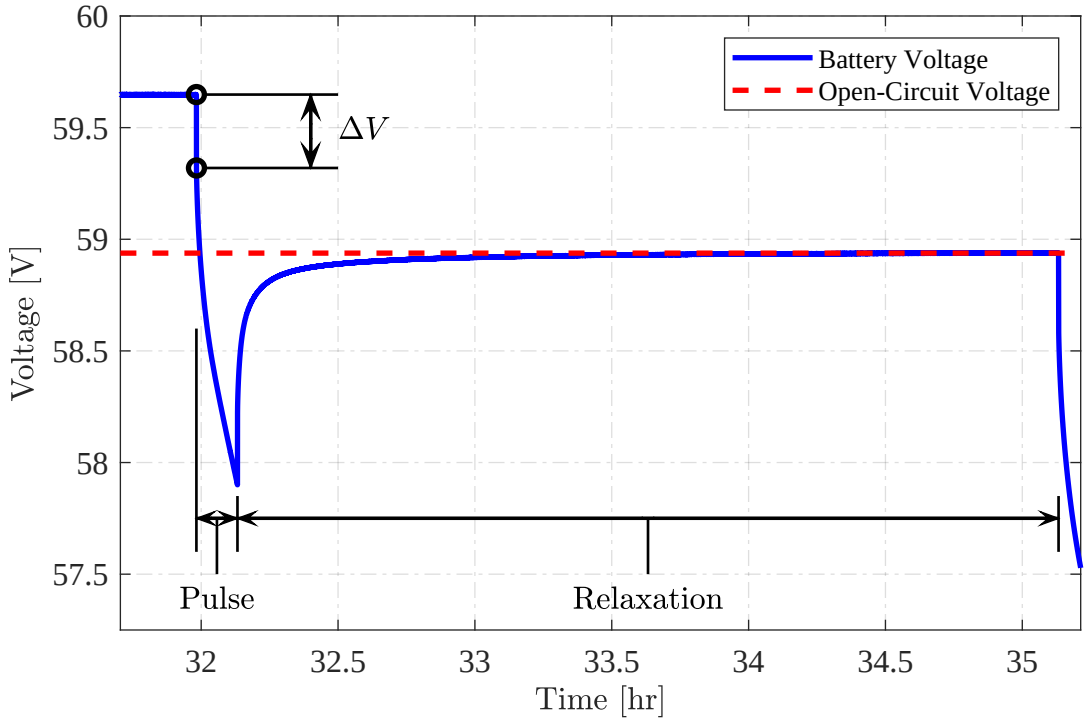


Figure 4.8: The battery voltage resulting from a single current pulse and relaxation period. The instantaneous voltage change ΔV, open circuit voltage V_{ocv} are labeled, and pulse and relaxation periods are labeled.

dynamics of the RC pairs are zero by the end of the relaxation periods, the voltage state of the first RC pair at the end of a current pulse t_{pulse} is given by

$$V_1(t_{pulse}) = I_1 R_1 \left(1 - e^{\frac{-t_{pulse}}{R_1 C_1}}\right). \tag{4.3}$$

Using (4.3) as an initial condition, the voltage state of the first RC pair during the relaxation period is given by

$$V_1(t) = I_1 R_1 \left(1 - e^{\frac{-t_{pulse}}{R_1 C_1}}\right)\left(1 - e^{\frac{-t}{R_1 C_1}}\right). \tag{4.4}$$

These trajectories can be solved using frequency domains methods or by the state transition matrix. Since the RC pair voltages share the same dynamics, the relaxation voltage trajectory V_{relax} is

$$V_{relax}(t) = I_1 R_1 \left(1 - e^{\frac{-t_{pulse}}{R_1 C_1}}\right)\left(1 - e^{\frac{-t}{R_1 C_1}}\right) + I_1 R_2 \left(1 - e^{\frac{-t_{pulse}}{R_2 C_2}}\right)\left(1 - e^{\frac{-t}{R_2 C_2}}\right), \tag{4.5}$$

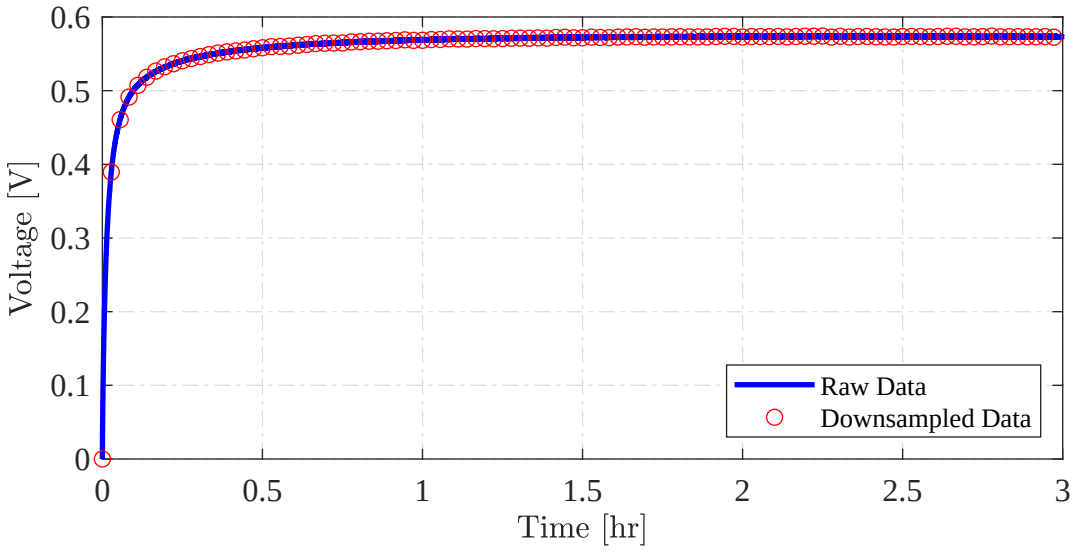


Figure 4.9: A comparison of downsampled and raw data for the relaxation voltage resulting from a current pulse. The instantaneous voltage change has been removed and the response has been moved to the origin.

To simplify the optimization problem, the data is downsampled and the relaxation periods resulting from each current pulse are isolated from the rest of the data set and moved to the origin (Figure 4.9).

The nonlinear grey-box model estimation tool in the System Identification Toolbox in MATLAB was used to identify the RC pair values. The optimization problem is setup as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{k=1}^{N} \left(V_{data}[k] - V_{relax}[k]\right)^2 \\ \text{subject to} \quad & \text{Equation 4.5,} \\ & \underline{x} \leq x \leq \bar{x}. \end{aligned} \tag{4.6}$$

where V_{data} is experimentally obtained data, $x = \begin{bmatrix} R_1 & R_2 & C_1 & C_2 \end{bmatrix}$, and $\underline{x}$ and $\bar{x}$ are highly conservative upper and lower bounds for each parameter used to constrain the search space. For example $\underline{x} = 0$ and $\bar{x} = 10^7$.

4.3.3 Results

The measured pack capacity between charge and discharge cycles are shown in Table 4.2. These values are slightly greater than the value reported by the battery datasheet. The differentiation in measured capacity is likely a result of the BMS, which is powered by the battery pack. Although the instantaneous BMS power draw is small, the total energy demand over a three day test may be significant.

The pack open circuit voltage and internal resistance curves are shown in Figure 4.11. The shape of the OCV curve is expected for this particular pack chemistry. Similarly, the exponential decay of the internal resistance as a function of SOC is expected as reported elsewhere in the literature [8].

The RC pair values for pack are shown in Figure 4.11. The general shape of the resistance and capacitance curves is somewhat consistent with results of another cell in the literature [8]. Some work has characterized the R and RC pair values as exponential or polynomial functions. Here, the OCV, R, and RC pair values are left as look-up tables (approximate linear behavior between samples). Two comparisons between the model and experimental data are shown Figure 4.10. By observation, the model matches the data well.

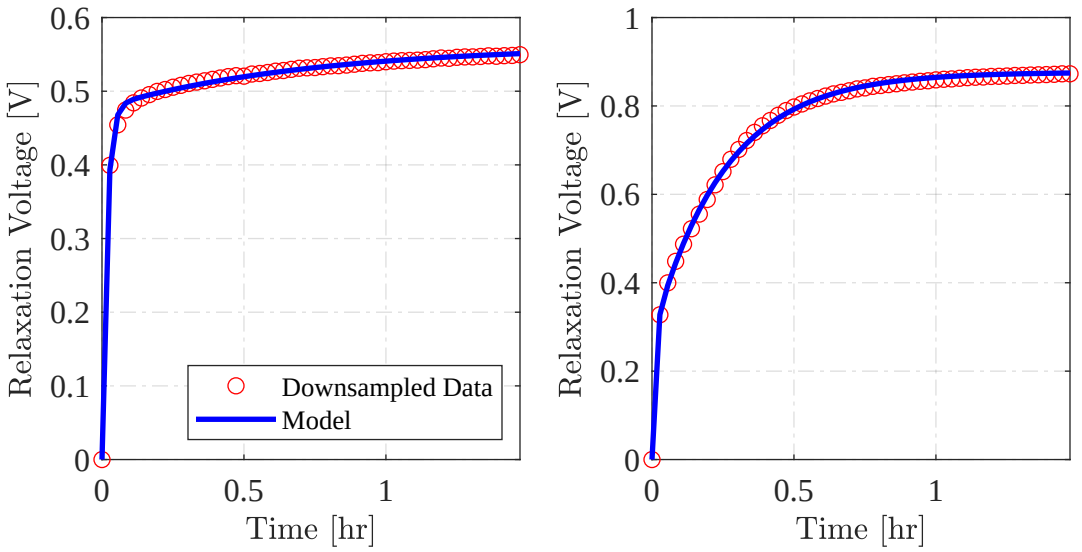


Figure 4.10: Two comparisons of the downsampled experimental data and model during a relaxation period.

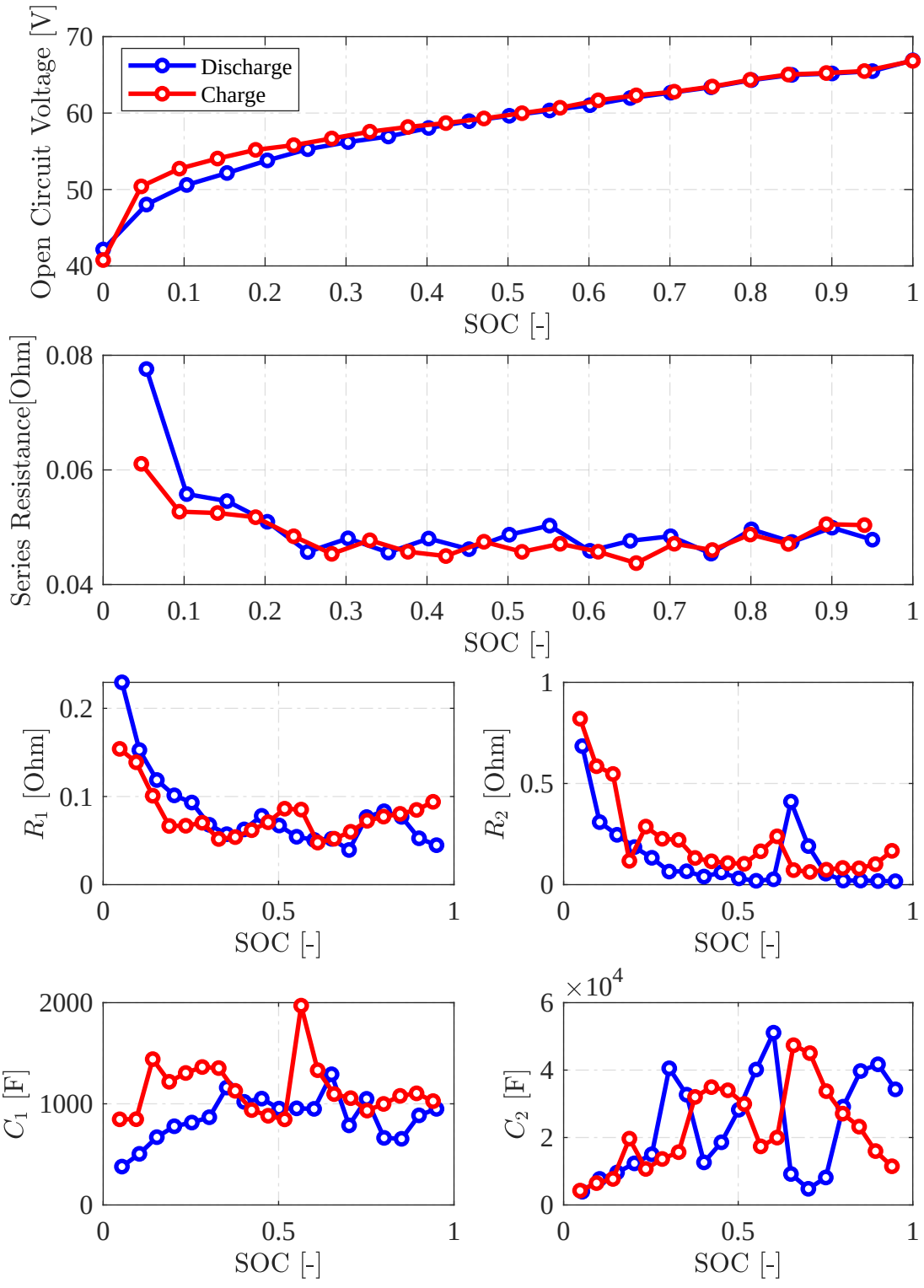


Figure 4.11: The battery open circuit voltage, internal resistance, and RC pair values as a function of state of charge for charge and discharge cycles.

4.4 Motor Validation

This section will describe the 4 tests used to validate the BLDC motor model: the voltage step, backdrive, friction, and coastdown tests [62]. Note that the following results are identified for a Δ wound motor. If characterizing a Y wound motor, the same procedures can be used with slight modifications to the analysis. Similarly, this set of tests can also be used to characterize a brushed DC motor. The identified motor parameters are summarized in Table 4.4.

Table 4.4: Propulsion motor and drivetrain parameters.

Propulsion Motor		
Parameter	Value	Standard Deviation
Coil Resistance $[m\Omega]$	29.2	0.26
Coil Inductance $[\mu H]$	48.7	0.80
Motor Constant $[mNm/A]$	123.8	0.53

Drivetrain (Propulsion Motor, Coupling, and Dynamometer Motor)		
Parameter	Value	Standard Deviation
Viscous Friction Constant $[Nm - s/rad]$	0.360×10^{-3}	–
Static Friction Constant $[Nm]$	0.228	–
Shaft Inertia $[kg/m^2]$	2.73×10^{-3}	0.08×10^{-3}

4.4.1 Voltage Step Test

4.4.1.1 Experimental Setup

The voltage step test is used to identify the motor's coil resistance R and inductance L as described in Section 2.3.2. Because the motor windings are modeled as RL branches, the concept of this test is to excite only the electrical dynamics and analyze the first-order dynamic response. The experimental setup schematic is shown in Figure 4.12 and necessary test equipment is listed in Table 4.5. When choosing components, the resistor value should be of the same order of magnitude as the expected motor resistance, which can be estimated using a ohmmeter. The relay should have a fast switch-open time such that the motor dynamics are not impacted by relay's time constant. Lastly, the motor shaft should be braked during this test to isolate the motor's electrical dynamics. Fixing the motor shaft can be achieved by mechanically braking the shaft or aligning the magnetic poles of the motor stator and rotor such that a DC current does not rotate the shaft. The pole alignment method was done in this thesis. The voltage step test procedure is defined below.

Procedure:

1. Connect the resistor across two of the motor terminals as shown in Figure 4.12.
2. Set the power supply current limit to 3A, close the relay, and let the current dynamics reach steady state.
3. Open the relay and use the oscilloscope to capture and save the resulting current waveforms.
4. Repeat steps (2) and (3) for current limits $I_{lim} = [3 : 8]$A in 1A increments (6 trials).
5. Repeat steps (2)-(4) for each connection combination between resistor and motor terminals. There are 3 combinations in total (18 total trials).

Note the power supply was set in current limiting mode with small current limits to prevent damage to the motor coils. The physical test setup is shown in Figure 4.13.

Table 4.5: Voltage step test equipment.

Item	Part Used	Purpose
Test Motor	Propulsion Motor Neu 8038-105	This is the device under test.
Shunt Resistor	0.01Ω Resistor	Used to measure motor terminal voltage.
Power Supply	Kiethley 2260B-80-27	Provides power to the motor.
Relay	Potter and Brumfield T9AP5D52-24	Used to create a step in terminal voltage.
Oscilloscope	Tektronix MSO 4034B	Reads and saves current waveforms.
Current Probes (x2)	Tektronix TCP0030A	Measures branch currents.

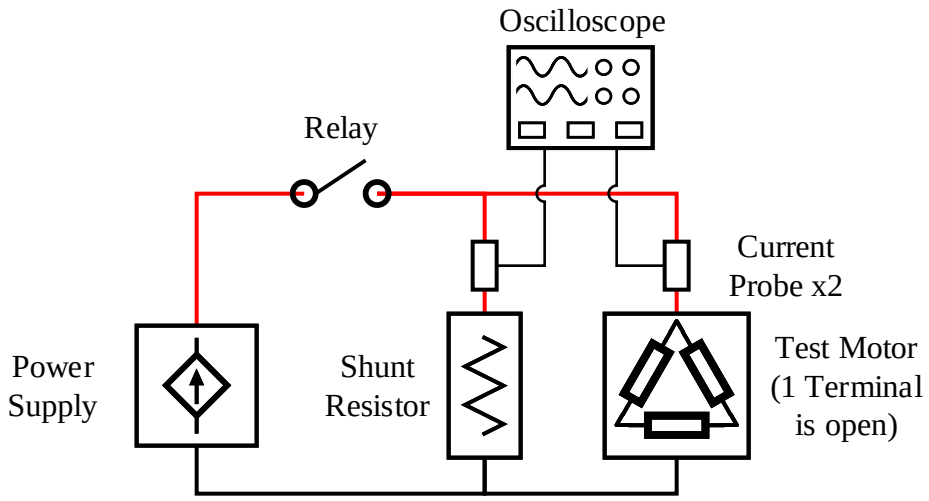


Figure 4.12: Wiring diagram for the motor voltage step test.

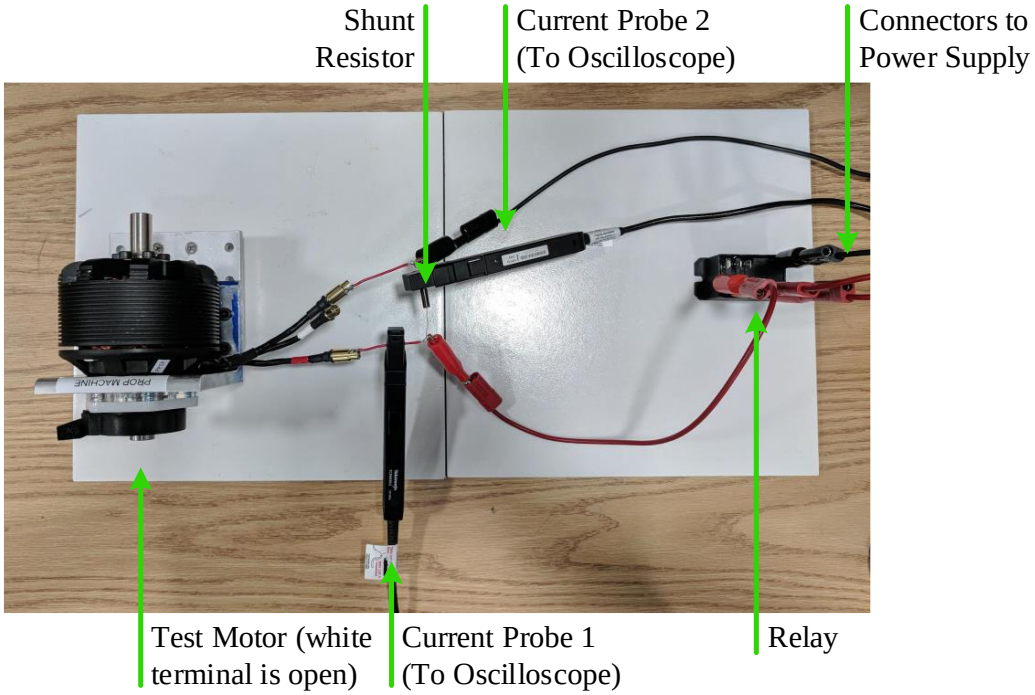


Figure 4.13: The physical test setup for the motor voltage step test of Figure 4.12.

4.4.1.2 Data Analysis Methods

The motor resistance and inductance values are determined using the grey-box model estimation tool in MATLAB. Since the motor shaft does not rotate during this test, the circuit current dynamics I Figure (4.14) are given by

$$\frac{2}{3}L\dot{I} = V - \left(\frac{2}{3}R + R_s\right) I \tag{4.7}$$

where V is the voltage across the shunt resistor R_s. The shunt resistor voltage is calculated using the current measurement and Ohm's law. The grey-box model estimation optimization problem is setup as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{k=1}^{N} \left(I_{data}[k] - I[k]\right)^2 \\ \text{subject to} \quad & \text{Equation 4.7}, \\ & \underline{x} \leq x \leq \bar{x}, \end{aligned} \tag{4.8}$$

where I_{data} is experimentally obtained current data from the current probe connected to the motor terminals, $x = \begin{bmatrix} R & L \end{bmatrix}$, and $\underline{x}$ and $\bar{x}$ are conservative upper and lower bounds for each parameter used to constrain the search space.

4.4.1.3 Results

A comparison between the model and experimental data is shown in Figure 4.15 and, by observation, the quality of the model is sufficient for control design. The average and standard deviation for the resulting coil resistance and inductance are given in Table 4.4.

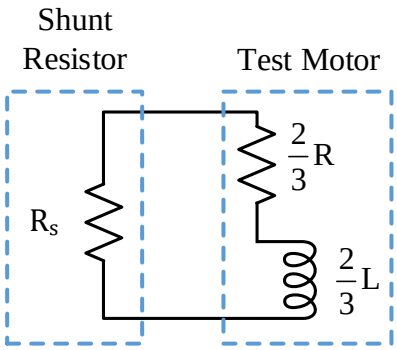


Figure 4.14: Equivalent circuit for the voltage step test.

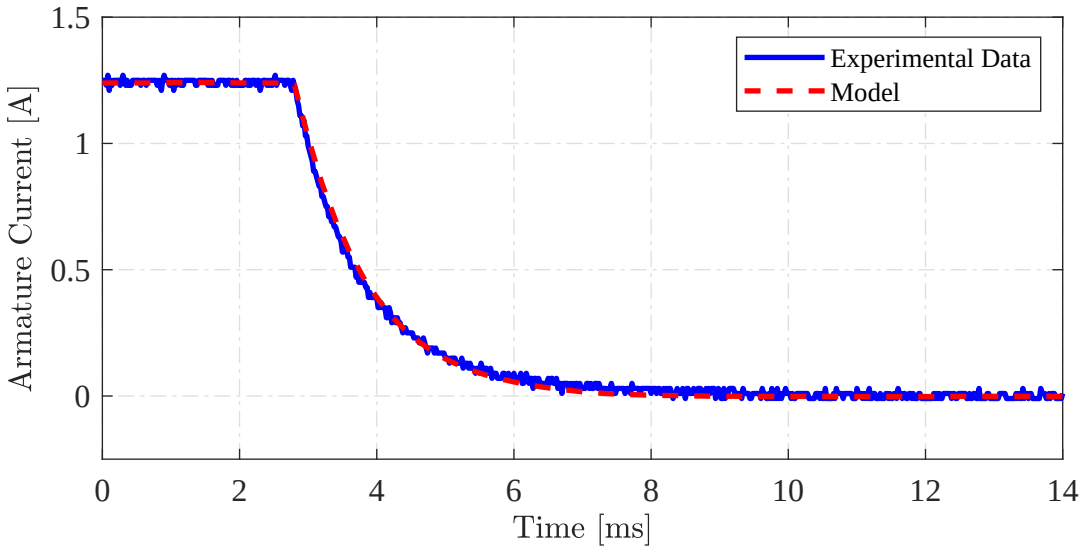


Figure 4.15: A comparison of experimental and model line currents from the voltage step test.

Table 4.6: Backdrive test equipment.

Item	Part Used	Purpose
Test Motor	Propulsion Motor Neu 8038-105	This is the device under test.
Drive Motor	Dynamometer Motor Neu 8038-105	Spins test motor at a constant speed.
Coupling	ZeroMax SC055R	Mechanically connects test and drive motor shafts.
Speed Controller	Hybrid ESC	Spins drive motor at constant speed.
Power Resistors (x3)	$200W$ 50Ω Resistors	Used to measure motor terminal voltage.
Power Supply	Kiethley 2260B-80-27	Provides power to the speed controller.
Oscilloscope	Tektronix MSO 4034B	Reads and saves voltage waveforms.
Voltage Probe	Tektronix THDP0200	Measures terminal voltages.

4.4.2 Backdrive Test

4.4.2.1 Experimental Setup

The backdrive test is used to identify the machine's motor constant k_v. The concept of this experiment is to spin the test motor at a constant speed and measure the generated voltage. The experimental setup schematic is shown in Figure 4.16 and necessary test equipment is provided in Table 4.6. Note that the second voltage probe in Figure 4.17 is a redundant measurement. The test motor and drive motor are mechanically coupled using a shaft coupling. The power resistors should be rated for the expected power generated by the test motor. The experimental procedure for the backdrive test is provided below.

Procedure:

1. Turn on the power supply.

2. Command the drive motor to 500rpm and let the shaft speed reach steady state.

3. Use the oscilloscope to capture and save the voltage waveform generated by the test motor.

4. Repeat steps (2) and (3) for steady state speed values $\omega_{ss} = [500 : 5000]$rpm in 500rpm increments (10 trials).

4.4.2.2 Data Analysis Methods

The motor constant is calculated by

$$k_v = p\frac{\sqrt{6}}{2}\frac{V_{pk}}{\omega_e}, \tag{4.9}$$

where p is the number of motor poles pairs, V_{pk} is the measured peak voltage across the power resistor, and ω_e is the electrical frequency of the measured voltage signal. The number of pole pairs can be obtained from a datasheet or calculated as the nearest whole number ratio between electrical and mechanical motor frequencies. The electrical frequency can be calculated by hand or a Fourier analysis. Note that $\frac{V_{pk}}{\omega_e}$ is commonly referred to as the motor's flux linkage [38].

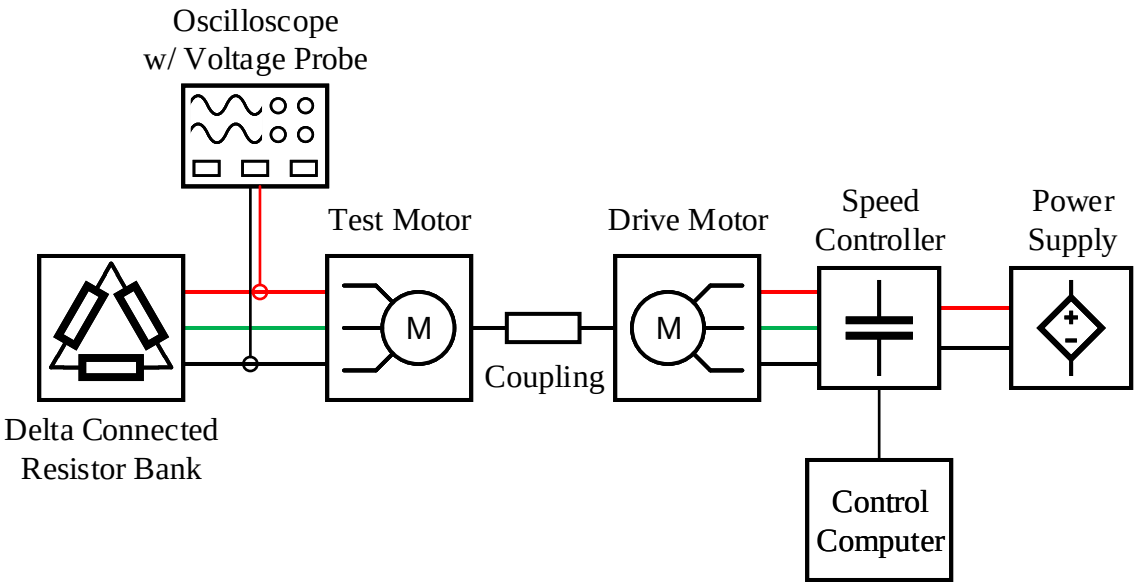


Figure 4.16: Experimental setup for the motor backdrive test.

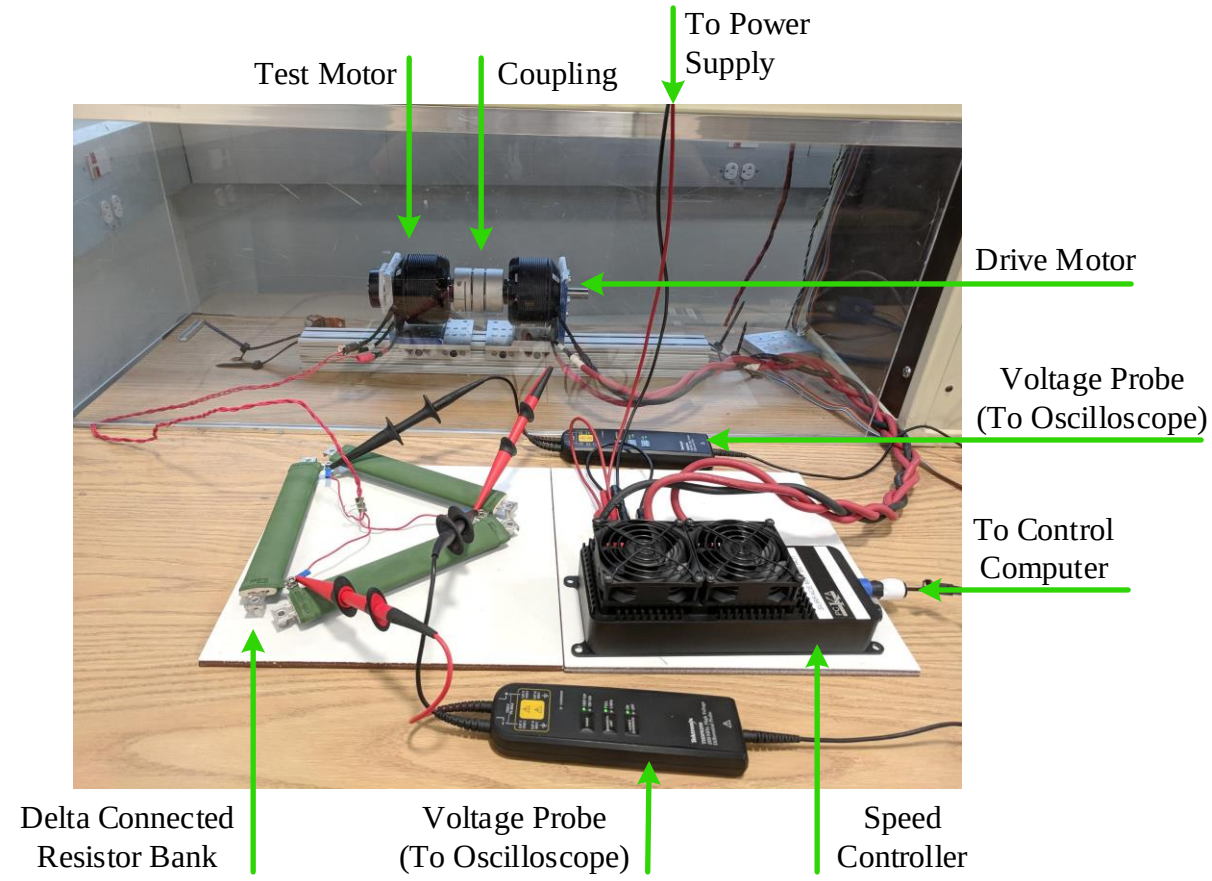


Figure 4.17: Physical experimental setup for the backdrive test of Figure 4.16.

4.4.2.3 Results

The results of a single trial is shown in Figure 4.18. The average motor constant and standard deviation between all trails is provided in Table 4.4. By observation of the standard deviation, the results are consistent across all trails.

4.4.3 Friction Test

4.4.3.1 Experimental Setup

The friction test is used to identify the machine's static and viscous friction coefficients c and b. Since the propulsion motor is coupled to the dynamometer on the testbed, this test actually characterizes the total friction on the shaft. The concept of this test is to drive

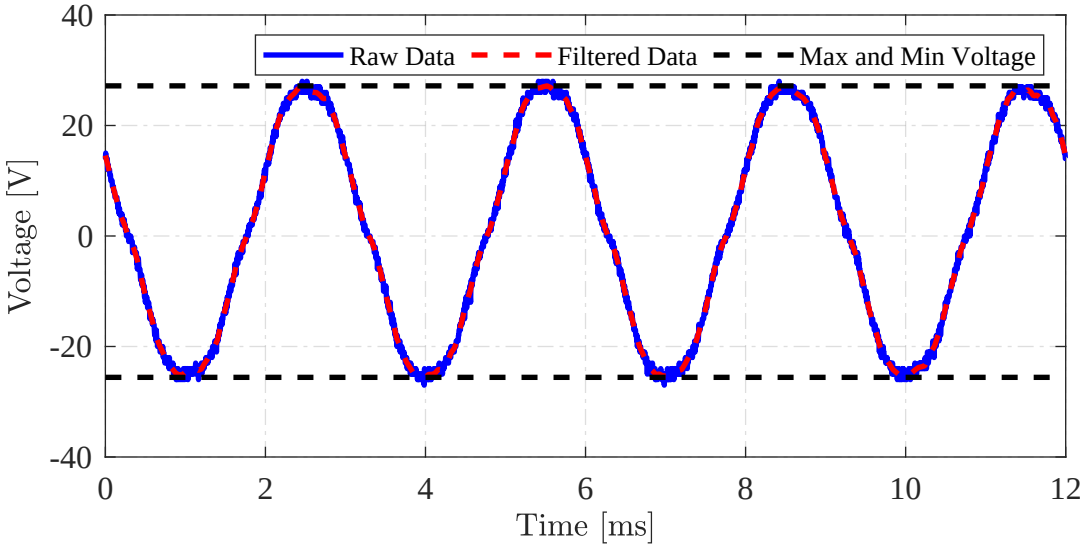


Figure 4.18: The resulting generated voltage waveform from the backdrive test for a single trial. The maximum and minimum generated voltages are labeled.

the motor and various shaft speeds with no applied load. In this case, friction is the only resistive force acting on the shaft. The experimental setup schematic is shown in Figure 4.19 and the necessary test equipment is listed in Table 4.7. LabVIEW records the q-axis motor current I_q and rotor speed ω reported by the hybrid ESC. The friction test procedure is provided below.

Procedure:

1. Connect the battery and start data acquisition.
2. Command the test motor to 2000rpm and let the system reach steady state.
3. Collect steady state data for 10 seconds.
4. Repeat steps (2) and (3) for steady state speed values $\omega_{ss} = [2000 : 4750]$rpm in 250rpm increments (12 trials).

Note that a more robust test procedure would use a power supply in place of a battery pack. However, when the system is under no mechanical load, the pack current draw is small and the battery essentially operates as a constant voltage source.

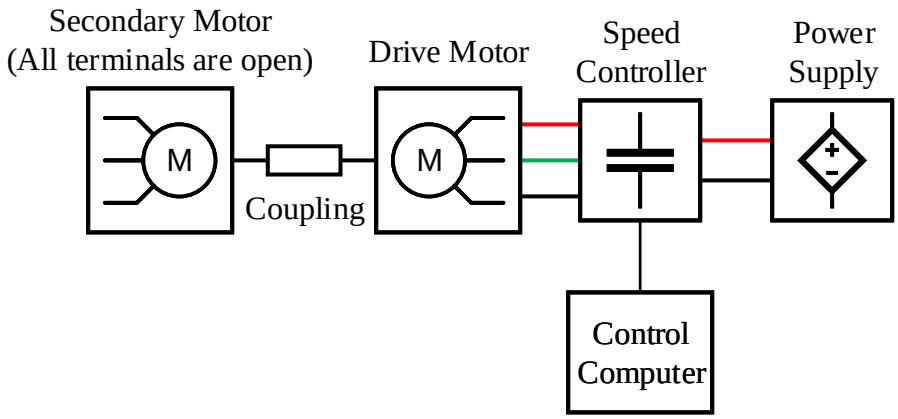


Figure 4.19: Experimental setup for the motor friction test.

Table 4.7: Friction test equipment.

Item	Part Used	Purpose
Test Motors	Propulsion and Dynamometer Motors Neu 8038-105	This is the system under test.
Coupling	ZeroMax SC055R	Mechanically connects test motors shafts.
Speed Controller	Hybrid ESC	Spins drive motor at a constant speed.
Power Supply	Battery Pack	Provides power to the speed controller.

4.4.3.2 Data Analysis Methods

At steady state, under no external mechanical load, (2.25a) simplifies to

$$k_v I_q = b\omega + c. \tag{4.10}$$

Because the motor constant, q-axis current, and rotor speed are known/measured, the friction coefficients can be calculated using a linear fit of torque ($k_v I_q$) versus rotor speed. The viscous friction coefficient is the slope of the fit and the static friction coefficient is the y-intercept of the fit.

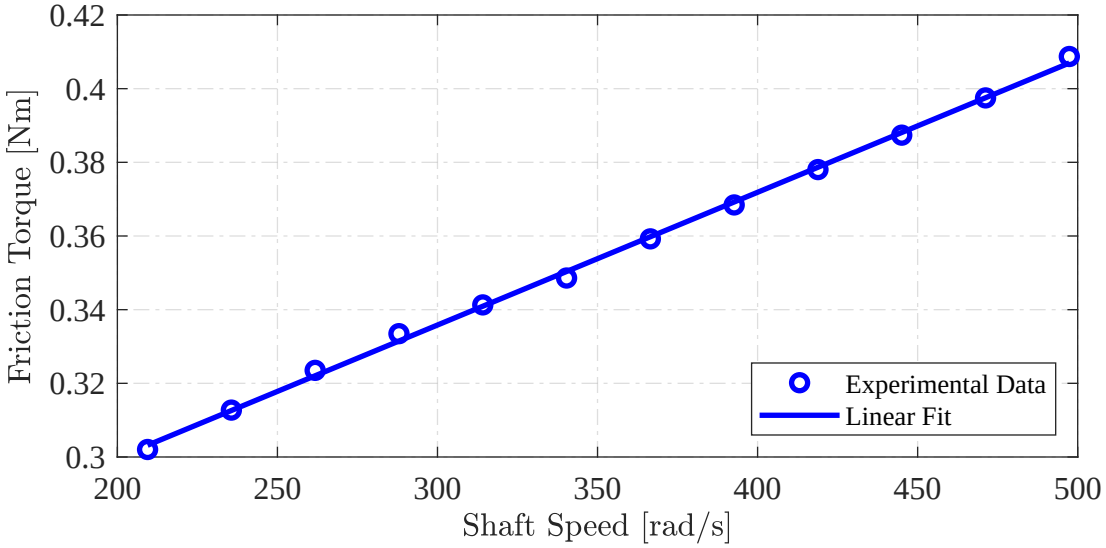


Figure 4.20: Friction torque versus shaft speed from the friction test.

4.4.3.3 Results

The resulting fit of torque versus speed data is shown in Figure 4.20 and by observation the fit is good. The viscous and static friction coefficients are reported in Table 4.4.

4.4.4 Coastdown Test

4.4.4.1 Experimental Setup

The coastdown test characterizes the total inertia J of both electric machines and the shaft coupling. In comparison to the voltage step test, the concept of the coastdown test is to isolate the mechanical dynamics by electrically disconnecting the motor. The experimental setup schematic is shown in Figure 4.21 and the necessary test equipment is listed in Table 4.6. LabVIEW reads the encoder output to report motor position and speed data. The test procedure is outlined below.

Procedure:

1. Connect the test motor to the speed controller via the quick disconnects.
2. Start data acquisition and command the motor to 1500rpm. Let the system reach steady state speed.
3. Once at steady state, simultaneously disconnect the three motor cables from the speed controller. In the setup described in Figure 4.21, the banana cables were aggressively pulled apart to create a clean step change.
4. Let the motor speed decrease to 0rpm, end data acquisition, and save the trial data.
5. Repeat steps (1)-(4) for steady state speed values $\omega_{ss} = [1500 : 4500]$ rpm in 500 rpm increments (7 trials).

Table 4.8: Coastdown test equipment.

Item	Part Used	Purpose
Test Motors	Propulsion and Dynamometer Motors (Neu 8038-105)	This is the system under test.
Coupling	ZeroMax SC055R	Mechanically connects test motors shafts.
Speed Controller	Hybrid ESC	Spins drive motor at constant speed.
Power Supply	Battery Pack	Provides power to the speed controller.
Rotary Encoder	US Digital E3-1024-625-IE-H-D-B	Measures shaft speed.
Quick Disconnects	3 Banana Cables	Used to electrically disconnect the drive motor.
DAQ	cDAQ Chassis with 9403 Card	Reads the encoder output.

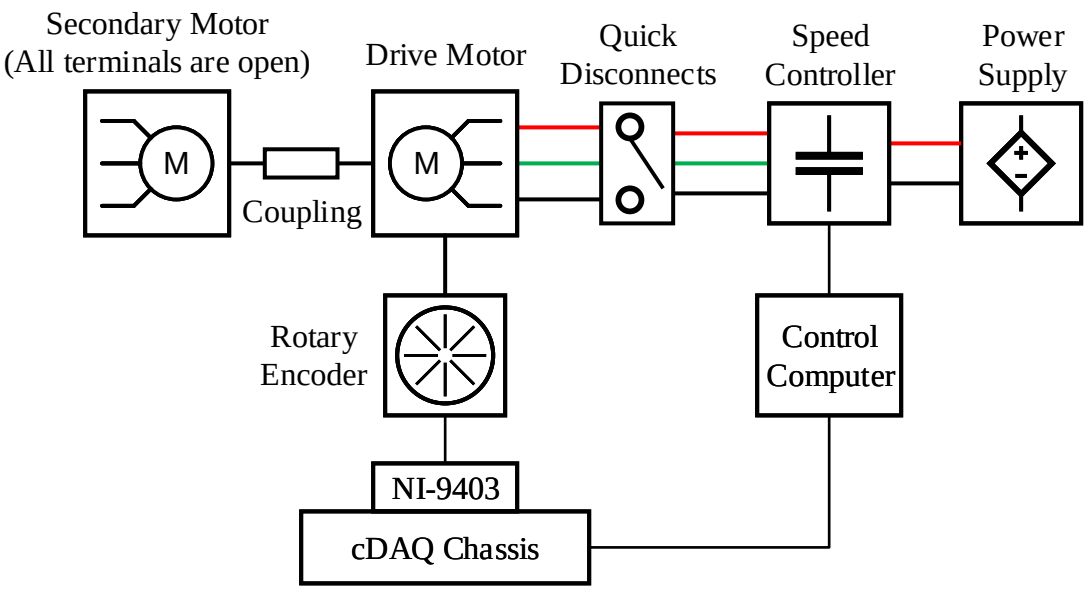


Figure 4.21: Experimental setup for the motor coastdown test.

4.4.4.2 Data Analysis Methods

The resulting angular speed dynamics ω when the motor terminals are opened at time t_0 is given by (4.11a). Using integration or frequency domain methods, the resulting speed trajectory is provided by (4.11b).

$$J\dot{\omega} = -b\omega - c, \quad \omega(t_0) = \omega_0, \tag{4.11a}$$

$$\omega = (\omega_0 + \frac{c}{b})e^{-\frac{b}{J}t} - \frac{c}{b}, \tag{4.11b}$$

$$\ln\left(\omega + \frac{c}{b}\right) = -\frac{b}{J}t + \ln\left(\omega_0 + \frac{c}{b}\right). \tag{4.11c}$$

By observation, the transformed state trajectory (4.11c) is given in the form of a linear equation with slope $m = -b/J$. Therefore, a linear fit is applied to the linear region of the transformed angular speed data recorded by the encoder. The motor inertia is given by

$$J = -\frac{b}{m}. \tag{4.12}$$

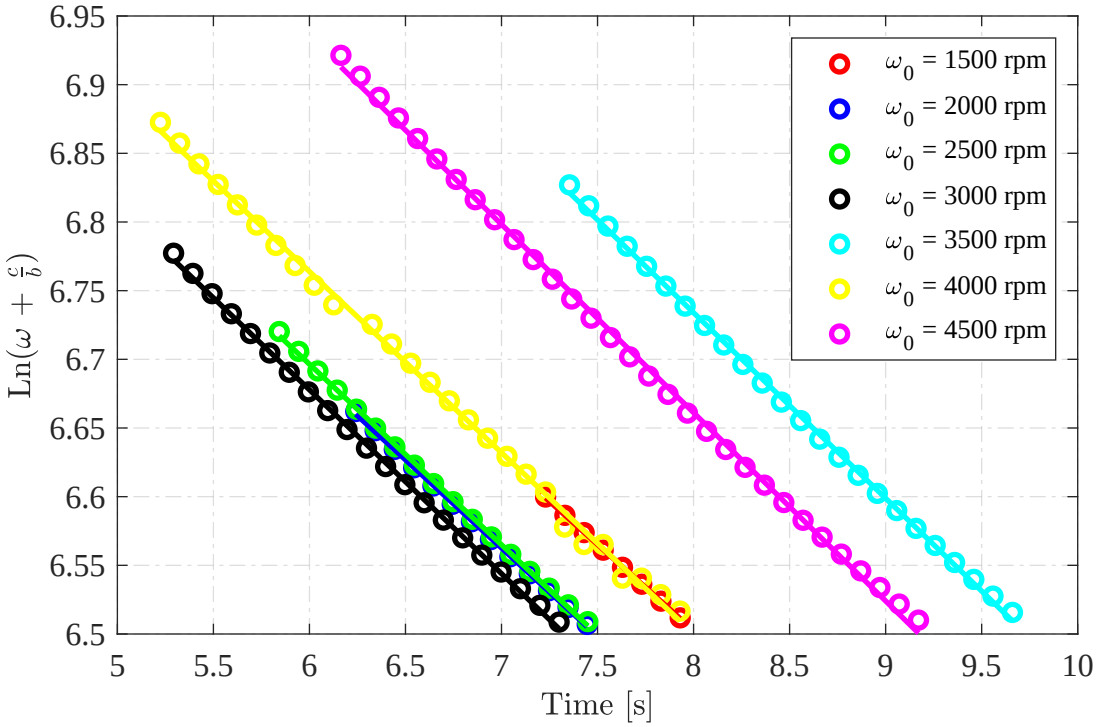


Figure 4.22: The experimental data and linear fit for each trial of the motor coastdown test.

4.4.4.3 Results

Raw data and the corresponding linear fit for each trial is provided in Figure 4.22 and by observation the fits are good. The average shaft inertia and standard deviation between all trials is reported in Figure 4.4.

4.5 Inverter and Controller Dynamics Validation

4.5.1 Experimental Setup

Inverter losses and control gains are identified simultaneously because the inverter only operates with closed loop control. As seen by the inverter and controller analysis of Sections 2.3.3.2 and 3.2.1, there are a total of 3 parameters to identify: the inverter loss R_i and controller control gains $K_{p,\omega}$ and $K_{I,\omega}$. As seen by Figure 4.23, the testbed's ESS, drivetrain, and braking subsystems are connected to conduct the test. A drive profile of 20 random

steady state speed and load torque commands is developed using band-limited white noise (Figure 4.24). White noise is used to generate a sufficiently rich data set. LabVIEW is used to save the relevant measurements reported by the ESCs. The test procedure is described below.

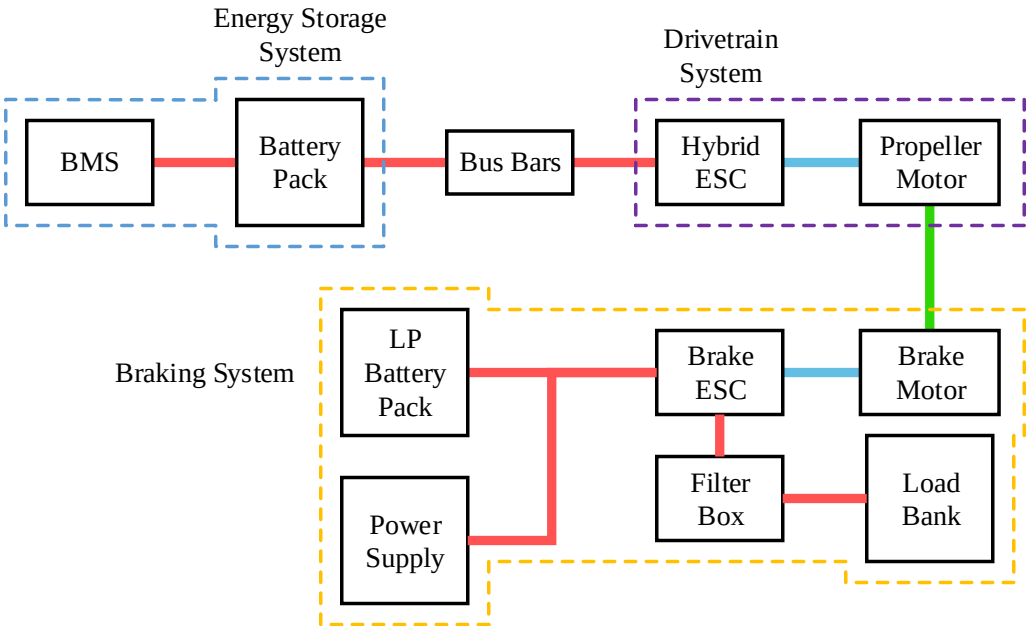


Figure 4.23: Experimental setup for the inverter and controller dynamic parameter identification test.

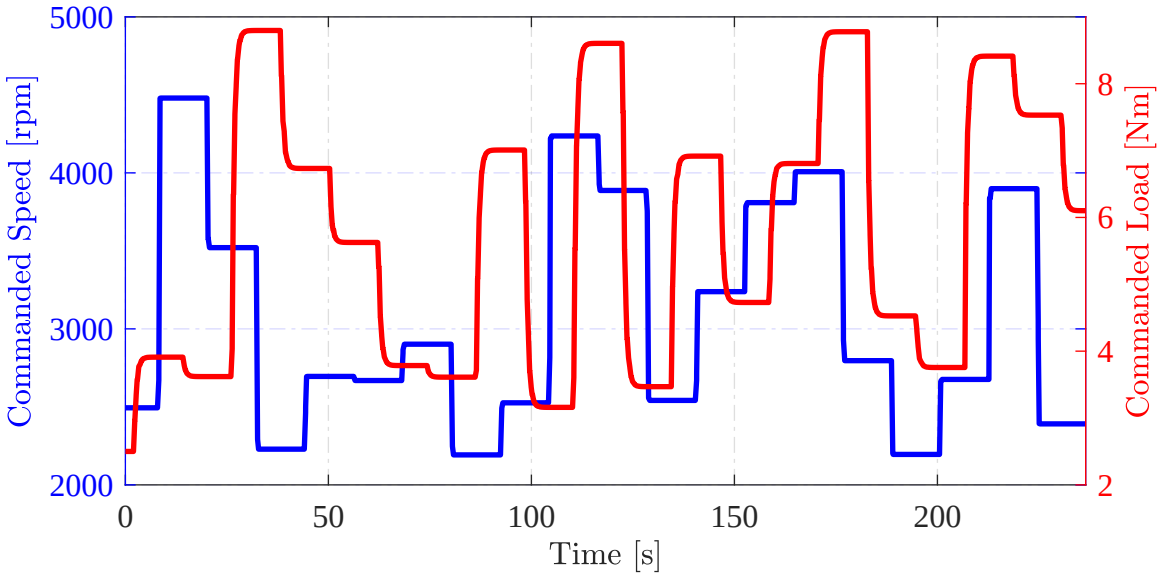


Figure 4.24: Shaft speed and load torque commands used to identify the inverter losses and control gains.

Procedure:

1. Begin data acquisition and use LabVIEW to pass the speed and load profiles to the ESCs.

2. When the profile ends, stop data acquisition and save the test data.

4.5.2 Data Analysis Methods

The grey-box model estimation tool is used to identify the 3 unknown parameters. From the analysis provided in Section 2.3.3.2, the plant dynamics are given by

$$L\dot{I}_{q,m} = V_{q,m} - RI_{q,m} - k_v\omega, \tag{4.13a}$$

$$J\dot{\omega} = k_v I_{q,m} - b\omega - c - \tau, \tag{4.13b}$$

$$V_{q,i} = m\sqrt{\frac{3}{2}}V_{dc} - mR_i I_{q,i}^2, \tag{4.13c}$$

$$I_{dc} = m\sqrt{\frac{3}{2}}I_{q_i}, \tag{4.13d}$$

$$I_{q,i} = \sqrt{3}I_{q,m}, \tag{4.13e}$$

$$V_{q,m} = \sqrt{3}V_{q,i}, \tag{4.13f}$$

$$\tag{4.13g}$$

where I and V are current and voltage states respectively, ω is the shaft speed, L and J are motor coil inductance and shaft inertia respectively, R is the motor resistance, k_v is the motor constant, b and c are viscous and static friction constant, τ is the motor load torque, the subscript q denotes a q-axis value, the subscripts m and i denote a motor and inverter quantity respectively, and the subscript dc denotes a DC-link value. Note that (4.13a) and (4.13b) and are the motor dynamics, (4.13c) and (4.13d) are the inverter "dynamics", and (4.13e) and (4.13f) convert between wye and delta values. Using equations (3.1), (3.2), (3.3b) and the controller description in Section 3.2.1, the control dynamics are defined as

$$e = \omega_{ref} - \omega, \tag{4.14a}$$

$$\tau_{ref} = \left(K_{p,\omega} e + \int K_{I,\omega} e \, dt \right), \tag{4.14b}$$

$$V_{q,i,ref} = \frac{1}{\sqrt{3}} \left(\frac{R}{k_v} \tau_{ref} + k_v \omega \right), \tag{4.14c}$$

$$m = \frac{V_{q,i,ref}}{\sqrt{\frac{3}{2}} V_{dc}}, \tag{4.14d}$$

where e is the error signal, ω_{ref} is the commanded motor speed, τ_{ref} is the calculated torque reference, $V_{q,i,ref}$ is the voltage reference, and m is the inverter modulation signal.

The grey-box model identification optimization problem is posed as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{k=1}^{N} \left(I_{dc,data}[k] - I_{dc}[k] \right)^2 + \left(\omega_{data}[k] - \omega[k] \right)^2 \\ \text{subject to} \quad & \text{Equation 4.13}, \\ & \text{Equation 4.14}, \\ & \underline{x} \leq x \leq \bar{x}, \end{aligned} \tag{4.15}$$

where $I_{dc,data}$ and ω_{data} are the DC current and speed data reported by the hybrid ESC, and $x = \begin{bmatrix} R_s & K_p & K_I \end{bmatrix}$, and $\underline{x}$ and $\bar{x}$ are conservative upper and lower bounds for each parameter used to constrain the search space.

4.5.3 Results

The model is validated using a different drive profile than described in Figure 4.24 because we want to show that the model predicts system behavior outside the state-space that is was characterized in. A comparison between the experimental and model motor speed and DC-link currents is shown in Figure 4.25. By observation, most of the steady state and transient behavior matches well. These results are sufficient for application in a model based controller. The estimated parameter values are shown in Table 4.9.

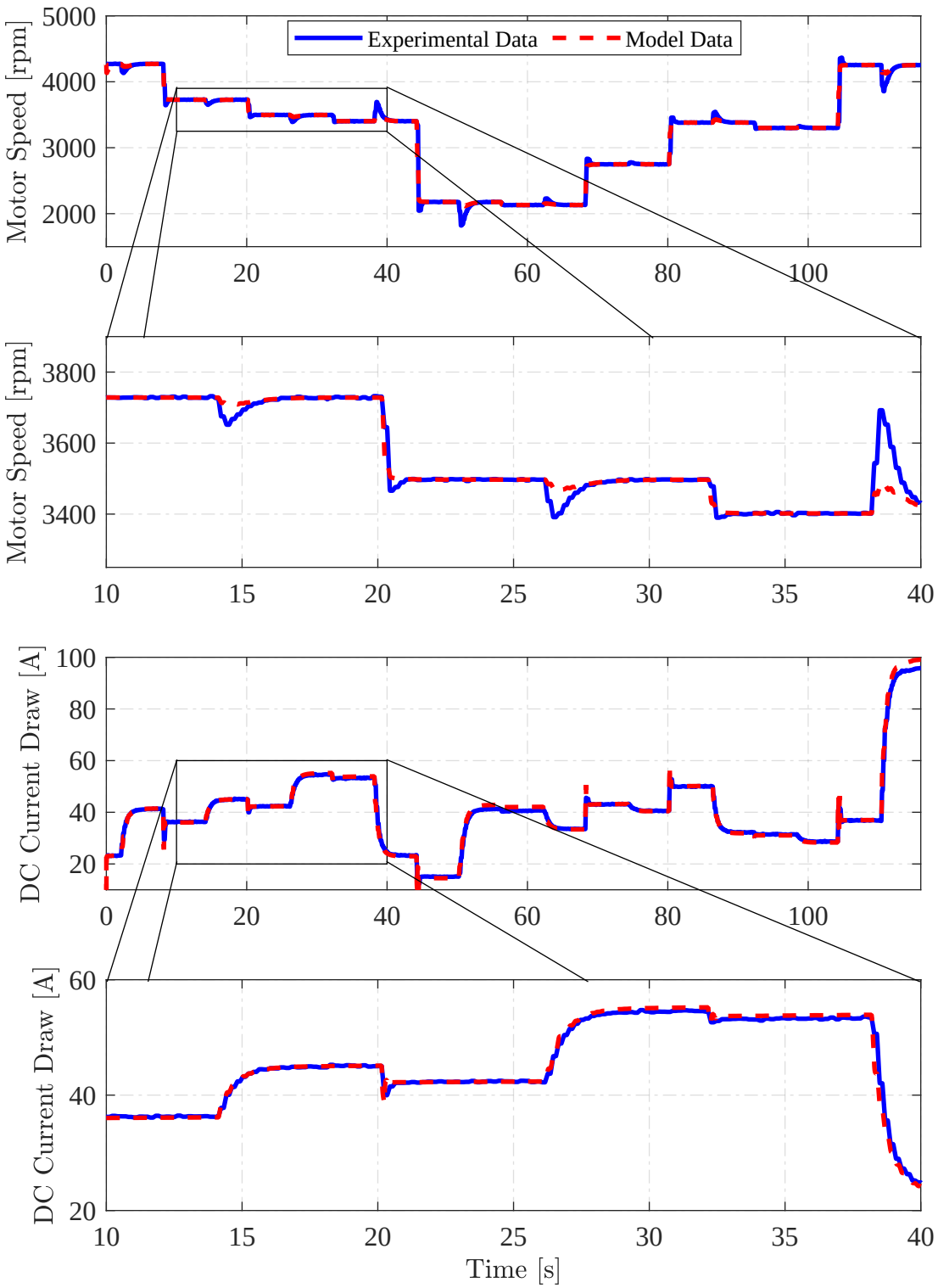


Figure 4.25: Comparison of experimental and model data for motor shaft speed and inverter DC current draw. The 2nd and 4th plots magnify sections of the 1st and 3rd plots to better highlight the transient behavior.

Table 4.9: Propulsion motor inverter loss and speed controller gains

Inverter Loss $[m\Omega]$	Proportional Gain [-]	Integral Gain [-]
80.8	0.96	1.50

4.6 Buck-Boost Converter Validation

4.6.1 Experimental Setup

There are three buck-boost converter channels on the testbed. The procedure for identifying the three parameters (input loss R_u, constant loss R_c, diode forward voltage V_d) of each channel is identical. The testbed is setup with the ESS and DCDC subsystems connected as shown in Figure 4.26 and the test equipment is listed in Table 4.10. Similar to 4.24, a voltage profile of 12 random voltage and current load commands is developed using band-limited white noise (Figure 4.27). LabVIEW is used to save the relevant measurements reported by the ESC and the shunt resistor. The test procedure is listed below.

Procedure:

1. Begin data acquisition and use LabVIEW to pass the voltage profile to the voltage step ESC.

2. When the profile ends, stop data acquisition and save the test data.

Table 4.10: Buck-boost converter identification test equipment.

Item	Part Used	Purpose
Buck Boost Converter	DCDC ESC and Filter Box	This is the system under test.
Power Supply	Battery Pack	Provides power to the voltage step ESC.
Resistor Bank	1600W 1Ω Equivalent Resistor	Load for the voltage steps system.

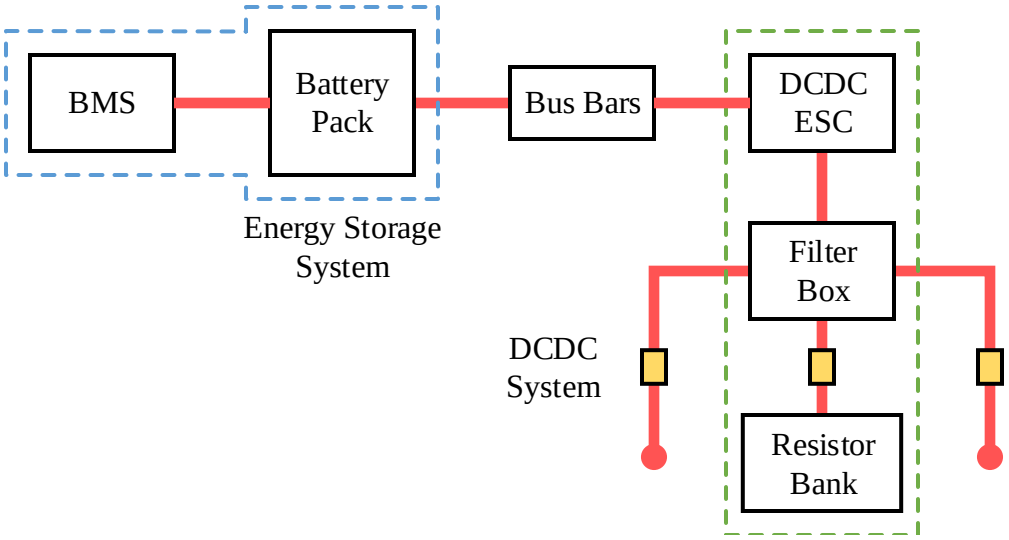


Figure 4.26: Experimental setup for the buck boost converter parameter identification test.

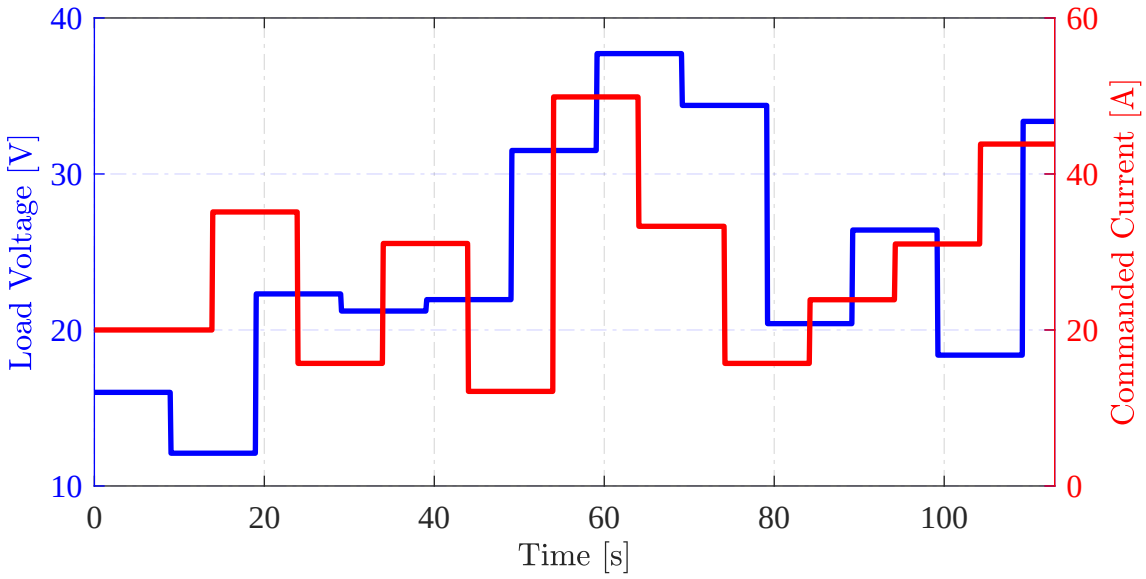


Figure 4.27: Voltage command used to identify the buck-boost converter losses.

4.6.2 Data Analysis Methods

The (approximately) steady state portion of each voltage step reference is analyzed using the grey-box model estimation tool. The output voltage state of the converter, as it was described in Section 2.3.3.1, is

$$V_{out} = uV_{dc} - (1-u)V_d - uR_uI_{out} - R_cI_{out}, \tag{4.16}$$

where V_{out} and I_{out} are the output voltage and current of the converter, u is the converter duty cycle, V_{dc} is the DC-link voltage, V_d is the diode forward voltage, and $R_u = R_s - R_D$ and $R_c = R_L + R_D$ are as defined in Section 2.3.3.1. The grey-box model estimation optimization problem is defined as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{k=1}^{N} \left(V_{out,data}[k] - V_{out}[k]\right)^2 \\ \text{subject to} \quad & \text{Equation 4.16}, \\ & \underline{x} \leq x \leq \bar{x}, \end{aligned} \tag{4.17}$$

where $x = \begin{bmatrix} V_D & R_u & R_e \end{bmatrix}$, $V_{out,data}$ is the voltage across the resistor bank and $\underline{x}$ and $\bar{x}$ are conservative upper and lower bounds for each parameter used to constrain the search space. The voltage is calculated using the resistor bank resistance and measured current.

4.6.3 Results

Once characterized, the model is validated using a load profile that differs from the profile voltages and currents for each channel are described in Figure 4.27. By observation the model is sufficient for its intended use. The identified parameters for each channel are provided in Table 4.11, where it is observed that there are some differences in channel losses. However, the losses across channels are of the same order of magnitude, so the result should be sufficient for control design.

In Section 4.5, the speed control gains of the inverter were identified. In the analysis of the inverter controller 3.2.1, it was assumed that the electrical control dynamics are fast enough (faster than the DAQ 10Hz sampling rate) to be treated as static. The same concept is applied here. However, instead of entirely neglecting the electrical controller dynamics, control gains are hand tuned such that the system dynamics settle in less than 0.1 seconds (10Hz). The control gains for the buck-boost converter control loops are provided in Table 4.12.

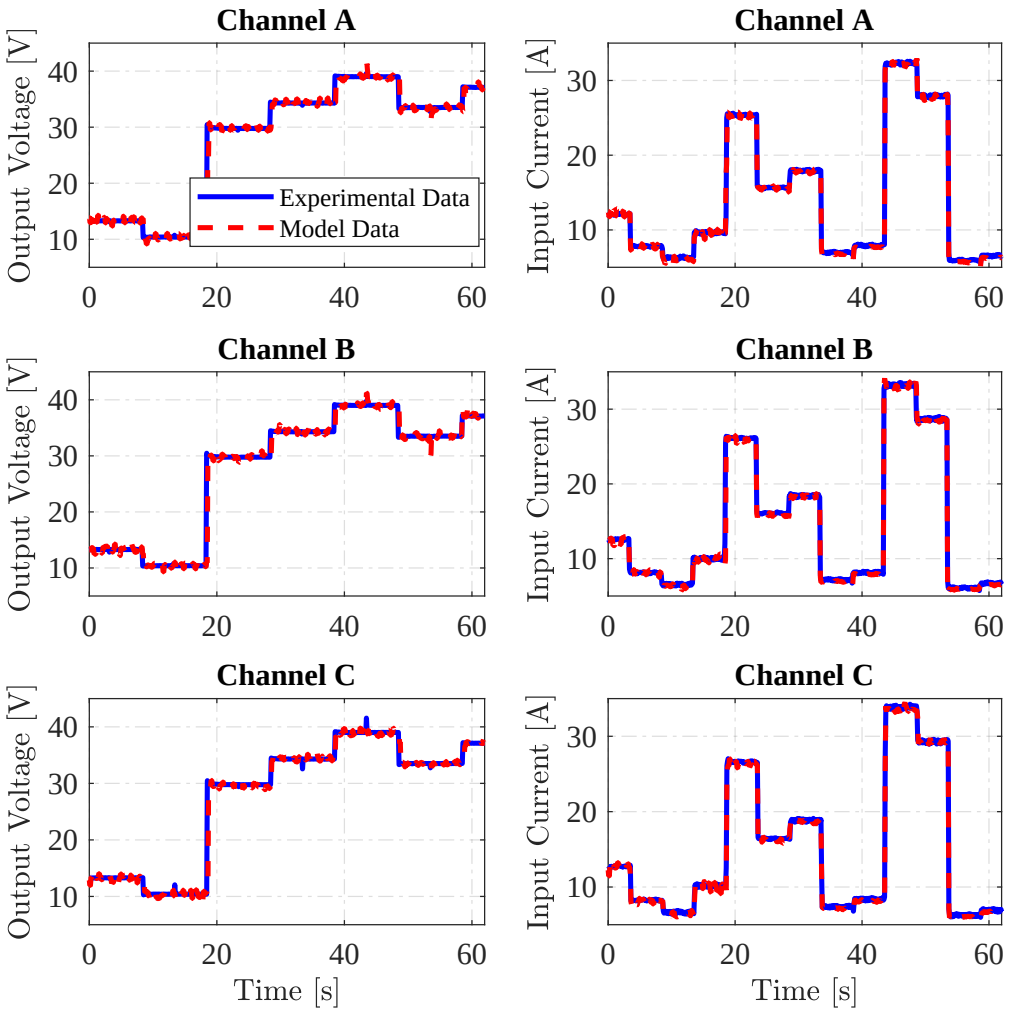


Figure 4.28: Comparison of experimental and model output data for the buck boost converter.

Table 4.11: Voltage step ESC and filter box identified losses.

Channel	Forward Voltage $[mV]$	Constant Loss R_c $[m\Omega]$	Input Loss R_u $[m\Omega]$
A	397	24	24
B	587	19	29
C	475	20	27

Table 4.12: Estimated buck-boost current regulator control gains.

Proportional Gain [-]	Integral Gain [-]
3.91×10^{-5}	0.21

4.7 Genset Validation

4.7.1 Experimental Setup

The genset is entirely characterized by the its input gain K, time constant τ, and fuel consumption coefficients a_i as described in Section 2.3.4. The experimental setup is shown in Figure 4.29. In this test, the genset is commanded to produce a specified amount of current and then the propulsion subsystem is actuated such that the battery current is zero. This process is repeated for multiple bus voltages. The test procedure is described below.

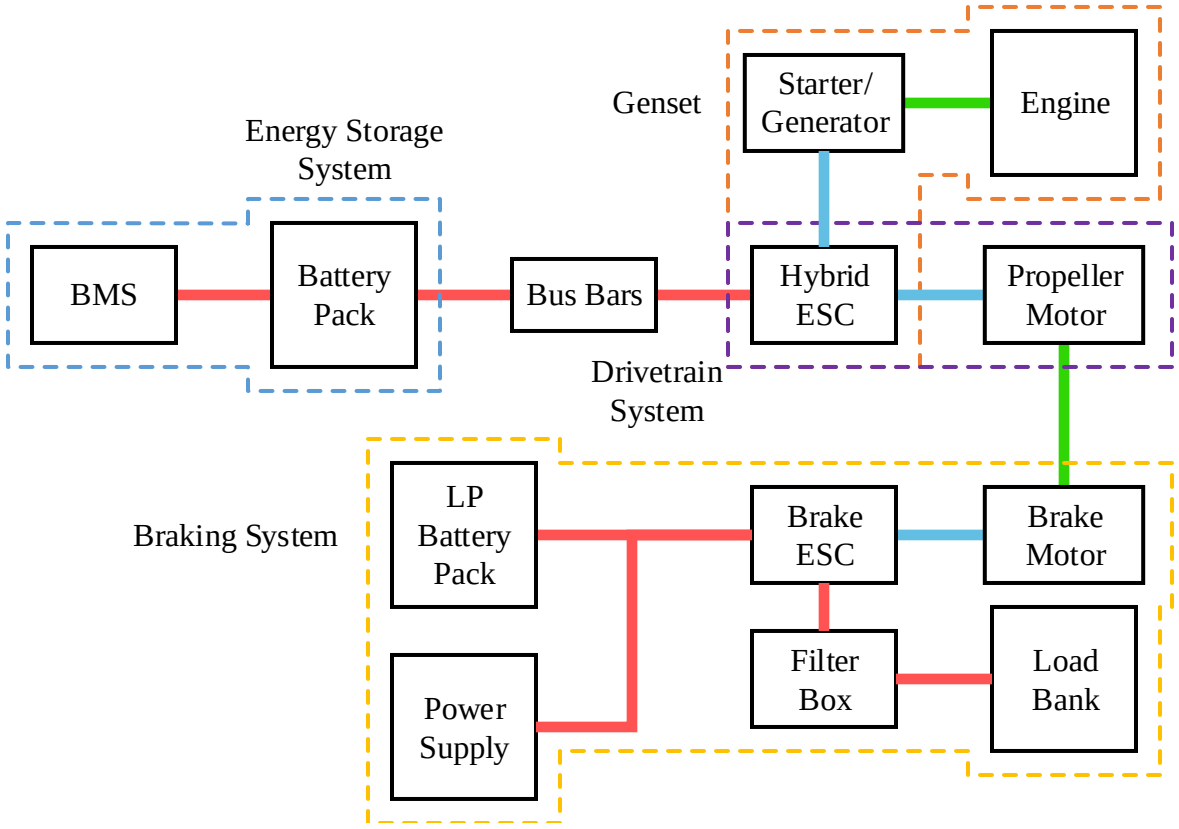


Figure 4.29: Experimental setup for the genset parameter identification test.

Procedure:

1. Charge or discharge the battery such that the bus voltage reaches approximately 50V.
2. Start the genset and command the genset to produce 5A.
3. Command the prop motor speed or dyno motor torque such that the battery current is approximately 0A.
4. Collect steady state data for approximately 30 seconds.
5. Repeat steps (2)-(4) for desired genset currents $I_d = [5 : 70]$A in 5A increments. Note that the engine has an upper limit on power production so larger commanded currents may not be feasible. In those cases, just move to step (5).
6. Turn off the genset.
7. Repeat steps (1)-(6) for bus voltages $V = [50 : 65]$V in 2.5V increments.

4.7.2 Data Analysis Methods

Due to ripple in the engine torque and controller tuning, the resulting genset current is highly periodic. This phenomenon is best illustrated by Figure 4.30 where, for a constant commanded current, there is about a 10A peak-to-peak variation in the actual genset current. For this reason, the "steady state" for each command is averaged over the 30 second data acquisition interval. The subsequent analysis utilizes the averaged values.

As mentioned previously, the engine has an upper limit on its power production. This translates to a non-constant upper limit on the achievable genset current because the bus voltage changes between trials. The current upper limit $\bar{I}_d$ is assumed linear in the operational domain and is a function of the bus voltage

$$\bar{I}_d = b_1 V + b_0, \tag{4.18}$$

where $b = \{b_1, b_0\}$ are constants. Since $\bar{I}_d$ was determined for each bus voltage, the coefficients b can be determined by linear least squares.

Next, it is desirable to command a control signal $u \in [0, \bar{u}]$ instead of I_d. The commanded currents are normalized by the max commanded current resulting from (4.18)

$$u = \frac{I_d}{\max\left(\bar{I}_d\left(V\right)\right)} \quad \forall \quad V \in [50:65]. \tag{4.19}$$

The bounds on V are required because data was collected on a closed domain. Additionally, the bounding coefficients b should be scaled such that (4.18) can be described by $\bar{u}(V)$ instead of $\bar{I}_d(V)$.

It must be checked that the desired/commanded genset current is physically achieved. Assuming that the steady state genset current is a linear function of the input $I = Ku$, a linear fit is applied to the experimental data. Notice that K is indeed the control gain.

Lastly, the SFC coefficients must be determined. Based on the experimental data, the max genset power is approximately 3.3kW. The genset SFC is assumed to be minimized at approximately 75% max genset power with a minimum SFC of 200g/kW-h. Additionally, the engine SFC is assumed to be maximized at 0kW with a maximum SFC of 600g/kW-h. Based on these two assumptions, a quadratic and convex surface can be generated to represent the genset SFC.

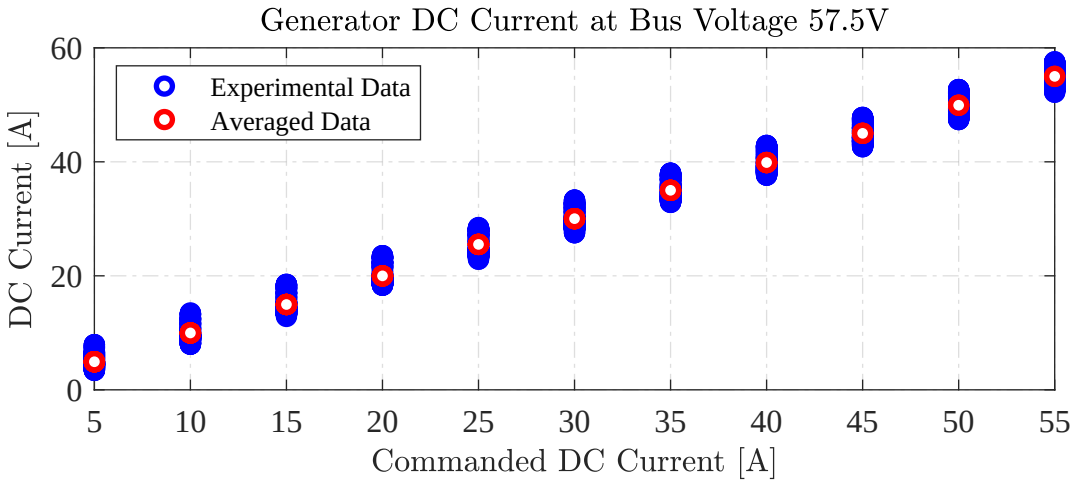


Figure 4.30: A comparison the the experimental and average genset current values at a 57.5V bus voltage.

4.7.3 Results

The results of each genset fit are highlighted in Figure 4.31. By observation of the top plot of Figure 4.31, the fit is sufficient. It also becomes obvious how the max genset DC current changes as a function of the bus voltage. The genset model parameters are listed in Table 4.13. The time constant was hand tuned to best fit transient data because0 a steady-state analysis was used to characterize the genset.

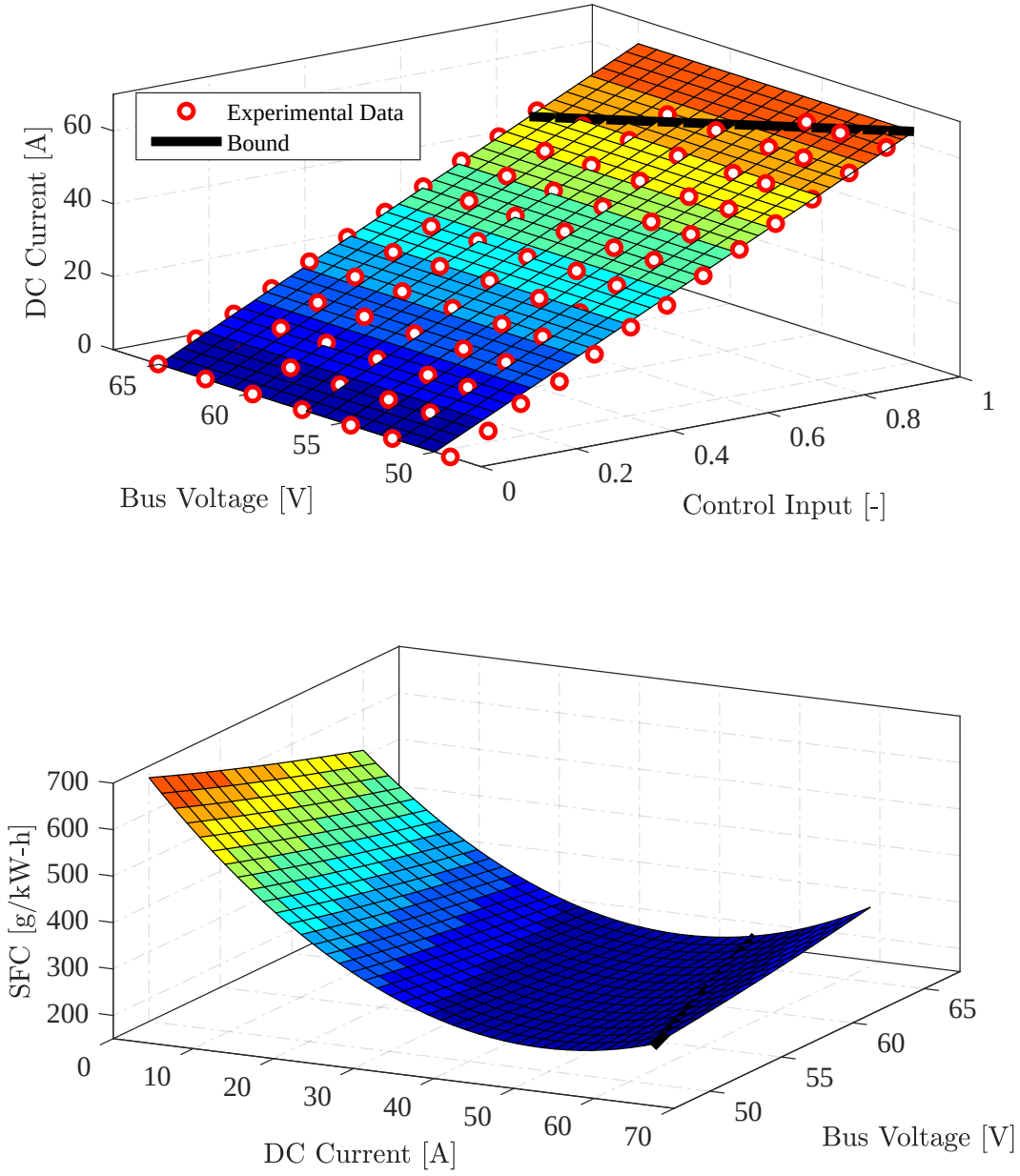


Figure 4.31: The genset current and specific fuel consumption maps.

Table 4.13: Experimentally characterized parameters for the genset model.

Input Gain [A]	Input Bound Coefficients	SFC Coefficients	Time Constant [s]
-63.3	$\begin{Bmatrix} b_1 = -15.1 \times 10^{-3} \\ b_0 = 1.76 \end{Bmatrix}$	$\begin{Bmatrix} a_1 = 0.197 \\ a_2 = 0.089 \\ a_3 = 1.50 \times 10^3 \\ a_4 = 0.247 \\ a_5 = -31.7 \\ a_6 = -21.0 \end{Bmatrix}$	0.5

4.8 Processor Load Validation

The processor load is experimental validated by removing the system battery and connecting a power supply to the main bus. The power supply is commanded to 9 steady state voltages ($V = [45 : 65]$V in 2.5V increments) and the steady state current is recorded. A linear fit characterizing the processor load as a function of the bus voltage is applied to resulting data points. The resulting fit in shown in Figure 4.32. The characterized processor load coefficients as described in Section 2.3.7 are $\{a_1, a_2\} = \{-5.2 \times 10^{-3}, 0.71\}$.

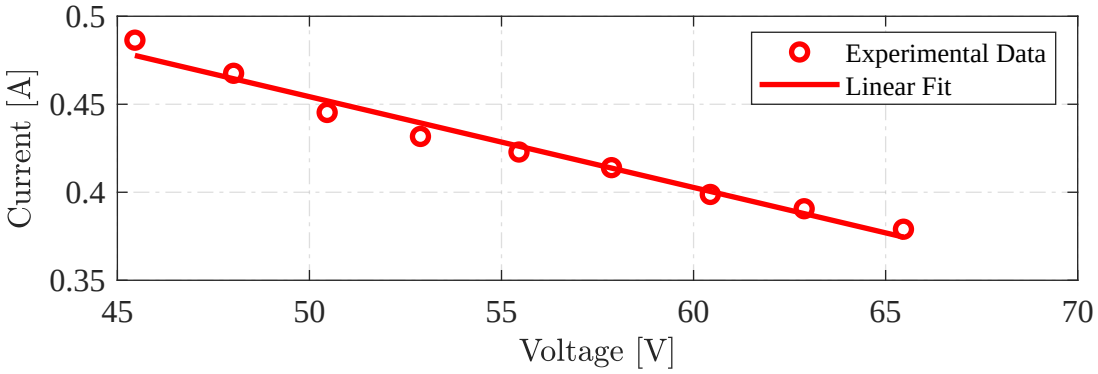


Figure 4.32: A linear fit relating the ESC bus voltage to the ESC processor load.

4.9 Open-Loop System Validation

The full system model without vehicle body dynamics is experimentally validated in open-loop. The experimental system is passed the random set of prop speed, load current, load torque, and genset input commands shown in Figure 4.33. The same set of inputs are passed to the system model and states are compared in Figure 4.34. By observation, the model dynamics match the experimental data very well. There appears to be some error in the battery voltage, however, the error may be the result of poor initial conditions for the model. Additionally, the genset model does not capture the startup transient at 50 seconds. Neglecting the transient should be sufficient for control design because the startup dynamics are less than 0.5 seconds. Lastly, there is a small amount of steady state mismatch in the load current state. Overall, the modeling error is small and this model should be sufficient for control design.

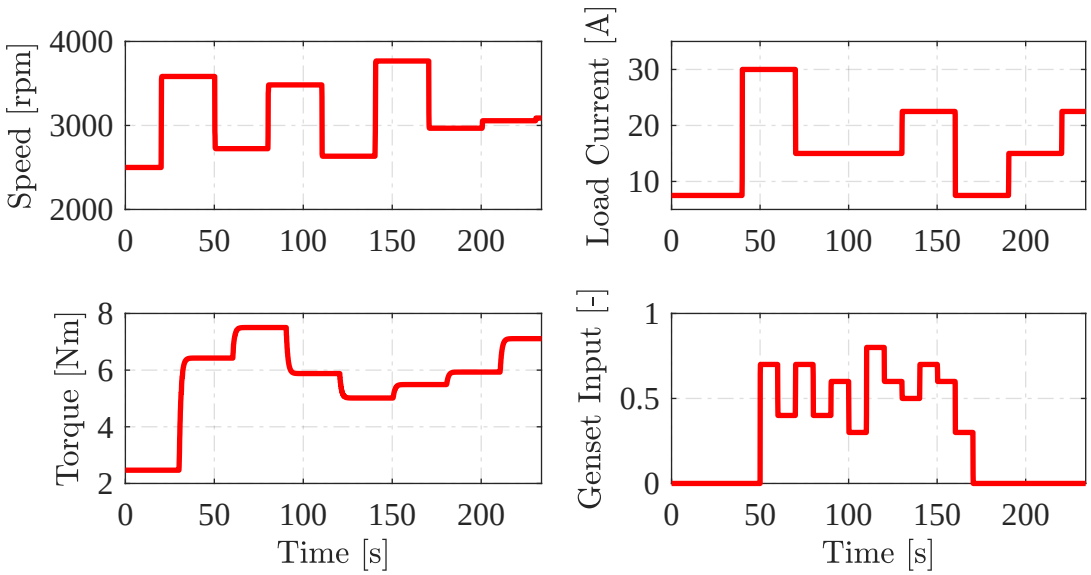


Figure 4.33: Random set of open-loop inputs used for the system model validation process.

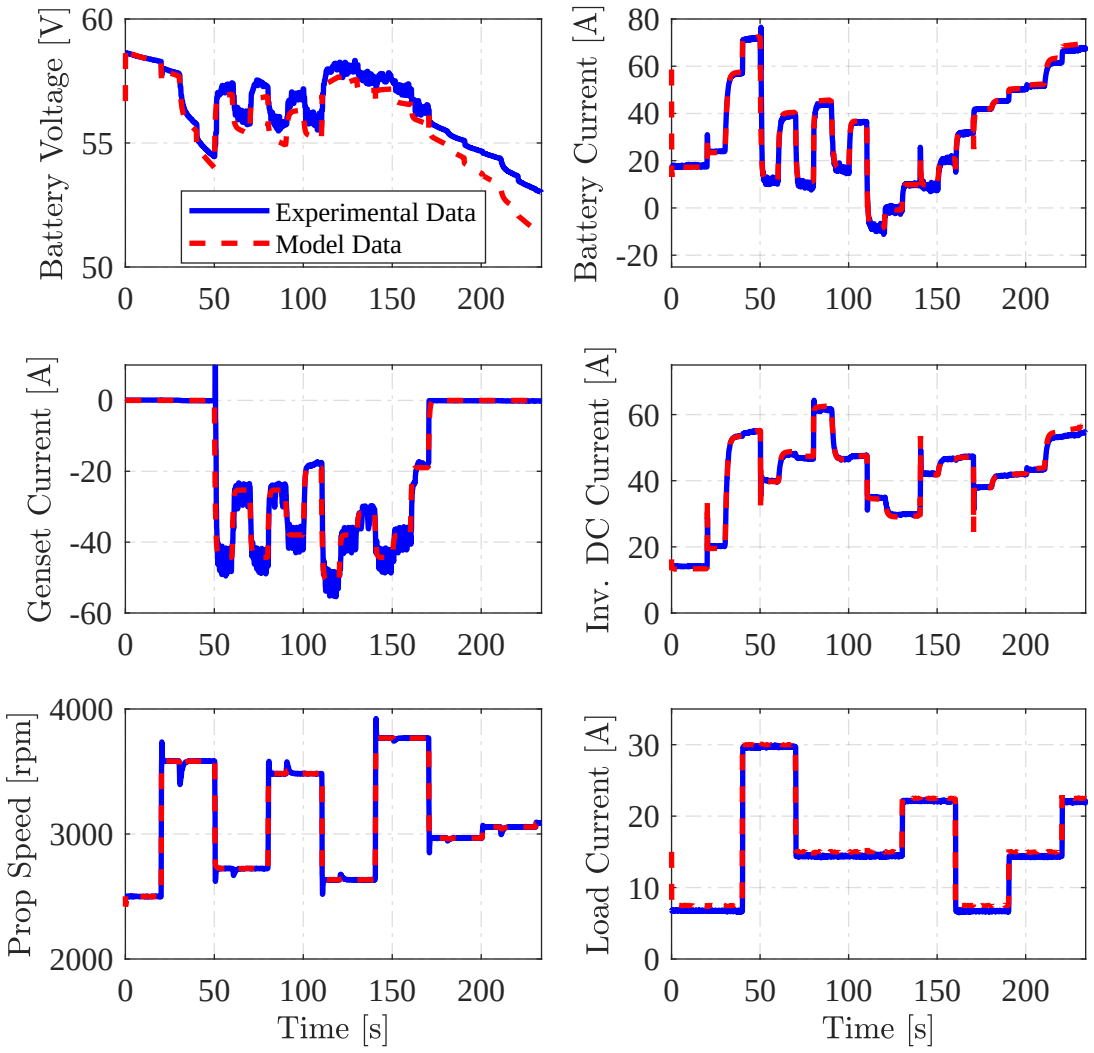


Figure 4.34: A comparison between open-loop experimental and model data for 6 selected states.

4.10 Conclusion

This chapter provided a description, characterization methods, and validation efforts for the hybrid electric UAV powertrain testbed. First, a description of the testbed, its subsystems, and communication methods was provided. The testbed description was followed by characterization methods for individual components and subsystems on the testbed (battery, motor, inverter, converter, genset, and processor load). These characterization methods

utilized analytic solutions and optimization programs to identify parameters. The fully characterized system model was validated against experimental data with good matching between states.

Chapter 5

Controller Results and Discussion

5.1 Background

The experimentally validated system model in Chapter 4 was used to tune the control algorithms designed in Chapter 3. After the controllers are verified in simulation, they are experimentally validated on the experimental hardware. This two-step process facilities rapid control development and demonstrates that the hardware can be operated safely. Additionally, the experimental validation step is necessary to show that simulation results translate well to a physical system and that the controllers are capable of operating in real-time.

The remainder of this chapter will demonstrate the results of executing this two-step control design process. Controller evaluation metrics are presented in Section 5.2. In Section 5.3, the parameters and mission profile for each control formulation are described. The simulation results of the three control designs are compared and evaluated in Section 5.4. Lastly, Section 5.5 provides the experimental validation results for each controller

5.2 Figures of Merit

Each controller design will be evaluated on 3 quantitative metrics and 1 qualitative metric. The metrics and their purpose are described below.

- *Performance* - In this work, performance is described as the controller's ability to meet mission objectives. Numerically, performance is represented by taking the 2-norm of the difference between the desired reference and the actual system state,

$$\mathcal{P} = \|x_{ref} - x\|_2^2 . \tag{5.1}$$

- *Reliability* - Reliability is a safety metric defined as the largest constraint violation for the battery state of charge and current. Reliability is important to control design to preserve the longevity of the system. This metric is evaluated by taking the infinity norm of all violations.

$$\mathcal{R} = \|s\|_\infty \tag{5.2}$$

  where s is the slack term of a state as described in (3.15).

- *Efficiency* - The efficiency metric describes the fuel consumption of the genset subsystem. Minimizing fuel consumption has a positive environmental impact and can potentially increase range. Numerically, the fuel consumption is quantified by

$$\mathcal{E} = \int_0^T sfcP_{gen}dt, \tag{5.3}$$

  where T is the total mission time, sfc is the fuel consumption of the genset subsystem, and P_{gen} is the instantaneous power of the genset subsystem.

- *Adaptability* - Adaptability is a qualitative metric used to describe how well a control design can be adapted to a new system architecture or mission. Adaptability is important because UAVs are often reconfigured for mission-specific designs [3]. If a controller is not well-adaptable, a new controller would have to designed for each iteration of the system architecture or change of mission.

5.3 Mission and Controller Parameters

Each controller is evaluated against the same mission profile with the same state and input bounds. The mission profile consists of a vehicle velocity and avionic load profile and is described by Figure 5.1. The velocity profile consists of a take-off, cruise, dash, and loiter segments. The avionic load profile can be described by the sum of two separate loads operating at 40V. The first load is a constant 12.5A load plus a 15A pulse load (120s period) representative of the base system power and communication requirements. The second load is a 20A load that starts at the onset of the loiter mission segment and could represent the

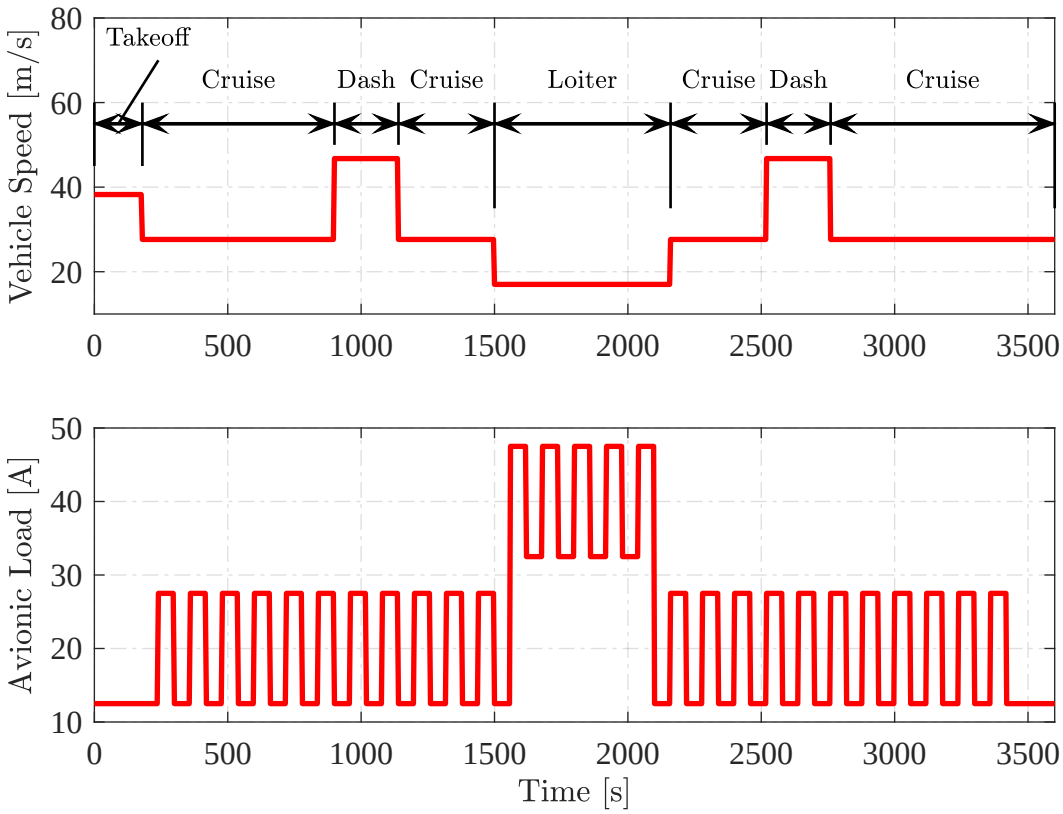


Figure 5.1: The mission vehicle velocity and avionic load profile.

main payload. The concept of the mission is that the unmanned vehicle flies to a specific location, collects or communicates data, and then leaves.

The state and input bounds for all control designs are provided in Table 5.1. The battery state of charge, battery current, and input bounds are physical constraints of the system. Note that the inverter input is constrained to the range of $[0.30, 0.70]$ to prevent the optimization program from choosing unreasonable inputs. This conservative range was chosen based on closed-loop evaluation of the system model. The prop speed upper bound is a physical limitation of the hardware. However, the lower bound is imposed because the model loses accuracy below that threshold. At low speeds, d-axis current is injected into the motor which violates one of the modeling assumptions (Section 2.3.2.2). The inverter and converter lower state bounds are required because regeneration is not enabled on the physical hardware. The vehicle speed constraint is used to prevent the optimizer from predicting a negative vehicle velocity in a fringe case.

Table 5.1: State and input bounds for all control designs for the system described in Chapter 2.

State/Input	Description	Lower Bound	Upper Bound
State 1	Battery SOC [-]	0.3	0.9
State 6	Battery Current [A]	-10.5	105
State 8	Inverter Current [A]	1	1000
State 12	Converter Current [A]	1	1000
State 18	Inverter DC Current In [A]	1	95
State 23	Prop Speed [rpm]	1500	5000
State 24	Vehicle Speed [m/s]	1	100
Input 6	Inverter Input [-]	0.30	0.70
Input 7	Converter Input [-]	0.01	0.99
Input 8	Genset Input [-]	0	1

Table 5.2: Baseline controller parameters (0.2 second update interval).

Power Share Controller			
$\hat{q}$ [-]	$\check{q}$ [-]	$\tilde{q}$ [-]	Dwell Time [s]
0.4	0.4	0.5	30

Vehicle Speed Regulator			
Proportional Gain [-]	Integral Gain [-]	Damping Ratio [-]	Cutoff Frequency [rad/s]
1.03×10^5	4.49×10^5	1	0.06

5.3.1 Baseline Controller

The baseline controller is composed of the power share controller and vehicle speed regulator. Each controller has 4 tuning parameters and their respective values are shown in Table 5.2. The baseline controller has an update interval of 0.2 seconds.

5.3.2 Centralized Controller

The centralized controller is tuned by varying the time-step, control horizon, controller weightings in the cost function ((3.15a) and (3.15b)), and by adjusting the preview horizon of the time-varying state bound. Each cost and the associated state is tabulated in Table 5.3.

The time step and prediction horizon of the controller is 3 seconds and 15 steps respectively (45 second preview). Two sets of time-vary state bounds are generated with preview horizons of 4 and 10 minutes (Figure 5.2) in which the 10 minute bound is more conservative and weighted less than the 4 minute bound. The conservatism is apparent from Figure 5.2 in which the 10 minute preview bound is always greater than or equal to the 4 minute preview bound. By utilizing two bounds with different cost function weightings, the controller can be tuned to better plan the battery state of charge. Additionally, the inverter and converter inputs are required to stay within a $\delta = 0.1$ tube in reference to (3.15j).

Table 5.3: Centralized controller cost function weightings (3 second time-step, 15 step horizon).

	State	Description	Cost
Reference Tracking	State 24	Vehicle Speed	35
	State 27	Avionic Load Current	75
State Constraints	State 1	Battery SOC	10^6
	State 6	Battery Current	10
	State 8	Inverter Current	1
	State 12	Converter Current	1
	State 18	Inverter DC Current In	10^4
	State 23	Prop Speed	100
	State 24	Vehicle Speed	1
TV State Constraints	State 1	10 Minute Preview	10^3
	State 1	4 Minute Preview	10^6
Derivative Costs	State 23	Prop Speed	5
	State 24	Vehicle Speed	250
	Input 8	Engine Input	250
Miscellaneous	N/A	Fuel Cost	1600
	N/A	Switching Cost	600

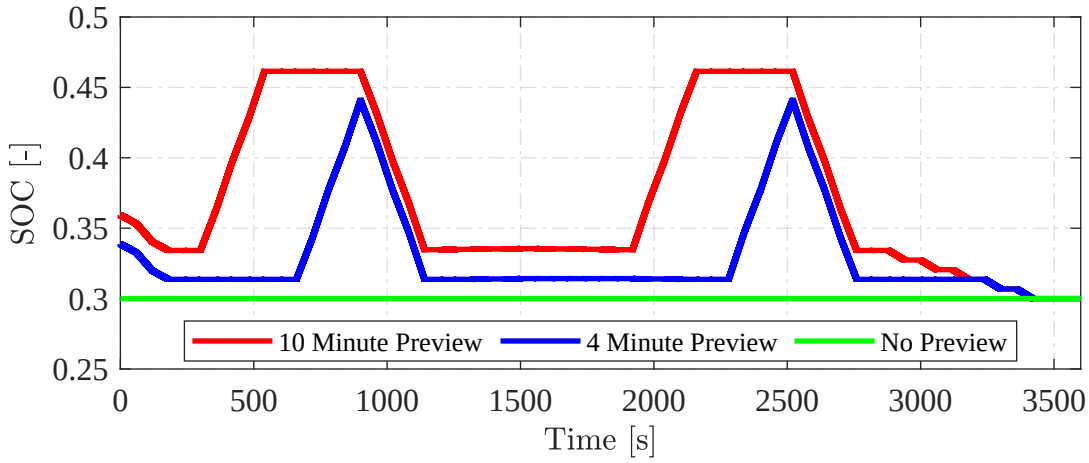


Figure 5.2: A 10 and 4 minute time-varying battery SOC state bound for the mission described in Figure 5.1. The lower bound with no preview is shown for comparison.

5.3.3 Hierarchical Controller

The hierarchical controller is tuned by varying the time-step, prediction horizon, and cost function weightings of the two model predictive controllers described by (3.16). The time-step and prediction horizon for the upper level controller is 60 seconds and 10 steps (10 minute preview), while the time-step and prediction horizon for the lower level controller is 3 seconds and 15 steps (45 second preview) (Table 5.4). The preview horizon of the upper level controller was chosen to match the long preview horizon of the time-varying state bound of the centralized controller. The preview horizon of the lower level controller was chosen for similar reasons. Again, the inverter and converter inputs are required to stay within a $\delta = 0.1$ tube (3.16j) at the lower level. The tube constraint is ignored at the upper level.

Note that the lower level and centralized controller cost function weightings are nearly identical. The only difference between the two control designs is the implementation of the battery state of charge planning. Also see that the upper level controller is tuned to only track references while respecting the battery SOC bound and minimizing fuel consumption. The inclusion of additional control objectives may degrade the controller performance.

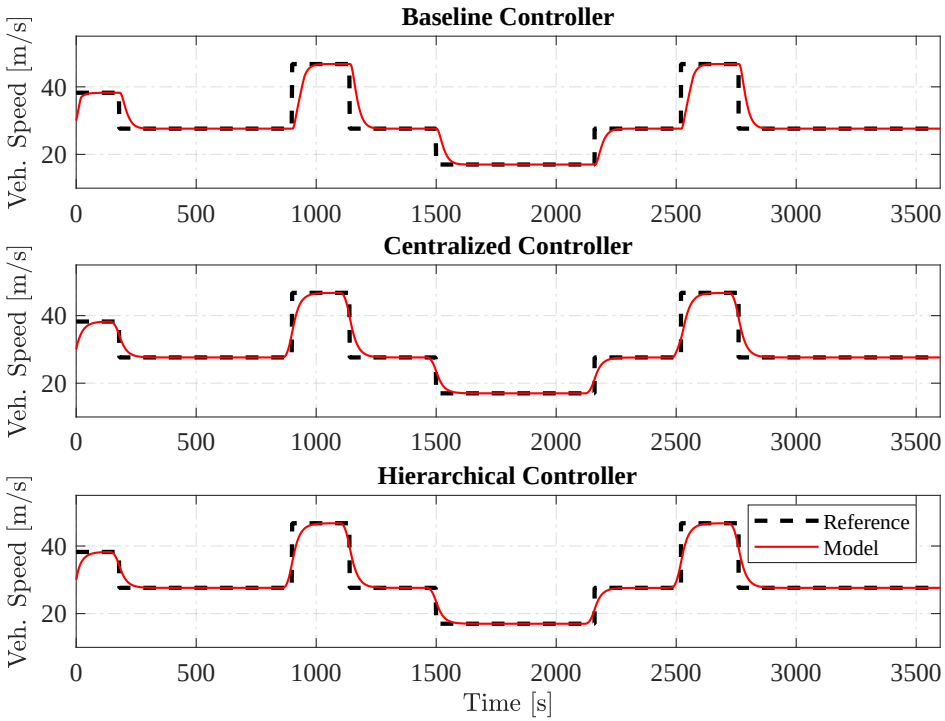
Table 5.4: Hierarchy upper and lower level controller cost function weightings. Upper level time horizon is 30 second time-step, 20 step horizon. Lower level time horizon is 3 second time-step, 15 step horizon.

	State	Description	Lower Level Cost	Upper Level Cost
Reference Tracking	State 24	Vehicle Speed	35	10^4
	State 27	Avionic Load Current	75	5×10^3
State Constraints	State 1	Battery SOC	10^6	10^6
	State 6	Battery Current	10	0
	State 8	Inverter Current	1	0
	State 12	Converter Current	1	0
	State 18	Inverter DC Current In	10^4	0
	State 23	Prop Speed	100	0
	State 24	Vehicle Speed	1	0
TV State Constraints	State 1	SOC Bound	10^3	N/A
Derivative Costs	State 23	Prop Speed	5	0
	State 24	Vehicle Speed	250	0
	Input 8	Engine Input	250	0
Miscellaneous	N/A	Fuel Cost	1600	10
	N/A	Switching Cost	600	10^4

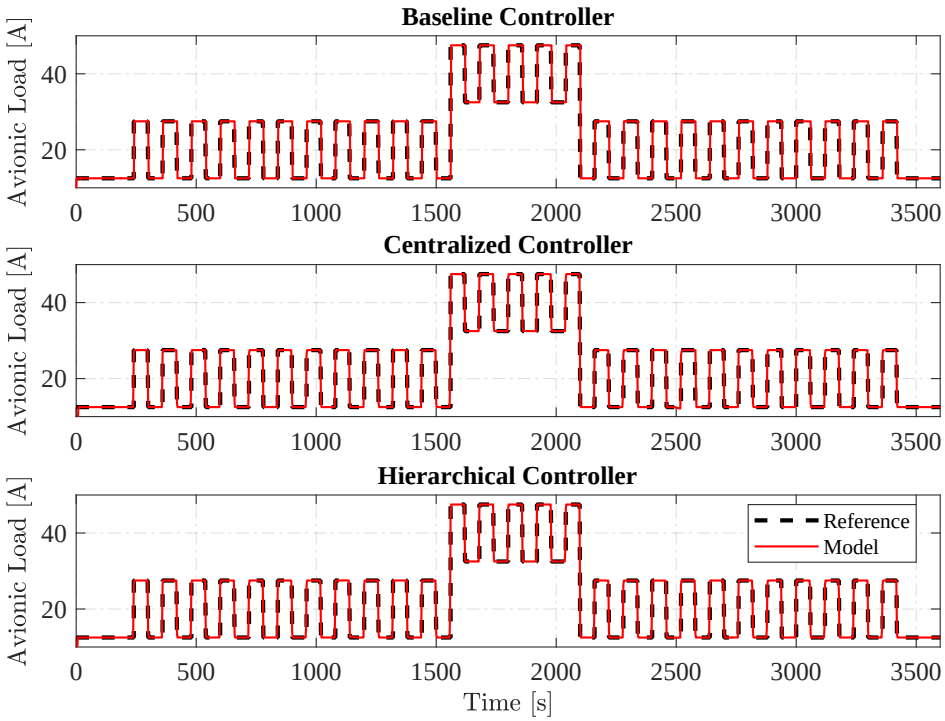
5.4 Simulation Results

First, the reference tracking (performance objective) results for each controller are compared in Figure 5.3. By observation, each controller can successfully track state references. The proactive nature of the centralized and hierarchical controllers can be observed in the vehicle speed plot comparisons.

The battery state of charge and battery current states are compared in Figure 5.4 to evaluate the reliability of each control design. Each controller was able to maintain the battery state of charge within the predefined state bounds. The baseline controller had a single major battery current state violation at approximately 2100s when the avionic load current had a large step decrease. The violation is a result of the reactive nature



(a) Simulated vehicle velocity tracking result for each control design.



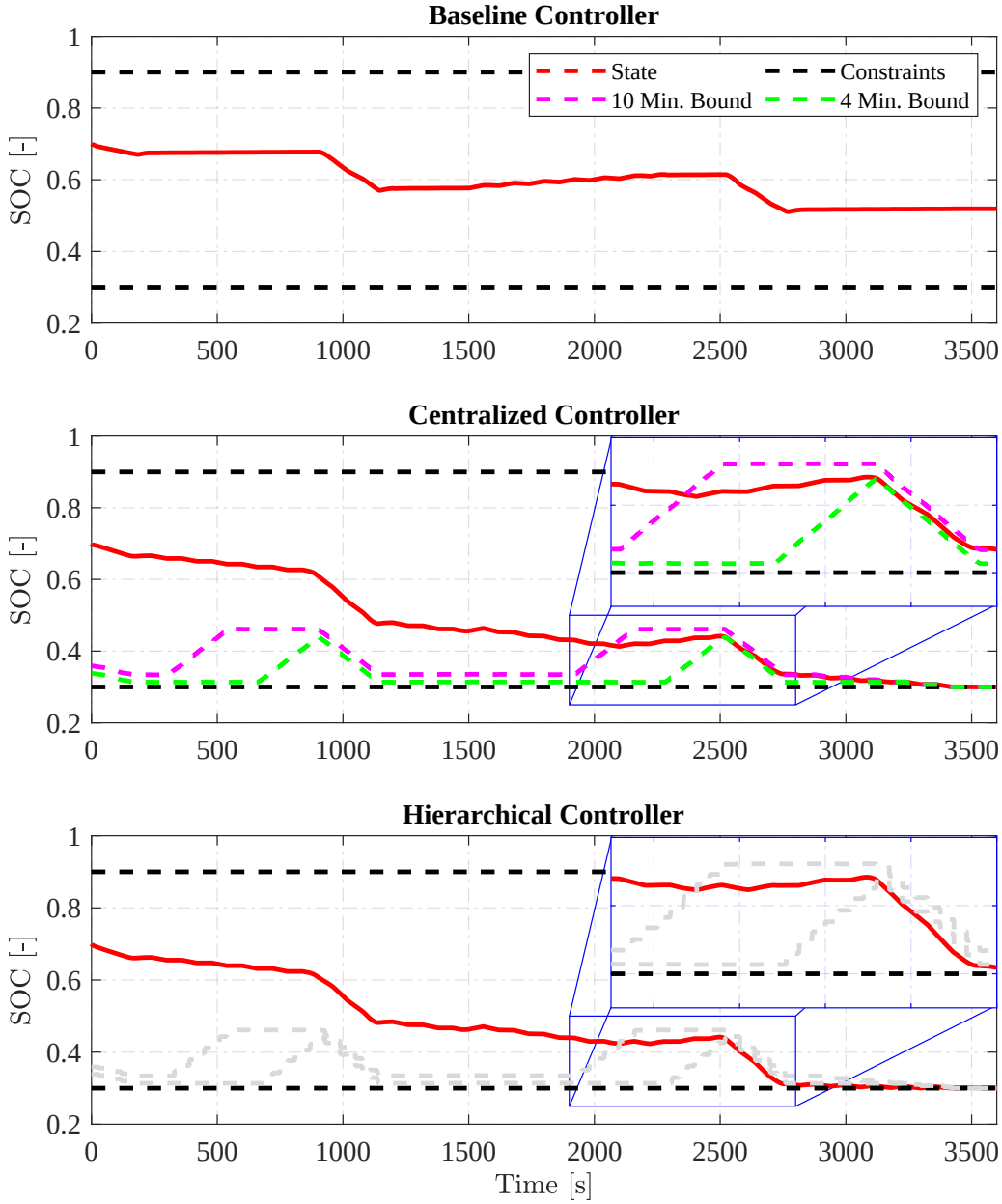
(b) Simulated avionic load current tracking result for each control design.

Figure 5.3: The simulated (a) vehicle velocity and (b) avionic load current tracking results for each control design.

of the baseline controller since the dynamic genset takes time to decrease its power output. However, the state violation is small in magnitude and time. The centralized and hierarchical controllers have significantly smaller current state violations. The violation is likely non-zero because of the relatively small penalty on that slack variable in the controller parameters. Table 5.5 lists the reliability metric for each controller.

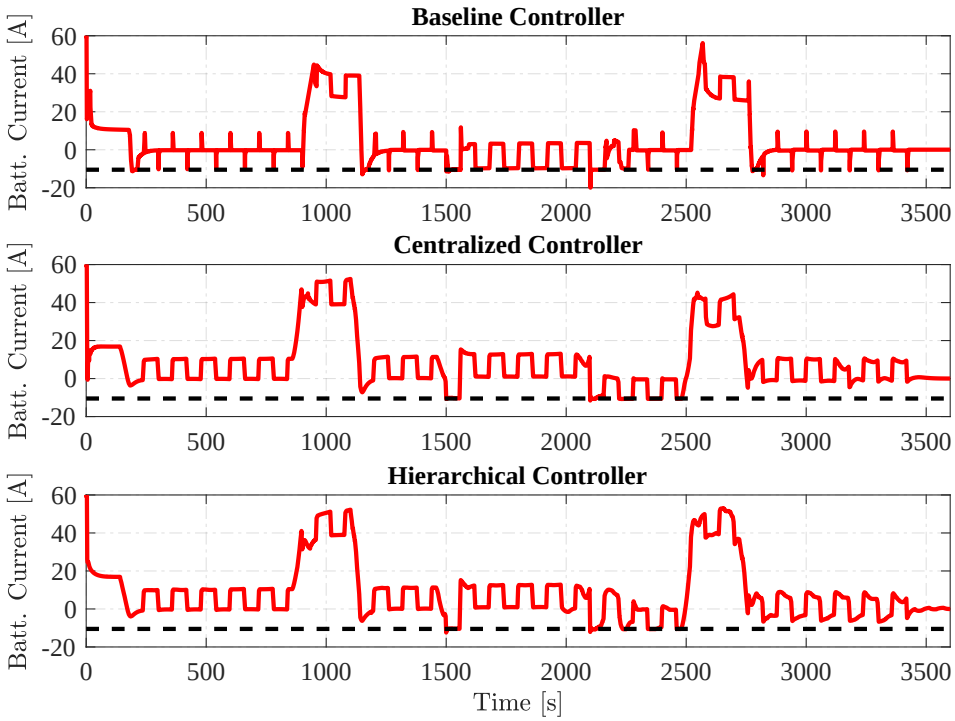
The evaluation of the fuel consumption (and genset operation) is best described by Figure 5.5. Note that the minimum SFC is 200 $g/kW-h$. It is apparent the baseline controller operates the genset aggressively in comparison to the centralized and hierarchical controllers because the genset current is consistently higher and the SFC is commonly operated away from optimal. In contrast, the centralized and hierarchical controllers are more conservative when using the genset. This is illustrated by the SFC traces where the advanced control designs maintain the genset near optimal operation for most of the mission. Lastly, observe that the advanced control designs decrease the genset current output at approximately 1100, 1500, 2100, and 2750 seconds. The controllers anticipate a large decrease in the total system load (decrease in vehicle speed or avionic load) and decreases the genset current output such that the battery is not charged at high rates. The baseline controller has constraint violations because it lacks this anticipatory behavior. The fuel consumption of each control design is outlined in Table 5.5.

Overall, each controller performs as designed. The baseline controller is more conservative in utilizing the battery in order to sustain the pack at a higher charge whereas the centralized and hierarchical controllers better utilize the full depth of charge of the battery. Figure 5.6 and Table 5.5 compare each control design based on the performance, reliability, and efficiency figures of merit. By observation, both the centralized and hierarchical controllers use 88% (12.5% improvement) of the fuel used by the baseline controller while exhibiting significant improvements in both speed tracking and minimizing the current constraint violations. When comparing the hierarchical and centralized designs, the hierarchical is marginally more economical. This improvement is a result of the over-conservative nature of the time-varying SOC bound implemented in the centralized controller. Because the hierarchy determines the bound in real-time using a dynamic model, it can better calculate an appropriate lower bound. However, it is important to note the hierarchical controller is



(a) Simulated battery state of charge state trajectory for each control design. The time-vary state bounds are grey in the hierarchical result because the hierarchy does not have knowledge of those bounds.

Figure 5.4: The simulated (a) battery SOC and (b) current state trajectories for each control design.



(b) Simulated battery current state trajectory for each control design.

Figure 5.4 (cont.): The simulated (a) battery SOC and (b) current state trajectories for each control design.

using a linearized model of a non-linear system so the bound output by the upper level in the hierarchy may not provide any guarantees. Furthermore, even though the upper level of the hierarchy is not given explicit knowledge of the time-varying SOC bound, it still manages to predict that the battery needs to recharge before the dash segment. This is best illustrated by Figures 5.4a and 5.5 where the genset begins to ramp up to recharge the battery at approximately 2200 seconds. The average (and max) computation time of the centralized controller is approximately 0.29 (max 0.66) seconds. The average (and max) computation times of the hierarchy upper and lower level controllers are 0.23 (max 0.37) and 0.30 (max 0.39) seconds. By observation, the controllers are running orders of magnitude faster than real-time. Although more complexity could be added to the control designs, it may be advantageous to maintain a lower computation cost such that the controllers could be run on cheaper or lower power processors.

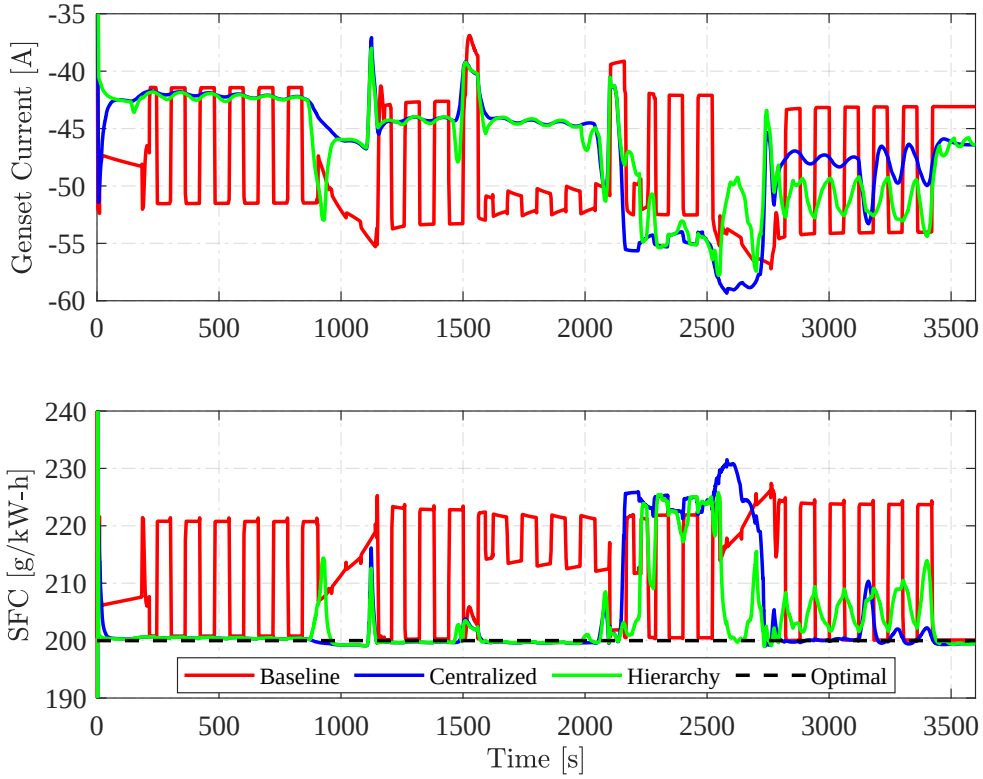


Figure 5.5: Simulation comparison of the genset current and SFC for each control design.

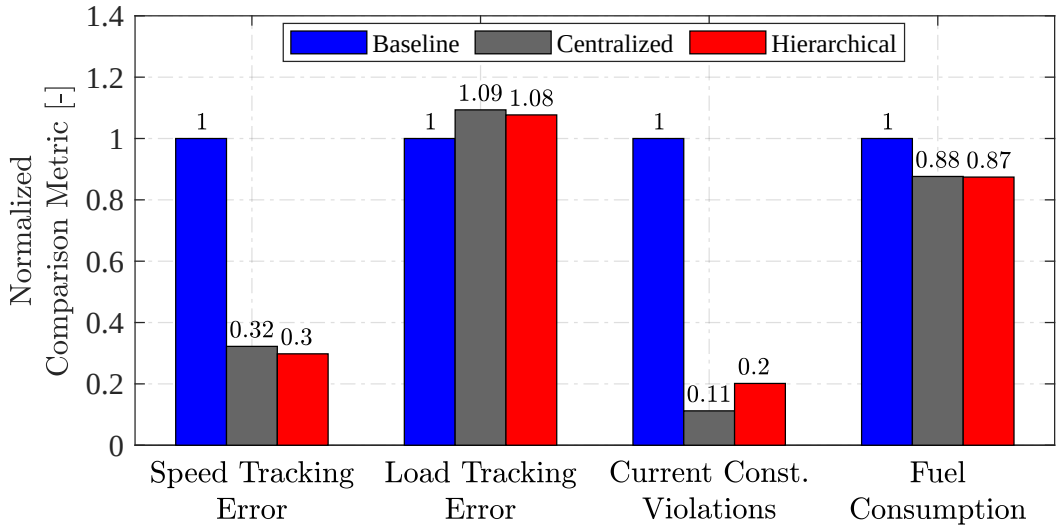


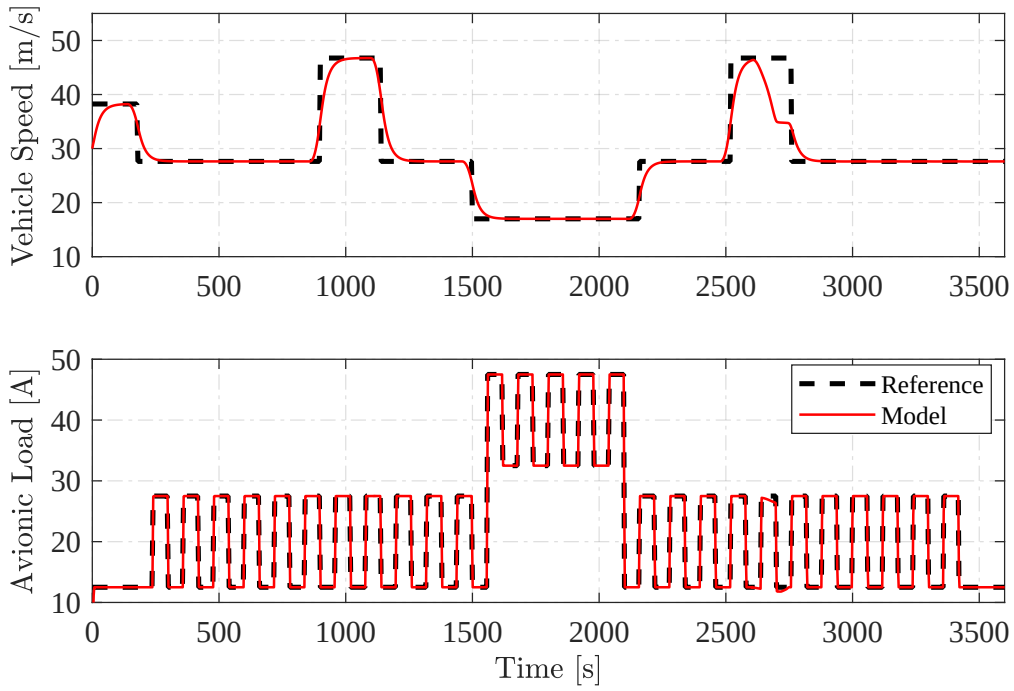
Figure 5.6: Relative comparison between each control design on the figures of merit for the simulated system. The battery SOC constraint violation was neglected because all controllers respected that constraint.

Table 5.5: Figure of merit comparison for each simulated control design.

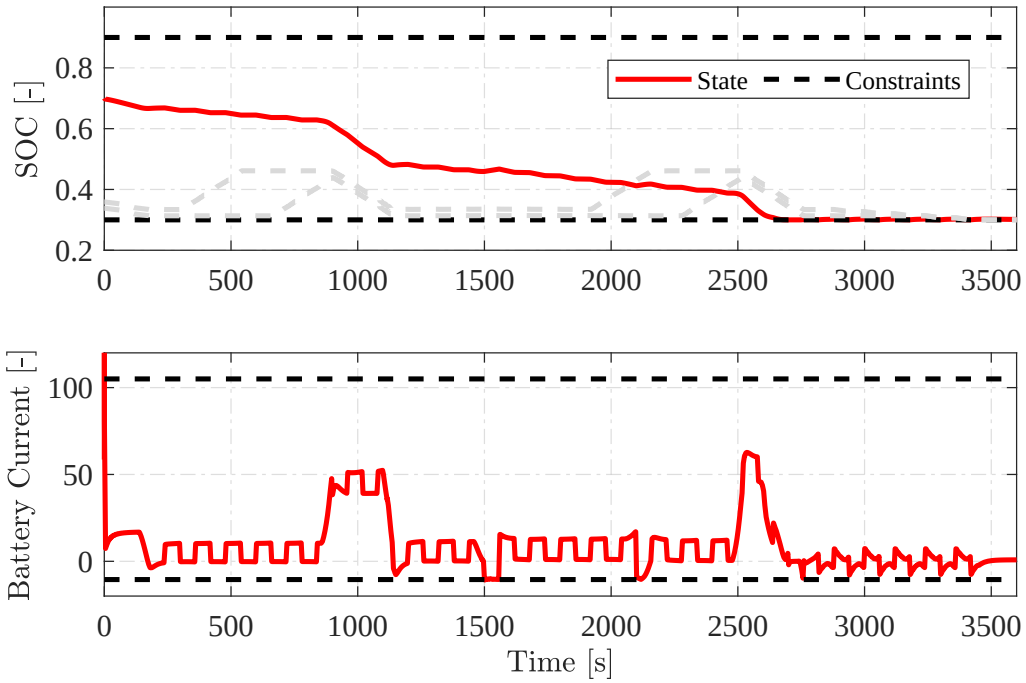
Figure of Merit	State	Baseline	Centralized	Hierarchy
Performance	Vehicle Speed $[m/s]$	4.1×10^5	1.3×10^5	1.2×10^5
	Avionic Load $[A]$	1.0×10^5	1.1×10^5	1.1×10^5
Reliability	Battery SOC $[-]$	0	0	0
	Battery Current $[A]$	-9.5	-1.1	-1.9
Efficiency	Fuel Consumption $[kg]$	626.6	549.1	547.8

Lastly, we claim that the baseline controller is less adaptable than either advanced control design. The baseline controller works well for this architecture, but may not be suitable for other vehicle architectures. For example, this system has a large battery pack (approximately 6kW) and a small genset (approximately 3kW). Because the genset cannot provide peak power, the power share controller must be charge sustaining. Whereas if the genset was large enough to provide peak power, a charge depleting strategy may be more optimal and a new controller would need to be developed. With the advanced control architectures, a simple re-tuning of the same control design would suffice. Furthermore, as already mentioned, the centralized control design was adapted to a hierarchical design by changing only the battery SOC coordination algorithm. Lastly, we claim that the hierarchical design is more adaptable than the centralized design. Although not shown, if objectives change during the mission, the hierarchy can adapt by calculating new SOC bounds while the centralized controller SOC bounds would not update. The modularity and adaptability aspect is quite useful when designing large distributed controllers.

This work presented two methods for integrating long-term mission planning into a fast updating predictive controller design: the time-varying SOC bound algorithm and hierarchical controller. To prove that long-term mission planning is required for these hybrid systems, the centralized controller from Section 3.4 is re-evaluated. However, in this evaluation, the centralized controller uses only the "No Preview" trace shown in Figure 5.2. The results are illustrated in Figure 5.7. By observation, the controller fails the mission at approximately 2600 seconds because it chooses to prematurely decrease the vehicle velocity in order to respect the battery state of charge constraint.



(a) Simulated centralized controller reference tracking result.



(b) Simulated centralized controller battery SOC and current trajectory.

Figure 5.7: The simulated (a) tracking and (b) constrained state results for the centralized controller without knowledge of the time-varying SOC bound.

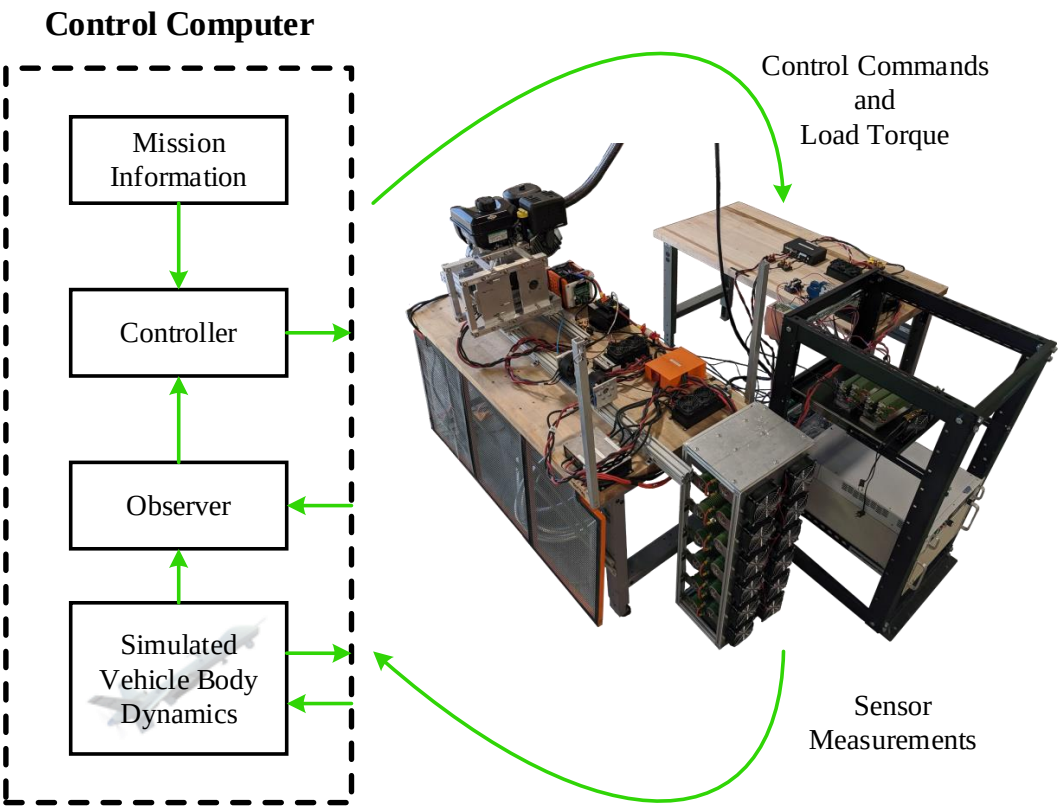


Figure 5.8: The structure of the hardware-in-the-loop setup for the hybrid electric UAV powertrain testbed.

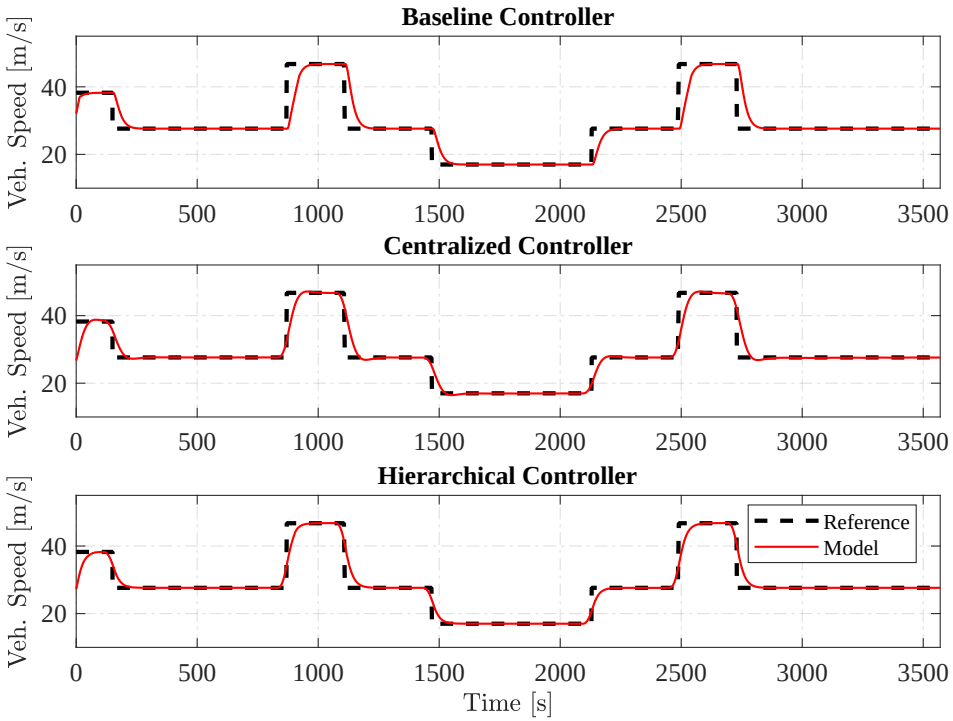
5.5 Experimental Results

The experimental system is operated via a hardware-in-the-loop (HIL) setup best described by Figure 5.8. As mentioned in Chapter 4, the control computer communicates with the testbed's ESCs at a rate of 10Hz. At every update, the control computer sends prop speed, avionic load current, genset, and load torque commands to the ESCs and receives sensor data. The sensor data is immediately communicated to MATLAB (via UDP) which is running the controller, observer, and vehicle body dynamics. The simulated vehicle body dynamics block is input the instantaneous prop speed and outputs a load torque which is sent back to LabVIEW. Measurement information is passed to the observer which estimates the system states. State information is passed to the controller which determines prop speed, avionic load current, and genset commands which are sent back to LabVIEW to be applied to the testbed. To run in real-time, the controllers are run using MATLAB parallel processing.

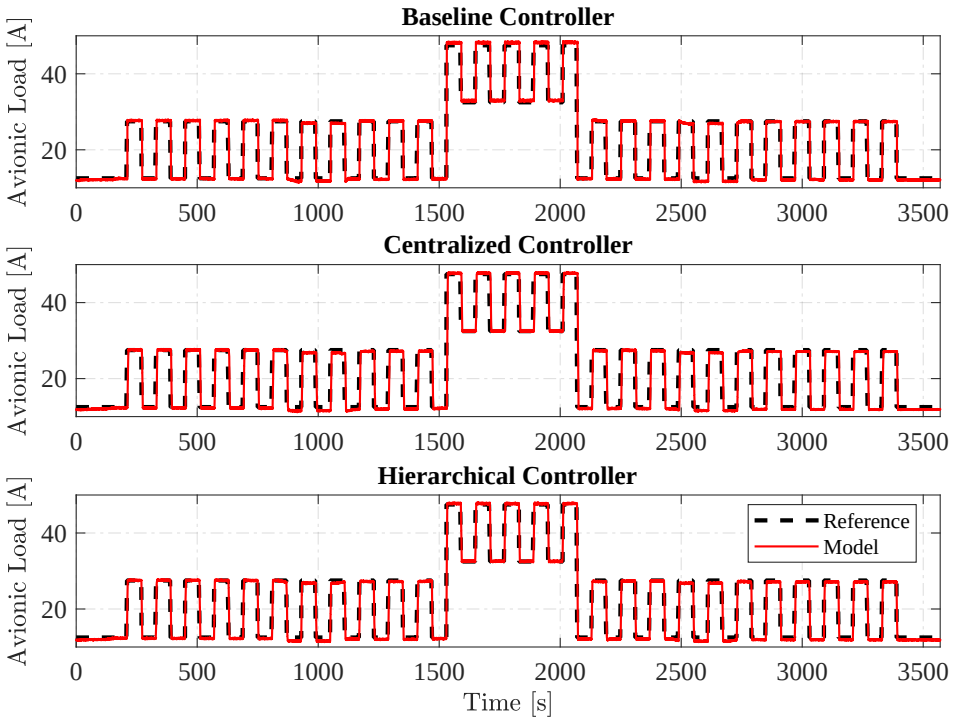
The controllers are formulated using YALMIP [65] and solved using Gurobi [66].

Each control design was experimentally validated using the testbed described in Chapter 4. In addition to the figures of merit, it is important to see that the controllers make similar operating decisions when translated to experimental hardware. First, the performance of each controller is demonstrated in Figure 5.9. By observation, each controller adequately tracks references. The reliability of each controller is illustrated by Figure 5.10. Similar to the simulated result, each controller maintains the battery within state bounds but there are some battery current violations.

Lastly, the efficiency of each control design is demonstrated in Figure 5.11. Note that the estimated genset current is shown here for clarity. Again, the experimental results compare well to the simulated results. The baseline controller aggressively utilizes the engine while the advanced controllers are more conservative. In the advanced control designs, the controller still chooses to decrease the genset current output before large decreases in system load. The main difference between experimental and simulated results is the centralized controller decision to turn off the engine in anticipation of the decrease in vehicle speed at approximately 1150 seconds. In this scenario the cost to turn off the engine was likely lower than the cost to operate the engine at a higher SFC due to imperfect estimation or slightly different state trajectories.

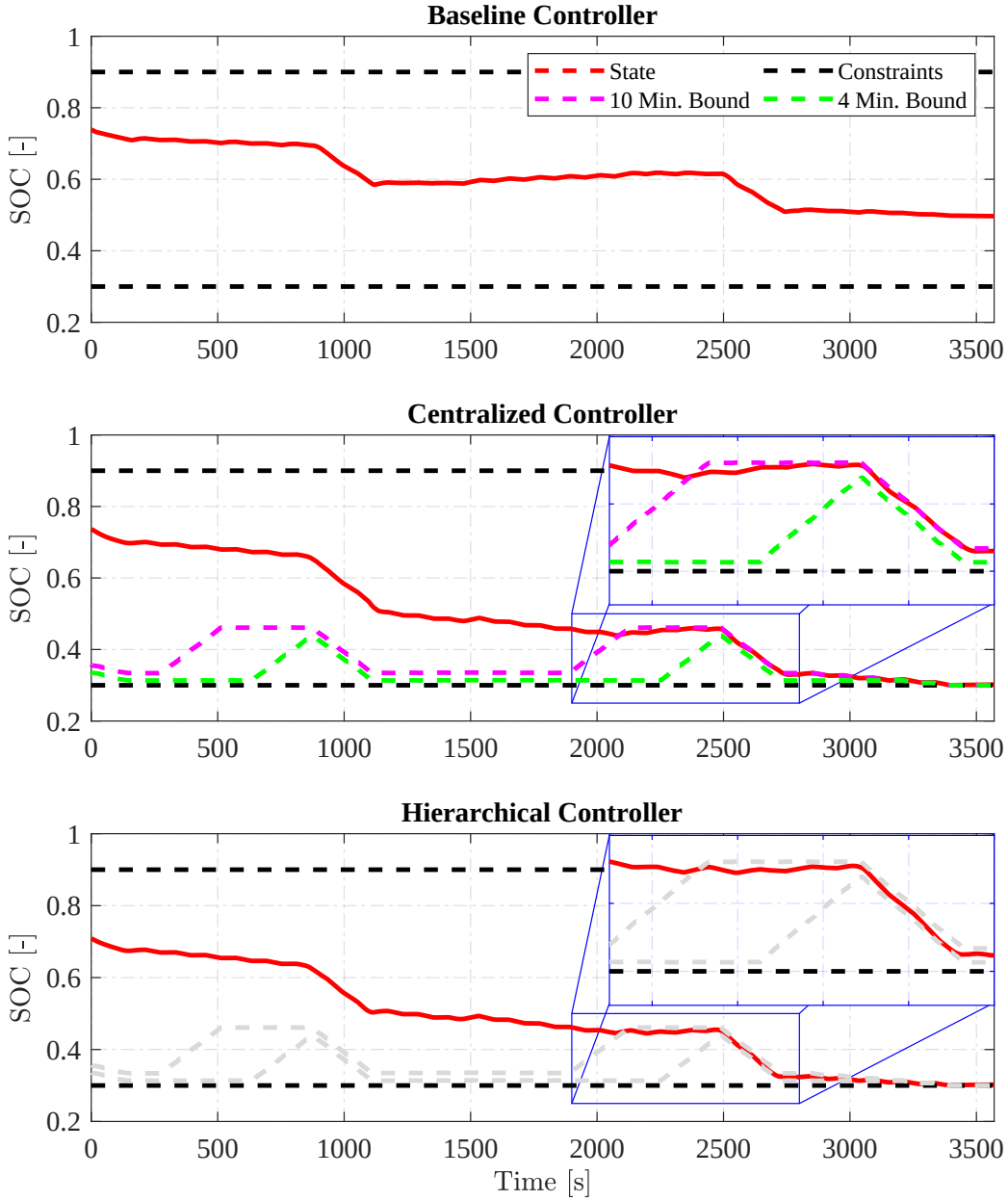


(a) Experimental vehicle velocity tracking result for each control design.



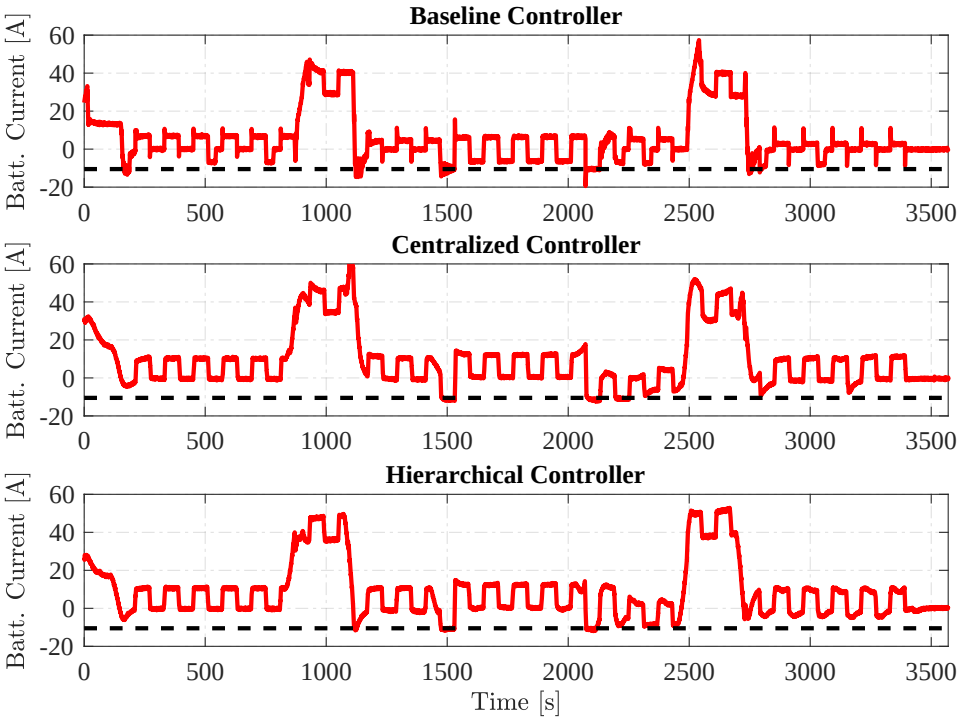
(b) Experimental avionic load current tracking result for each control design.

Figure 5.9: The experimental (a) vehicle velocity and (b) avionic load current tracking results for each control design.



(a) Experimental battery state of charge state trajectory for each control design. The time-vary state bounds are grey in the hierarchical result because the hierarchy does not have knowledge of those bounds.

Figure 5.10: The experimental (a) battery SOC and (b) current state trajectories for each control design.



(b) Experimental battery current state trajectory for each control design.

Figure 5.10 (cont.): The experimental (a) battery SOC and (b) current state trajectories for each control design.

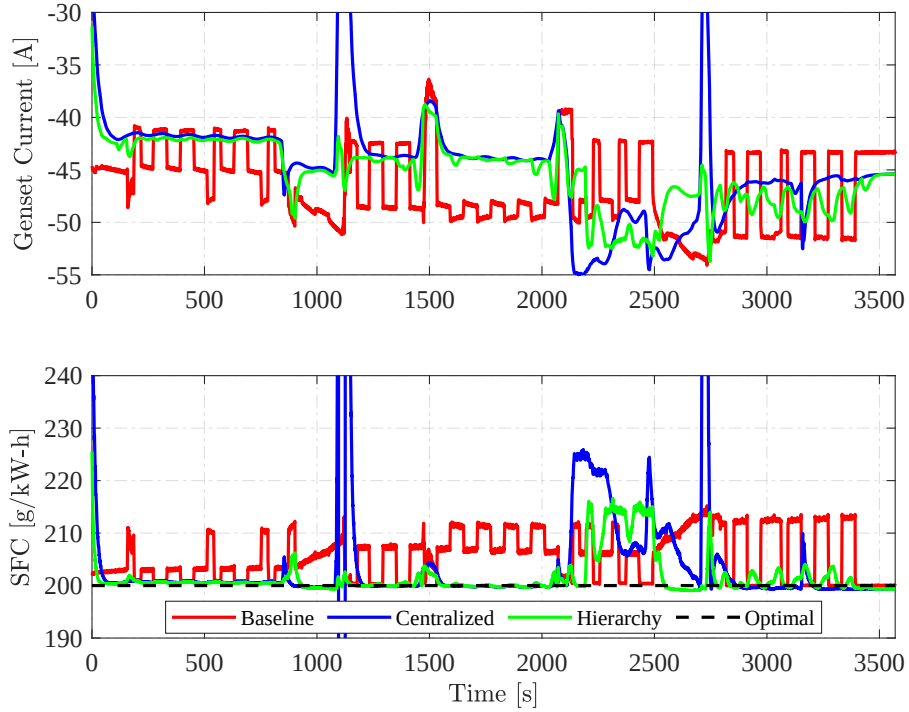


Figure 5.11: Experimental comparison of the genset current and SFC for each control design.

Table 5.6: Figure of merit comparison for each experimental controller validation.

Figure of Merit	State	Baseline	Centralized	Hierarchy
Performance	Vehicle Speed $[m/s]$	4.1×10^5	1.7×10^5	1.3×10^5
	Avionic Load $[A]$	2.2×10^5	1.3×10^5	1.1×10^5
Reliability	Battery SOC $[-]$	0	0	0
	Battery Current $[A]$	-8.9	-2.0	-1.5
Efficiency	Fuel Consumption $[kg]$	575.5	521.9	528.7

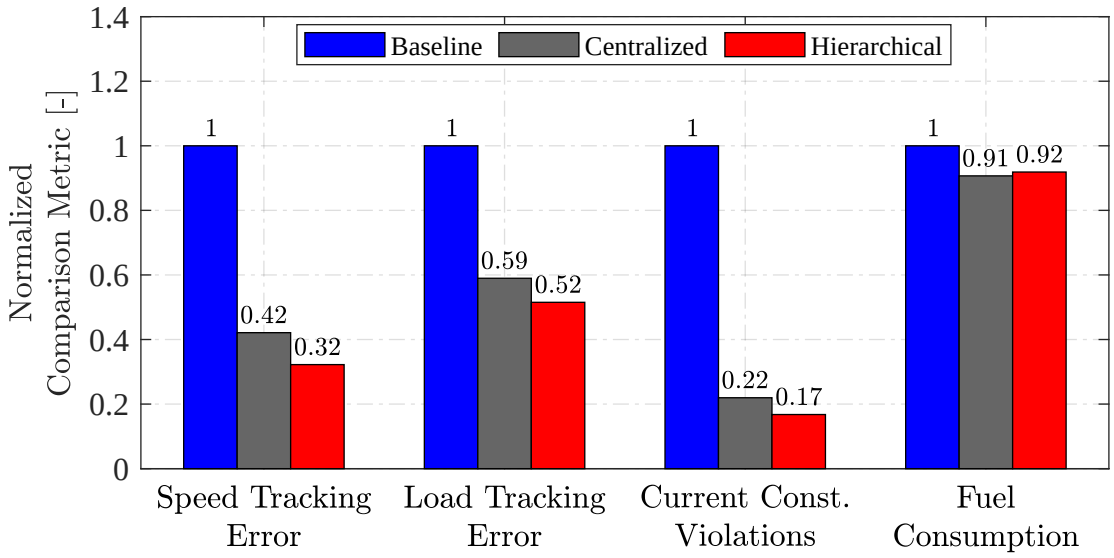


Figure 5.12: Relative comparison between each control design on the figures of merit for the experimental system. The battery SOC constraint violation was neglected because all controllers respected that constraint.

Overall, the controller results translated well to experimental hardware. Figure 5.12 and Table 5.6 compare each control design on the figures of merit. Similar to the simulation results, it is observed that while consuming less fuel, the advanced control designs are more capable at reference tracking and minimizing constraint violations. Here, the advanced controllers use about 91% (9% improvement) of the fuel consumed by the baseline design. It is also especially important to note the effect of mission planning for the centralized and hierarchical control designs. Focusing on Figures 5.10a and 5.11, the centralized controller chooses to increase the genset current output once it enters the time-varying state bound. Although less pronounced, the hierarchical controller increases the genset output to sustain

pack charge once it has preview of the dash segment. A comparison of the figures of merit is listed in Table 5.6. Note that this comparison is included for sake of completeness and that the reader should be aware that there is some variation in the system state initialization (ex: battery SOC). For example, experimentally, the centralized controller consumes less fuel. However, close inspection of Figure 5.10a would indicate that the centralized controller also started at a higher state of charge and would require less fuel to complete the mission.

5.6 Conclusion

This chapter experimentally validated the control designs introduced in Chapter 3 on the experimental platform described in Chapter 4. After the mission and controller parameters were tabulated, simulation results were used to verify the control designs. Simulation results illustrated that the advanced controllers consumed approximately 12.5% less fuel with improved reference tracking and less significant constraint violations in comparison to the baseline design. Furthermore, a brief case study was used to highlight the importance of integrating long term mission planning in the control design. These controllers were validated on the experimental hardware where it was observed again that the advanced controllers yielded a higher performing, more reliable, and more efficient system. The simulated results translated well to experiments.

Chapter 6

Conclusion and Future Work

6.1 Summary of Research Contributions

The increasing trend of electrification has been motivated by the promise of more capable and efficient mobile systems. Novel control algorithms must be developed to accommodate this increasingly large and complex class of vehicles. This thesis utilizes a graph-based modeling framework and predictive control strategies to rapidly design and evaluate model-based controllers for a hybrid unmanned aerial vehicle. The models and controllers are also experimental validated on a novel hybrid electric UAV powertrain testbed.

Chapter 2 introduces an adaptation of the graph-based modeling framework to capture interactions between electrical, mechanical, and thermal system dynamics. A desired system architecture was described and graph-based models for each component and subsystem were formulated. A novel system composition method was introduced to facilitate the development of system-level graph models from a set of core component graph models. Lastly, the composition algorithm was used to develop a hybrid UAV system model. This model was used to facilitate control design in Chapter 3.

Chapter 3 focused on control design. First, models of the testbed embedded controllers were introduced. A baseline controller consisting of a PI vehicle speed regulator and rule-based power share controller were developed. The power share controller was developed to sustain pack charge because the genset could not provide peak power. Two advanced control designs were developed that address challenges of the application of model predictive control to hybrid electric systems with multi-timescale dynamics. Both advanced controllers utilized the same controller to regulate faster system dynamics. Long time horizon mission planning for the centralized controller was implemented via a battery SOC bounding algorithm that

runs prior to mission start. The hierarchical controller planned the battery SOC in real-time via a model predictive controller with a long time horizon.

Chapter 4 introduces a novel hybrid electric UAV powertrain testbed and highlights experimental validation of the component, system, and controller models described in Chapters 2 and 3. The main contribution of this chapter is a detailed aggregation of system identification methods for components of a hybrid system. The results show good validation between the system model and experimental hardware.

Chapter 5 provides simulated and experimental controller validation results. The simulation results show that both advanced control designs offer comparable or better system performance and reliability in comparison to the baseline. Most notable is the advanced controllers' 12.5% improvement in fuel economy over the baseline design. These simulated results were validated on the experimental hardware where it was observed that the advanced control designs offer significant improvements over the baseline design in fuel economy, reliability, and similar performance. Although the hierarchical and centralized controllers were similar in terms of performance, reliability, and efficiency, we claimed that the hierarchical design in more adaptable. The increased adaptability of the hierarchical controller makes it more robust to real-time changes in mission objectives.

6.2 Future Work

This thesis provides a satisfactory analysis of the dynamics and control for the application to a hybrid UAV. However, future work should consider the control of more complex system dynamics with a theoretic emphasis on controller robustness.

6.2.1 Electro-Thermal Interactions

This work mainly considered the electro-mechanical powertrain dynamics. However, it is well known that the dynamics of the electro-mechanical system is strongly influenced by the thermal system state. To provide a more complete analysis of mobile energy systems, the coupling and coordinated control of both energy domains must be considered. Although

mostly neglected in this work, the thermal dynamics of the component models in Chapter 2 will provide means to couple electro-mechanical and thermal systems.

6.2.2 All Electric Operation

Switching between all electric and hybrid operation is a unique ability inherent to hybrid electric systems. There may be specific locations or times during a mission when an aircraft may prefer all electric operation such as near crowded urban areas so as to minimize pollution or noise. However, if electric propulsion is required for a specific mission segment there must exist some minimum battery energy available. Calculating the minimum SOC and planning and optimal trajectory to reach that state is particularly challenging.

6.2.3 Controller Robustness

A formal analysis of controller robustness is a necessary step in control design. Although aircraft missions are thoroughly planned a priori, mission objectives may suddenly change and there are still unknown flight conditions (e.g. wind speeds). Some analysis is required to show that an aircraft will successfully complete a mission in the presence of such disturbances.

References

[1] X. Roboam, B. Sareni, and A. Andrade, "More Electricity in the Air: Toward Optimized Electrical Networks Embedded in More-Electrical Aircraft," *IEEE Industrial Electronics Magazine*, vol. 6, no. 4, pp. 6–17, 2012.

[2] B. Sarlioglu and C. T. Morris, "More Electric Aircraft: Review, Challenges, and Opportunities for Commercial Transport Aircraft," *IEEE Transactions on Transportation Electrification*, vol. 1, no. 1, pp. 54–64, 2015.

[3] C. T. Aksland, T. W. Bixel, L. C. Raymond, M. A. Rottmayer, and A. G. Alleyne, "Graph-Based Electro-Mechanical Modeling of a Hybrid Unmanned Aerial Vehicle for Real-Time Applications," in *American Control Conference*. American Automatic Control Council, 2019.

[4] M. A. Williams, "A Framework for the Control of Electro-Thermal Aircraft Power Systems," Ph.D. dissertation, University of Illinois Urbana-Champaign, Urbana, IL, 2017.

[5] B. J. Brelje and J. R. R. A. Martins, "Progress in Aerospace Sciences Electric, Hybrid, and Turboelectric Fixed-Wing Aircraft : A Review of Concepts, Models, and Design Approaches," *Progress in Aerospace Sciences*, vol. 104, pp. 1–19, 2019. [Online]. Available: https://doi.org/10.1016/j.paerosci.2018.06.004

[6] P. C. Vratny, H. Kuhn, and M. Hornung, "Influences of Voltage Variations on Electric Power Architectures for Hybrid Electric Aircraft," *CEAS Aeronautical Journal*, vol. 8, no. 2017, pp. 31–43, 2017.

[7] S. G. Garrow, C. T. Aksland, S. Sharma, and A. G. Alleyne, "Integrated Modeling for Battery Electric Vehicle Transcritical Thermal Management System," in *American Control Conference*, 2018.

[8] H. E. Perez, J. B. Siegel, X. Lin, and A. G. Stefanopoulou, "Parameterization and Validation of an Integrated Electro-Thermal Cylindrical LFP Battery Model," in *ASME 5th Annual Dynamic Systems and Control Conference*, 2012.

[9] R. H. Jansen, C. Bowman, A. Jankovsky, R. Dyson, and J. Felder, "Overview of NASA Electrified Aircraft Propulsion Research for Large Subsonic Transports," in *53rd AIAA/SAE/ASEE Joint Propulsion Conference*, 2017.

[10] H. A. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, and I. V. Kolmanovsky, "Predictive Energy Management of a Power-Split Hybrid Electric Vehicle," in *American Control Conference*, 2009.

[11] B. P. Rasmussen and A. G. Alleyne, "Dynamic Modeling and Advanced Control of Air Conditioning and Refrigeration Systems," Ph.D. dissertation, University of Illinois Urbana-Champaign, Urbana, IL, 2005.

[12] M. Kania, J. Koeln, A. Alleyne, K. McCarthy, N. Wu, and S. Patnaik, "A Dynamic Modeling Toolbox for Air Vehicle Vapor Cycle Systems," in *SAE Power Systems Conference*, 2012.

[13] "MATLAB Simscape Toolbox 2016a," Natick Massachusetts, United States.

[14] M. Williams, S. Sridharan, S. Banerjee, C. Mak, C. Pauga, P. Krein, A. Alleyne, A. Jacobi, and S. D. Urso, "PowerFlow: A Toolbox for Modeling and Simulation of Aircraft Systems," in *SAE Technical Paper*, 2015.

[15] N. Jalil, N. Kheir, and M. Salman, "A Rule-Based Energy Management Strategy for a Series Hybrid Vehicle," in *American Control Conference*, 1997.

[16] F. R. Salmasi, "Control Strategies for Hybrid Electric Vehicles: Evolution, Classification, Comparison, and Future Trends," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 5, pp. 2393–2404, 2007.

[17] W. Shabbir, "Control Strategies for Series Hybrid Electric Vehicles," Ph.D. dissertation, Imperial College London, 2015.

[18] R. M. Patil, Z. Filipi, and H. K. Fathy, "Comparison of Supervisory Control Strategies for Series Plug-In Hybrid Electric Vehicle Powertrains Through Dynamic Programming," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 502–509, 2014.

[19] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2011.

[20] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control : Theory and Design*, 2nd ed. Nob Hill Publishing, 2016.

[21] D. J. Docimo, H. C. Pangborn, and A. G. Alleyne, "Hierarchical Control for Electro-Thermal Power Management of an Electric Vehicle Powertrain," in *ASME Dynamic Systems and Control Conference,*, 2018.

[22] M. S. Elliott and B. P. Rasmussen, "A Model-Based Predictive Supervisory Controller for Multi-Evaporator HVAC Systems," in *American Control Conference*. IEEE, 2009.

[23] L. Würth, J. B. Rawlings, and W. Marquardt, "Economic Dynamic Real-Time Optimization and Nonlinear Model-Predictive Control on Infinite Horizons," *IFAC Proceedings Volumes*, vol. 42, no. 11, pp. 219–224, 2009.

[24] M. J. Risbeck and J. B. Rawlings, "Economic Model Predictive Control for Time-Varying Cost and Peak Demand Charge Optimization," *IEEE Transactions on Automatic Control*, 2019.

[25] J. P. Koeln, H. C. Pangborn, M. A. Williams, M. L. Kawamura, and A. G. Alleyne, "Hierarchical Control of Aircraft Electro-Thermal Systems," *IEEE Transactions on Control Systems Technology*, 2019.

[26] P. Koeln, Justin, "Hierarchical Power Management in Vehicle Systems," Ph.D. dissertation, University of Illinois Urbana-Champaign, Urbana, IL, 2016.

[27] H. C. Pangborn, "Hierarchical Control for Multi-Domain Coordination of Vehicle Energy Systems with Switched Dynamics," Ph.D. dissertation, University of Illinois Urbana-Champaign, Urbana, IL, 2019.

[28] D. Kim and J. E. Braun, "Hierarchical Model Predictive Control Approach for Optimal Demand Response for Small/Medium-sized Commercial Buildings," in *American Control Conference*. AACC, 2018.

[29] H. Pangborn, A. G. Alleyne, and N. Wu, "A Comparison between Finite Volume and Switched Moving Boundary Approaches for Dynamic Vapor Compression System Modeling," *International Journal of Refrigeration*, vol. 53, pp. 101–114, 2015.

[30] J.-l. Lin and T. Yeh, "Modeling , Identification and Control of Air-Conditioning Systems," *International Journal of Refrigeration*, vol. 30, pp. 209–220, 2007.

[31] T. Ersal, H. K. Fathy, and J. L. Stein, "Structural Simplification of Modular Bond-Graph Models Based on Junction Inactivity," *Simulation Modelling Practice and Theory*, vol. 17, pp. 175–196, 2009.

[32] D. Docimo and A. G. Alleyne, "Electro-Thermal Graph-Based Modeling for Hierarchical Control with Application to an Electric Vehicle," in *2nd IEEE Conference on Control Technology and Applications*, 2018.

[33] H. C. Pangborn, J. P. Koeln, M. A. Williams, and A. G. Alleyne, "Experimental Validation of Graph-Based Hierarchical Control for Thermal Management," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 140, 2018.

[34] C. T. Aksland and A. G. Alleyne, "Experimental Model and Controller Validation for a Series Hybrid Unmanned Aerial Vehicle," 2020, (Under Review).

[35] D. B. West, *Introduction to Graph Theory*, 2nd ed. Pearson Education, 2009.

[36] P. J. Tannous, S. R. Peddada, J. T. Allison, T. Foulkes, R. C. Pilawa-Podgurski, and A. G. Alleyne, "Model-Based Temperature Estimation of Power Electronics Systems," *Control Engineering Practice*, vol. 85, no. 2019, pp. 206–215, 2019.

[37] B. J. M. Kolly, J. Panagiotou, and B. A. Czech, "The Investigation of a Lithium-Ion Battery Fire Onboard a Boeing 787 by the US National Transportation Safety Board," National Transportation Safety Board, Tech. Rep., 2004.

[38] A. E. Fitzgerald, C. Kingsley, and S. D. Umans, *Electric Machinery*, 6th ed. McGraw Hill, 2010.

[39] M. Zeraoulia, M. E. H. Benbouzid, and D. Diallo, "Electric Motor Drive Selection Issues for HEV Propulsion Systems: A comparative study," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 6, pp. 1756–1764, 2006.

[40] S. P. Paul Krause, Oleg Wasynczuk, Scott Sudhoff, *Analysis of Electric Machinery and Drive Systems*, 3rd ed., M. E. El-Hawary, Ed. Wiley, 2002.

[41] MathWorks, "Park Transfrom," 2019. [Online]. Available: https://www.mathworks.com/help/physmod/sps/ref/parktransform.html

[42] M. Soja, M. Ikic, M. Benjanin, and M. Radmanovic, "Improving Efficiency of Power Electronics Converters," *Electronics*, vol. 14, no. 2, pp. 37–42, 2010.

[43] J. Rodríguez, J. Pontt, C. Silva, P. Cortés, U. Amman, and S. Rees, "Predictive Current Control of a Voltage Source Inverter," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, 2004.

[44] O. Ellabban, J. V. Mierlo, P. Lataire, and B. Elsene, "Comparison Between Different PWM Control Methods for Different Z- Source Inverter Topologies Keywords Review of PWM Control Methods for ZSI Simple Boost Control Modified Space Vector PWM (MSVPWM) Control," in *13th European Conference on Power Electronics and Applications*, 2009.

[45] C. J. Oglieve, M. Mohammadpour, and H. Rahnejat, "Optimisation of the Vehicle Transmission and the Gear-Shifting Strategy for the Minimum Fuel Consumption and the Minimum Nitrogen Oxide Emissions," in *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 2017.

[46] W. T. Suit, "Aerodynamic parameters of the Navion airplane extracted from flight," Langley Research Center, Tech. Rep., 1972.

[47] M. R. Hossain, D. G. Rideout, and D. N. Krouglicof, "Bond graph dynamic modeling and stabilization of a quad-rotor helicopter," in *Spring Simulation Multiconference*, 2010, pp. 1–9.

[48] G. J. Flynn, "The Physics of Aircraft Flight," American Institution of Physics Teachers, Tech. Rep., 1987.

[49] C. E. Laird and A. G. Alleyne, "A Hybrid Electro-Thermal Energy Storage System For High Ramp Rate Power Applications," in *Dynamic Systems and Control Conference*, 2019, pp. 1–9.

[50] R. Whitt, B. Nafis, D. Huitink, Z. Yuan, A. Deshpande, B. Narayanasamy, and F. Luo, "Heat Transfer and Pressure Drop Performance of Additively Manufactured Polymer Heat Spreaders for Low-Weight Directed Cooling Integration in Power Electronics," in *18th IEEE ITherm Conference*. IEEE, 2019.

[51] D. J. Docimo, Z. Kang, K. A. James, and A. G. Alleyne, "A Novel Framework for Simultaneous Topology and Sizing Optimization of Complex, Multi-Domain Systems-of-Systems," *ASME Journal of Mechanical Design*, 2020.

[52] S. R. Peddada, D. R. Herber, H. C. Pangborn, A. G. Alleyne, and J. T. Allison, "Optimal Flow Control and Single Split Architecture Exploration for Fluid-Based Thermal Management," *Journal of Mechanical Design, Transactions of the ASME*, vol. 141, 2019.

[53] T. O. Deppen and A. G. Alleyne, "A Model Predictive Framework For Thermal Management of Aircraft," in *Dynamic Systems and Control Conference*, 2015.

[54] N. Jain and B. M. Hencey, "Increasing Fuel Thermal Management System Capability via Objective Function Design," in *American Control Conference*. American Automatic Control Council (AACC), 2016.

[55] S. Sharma, "Extremum Seeking Control of Battery Powered Vapor Compression Systems for Vehicles," M.S. thesis, Univertity of Illinois Urbana-Champaign, Urbana, IL, 2018.

[56] J. P. Koeln and A. G. Alleyne, "Event-Based Hierarchical Control for Power Flow in Vehicle Systems," in *American Control Conference*. American Automatic Control Council (AACC), 2016.

[57] Z. Yu, "Space-Vector PWM With TMS320C24x/F24x Using Hardware and Software Determined Switching Patterns," Texas Instruments, Tech. Rep. March 1999, 1999.

[58] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein, "A Stochastic Optimal Control Approach for Power Management in Plug-In Hybrid Electric Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 545–555, 2011.

[59] M. Williams, "A Hierarchical Control Strategy For Aircraft Thermal Systems," M.S. thesis, University of Illinois Urbana-Champaign, Urbana, IL, 2014.

[60] R. v. d. M. Eric Wan, *Kalman Filtering and Neural Networks Chapter 7*, 1st ed., S. Haykin, Ed. John Wiley & Sons, 2008.

[61] G. L. Plett, "Sigma-Point Kalman Filtering for Battery Management Systems of LiPB-Based HEV Battery Packs. Part 1: Introduction and State Estimation," *Journal of Power Sources*, vol. 161, no. 2006, 2006.

[62] B. Hieb and H. Hofmann, "Parameterizing and Verifying a Permanent Magnet Synchronous Motor Model," 2012. [Online]. Available: https://bit.ly/2rpmbrn

[63] J. T. Zumberge, "Validation of a DC-DC Boost Circuit Model and Control Algorithm," Ph.D. dissertation, University of Dayton, Dayton, OH, 2015.

[64] A. Gangopadhyay and P. Meckl, "Modeling, Validation and System Identification of a Natural Gas Engine," in *American Control Conference*, vol. 1, no. June, 1997.

[65] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," *IEEE International Symposium on Computer-Aided Control System Design*, pp. 284–289, 2004.

[66] G. O. LLC, "Gurobi Optimizer Reference Manual," 2019. [Online]. Available: http://www.gurobi.com

[67] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, 1st ed. Cambridge University Press, 1992.

Appendix A

Multi-Domain Graph Development

A.1 System Composition

A motor controlled by a converter is a simple example highlighting the composition process of a multi-domain graph model. The graphs of each component described in Sections 2.3.2 and 2.3.3.1 are repeated below with updated vertex and edge labels useful for the following discussion (Figure A.1). Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be defined as the component graphs for the converter and motor respectively. The incidence matrix for each graph model is given by

$$M_1 = \begin{bmatrix} -1 & 1 & 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{A.1a}$$

$$M_2 = \begin{bmatrix} -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{A.1b}$$

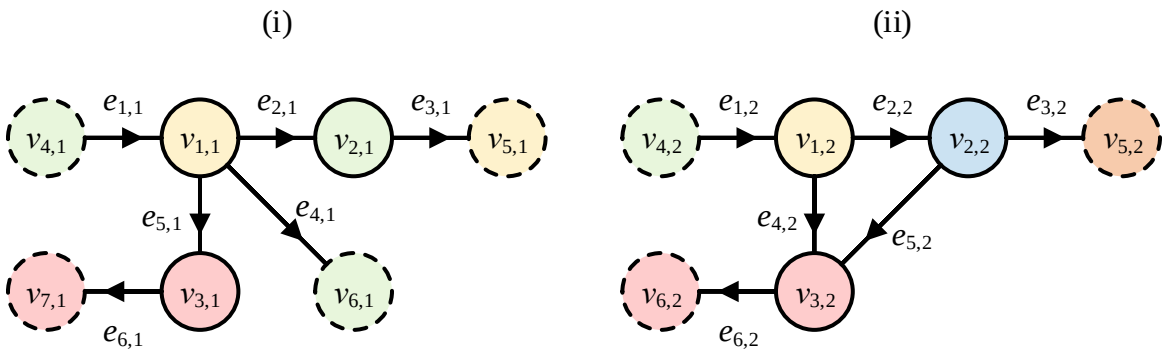


Figure A.1: The (i) buck-boost converter and (ii) motor graph models.

The state vector, capacitance vector, power flow coefficient, and property look-up coefficient vector for each component were defined in Sections 2.3.2 and 2.3.3.1. The following discussion will step through the processes described in Section 2.4.

We aim to develop a system graph model for a motor controlled by a converter. Assume that both components exist in the same ambient environment. Because the converter is electrically connected to the motor, it should be intuitive that the electrical power out of the converter along edge $e_{3,1}$ should be equivalent to the electrical power entering the motor along edge $e_{1,2}$. Similarly, the components are placed in the same ambient environment so they should share the same ambient temperature or cooling state (vertices $v_{7,1}$ and $v_{6,2}$).

Based on the description in Section 2.4, we have a set of 2 component graphs $\mathcal{C} = (\mathcal{G}_1, \mathcal{G}_2)$. The set of graph vertices and edges are defined by

$$\mathcal{V}_1 = \{v_{1,1},\ v_{2,1},\ v_{3,1},\ v_{4,1},\ v_{5,1},\ v_{6,1},\ v_{7,1}\}\,, \tag{A.2a}$$

$$\mathcal{E}_1 = \{e_{1,1},\ e_{2,1},\ e_{3,1},\ e_{4,1},\ e_{5,1},\ e_{6,1}\}\,, \tag{A.2b}$$

$$\mathcal{V}_2 = \{v_{1,2},\ v_{2,2},\ v_{3,2},\ v_{4,2},\ v_{5,2},\ v_{6,2}\}\,, \tag{A.2c}$$

$$\mathcal{E}_2 = \{e_{1,2},\ e_{2,2},\ e_{3,2},\ e_{4,2},\ e_{5,2},\ e_{6,2}\}\,, \tag{A.2d}$$

The component graph vertex and edges sets are combined according to Section 2.4.2.

$$\chi = \{v_{1,1},\ v_{2,1},\ v_{3,1},\ v_{4,1},\ v_{5,1},\ v_{6,1},\ v_{7,1},\ v_{1,2},\ v_{2,2},\ v_{3,2},\ v_{4,2},\ v_{5,2},\ v_{6,2}\}, \tag{A.3a}$$

$$\bar{\chi} = \{v_{1,1},\ v_{2,1},\ v_{3,1},\ v_{1,2},\ v_{2,2},\ v_{3,2}\}, \tag{A.3b}$$

$$\underline{\chi} = \{v_{4,1},\ v_{5,1},\ v_{6,1},\ v_{7,1},\ v_{4,2},\ v_{5,2},\ v_{6,2}\}, \tag{A.3c}$$

$$\Xi = \{e_{1,1},\ e_{2,1},\ e_{3,1},\ e_{4,1},\ e_{5,1},\ e_{6,1},\ e_{1,2},\ e_{2,2},\ e_{3,2},\ e_{4,2},\ e_{5,2},\ e_{6,2}\}. \tag{A.3d}$$

Next, the user defined connection sets Λ and Σ are developed based of the desired system model. Based on the previous discussion, there is one edge equivalency and three vertex equivalencies. The edge equivalency is apparent since the power leaving the converter is equivalent to the power entering the motor. Two of the vertex equivalencies result from the edge connection and the third vertex equivalency is a result of the shared ambient temperature state. Mathematically these equivalencies are defined by

$$\Lambda = \{\{v_{2,1},\ v_{4,2}\}, \{v_{5,1},\ v_{1,2}\}, \{v_{7,1},\ v_{6,2}\}\}, \tag{A.4a}$$

$$\Sigma = \{\{e_{3,1},\ e_{1,2}\}\}. \tag{A.4b}$$

The final preparatory step is to partition Λ, χ, and Ξ.

$$\bar{\Lambda} = \{\{v_{2,1},\ v_{4,2}\}, \{v_{5,1},\ v_{1,2}\}\}, \tag{A.5a}$$

$$\underline{\Lambda} = \{\{v_{7,1},\ v_{6,2}\}\}, \tag{A.5b}$$

$$\hat{\chi} = \{v_{2,1},\ v_{5,1},\ v_{7,1},\ v_{4,2},\ v_{1,2},\ v_{6,2}\}, \tag{A.5c}$$

$$\hat{\Xi} = \{e_{3,1},\ e_{1,2}\}, \tag{A.5d}$$

$$\tag{A.5e}$$

Now, according the definitions of (2.58) and (2.59) in Section 2.4.2, the vertex and edge property map partitions are developed.

$$V^{\bar{t}\bar{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T, \tag{A.6a}$$

$$V^{\bar{t}c} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \tag{A.6b}$$

$$V^{tc} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \tag{A.6c}$$

$$V^{t\bar{c}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T, \tag{A.6d}$$

$$E^{\bar{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \tag{A.6e}$$

$$E^{c} = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T. \tag{A.6f}$$

The partitioned vertex and edge property maps can be combined according to (2.58) and (2.59) and the resulting system level incidence matrix M_s is calculated using (2.57). The system graph is shown in Figure A.2. Equation (2.56) can be used to map other component properties to system properties.

$$M_s = \begin{bmatrix} -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A.7}$$

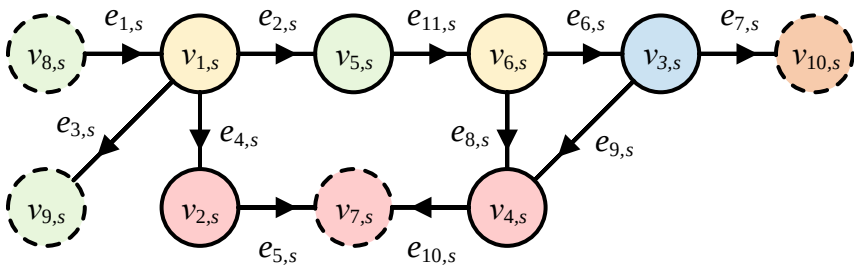


Figure A.2: The graph model for a motor controlled by a converter in a shared ambient environment.

A.2 Modified Graph Formulation

The previous section described the process to develop a system model in the generic graph formulation. However, the model must be transformed into the modified graph formulation before it can be used for control design. The purpose of this section is to illustrate how a generic graph model can be transformed into a modified graph model. We could reuse the converter-motor example to describe the process. However, it is a large system and that may lead to some confusion. Instead we opt for a simpler two state system described by Figure A.3 with the following graph properties.

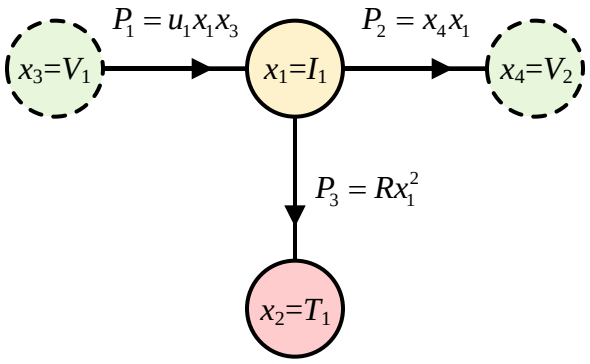


Figure A.3: A sample graph used to illustrate the process of developing a modified graph model. The states and power flows are defined in the figure.

$$x = \begin{bmatrix} I_1 & T_1 \end{bmatrix}^T, \tag{A.8a}$$

$$C = \begin{bmatrix} L_1 I_1 & C_T \end{bmatrix}^T, \tag{A.8b}$$

$$c = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \tag{A.8c}$$

$$f = \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{A.8d}$$

The incidence matrix (2.4) and power flow vector (2.9) for this system are given by

$$M = \begin{bmatrix} -1 & 1 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{A.9a}$$

$$P = \begin{bmatrix} F_1(x_1^{tail}, x_1^{head}, u_1) = u_1 x_1 x_3 \\ F_2(x_2^{tail}, x_2^{head}, 0) = x_4 x_1 \\ F_3(x_3^{tail}, x_3^{head}, 0) = R x_1^2 \end{bmatrix} \tag{A.9b}$$

For simplicity, only consider the thermal domain ($\mathcal{T}_v = 1$), voltage domain ($\mathcal{T}_v = 2$), and current domain ($\mathcal{T}_v = 3$). The vertex type vector for this system is $\mathcal{T}_v = \begin{bmatrix} 3 & 1 & 2 & 2 \end{bmatrix}^T$. It follows from (2.10b) and (2.11) that the modified capacitance and incidence matrix are given by

$$C^{\ddagger} = \begin{bmatrix} \dfrac{L_1 I_1}{I_1} & 0 \\ 0 & C_T \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ 0 & C_T \end{bmatrix} \tag{A.10a}$$

$$M^{\ddagger} = \overbrace{\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right.}^{M_1^{\ddagger}} \overbrace{\left|\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{array}\right.}^{M_2^{\ddagger}} \overbrace{\left.\begin{array}{ccc} -1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]}^{M_3^{\ddagger}} \tag{A.10b}$$

Next, the generic power flow vector can be converted to the modified power flow vector by utilizing (2.12).

$$P^{\ddagger} = \left[\begin{array}{c} F_1(x_1^{tail}, x_1^{head}, u_1) \\ F_2(x_2^{tail}, x_2^{head}, 0) \\ F_3(x_3^{tail}, x_3^{head}, 0) \\ \hline F_1(x_1^{tail}, x_1^{head}, u_1)/x_1^{tail} \\ F_2(x_2^{tail}, x_2^{head}, 0)/x_2^{head} \\ 0 \\ \hline F_1(x_1^{tail}, x_1^{head}, u_1)/x_1^{head} \\ F_2(x_2^{tail}, x_2^{head}, 0)/x_2^{tail} \\ F_3(x_3^{tail}, x_3^{head}, 0)/x_3^{tail} \end{array}\right] = \left[\begin{array}{c} u_1 x_1 x_3 \\ x_4 x_1 \\ R x_1^2 \\ \hline u_1 x_1 \\ x_1 \\ 0 \\ \hline u_1 x_3 \\ x_4 \\ R x_1 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c} a \\ b \\ c \end{array}}\right\} P_1^{\ddagger} \\ \left.\vphantom{\begin{array}{c} a \\ b \\ c \end{array}}\right\} P_2^{\ddagger} \\ \left.\vphantom{\begin{array}{c} a \\ b \\ c \end{array}}\right\} P_3^{\ddagger} \end{array} . \tag{A.11}$$

Based on the modified power flow equation (2.13), the modified power flow coefficients are given by

$$
b = \begin{array}{c} \\ e_{1,1} \\ e_{1,2} \\ e_{1,3} \\ e_{2,1} \\ e_{2,2} \\ e_{2,3} \\ e_{3,1} \\ e_{3,2} \\ e_{3,3} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} \end{array} \\
\left[\begin{array}{ccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\end{array}, \tag{A.12}
$$

where edge $e_{i,j}$ corresponds to the modified power flow $P^{\ddagger}_{i,j}$. The combination of (A.10) and (A.11) provide sufficient information to utilize the model for control design.

Appendix B

Runge-Kutta 4 Discretization

Runge-Kutta 4 (RK4) is the fourth-order variation of the more generalized Runge-Kutta method [67]. For the following initial value problem

$$\dot{x} = f(t, x), \quad x(t_0) = x_0, \tag{B.1}$$

the RK4 algorithm is as follows

$$x_{n+1} \approx x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}, \tag{B.2a}$$

$$t_{n+1} = t_n + \Delta t, \tag{B.2b}$$

$$k_1 = \Delta t f(t_n, x_n), \tag{B.2c}$$

$$k_2 = \Delta t f(t_n + \frac{\Delta t}{2}, x_n + \frac{k_1}{2}), \tag{B.2d}$$

$$k_3 = \Delta t f(t_n + \frac{\Delta t}{2}, x_n + \frac{k_2}{2}), \tag{B.2e}$$

$$k_4 = \Delta t f(t_n + \Delta t, x_n + k_3), \tag{B.2f}$$

$$\tag{B.2g}$$

for step size Δt and index $n \in \{0, 1, 2, \ldots\}$. Note the approximation in (B.2a) because there exist higher order terms $\mathcal{O}(\Delta t^5)$.

We seek to discretize the continuous time linear DAE system. Consider the RK4 discretization of the state dynamics defined in (2.19a)

$$\begin{bmatrix} x_{(k+1),d} \\ 0 \end{bmatrix} = A_{r1} \begin{bmatrix} x_{k,d} \\ x_{k,a} \end{bmatrix} + A_{r2} x_k^t + B_{r1} u_k + B_{r2} + D_r P_k^s \quad \text{where,} \tag{B.3a}$$

$$A_{r1} = \begin{bmatrix} P\left(\bar{A}'_{1,d}\right) & Q\left(\bar{A}'_{1,a}, \bar{A}'_{1,d}\right) \\ \underline{A}'_{1,d} & \underline{A}'_{1,a} \end{bmatrix}, \tag{B.3b}$$

$$A_{r2} = \begin{bmatrix} Q\left(\bar{A}'_{2}, \bar{A}'_{1,d}\right) \\ \underline{A}'_{2} \end{bmatrix}, \tag{B.3c}$$

$$B_{r1} = \begin{bmatrix} Q\left(\bar{B}'_{1}, \bar{A}'_{1,d}\right) \\ \underline{B}'_{1} \end{bmatrix}, \tag{B.3d}$$

$$B_{r2} = \begin{bmatrix} Q\left(\bar{B}'_{2}, \bar{A}'_{1,d}\right) \\ \underline{B}'_{2} \end{bmatrix}, \tag{B.3e}$$

$$D_{r} = \begin{bmatrix} Q\left(\bar{D}', \bar{A}'_{1,d}\right) \\ \underline{D}' \end{bmatrix}, \tag{B.3f}$$

$$P(Z) = \frac{1}{6}\left(6\Delta t Z + 3\Delta t^2 Z^2 + +\Delta t^3 Z^3 + \frac{1}{4}\Delta t^4 Z^4\right) + I, \tag{B.3g}$$

$$Q(Y, Z) = \frac{1}{6}\left(6\Delta t Y + 3\Delta t^2 Y Z + +\Delta t^3 Y Z^2 + \frac{1}{4}\Delta t^4 Y Z^3\right). \tag{B.3h}$$

where $I \in \mathbb{R}^{N_d \times N_d}$ and N_d is the total number of dynamic states.

Appendix C

Observer Algorithms

C.1 Definitions

In this section we define terminology and notation useful for the following Kalman filtering algorithms. First, the following algorithms will utilize the same general state-space model

$$x_k = f\left(x_{k-1}u_{k-1}, w_{k-1}, k-1\right), \tag{C.1a}$$

$$y_k = h\left(x_k, u_k, v_k, k\right), \tag{C.1b}$$

where k is the time index, $x \in \mathbb{R}^N$ is the state vector, u is the input vector, w is the process noise/disturbance, y is the output vector, and v is the sensor noise. The notation $a_{i|j}$ indicates the numerical value of variable a at index i given information at index j. An estimate of variable a is signified with a tilde $\tilde{a}$. The operator $\mathbb{E}(a)$ is the expected value of variable a. We assume additive, zero-mean, process and sensor noise with covariance Q and R (i.e. $w_k \sim N(0, Q)$ and $v_k \sim N(0, R)$) . The error covariance is defined by the matrix P.

C.2 Central-Difference Kalman Filter Algorithm

A central-difference Kalman filter is easy to tune since there is a single tuning parameter h. The choice of h defines the following five weightings used in the algorithm,

$$\gamma = h, \tag{C.2a}$$

$$\alpha_0^{(m)} = \frac{h^2 - N}{h^2}, \tag{C.2b}$$

$$\alpha_j^{(m)} = \frac{1}{2h^2}, \tag{C.2c}$$

$$\alpha_0^{(c)} = \frac{h^2 - N}{h^2}, \tag{C.2d}$$

$$\alpha_j^{(c)} = \frac{1}{2h^2}, \tag{C.2e}$$

where $N = \dim\{x\}$ and $j \in [1 : 2N]$. See that $\alpha^{(m)}$ and $\alpha^{(c)}$ are vectors. From [61], the default value for h is typically $\sqrt{3}$. In this work $h = 1.2$. The following algorithm reiterates the central-difference Kalman filter as described in [60, 61].

Algorithm:

1. **Initialization:**

   Initialize the state estimate and error covariance.

$$\tilde{x}_{0|0} = \mathbb{E}\left(x_0\right), \tag{C.3a}$$

$$P_{0|0} = \mathbb{E}\left(\left(x_0 - \tilde{x}_{0|0}\right)\left(x_0 - \tilde{x}_{0|0}\right)^T\right). \tag{C.3b}$$

2. **Calculate Sigma Points:** Repeat steps (2)-(6) for $k \in \{1, 2, \ldots\}$.

   Calculate the sigma points $\mathcal{X}$

$$\mathcal{X}_{k-1|k-1} = \begin{bmatrix} \tilde{x}_{k-1|k-1} & \tilde{x}_{k-1|k-1} + \gamma\sqrt{P_{k-1|k-1}} & \tilde{x}_{k-1|k-1} - \gamma\sqrt{P_{k-1|k-1}} \end{bmatrix} \tag{C.4}$$

   where the matrix square root can be implemented via Cholesky Factorization. Note that $\mathcal{X} \in \mathbb{R}^{N \times (1+2N)}$.

3. **Time Update (Prediction) Step:**

   Predict the system sigma point state using the process model (C.1a). Then update the system state estimate and error covariance.

$$\mathcal{X}'_{k|k-1} = f\left(\mathcal{X}_{k-1|k-1}, u_{k-1}, 0, k-1\right), \tag{C.5a}$$

$$\tilde{x}_{k|k-1} = \sum_{i=0}^{2N} \alpha_i^{(m)} \mathcal{X}'_{i,k|k-1}, \tag{C.5b}$$

$$P_{k|k-1} = \sum_{i=0}^{2N} \alpha_i^{(c)} \left(\mathcal{X}'_{i,k|k-1} - \tilde{x}_{k|k-1}\right)\left(\mathcal{X}'_{i,k|k-1} - \tilde{x}_{k|k-1}\right)^T + Q. \tag{C.5c}$$

4. **Sigma Point Augmentation:**

   Based on the literature, any 1 of the following 3 equations can be used to augment the sigma point states. Note that if the third option is chosen, $N \to 2N$ and the weightings α should be recalculated.

$$\mathcal{X}_{k|k-1} = \mathcal{X}'_{k|k-1}, \tag{C.6a}$$

$$\mathcal{X}_{k|k-1} = \begin{bmatrix} \tilde{x}_{k|k-1} & \tilde{x}_{k|k-1} + \gamma\sqrt{P_{k|k-1}} & \tilde{x}_{k|k-1} - \gamma\sqrt{P_{k|k-1}} \end{bmatrix} \tag{C.6b}$$

$$\mathcal{X}_{k|k-1} = \begin{bmatrix} \mathcal{X}'_{k|k-1} & \mathcal{X}'_{0,k|k-1} + \gamma\sqrt{Q} & \mathcal{X}'_{0,k|k-1} - \gamma\sqrt{Q} \end{bmatrix}. \tag{C.6c}$$

5. **Output Estimation Step:**

   The sigma point output $\mathcal{Y}$ is calculated using the measurement model (C.1b). Then estimate the system output.

$$\mathcal{Y}_{k|k-1} = h\left(\mathcal{X}_{k|k-1}, u_k, 0, k\right), \tag{C.7a}$$

$$\tilde{y}_{k|k-1} = \sum_{i=0}^{2N} \alpha_i^{(m)} \mathcal{Y}_{i,k|k-1}. \tag{C.7b}$$

6. **Measurement Update (Correction) Step:**

   The observer gain L_k is calculated and used to update the state estimate and error covariance. Here, $\tilde{x}_{k|k}$ is the state estimate output by the observer at time index k.

$$U_k = \sum_{i=0}^{2N} \alpha_i^{(c)} \left(\left(\mathcal{Y}_{i,k|k-1} - \tilde{y}_{k|k-1}\right)\left(\mathcal{Y}_{i,k|k-1} - \tilde{y}_{k|k-1}\right)^T\right) + R, \tag{C.8a}$$

$$V_k = \sum_{i=0}^{2N} \alpha_i^{(c)} \left(\left(\mathcal{X}_{i,k|k-1} - \tilde{x}_{k|k-1}\right)\left(\mathcal{Y}_{i,k|k-1} - \tilde{y}_{k|k-1}\right)^T\right), \tag{C.8b}$$

$$L_k = V_k U_k^{-1}, \tag{C.8c}$$

$$\tilde{x}_{k|k} = \tilde{x}_{k|k-1} + L_k\left(y_k - y_{k|k-1}\right), \tag{C.8d}$$

$$P_{k|k} = P_{k|k-1} - L_k U_k K_k^T. \tag{C.8e}$$

C.3 Extended Kalman Filter Algorithm

The extended Kalman filter (EKF) is an adaptation of the linear Kalman filter (LKF) for nonlinear systems in which the nonlinear system is linearized at each update. The algorithm is provided below [61].

Algorithm:

1. **Initialization:**

   Initialize the state estimate and error covariance.

$$\tilde{x}_{0|0} = \mathbb{E}\left(x_0\right), \tag{C.9a}$$
$$P_{0|0} = \mathbb{E}\left(\left(x_0 - \tilde{x}_{0|0}\right)\left(x_0 - \tilde{x}_{0|0}\right)^T\right). \tag{C.9b}$$

2. **Time Update (Prediction) Step:** Repeat steps (2) and (3) for $k \in \{1, 2, \ldots\}$.

   Linearize the nonlinear process model (C.1a), predict the system state, and update the error covariance.

$$A = \left.\frac{\partial f}{\partial x}\right|_{x=\tilde{x}_{k-1|k-1}, u=u_{k-1}, w=0, k=k-1}, \tag{C.10a}$$
$$\tilde{x}_{k|k-1} = f\left(\tilde{x}_{k-1|k-1}, u_{k-1}, 0, k-1\right), \tag{C.10b}$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q. \tag{C.10c}$$

3. **Measurement Update (Correction) Step:**

   Linearize the measurement model (C.1b), calculate the observer gain L_k, and correct the state estimate and error covariance.

$$C = \left.\frac{\partial h}{\partial x}\right|_{x=\tilde{x}_{k|k-1}, u=u_k, v=0, k=k}, \tag{C.11a}$$

$$L_k = P_{k|k-1}C^T \left(CP_{k|k-1}C^T + R\right)^{-1}, \tag{C.11b}$$

$$\tilde{x}_{k|k} = \tilde{x}_{k|k-1} + L_k \left(y_k - h(\tilde{x}_{k|k-1}, u_k, 0, k)\right), \tag{C.11c}$$

$$P_{k|k} = (I - L_k C)P_{k|k-1}. \tag{C.11d}$$

Appendix D

Testbed Parts List

This section lists the hardware on the hybrid electric UAV powertrain testbed in Tables D.1 and D.2. The powertrain components are shown in Figure D.1 and a complete view of the testbed is shown in Figure D.2.

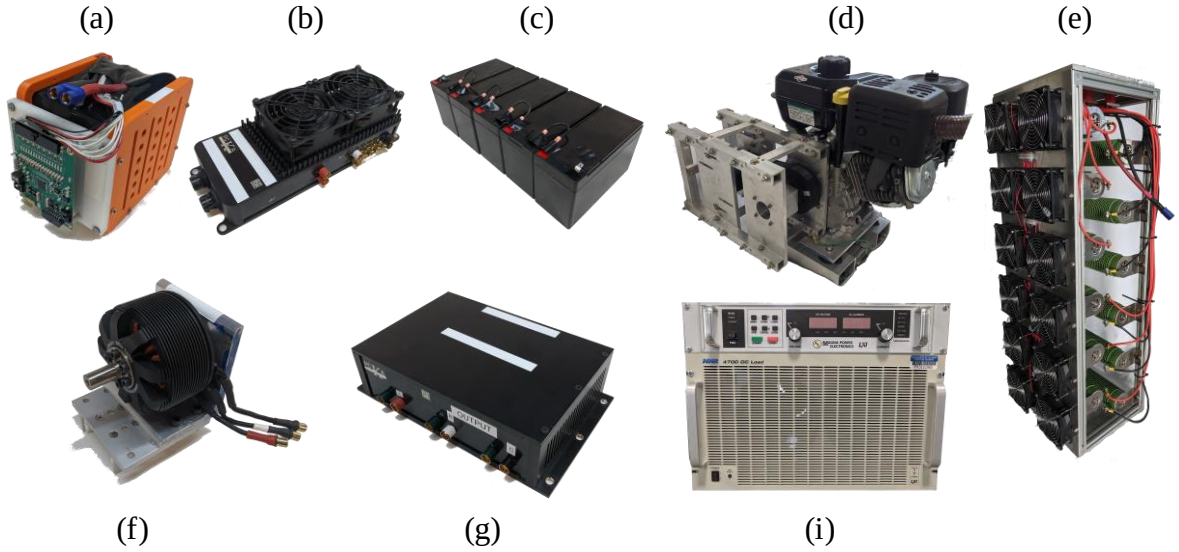


Figure D.1: The components that compose the testbed powertrain. Shown above are the (a) battery pack, (b) electronic speed controller, (c) low power battery pack, (d) engine with starter/generator, (e) load bank, (f) propulsion/dynamometer motor, (g) filter box, and (f) (top) power supply and (bottom) electronic load.

Table D.1: The UAV testbed powertrain parts list.

Component	Manufacturer	Details
Battery Pack	Custom	16S7P Samsung 18650 Cells
Low Power Battery Pack	Expert Power	5S1P EXP1270 Lead Acid Cells https://bit.ly/2RyIIMI
Hybrid ESC	PC Krause and Associates	-
Brake ESC	PC Krause and Associates	-
DCDC ESC	PC Krause and Associates	-
Filter Box (x2)	PC Krause and Associates	-
Propeller Motor	Neu Motors	Series 8038-105 https://bit.ly/2sRBWHE
Brake Motor	Neu Motors	Series 8038-105 https://bit.ly/2sRBWHE
Starter/Generator Motor	Neu Motors	Series 8038-140 https://bit.ly/2sRBWHE
Power Resistors (x12)	TE Connectivity	1Ω 2S6P https://bit.ly/2E0zFfK
Engine	Briggs and Stratton	19N1 Series https://bit.ly/2s7OEBX
Power Supply	Magna-Power	4kW Rating (XR400-10.0 Series Supply) https://bit.ly/2LAnTgk
Electronic Load	NH Research	3kW Rating (4700-3-TP Series) https://bit.ly/2LCKadh

Table D.2: The UAV testbed sensing and controls parts list.

Component	Manufacturer	Details
BMS	Texas Instruments	bq76PL455A-Q1 Evaluation Board https://bit.ly/36hhApE
CompactDAQ	National Instruments	4 slots (cDAQ-9174) https://bit.ly/353scbH
Voltage Input Module	National Instruments	Reads shunt resistors (NI-9205) https://bit.ly/2P021wO
Digital I/O Module	National Instruments	Controls relays (NI-9403) https://bit.ly/2RyJwkI
High Current Shunt (x2)	Rideon	200A rating (RSB-200-100) https://bit.ly/2PkviBe
Low Current Shunt (x4)	Rideon	100A rating (RSB-100-100) https://bit.ly/2qxXV5I
USB to CAN Adapter	Intrepid Control Systems	Facilitates CAN communication https://bit.ly/2YvlkBg
Control Computer	Dell	7^{th} Gen Intel Core i7 with 8GB RAM

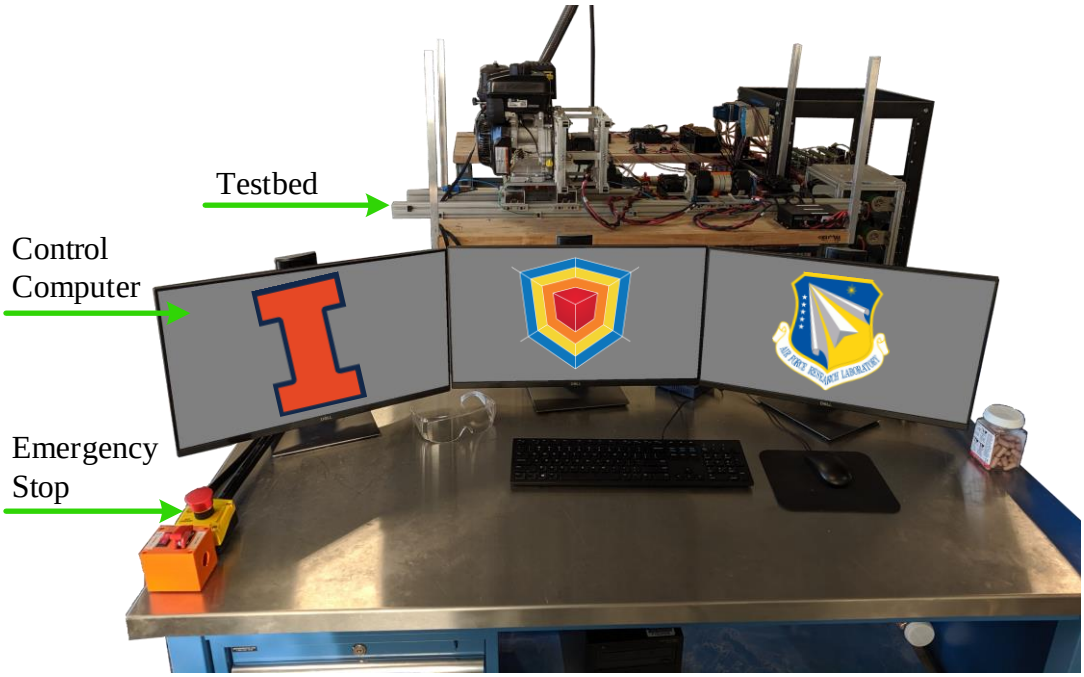


Figure D.2: Additional photo showing the testbed with the control computer.

Appendix E

MATLAB Code

The author has selected to share some code that he believes may be useful to other researchers. This code includes functions for battery parameter identification , optimal controller development, and motor frequency identification.

E.1 Battery Parameter Identification Functions

```
%% Battery Identification
% this code uses current pulse data to fully characterize a second order
% battery model. All that is required is the experiments time vector,
% battery voltage, and battery current. The code (for the most part)
%automates the process of calculating all 7 battery parameters from the
%single data set. Some thresholding/tuning of the code is required, but
%those tuning knobs are specifically listed in the code.

% developed by Christopher T. Aksland at the University of Illinois at
% Urbana–Champaign (5/1/2019)
%%
clear all
close all

load dataAll

plotTest = 0; % set to 1 plot raw data
plotSplit = 0; % set to 1 to plot split data
plotOCV = 0; % set to 1 to plot OCV curves
plotR = 0; % set to 1 to plot internal resistance curves
plotDyn = 0; % set to 1 to plot data with the initial voltage change ...
    removed

plotSample = 0; % set to 1 to check gradient sampled data
runOpt = 1; % run optimization section of the code
plotFit = 1; % set to 1 to plot RC fits
plotRC = 0; % set to 1 to plot RC curves

```

```
TestTime = data(:,1);
Current  = data(:,2);
Voltage  = data(:,3);

%% Reduce out voltage steps from internal resistance
%this block of code finds all voltage changes from the first edge of the step
cnt = 0; %how many edges are count
pSt = zeros(100,1); % stores start point
pEn = zeros(100,1); % stores end point
thresh = .07; % [V] HAND TUNED threshold for determining if the voltage change is a consequence of a pulse
for i = 2:length(Voltage)-1 % cycle through voltage data
    sn = sign(i - TestTime(end)); %denotes whether the cycle is charge or discharge
    dV = Voltage(i+1) - Voltage(i); % voltage between next and current data point
    if (abs(dV) > thresh) && (dV*sn > 0) % if the change is greater than a threshold, store the current and next data points
        cnt = cnt + 1;
        pSt(cnt) = i;
        pEn(cnt) = i+1;
        if cnt > 1
            if pSt(cnt) == pEn(cnt-1)
                pEn(cnt-1) = i+1;
                cnt = cnt -1;
            end
        end
    end
end
% removes excess zeros
rem = 2; %remove points from end
pSt = pSt(1:cnt-rem);
pEn = pEn(1:cnt-rem);
pdV = Voltage(pEn) - Voltage(pSt);

%this block of code is mostly same as above however it applies to the second step
%edge. the only change is that a secondary threshold is used. The secondary
%threshold is associated with the voltage change from the previous section of code
%ie. the voltage change from the rising edge of 1 pulse should be similar
%in magnitude to the voltage change from the falling edge of the previous
%pulse
cnt = 0;
p = 1;
rSt = zeros(100,1);
rEn = zeros(100,1);
rdV = zeros(100,1);
i = 1;
while i < length(Voltage)-1
    sn = sign(i - TestTime(end));
    dV = Voltage(i+1) - Voltage(i);
```

```
    if (abs(dV) > thresh) && (dV*sn < 0)
        cnt = cnt + 1;
        rSt(cnt) = i;
        rEn(cnt) = i+1;
        if cnt > 1
            if rSt(cnt) == rEn(cnt-1)
                rEn(cnt-1) = i+1;
                cnt = cnt -1;
            end
        end
    end
    i = i+1;
end
rem = 0; %remove points from end
rSt = rSt(1:cnt-rem);
rEn = rEn(1:cnt-rem);
rEn(end) = rEn(end)-1; % manual edit
rdV = Voltage(rEn) - Voltage(rSt);

% reshape data to fit into a single vector
St = reshape([pSt,rSt]',2*length(pSt),1);
En = reshape([pEn,rEn]',2*length(pEn),1);

% plot results
if plotTest == 1
    plot(TestTime,Voltage,'linewidth',2)
%     plot(Voltage,'linewidth',2)
    hold on
    plot(TestTime(St),Voltage(St), 'o','MarkerEdgeColor','r','linewidth',2)
    plot(TestTime(En),Voltage(En), 'o','MarkerEdgeColor','g','linewidth',2)
    ylabel('Voltage [V]')
    xlabel('Time [s]')
    axis([0 TestTime(end) 40 70])

%     figure
%     plot(TestTime,Current)
%     for i = 1:18
%         line([St(2*i-1)-1 St(2*i-1)-1],[0 -7],'Color',[1 0 0])
%         line([St(2*i+1)-1 St(2*i+1)-1],[0 -8],'Color',[0 1 0])
%         line([St(i) St(i)],[0 -7],'Color',[1 0 0])
%     end
end

%% Split up curves
% this code splits up the raw data by each distinct 'period' this is
% relatively simple because the previous section found the starting point
% for each 'period'. The data is divided at those points
pFin = 217197; %HAND TUNED. index for the last point in the final 'good' ...
    relaxtion curve (unused)
sect{1} = [TestTime(1:St(1)) Voltage(1:St(1))]; %gets first section
for i = 1:2*cnt-1 %split intermediate sections
    sect{i+1} = [TestTime(St(i):St(i+1)) Voltage(St(i):St(i+1))];
end
```

```
126 sect{2*cnt+1} = [TestTime(St(2*cnt):pFin) Voltage(St(2*cnt):pFin)]; %get ...
       last relaxation section
127 sect{2*cnt+2} = [TestTime(pFin:end) Voltage(pFin:end)]; %rest of data ...
       (unused)
128
129 % plot results
130 if plotSplit == 1
131     figure
132     for i = 1:2*cnt+2
133         plot(sect{i}(:,1),sect{i}(:,2),'linewidth',2)
134         hold on
135     end
136
137     plot(TestTime(St),Voltage(St),'o','MarkerEdgeColor','r')
138     hold on
139     plot(TestTime(En),Voltage(En),'o','MarkerEdgeColor','g')
140     ylabel('Voltage [V]')
141     xlabel('Time [s]')
142     axis([0 TestTime(end) 40 70])
143 end
144
145 %% Get OCV Curve
146 capD = sum(abs((Current(1:end-1)+Current(2:end)) ...
       .*(TestTime(1:end-1)-TestTime(2:end)))) /(2*3600); % HAND TUNED ...
       battery capacity from discharge
147
148 SOC_D = linspace(0,1,21);
149
150 Q = zeros(100,1);
151 Q(1) = sum(abs((Current(1:St(1)-2)+Current(2:St(1)-1)) ...
       .*(TestTime(1:St(1)-2)-TestTime(2:St(1)-1))))/(2*3600);
152 for i = 1:length(St)/2-1
153     Q(i+1) = sum(abs((Current(St(2*i-1)-1:St(2*i+1)-2) ...
           +Current(St(2*i-1):St(2*i+1)-1)) ...
           .*(TestTime(St(2*i-1)-1:St(2*i+1)-2) ...
           -TestTime(St(2*i-1):St(2*i+1)-1)))) /(2*3600); % HAND TUNED ...
           battery capacity from discharge
154 end
155 Q = Q(1:cnt);
156 i = i+1;
157 Q(end+1) = sum(abs((Current(St(2*i-1)-1:pFin-1)+Current(St(2*i-1):pFin)) ...
       .*(TestTime(St(2*i-1)-1:pFin-1)-TestTime(St(2*i-1):pFin)))) ...
       /(2*3600); % HAND TUNED battery capacity from discharge
158 Q(end+1) = sum(abs((Current(pFin:end-1)+Current(pFin+1:end)) ...
       .*(TestTime(pFin:end-1)-TestTime(pFin+1:end)))) /(2*3600); % HAND ...
       TUNED battery capacity from discharge
159 % sum(Q)
160 del = cumsum(Q);
161 del(1) = 0;
162 SOC_D = 1 - del/capD;
163 SOC_D = fliplr(SOC_D');
164 SOC_D(1) = 0;
165
```

```
166 pts = 60; %HAND TUNED how many data points are averaged
167 for i = 1:cnt + 1
168     OCV(i) = mean(sect{2*i-1}(end-pts:end)); %average steady state …
            section of each curve
169 end
170 OCV(i+1) = mean(sect{end}(end-pts:end));
171 OCV = fliplr(OCV);
172
173 % plot results
174 if plotOCV == 1
175     figure
176     plot(SOC_D,OCV,'linewidth',2)
177     hold on
178     ylabel('OCV [V]')
179     xlabel('SOC')
180 end
181
182 %% Find series Resistances
183 I = 7; %"HAND TUNED" may differ depending on test
184 Rs_relax = fliplr((abs(rdV)/I)'); % determine R_s during relatxtion period
185 % Rs_step  = abs(pdV)/I; % determine R_s during pulsed period
186
187 % plot results
188 if plotR == 1
189     figure
190     plot(SOC_D(2:end-1),Rs_relax,'Color','b')
191 %     hold on
192 %     plot(SOC_D(3:end),Rs_step,'Color','r')
193     xlabel('SOC []')
194     ylabel('Rs [Ohm]')
195 %     legend('Relax Discharge','Pulse Discharge')
196 end
197
198
199 %% Create Section for only dynamic region
200 % this section removes the intial voltage change from the raw data. this
201 % will make the optimization better
202 sect_dyn = cell(1,length(sect)); %stores dynamic data
203 sect_dyn{1} = sect{1};
204 del = En-St; %denotes how many data points need to be removed from the …
       raw data
205 for i = 2:length(sect)-1;
206     sect_dyn{i} = [sect{i}(1+del(i-1):end,1) sect{i}(1+del(i-1):end,2)]; …
            %modify data
207     %plot results
208     if 0
209         figure
210         plot(sect{i}(:,1),sect{i}(:,2))
211         hold on
212         plot(sect_dyn{i}(:,1),sect_dyn{i}(:,2))
213     end
214 end
215 sect_relax = cell(1,length(sect)/2); %stores dynamic data
```

```
sect_dyn_relax = cell(1,length(sect)/2); %stores dynamic data

for i = 1:length(sect)/2;
    sect_relax{i} = sect{2*i-1}; %stores dynamic data
    sect_dyn_relax{i} = sect_dyn{2*i-1}; %stores dynamic data

    if plotDyn == 1
        figure
        plot(sect_relax{i}(:,1),sect_relax{i}(:,2))
        hold on
        plot(sect_dyn_relax{i}(:,1),sect_dyn_relax{i}(:,2))
    end
end

%% Downsampling Sampling
% Optimizing with large data sets is taxing. a gradient sampling method is
% used. regular sample is ineffective because the data has varying
% timescales. filtering is not ideal because it can add too much phase
% shift (also it still yields too many data points for optimization)
% Ammendment: greybox models only solve with uniformly sampled data, so ...
    just
% use 'downsample()'
VThresh = .025; %HAND TUNED threshold for voltage change
tThresh = 400; %HAND TUNED threshold for change in index (ie if the ...
    voltage doesn't change much, sample every x points)
sect_gs = cell(length(sect_relax),1); %stores sampled data
sect_gs{1} = sect_dyn_relax{1};

for i  = 2:length(sect_relax);
%     sect_gs{i} = gradientSample(sect_dyn_relax{i},VThresh,tThresh); % ...
    gradient sample DONT USE
    sect_gs{i} = [downsample(sect_dyn_relax{i}(:,1),100), ...
        downsample(sect_dyn_relax{i}(:,2),100)]; % sample data
    % plot results
    if plotSample == 1
            figure
            plot(sect_dyn_relax{i}(:,1),sect_dyn_relax{i}(:,2))
            hold on
            scatter(sect_gs{i}(:,1),sect_gs{i}(:,2),'linewidth',2)
    end
end

%% Get Pulse Width
% get width of each current pulse. self explanatory hopefully
% plot(TestTime,Current)
% hold on
% scatter(TestTime(St),Current(St))

t_pulse = zeros(length(sect_gs),1);
t_r = zeros(length(sect_gs),1);
I_avg = zeros(length(sect_gs),1);

t_pulse(1) = 0;
```

```
t_r(1) = TestTime(St(1));
I_avg(1) = 0;

for i = 1:length(sect_gs) - 2
    t_pulse(i+1) = TestTime(St(2*i)) - TestTime(St(2*i-1));
    I_avg(i+1) = mean(Current(St(2*i-1):St(2*i)));
    t_r(i+1) = TestTime(St(2*i+1)) - TestTime(St(2*i));
end
i = i+1;
t_pulse(end) = TestTime(St(2*i)) - TestTime(St(2*i-1));
I_avg(end) = mean(Current(St(2*i-1):St(2*i)));
t_r(end) = TestTime(pFin) - TestTime(St(2*i));


%% Get RC Pairs
param = cell(1,length(sect_gs)); % stores parameters
V_save = cell(length(sect_gs),1);

LB = [.0000001 .0000001 1 100]; %HAND TUNED lower bounds for RC pairs ...
    [R1 R2 tau1 tau2] (avoid 0)
UB = [1 1 500 10000]; %HAND TUNED upper bounds for RC pairs [R1 R2 tau1 ...
    tau2]

load param_D
param{1} = [.0378;.1897; 22.944; 887.7172];

if runOpt == 1;
    for i = 2:length(sect_gs) %loop for each relaxation period
        tdata = sect_gs{i}(1:end,1) - sect_gs{i}(1,1); %time data for ...
            the period brought to the origin
        ydata = sect_gs{i}(1:end,2) - sect_gs{i}(1,2); %voltage data for ...
            the period brough to the origin
        sn = -1; % denotes charge or discharge (1)

        % stores relavant data to the optimization problem in OPT
        Opt.tdata   = tdata;
        Opt.ydata   = ydata;
        Opt.cnt     = length(Opt.tdata); %unused
        Opt.t_pulse = t_pulse(i);
        Opt.I       = -sn*abs(I_avg(i));
        Opt.LB      = LB;
        Opt.UB      = UB;
        Opt.lv      = 1;
        Opt.lx      = .75;
        Opt.x0      = param{i-1};

        [R1 R2 tau1 tau2] = RC_Fit_opt(Opt,1);
        param{i} = [R1;R2;tau1;tau2];

        % get fitted soltion V and plot results to compare
        V = R1*Opt.I*(1-exp(-Opt.t_pulse/tau1)) ...
            *(1-exp(-Opt.tdata/tau1)) + ...
            R2*Opt.I*(1-exp(-Opt.t_pulse/tau2)) *(1-exp(-Opt.tdata/tau2));
```

```
312         V_save{i} = [Opt.tdata,V];
313         if plotFit == 1
314             figure
315             plot(sect_dyn_relax{i}(1:end,1) - sect_dyn_relax{i}(1,1), ...
                    sect_dyn_relax{i}(1:end,2) - sect_dyn_relax{i}(1,2))
316             hold on
317             scatter(Opt.tdata,Opt.ydata)
318             hold on
319             plot(Opt.tdata,V,'linewidth',2)
320             xlabel('Time [s]')
321             ylabel('Voltage [V]')
322             if sign(sn) < 0
323                 legend('Raw Data','Sampled ...
                        Data','Fit','location','SouthEast')
324             else
325                 legend('Raw Data','Sampled ...
                        Data','Fit','location','NorthEast')
326             end
327         end
328         i
329     end
330 end
```

```
1  function [R1, R2, tau1, tau2] = RC_Fit_opt( Opt,plotResult)
2  % This function sets up and solves a grey box identifcation problem for
3  % battery identification
4  % Opt: Structure containing relevant data for the optimization program
5
6  % developed by Christopher T. Aksland at the University of Illinois at
7  % Urbana-Champaign (5/1/2019)
8
9  %% %%%%%%%% Optimization Setup %%%%%%%%%%%
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 % sample rate/ time step
12 ts = (Opt.tdata(2)-Opt.tdata(1));
13
14 % creates data file. shouldn't need to change
15 z = iddata(Opt.ydata, [], ts, 'Name', 'Experimental'); %create data file ...
       for optimizaiton
16 z.Tstart = 0; %start time
17
18 % file name for code describing the model structure. this is a .m file
19 FileName      = 'RC_Fit';
20
21 % Model orders [ny nu nx]. [number of matched states, "inputs", and states]
22 Order         = [1 0 0];
23
24 % Initial parameters.
25 Parameters    =  [Opt.x0;Opt.I;Opt.t_pulse];
26
27 % Initial states
28 InitialStates = [];
```

```
29
30 % Denotes our model is a continuous time system. don't change
31 Ts              = 0;
32
33 % creates non-linear greybox object
34 nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts, 'Name', ...
       'Fit');
35 nlgr.SimulationOptions.Solver = 'ode23tb';
36 nlgr.Parameters(1).Name = 'R1';
37 nlgr.Parameters(2).Name = 'R2';
38 nlgr.Parameters(3).Name = 'tau1';
39 nlgr.Parameters(4).Name = 'tau2';
40 nlgr.Parameters(5).Name = 'I';
41 nlgr.Parameters(6).Name = 't_pulse';
42 % nlgr.SimulationOptions.AbsTol = 1e-2;
43 % nlgr.SimulationOptions.RelTol = 1e-2;
44
45 %% %%% Setup Constraints and Test Initial Guesses %%%
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47
48 % bounds for regularized optimization variables
49 for i = 1:4 %
50     nlgr.Parameters(i).Minimum = Opt.LB(i);
51     nlgr.Parameters(i).Maximum = Opt.UB(i);
52 end
53 nlgr.Parameters(5).Fixed = 1;
54 nlgr.Parameters(6).Fixed = 1;
55
56 % plot response with given initial guesses
57 if 0
58     figure
59 %     optSim = compareOptions('InitialCondition',IC_x);
60 %     compare(z, nlgr,optSim);
61     compare(z, nlgr);
62 end
63
64 %% %%%%%%%%%%%%%%% Optimization %%%%%%%%%%%%%%%
65 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66
67 opt = nlgreyestOptions('Display', 'on'); % opens window to provide ...
       optimization updates
68 opt.SearchMethod = 'lm'; % type 'help nlgreyestOptions' to look up other ...
       search methods
69 opt.SearchOption.MaxIter = 20; %max iteration count
70
71 % call optimizer
72 nlgr = nlgreyest(z, nlgr, opt);
73
74 % plot true and fitted response
75 if plotResult
76     figure
77     compare(z, nlgr);
78 end
```

```
79
80 R1 = nlgr.Parameters(1).Value;
81 R2 = nlgr.Parameters(2).Value;
82 tau1 = nlgr.Parameters(3).Value;
83 tau2 = nlgr.Parameters(4).Value;
84
85 end
```

```
1  function [ dx, y ] = RC_Fit( t, x, u, R1, R2, tau1, tau2, I, t_pulse, ...
       varargin )
2  %this function calulates the battery volatge dynamics for application in
3  %the grey box identification tool
4
5  % developed by Christopher T. Aksland at the University of Illinois at
6  % Urbana-Champaign (5/1/2018)
7
8  % voltage trajectories
9  V1 = R1*I*(1-exp(-t_pulse/tau1))*(1-exp(-t/tau1));
10 V2 = R2*I*(1-exp(-t_pulse/tau2))*(1-exp(-t/tau2));
11
12 % Output
13 dx = [];
14 y = V1 + V2;
15
16 end
```

E.2 Model Predictive Controller Formulation Function

```
1  function [ Output ] = Controller_Gen( Input )
2  % this function formulates the controller as an optimization program using
3  % YALMIP. The variable Input is a structure containing controller
4  % parameters. The optimization program is stored as the function
5  % "Controller". Call Controller to solve the optimization.
6
7  % developed by Christopher T. Aksland at the University of Illinois at
8  % Urbana-Champaign (9/1/2019)
9
10 Output = Input;
11
12 %% Decision Variables
13 x_    = sdpvar(repmat(Output.Nv, 1, Output.horizon+1), ...
       ones(1,Output.horizon+1));
14 u_    = sdpvar(repmat(Output.Nu, 1, Output.horizon), ...
       ones(1,Output.horizon));
```

```
15 s_    = sdpvar(repmat(length(Output.ind_x_bounds), 1, Output.horizon), ...
       ones(1,Output.horizon));
16 stv_  = sdpvar(repmat(length(Output.ind_x_bounds_tv), 1, ...
       Output.horizon), ones(1,Output.horizon));
17
18 u0_   = sdpvar(repmat(Output.Nu, 1, 1), ones(1,1));
19 sv_   = sdpvar(repmat(Output.Nsv, 1, Output.horizon), ...
       ones(1,Output.horizon));
20 xref_ = sdpvar(repmat(length(Output.ind_x_ref), 1, Output.horizon), ...
       ones(1,Output.horizon));
21 xbnd_ = sdpvar(repmat(length(Output.ind_x_bounds_tv), 1, ...
       Output.horizon), ones(1,Output.horizon));
22 temp_ = sdpvar(repmat(Output.Nsv-1, 1, Output.horizon), ...
       ones(1,Output.horizon));
23
24 Az_   = sdpvar(Output.Nv,Output.v_tot,'full');
25 B1z_  = sdpvar(Output.Nv,Output.Nu,'full');
26 B2z_  = sdpvar(repmat(Output.Nv, 1, 1), ones(1,1));
27
28 Ie_   = sdpvar(repmat(1, 1, Output.horizon+1), ones(1,Output.horizon+1));
29 ue_   = sdpvar(repmat(1, 1, Output.horizon), ones(1,Output.horizon));
30 ue0_  = sdpvar(repmat(1, 1, 1), ones(1,1));
31 ubnd_ = sdpvar(repmat(1, 1, 1), ones(1,1));
32
33 m_    = binvar(repmat(2,1,Output.horizon),ones(1,Output.horizon));   % ...
       Binary variable for mode selection
34 m0_   = binvar(repmat(2,1,1),ones(1,1));   % previous switch state
35 SFCJ_ = sdpvar(repmat(1, 1, Output.horizon), ones(1,Output.horizon));
36 V_bus  = sdpvar(repmat(1, 1, Output.horizon), ones(1,Output.horizon));
37
38 %% Objective Function
39 objs = 0;
40 for k = 1:Output.horizon
41
42     %state tracking
43     objs = objs + norm((Output.l_v*(x_{k+1}(Output.ind_x_ref) - ...
           xref_{k})),2)^2;
44
45     %slack penalty
46     objs = objs + norm(Output.l_s*s_{k},2)^2;
47     %slack penalty for time varying bounds
48     objs = objs + norm(Output.l_s_tv*stv_{k},2)^2;
49
50     %rate of change of state penalty
51     objs = objs + norm(Output.l_dx*(x_{k+1}(Output.ind_x_dx) - ...
           x_{k}(Output.ind_x_dx)),2)^2;
52
53     %fuel cost penalty (note the normalizing term Output.Eng.maxE(2))
54     SFCJ_{k} = [-Ie_{k+1} V_bus{k} m_{k}(2)]*Output.Eng.S*[-Ie_{k+1} ...
           V_bus{k} m_{k}(2)]';
55     objs = objs + Output.l_f*SFCJ_{k}/Output.Eng.SFC_max;
56 end
57
```

```
%switching penalty
objs = objs + Output.l_m*(norm(m_{1}(1)-m0_(1),2))^2;
%rate of change of input penalty
objs = objs + norm(Output.l_du*(ue_{1} - ue0_),2)^2;
for k = 2:Output.horizon
    objs = objs + norm(Output.l_du*(ue_{k} - ue_{k-1}),2)^2;
    objs = objs + Output.l_m*(norm(m_{k}(1)-m_{k-1}(1),2))^2;
end

%% Constraints
cons = [];

for k = 1:Output.horizon

    % dynamic states
    cons = [cons, x_{k+1}(Output.dyn) == ...
        Az_(1:Output.Ndyn,:)*[x_{k}(Output.dyn);x_{k+1}(Output.alg); ...
        sv_{k}] + B1z_(1:Output.Ndyn,:)*u_{k} + B2z_(1:Output.Ndyn)];
    % algebraic states
    cons = [cons, 0 == Az_((Output.Ndyn+1):end,:)*[x_{k}(Output.dyn); ...
        x_{k+1}(Output.alg);sv_{k}] + B1z_((Output.Ndyn+1):end,:)*u_{k} + ...
        B2z_((Output.Ndyn+1):end)];

%     % rate of change of state constraints
%     for i = 1:length(Output.ind_x_dx_cons) %bounded states
%         cons = [cons, -Output.x_dx_cons(i) ≤ ...
    ((x_{k+1}(Output.ind_x_dx_cons(i)) - ...
    x_{k}(Output.ind_x_dx_cons(i)))/Output.dt) ≤ Output.x_dx_cons(i)];
%     end

%     other constrains
    cons = [cons, s_{k} ≥ 0]; %slack
    cons = [cons, stv_{k} ≥ 0]; %slack
    for i = 1:length(Output.ind_x_bounds) %bounded states
        cons = [cons, Output.x_min(i)-s_{k}(i) ≤ ...
            x_{k+1}(Output.ind_x_bounds(i)) ≤ Output.x_max(i)+s_{k}(i)];  ...
            %should be ≤ or ≥ %understand why slack is included
    end
    for i = 1:length(Output.ind_x_bounds_tv) %time varying bounded states
        cons = [cons, xbnd_{k}(i)-stv_{k}(i) ≤ ...
            x_{k+1}(Output.ind_x_bounds(1))];  %should be ≤ or ≥ ...
            %understand why slack is included
    end

    % all input constraints
    if k > 1
        cons = [cons; sum(m_{k}) == 1];

        %bounded inputs
        for i = 1:length(Output.u_min)
            cons = [cons, Output.u_min(i) ≤ ...
                u_{k}((Output.ind_u_bounds(i))) ≤ Output.u_max(i)];  ...
                %should be ≤ or ≥
```

```
99          end
100
101         %constant inputs
102         for i = 1:length(Output.u_cons)
103             cons = [cons, 1 == u_{k}((Output.ind_u_cons(i)))];  %should …
                    be ≤ or ≥
104         end
105
106         %rate of change of inputs
107         for i = 1:length(Output.ind_u_Δ)
108             cons = [cons,  u0_(Output.ind_u_Δ(i))-Output.u_Δ(i) ≤ …
                    u_{k}(Output.ind_u_Δ(i)) ≤ …
                    u0_(Output.ind_u_Δ(i))+Output.u_Δ(i)];
109         end
110
111         % Engine Model Constraints
112         % in mode 1, engine input is zero and current production is 0
113 %         cons = [cons, implies(m_{k}(1), [ue_{k} == 0, Ie_{k+1} == 0 ])];
114         cons = [cons, implies(m_{k}(1), [ue_{k} == 0])];
115         % in mode 2, engine input and current production is free
116 %         cons = [cons, implies(m_{k}(2), [0 ≤ ue_{k} ≤  1, 0 == …
    Output.Eng.A1_z*Ie_{k+1} + Output.Eng.B1_z*ue_{k}])];
117         cons = [cons, implies(m_{k}(2), [0 ≤ ue_{k} ≤  1])];
118         % constrain engine to sink state vetex
119
120     end
121
122     % engine state dynamics and sink state equivalence
123     cons = [cons, 0 == Output.Eng.A1_z*Ie_{k+1} + Output.Eng.B1_z*ue_{k}];
124     cons = [cons, Ie_{k+1} == sv_{k}(2)];
125
126
127     % bus voltage for SFC implies statement
128     cons = [cons, implies(m_{k}(1), [V_bus{k} == 0])];
129     cons = [cons, implies(m_{k}(2), [V_bus{k} == …
            x_{k+1}(Output.Eng.x_V)])];
130
131 % % %     % big-M matrix constraints
132 % % %     cons = [cons, -101 ≤ Ie_{k+1} ≤ 1];
133 % % %     cons = [cons, -101 ≤ x_{k+1}(Output.Eng.x_V) ≤ 100];
134 % % %     cons = [cons, -1 ≤ ubnd_ ≤ 2];
135 % % %     cons = [cons, -1 ≤ ue_{k} ≤  2];
136 % % %     cons = [cons, -10000 ≤ V_bus{k} ≤  10000];
137
138 end
139
140 % bound the engine input
141 for k = 2:Output.horizon
142         cons = [cons, 0 ≤ ue_{k} ≤ ubnd_];
143 end
144
145 % constrain the switch to stay constant over the horizon
146 for k = 2:Output.horizon-1
```

```
147         cons = [cons, m_{k}(1) == m_{k+1}(1)];
148 end
149
150
151     %account for input delay on continuous plant
152     cons = [cons, u_{1}  == u0_];
153     cons = [cons, ue_{1} == ue0_];
154     cons = [cons, m_{1} == m0_];
155
156 % opts = sdpsettings('solver','gurobi'); %,'quadprog.TolFun',1e-16
157 opts = sdpsettings('solver','gurobi','gurobi.TimeLimit',Output.dt*.9); ...
       %,'quadprog.TolFun',1e-16
158
159 % replace/remove the sink vertex information relating to the engine ...
       current since
160 % the engine current is now chosen by the controller
161 % temp = cell(1,Output.horizon);
162 for k = 1:Output.horizon
163     cons = [cons, temp_{k} == sv_{k}([1,3:end])];
164 end
165 Output.Controller = optimizer(cons,objs,opts, ...
       {x_{1}(:),u0_,temp_{:},xref_{:},Az_,B1z_,B2z_,Ie_{1}(:),m0_, ...
       ubnd_,ue0_,xbnd_{:}}, [x_,u_,s_,Ie_,ue_,m_,sv_,stv_]);
166
167 end
```

E.3 Motor Frequency Identification Function

```
1  function [ DataOut, f_n ] = filterData( DataIn, res, n, fc_adj )
2  % this code was used to get frequency for the motor ID process by applying
3  % a fast fourier transform to the experimental data. The fft provides
4  % frequency data. This frequency data yields motor shaft speed and helps
5  % the design process for a zero-phase filter.
6
7  % DataIn: Data that should be filtered
8  % res: resolution for the FFT
9  % n: order of butterworth filter
10 % fc_adj: a cutoff frequency is found from the FFT. use this to adjust the
11 % the cutoff frequency of the filter
12 % this code finds the dominant frequency of a waveform using the FFT. A low
13 % pass butterworth filter is used to remove signal noise.
14
15 % developed by Christopher T. Aksland at the University of Illinois at
16 % Urbana-Champaign (5/1/2018)
17
18 L = length(DataIn(:,1)); % length of datset
19 T = DataIn(end,1) - DataIn(1,1); %total sample time
```

```
F = 1/T;

Y=fft(DataIn(:,2),L*res); % fast fourier transform, res impacts …
    resolution of the output (large res = more accurate)
P2 = abs(Y/L); %two sided frequency response
P1 = P2(1:(L/2+1));
P1(2:end-1) = 2*P1(2:end-1); % one sided frequency response
f = F*(0:(L/2))/res; %frquecy assocaited with P1
mod = 9;
[~,i] = max(P1(1+mod:end)); % get index of greatest amplitude frequency
i=i+mod;
f_n = f(i); %get frequency associated with
fc = f_n*fc_adj; %cutoff frequnnecy for filter
fs = L/(1/F); %sampling frequency
[b,a] = butter(n,fc/(fs/2)); %make filter
% DataF = filter(b,a,DataIn(:,2)); %filer data
DataF = filtfilt(b,a,DataIn(:,2)); %zero phase filter data

DataOut = [DataIn(:,1) DataF];
%
% figure
% plot(f,P1)
end
```