

© 2022 Philip Renkert

COMPONENT-BASED DESIGN OPTIMIZATION OF MULTIROTOR AIRCRAFT

BY

PHILIP RENKERT

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 Urbana-Champaign, 2022

Urbana, Illinois

Adviser:

Dr. Andrew Alleyne

# Abstract

Rising complexity of engineered systems, coupled with increasing specialization of companies and their design engineers, requires increasing degrees of coordination to ensure local design decisions are made in service of the system-level objective. Simultaneously, component-based design has become common practice in complex system design, and the problem of selecting components to optimize a system has gained traction in the literature. Existing approaches typically solve the discrete problem directly or parameterize the components and solve the problem in the continuous domain. This thesis develops a hybrid methodology for component-based design optimization that leverages continuous-domain information to efficiently search the discrete design space. For demonstration, the process is applied in two case studies: the maximization of a quadrotor's endurance per system price and the minimization of the time required for a planar quadrotor to complete a dynamic mission.

*To Mr. John Tinnin, who revealed to me the pleasure of figuring things out.*

# Acknowledgments

Part of my job as an advisor is helping find the path and then opening doors along the path for the student.

(Dr. Andrew Alleyne)

When I read this line in the first email I received from Dr. Alleyne, I was oblivious to the impact it was to have on my graduate experience. Dr. Alleyne's guidance is invaluable: it combines razor-sharp technical instruction with career and life advice backed by a sincere desire to see his students thrive. And he does not merely point students down paths like a trail sign, he clears the path and trailblazes new ones when necessary. When you give Dr. Alleyne an inch of directed enthusiasm, he opens miles of opportunity. Without Dr. Alleyne and the supportive, brilliant research group he has fostered, this thesis would not have been possible.

The Alleyne Research Group includes senior members Christopher Aksland and Cary Laird, and alumni Spencer Igram and Mindy Wagenmaker. They exemplify servant leadership and research excellence, and they gave generous amounts of advice and their valuable time to help me learn the ropes. My own cohort includes Reid Smith, Kayla Russel, Chris Urbansky, Frank Andujar Lugo, Kurtis Kuipers, and Dylan Charter. These friends and coworkers added a healthy amount of joy, energy, and laughter to the research grind. Figuring things out together is much more fun than figuring things out alone.

The National Science Foundation Engineering Research Center for Power Optimization of Electro-Thermal Systems (POETS) supported my work financially and in countless other ways. The POETS community generates a contagious excitement as they lay the technical foundation for electric mobility; its expertise and inspiration was invaluable to my research. No organization such as POETS could exist without an excellent administrative team. I'd specifically like to thank Jodi Gritten and Owen Doyle for their personal support and contribution to the center.

I'm grateful for my friends, family, and church family who carried me through the ups and downs of graduate school. And last but certainly not least,

I lift up my eyes to the mountains –  
where does my help come from?  
My help comes from the Lord,  
the Maker of of heaven and earth.

(Psalms 121:1-2)

# Table of contents

List of Abbreviations .....	vii
List of Symbols .....	ix
<b>1 Introduction .....</b>	<b>1</b>
<b>2 Modeling .....</b>	<b>14</b>
<b>3 CBDO Problem Formulation .....</b>	<b>26</b>
<b>4 Hybrid Optimization .....</b>	<b>29</b>
<b>5 Case Study: Multirotor Design Optimization .....</b>	<b>40</b>
<b>6 Conclusion .....</b>	<b>113</b>
References .....	116
Appendix A Park Transform .....	1
Appendix B Powertrain System Model Details .....	4
Appendix C AGILe Exported Model: Planar Quadrotor Powertrain .....	7
Appendix D Component Databases .....	10

# List of Abbreviations

MDO	Multidisciplinary Design Optimization
CBD	Component-Based Design
COTS	Commercial off-the-shelf
CBDO	Component-Based Design Optimization
UAV	Unmanned Aerial Vehicle
GA	Genetic Algorithm
ESC	Electronic Speed Controller
LQR	Linear-Quadratic Regulator
SQP	Sequential Quadratic Programming
FEA	Finite Element Analysis
ODE	Ordinary Differential Equation
DAE	Differential-Algebraic Equation
AE	Algebraic Equation
CCD	Control Co-Design
DT	Direct Transcription
NLP	Nonlinear program
DSS	Distance-Sorted Search
eVTOL	Electric Vertical Take-Off and Landing
LiPo	Lithium Polymer
OCV	Open-Circuit Voltage
SOC	State of Charge



RC	Resistor-Capacitor
DC	Direct Current
FOC	Field-Oriented Control
PMSM	Permanent-Magnet Synchronous Machine
RPM	Revolutions per Minute
CAD	Computer-Aided Design
KKT	Karush-Kuhn-Tucker
SLSQP	Sequential Least Squares Programming

# List of Symbols

## Common Symbols

$G$	Oriented graph
$V$	Set of vertices
$N_v$	Number of vertices in a graph
$v$	Graph vertex, voltage, or linear velocity
$E$	Set of edges
$N_e$	Number of edges in a graph
$e$	Graph edge or unit vector
$P$	Graph edge flow, set of design parameters, or number of poles
$p$	Centroid of design parameters or price
$C$	Capacitance, component database, or C-rating
$c$	Component
$M$	Incidence matrix defining graph structure, number of design parameters, or moment
$D$	Incidence matrix defining graph structure or diameter
$x$	State variable
$\xi$	State variable (optimization context)
$\Xi$	Discretized state trajectory
$a$	Algebraic state variable
$u$	Input variable
$d$	Disturbance variable or distance

$y$	Output variable
$\Phi$	Parameter variable
$\phi$	Single parameter or roll angle
$\Phi^d$	Design parameter variable
$\Phi^p$	Performance parameter variable
$\Phi_c$	Parameter value corresponding to component $c$
$N_\Phi$	Number of parameters
$f$	ODE function
$h$	Residual function or equality constraint function
$g$	Output function, inequality constraint function, or gravity
$g_P$	Physical or static constraint function
$g_D$	Dynamic constraint function
$i$	Selection index or current
$J$	Cost function or inertial tensor
$T$	Set of component types, transformation function, or thrust
$t$	Component type
$N_T$	Number of component types
$N_C$	Number of components
$x_c$	Control design variables
$X_c$	Discretized control design variables
$K$	Matrix of feedback control gains or scaling matrix
$B$	Convex hull of design parameters or friction factor
$B_s$	Smoothed convex hull of design parameters
$g_B$	Boundary constraint function
$\zeta$	Defect constraint function
$S$	Parameter surrogate function or set of all possible configurations
$N_S$	Number of possible configurations

$s$	Component configuration
$d_c$	Component distance metric
$d_s$	Configuration distance metric
$w$	Parameter weighting
$N_s$	Series cells
$N_p$	Parallel cells
$q$	Battery state of charge
$R$	Resistance
$\mathbf{R}$	Rotation matrix
$Q$	Battery cell capacity
$L$	Inductance or angular momentum
$\lambda$	Flux Linkage
$\omega$	Angular velocity
$\tau$	Torque
$G_a$	Gyroscopic torque
$K_\tau$	Motor torque constant
$kV$	Motor speed constant
$k_T$	Propeller thrust coefficient
$k_P$	Propeller power coefficient
$k_Q$	Propeller torque coefficient
$\rho$	Density
$\psi$	Yaw angle
$\theta$	Pitch angle
$\Theta$	Attitude
$m$	Mass
$\mathbf{r}$	Displacement vector
$I$	Moment of inertia

$\Sigma$	Sensitivity
$\mathbb{Z}_+$	Set of positive integers
$\mathbb{R}$	Set of real numbers

## Notation and Operators

$\mathcal{O}(\cdot)$	Order of Accuracy
<b>x</b>	Bold type indicates vector-valued variables or functions
$\tilde{x}$	Tilde indicates transformed variable or a low-fidelity approximator
$x^*$	Optimal variable value or sorted set
$\phi_x^t$	Parameter $x$ corresponding to component type $t$
$\mathbf{R}_a^b$	Rotation of frame $a$ relative to fixed frame $b$
$S_\phi^t$	Surrogate for parameter $\phi$ corresponding to component type $t$
$\Sigma_t^J$	Sensitivity of $J$ with respect to component type $t$
$[\cdot]_\times$	Skew symmetric operator
$\ \cdot\ $	Euclidian norm
$ \cdot $	Absolute value
$\otimes$	Outer product

# Chapter 1

## Introduction

In 1919, Samuel Northrup Castle placed an order for a Duesenberg Model A [1]. Castle was a large man with deep pockets to match; he had a love for automobiles and the means to acquire the very best. His quest for such a motorcar led him to brothers Fred and Augie Duesenberg, two self-taught engineers with a booming reputation for superior engines and race cars. The Model A was to be the first production vehicle from the Duesenberg Automobile and Motors Company. Fred would accept nothing short of perfection; he designed the vehicle to, in his words, “outclass, outrun, and outlast any car on the road” [2]. His meticulous tuning of the A’s 88 horsepower single-overhead-cam eight-cylinder engine delayed production for nearly two years [3]. As with other luxury automobiles of the era, the Model A was only available as a rolling chassis; the buyer would hire a coachbuilder to fashion the vehicle’s cabin. Mr. Castle requisitioned the Bender Body Company of Cleveland to produce a body that could accommodate his 7-foot 300-pound frame. By the time Castle took delivery of his Model A in Hawaii in 1921, the car had commanded a princely sum of around \$7,000 for the chassis and up to an additional \$6,000 for the coachwork [3].

An equally impressive vehicle was received under vastly different circumstances in 1918. It arrived not on a ship but on a railroad, packed into five or six wooden crates. A horse and wagon carried the crates to buyer Carl Maute’s home in Wolford, North Dakota, where he assembled the car himself. The vehicle’s price was \$450, or around 3% that of Castle’s ‘Duesy’ [3]. Maute’s new car was a Ford Model T, the realization of Henry Ford’s dream to make automobiles for the multitudes. Though Duesenberg’s Model A was unquestionably superior to the Model T in refined elegance, both vehicles were created with the same meticulous standards and attention to detail. The difference was in objectives: Duesenberg’s to build the best motorcar *in* the world and Ford’s to build the best motorcar *for* the world, a vehicle both extremely sturdy and extremely cheap. Pursuit of this goal led Ford to introduce the moving assembly line, which reduced production time from over twelve hours to 93 minutes

[4]. Ford designed the tools and equipment so they could be used by unskilled labor and ruthlessly simplified the vehicle and its production process. At a time when coachbuilders like Bender Body Company gave buyers full control over the form of their automobile, Ford refused to offer even a choice of color [5]. But from this austere simplicity of design sprung, paradoxically, a near limitless ability to modify the car to fit one's needs. In the New Yorker article 'Fairwell, My Lovely!', E.B. White recalls a time when the Sears Roebuck catalog contained a Ford gadget section larger than men's clothing and nearly as large as household furnishings [6]. He poetically describes the Model T's adaptability:

the purchaser never regarded his purchase as a complete, finished product. When you bought a Ford, you figured you had a start—a vibrant, spirited framework to which could be screwed an almost limitless assortment of decorative and functional hardware. Driving away from the agency, hugging the new wheel between your knees, you were already full of creative worry. A Ford was born naked as a baby, and a flourishing industry grew up out of correcting its rare deficiencies and combatting its fascinating diseases. Those were the great days of lily-painting. [6]

# FOR FORD CARS

Only those standard, thoroughly dependable Ford accessories that bring return sales and no grief are found on the following pages. And on each item we have used our tremendous buying power to secure such low costs that we in turn can name these unbeatable prices. That's why Butler Brothers is such a good place to buy Ford Supplies.

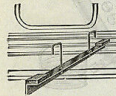


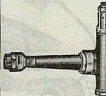


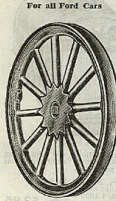


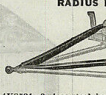


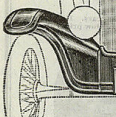
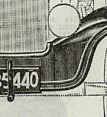




<p><b>RUNNING BOARD BRACE</b></p>  <p>4K716—1½" light in. steel angle bars, black enameled wood blocks riveted on each end to fit space under running board. With 16 in. flange over chassis, prevents sagging of rattling fenders. 1 in. hd. Lots of 6. Each, Out \$1.00</p>	<p><b>"W-X" SHOCK ABSORBER SETS</b></p>  <p>High grade malleable iron, black enameled, strong compression springs, secure easy riding qualities, prevents braking, will not squeak or rattle. Fully standard. Set of 4 in box with directions for attaching. 4K813—For touring car. Set, \$3.50</p>	<p><b>SIMPLEX "SPINNING" LOCK</b></p>  <p>4K8366—Case hardened steel "spin-off", equipped with case lock, with two keys, disengages, leaving steering wheel free to spin, easily installed without change in construction. Each in box with directions. Each, \$3.45</p>	<p><b>GUARANTEED SPINDLE BODIES</b></p>  <p>High grade, perfect alignment, requires no additional framing, hot treated and adapted to accurate gauge, special hardened steel cones, true finished, ready to install. 4K748—Right, No. 5994. Each, Out \$1.00 4K749—Left, No. 5995. Each, Out \$1.00</p>	<p><b>"A C" SPEEDOMETER</b></p>  <p>4K8771—A high grade rapid-acting instrument, fully guaranteed, 100,000 mile total register and 100 mile trip register, readable, rustless, black face with white markings. Full 2 1/2" dial included. Complete with driving shaft, gears and attachments. 1 in box. Each, \$10.50 Lots of 5, \$10.00 " 100, " 9.00</p>	<p><b>"LAZEAR" LOCK WHEEL</b></p>  <p>For All Model Fords</p> <p>4K8201—Combination one unit 17 in. steering wheel and lock, makes tire, nickel solder, set in Corbin lock, locks without key, approved by Underwriters' Laboratory, who allow 15% of insurance premium. 1 in box. Each, \$6.75</p>
<p><b>REPLACEMENT WHEELS</b></p>  <p>For all Ford Cars</p> <p>High grade malleable iron, thoroughly annealed, double arm type, cold rolled steel bolts, oil tempered surface. Set consists of four shock absorbers, layover, tie rod, cone plate with springs and all attachments and directions for installation. 2 set in carton. 4K8416—For Roadster and Touring cars. Set, \$8.50 4K8417—For Coupe and Sedans. Set, \$5.00</p>	<p><b>"W. C." SHOCK ABSORBER SETS</b></p>  <p>High grade malleable iron, thoroughly annealed, double arm type, cold rolled steel bolts, oil tempered surface. Set consists of four shock absorbers, layover, tie rod, cone plate with springs and all attachments and directions for installation. 2 set in carton. 4K8416—For Roadster and Touring cars. Set, \$8.50 4K8417—For Coupe and Sedans. Set, \$5.00</p>	<p><b>"LYON" SPRING BUMPER</b></p>  <p>4K9181—(Mfr. No. 8). Front, good quality steel, nickel with black trimmings, excellently finished. Additional 15% discount on all orders for 10 or more sets. Hangers, insurance companies allow 10% on collision insurance when cars are equipped with Lyon. 1 in crate, 10 lbs. Each, \$8.40</p>	<p><b>RADIUS ROD SUPPORTS</b></p>  <p>4K8211—2 piece steel, heavy gauge, black enameled, for 1910 and earlier model Ford cars. 1 in pkg. Each, \$2.00 4K8222—1 piece angle iron, 1 1/2 x 1/2 in., black enameled, correct mechanical construction, strong, durable, equipped with patented twin hook bolt. For all model Ford cars. 1 in pkg. Each, \$2.00</p>	<p><b>AXLE SHAFT</b></p>  <p>4K8345—Made from 40 carbon and 60 manganese steel S. A. E. Standard. Turned and ground on centers. Finished to micronometer gauge to insure accuracy. 1 in pkg., 9 lbs. Each, \$9.00</p>	<p><b>NON-RATTLING SOCKETS</b></p>  <p>Malleable iron, black enameled, hardened steel sockets, heavy spring takes up wear, prevents rattling, malleable screw plug with lock, order of coating and 1/2" roller rod sockets for complete set. 1 in box. 4K3428—Radius rod socket. Each, \$3.00</p>
<p><b>"MOSCO" BLACK ENAMELED STEEL FRONT APRON</b></p>  <p>Fits 1917 to 1920 Models</p> <p>4K9660—Heavy steel, reinforced corners, 2 coats black enamel on exterior. Compliant with bolts, nuts and lock washers and the instructions for attaching. Protects fenders, and both of mud guards. 1 in crate. Each, \$2.00</p>	<p><b>"PEERLESS" HONEYCOMB TYPE RADIATORS</b></p>  <p>Black enameled, honeycomb core practically indestructible and very strong and efficient. Large water capacity, cooling surface 50% more than type of core used on standard equipment. Guaranteed to keep cooling cool under the most severe conditions. Freezing will not harm it, particularly efficient in Ford cars when converted into Truck Units or Tractors. 1 in crate. 4K8485—One piece construction, 1910 and earlier Fords. Each, \$12.00 4K8486—Black enameled sheet, 1917 and later Fords. Each, \$12.00</p>	<p><b>FRONT AND REAR FENDERS</b></p>  <p>For all Ford Cars</p> <p>50 extra steel, 2 coats lustrous black enamel, baked on. Exact duplicate of factory equipment.</p> <p>4K8501—Front, 2 fenders, for 1917 and later Fords. 1 pr. in crate. Pr. \$6.50</p>	<p><b>DRIVE SHAFT</b></p>  <p>4K8346—Made from 40 point carbon and 60 point manganese steel S. A. E. Standard, turned and ground on centers. Finished to micronometer gauge, to insure accuracy. 1 in pkg. Each, \$12.50</p>	<p><b>"PARAMOUNT" COMMERCIAL REAR FENDER</b></p>  <p>The regular Ford chassis. Exact duplicate of factory equipment.</p> <p>4K8506—Heavy gauge steel, 2 coats lustrous black enamel, baked on. Matches from fenders of the Ford car. Easily attached. 1 pr. in crate. Pr. \$4.50</p>	<p><b>GUARANTEED FRONT AND REAR SPRINGS</b></p>  <p>For All Ford Cars</p> <p>4K8696—"Furbill" front spring, 2 bar, oil tempered, special analysis spring steel, black polished, guaranteed against defects in workmanship and material. 1 in pkg. Each, \$1.25</p> <p>4K8698—"Furbill" rear spring, high grade special analysis spring steel with spring steel strip, painted black, guaranteed against defects in workmanship and material. Weight 41 1/2 lbs. Each, \$4.25</p> <p>4K8695—"Flamingo" guaranteed front spring, made of special quality alloy steel, has no center bolt or center rib to weaken beam. Arch and wide shaped flash make black of bolt and 1/8" and 1/4" diameter bolts are furnished for one year against breakage and sagging at any point. 1 in pkg. Each, \$4.75</p>

Figure 1.1: 1922 advertisement for Model T Ford parts and accessories [7].

12 years after his purchase, this 'creative worry' struck Carl Maute. He used a hacksaw to remove the back seats and built a wooden bed to accommodate his needs as a farmer and carpenter [8]. It was now *his* Model T, capable of assisting Maute in his unique pursuits better than any vehicle rigidly designed for the general population. And should those pursuits change, his Model T could adapt in response.



## 1.1 Motivation

The species of car owner willing and able to modify their vehicle to suit their needs is not quite extinct, but is increasingly rare. The modern Tin Lizzy is made up of more silicon than tin. As many as 50 computers whir away in today's cars, and they only obey commands given by the subset of mechanics with the right electronic key. Where the Model T had around 1,481 individual parts [9], the modern automobile has upwards of 30,000 [10]. This increase in complexity is not unwarranted, as modern vehicles are far safer, better performing, and more efficient than vehicles of the previous century. But improvements gained by increasing complexity come at a cost of clarity, robustness, and flexibility. The more a system exceeds the limits of human understanding, the more difficult it is for humans to adapt it. A prospective car modder may have a clear objective in mind, increasing fuel economy, for instance, but determining which components can be modified or replaced in service of that goal without compromising the system's integrity is a mind-boggling challenge indeed. If Maute sought to modify a Ford from this decade, where would he start when wading through the modern equivalents of Sears Roebuck catalogs?

In designing systems for the modern Ford, or any sufficiently complex engineering product, the engineer faces a similar challenge. In response to increasing system complexity, the modern engineer has become a highly specialized member of a larger design team. Each engineer, or even subteam, has a myopic focus on a small piece of the complex system. With such necessarily limited perspectives, the efforts an engineer makes toward a sub-goal may not complement work of other engineers in achieving the system goal. The antidote to complexity is coordination: alignment of individuals' effort through formalized design processes. The most common and straightforward approach to system design is a sequential process whereby specialized departments work independently and consecutively toward a final design. As a design moves from one step of the process to the next, disciplinary designers work inside constraints handed down from the previous step, generate a subsystem optimized for their respective discipline, and hand down a new set of constraints more limiting than before. As a result, disciplines toward the end of the design process work within a severely constrained design space and the system that results may not be optimal. This standard practice has been disrupted by the emergence of multidisciplinary design optimization (MDO). MDO takes a larger perspective and considers the interaction between disciplines in a system-level optimization framework, maintaining a broader design space throughout the design process and coordinating the efforts of individual disciplines to optimize a system-level objective.

Not only has the engineer become more specialized in his work; engineering companies have become more specialized within their industries in response to pressures from the

economic Invisible Hand. Motivations for specializing include the ability of a business to focus on a small set of core, profit-generating operations; to more easily scale production and amortize production costs over a larger number of units; and to simplify regulation compliance tasks [11]. In the automotive world, OEMs like Ford purchase from Tier 1 suppliers like Bosch, Continental, and Delphi, who in turn purchase from Tier 2 suppliers including chip manufacturers like Intel or NVIDIA. The specificity and complexity of a system snowballs as it is passed from one supplier to the next. For the engineer, this means that clean-sheet component design is a shrinking sub-task in the design of an engineered system. The remaining sub-tasks are *component selection* problems: choosing from supplier catalogs the parts which, when integrated into a system, best serve the design objective.

### 1.1.1 Component-Based Design (CBD)

In this thesis, we will refer to the practice of designing systems primarily by integrating ready-made components as “Component Based Design” (CBD). In [12], Lee and Sangiovanni-Vincentelli provide a working definition of CBD: designs “obtained by assembling strongly encapsulated entities called ‘components’ equipped with concise and rigorous interface specifications.” CBD practices both increase a company’s ability to efficiently bring new systems to market and lends new capabilities to the systems themselves. One benefit on the production side is the ability to incorporate standard or commercial off-the-shelf (COTS) components into the system. When production quantities are low and the market demands a short development time, COTS components become quite appealing [13]. For small production runs, the fixed cost of developing a component are amortized over fewer units. These fixed costs are driven by design activities as well as testing, qualification, and the setup and tooling charges for manufacturing. Big vendors can typically amortize these development costs over a large number of units since their products can be used in a variety of systems. Another motivation for the use of COTS parts is the need for faster development times, a major goal for companies in competitive markets [14]. Unless a COTS component requires extensive modification to integrate, it is far faster to purchase the component than to build it in house. Finally, use of COTS components can reduce a business’ knowledge capital since they can lean on vendors’ specialized expertise. Benefits similar to those of COTS component use can be realized through internal component reuse. This is particularly applicable to large companies with the resources to produce components that can be integrated into multiple systems. The fundamental principle remains the same: once the development costs for a component have been incurred, by either a vendor or the system manufacturer, it is most efficient to incorporate that component in as many systems as possible.

The benefits of component-based design are not restricted to production; they can also be seen in the functionality of the resulting system. One such benefit is improved repairability, as COTS or high production volume components can be more easily acquired for replacement. CBD also opens the door to modular systems that can be rapidly reconfigured in response to changing missions. In manufacturing, trends toward mass customization and shorter product lifecycles prioritize the ability of factories to 1) quickly respond to changing customer requirements, 2) resiliently retool or adjust processes in response to unforeseen system failures or quality issues and 3) retrofit new technologies onto existing systems [15]. Manufacturing systems, therefore, must be reconfigurable and versatile. Another application for reconfigurable systems is found in mobile robotics. Systems designed to assist in uncertain events, emergency situations or military countermeasures, for example, must be able to adapt to changing job requirements on the fly. CBD does pose its own challenges like the necessity of rigorous interface specifications [12], but its widespread adoption indicates companies are willing to overcome these hurdles.

## 1.2 Background

### 1.2.1 Component-Based Design Optimization (CBDO)

The previous two sections illustrated two trends in engineering system design: 1) increasing system complexity requiring higher degrees of specialization and coordination and 2) an adoption of component-based system design to reduce development cost, respond more quickly to dynamic markets, and create more adaptable systems. Current MDO tools have greatly assisted in the design of complex systems but are ill-suited for component-based design modalities. Automating CBD with optimization algorithms, referred to in this thesis as component-based design optimization (CBDO), is challenging because the component selection problem is fundamentally discrete. The problem of discrete optimization is not new; academia and industry alike have invested significant effort in the development of algorithms for such problems. The simplest and most straightforward of the discrete optimization algorithms is the exhaustive search. Though guaranteed to find the global optimum at *some point*, to enumerate and test all possibilities is computationally intractable for all but the simplest of problems. Instead of searching the entire design space, one could use a greedy algorithm that makes sequential, locally-optimal decisions one variable at a time. This approach is scalable but does not generally find the global optimum since it ignores coupling among decision variables. Branch-and-bound algorithms are commonly used to solve integer programming problems due to their robustness and general applicability, but they are

typically inefficient and most suitable for linear problems [16]. The most efficient algorithms typically take advantage of specific problem structures. One very elegant solution to certain discrete optimization problems is dynamic programming, which is used frequently in fields such as optimal control. Dynamic programming can be applied to evolutionary problems where a future state can be predicted from the current state without any dependence on past states; that is, the process can be posed as a Markov chain. When this property holds, the problem can be solved recursively with smaller sub-problems being solved first followed by larger problems that use past solution information. Unfortunately, this class of algorithms cannot be applied to the general component selection problem because the decisions cannot be decomposed into smaller independent sub-problems [16]. We are left with stochastic methods such as simulated annealing and binary genetic algorithms. These are well-suited for nonconvex problems but struggle as problem dimensionality increases, especially when purely random processes are used to drive the algorithm’s evolution [16].

In addition to discrete CBDO’s clear computational challenges, its practical application is hindered by a lack of transparency. When design decisions are reduced to a set of selection indices, the algorithm can no longer convey information that helps explain *why* a particular design was chosen. The desire for humans to understand the decisions made by complex algorithms can be seen in the explainable artificial intelligence movement [17]. An intuitive explanation of why an algorithm arrived at a particular solution can be just as, if not more, valuable than the solution itself. For the designers, such knowledge could lead to a deeper understanding of the trade-offs between design decisions or help them catch errors in models or datasets. Just as the modeling process can lead to insights about a particular design, a transparent design optimization process can expose unforeseen possibilities.

Figure 1.2 compares a traditional continuous MDO development process with a discrete CBDO development process. In the design phase, the system model (A.2 and 1.3 in the figure) requires component parameters and performance specifications as inputs and outputs performance data required to evaluate objectives and constraints. In a traditional continuous design process, these parameters are the design variables and are fed directly into the system model. In the component-based design modality where discrete components are the design variables, the system model (block 1.3) is augmented by component catalog (block 1.2) that converts selection indices into the parameters required by the model. In a manual system design approach, blocks A.1 and 1.1 would be replaced by an individual designer or design team that selects parameters or components based on experience, intuition, heuristics, or other more advanced analysis. MDO (block A.1) and CBDO (block 1.1) automate the design processes with algorithms to select parameters values (MDO) or components (CBDO) that optimize an objective.

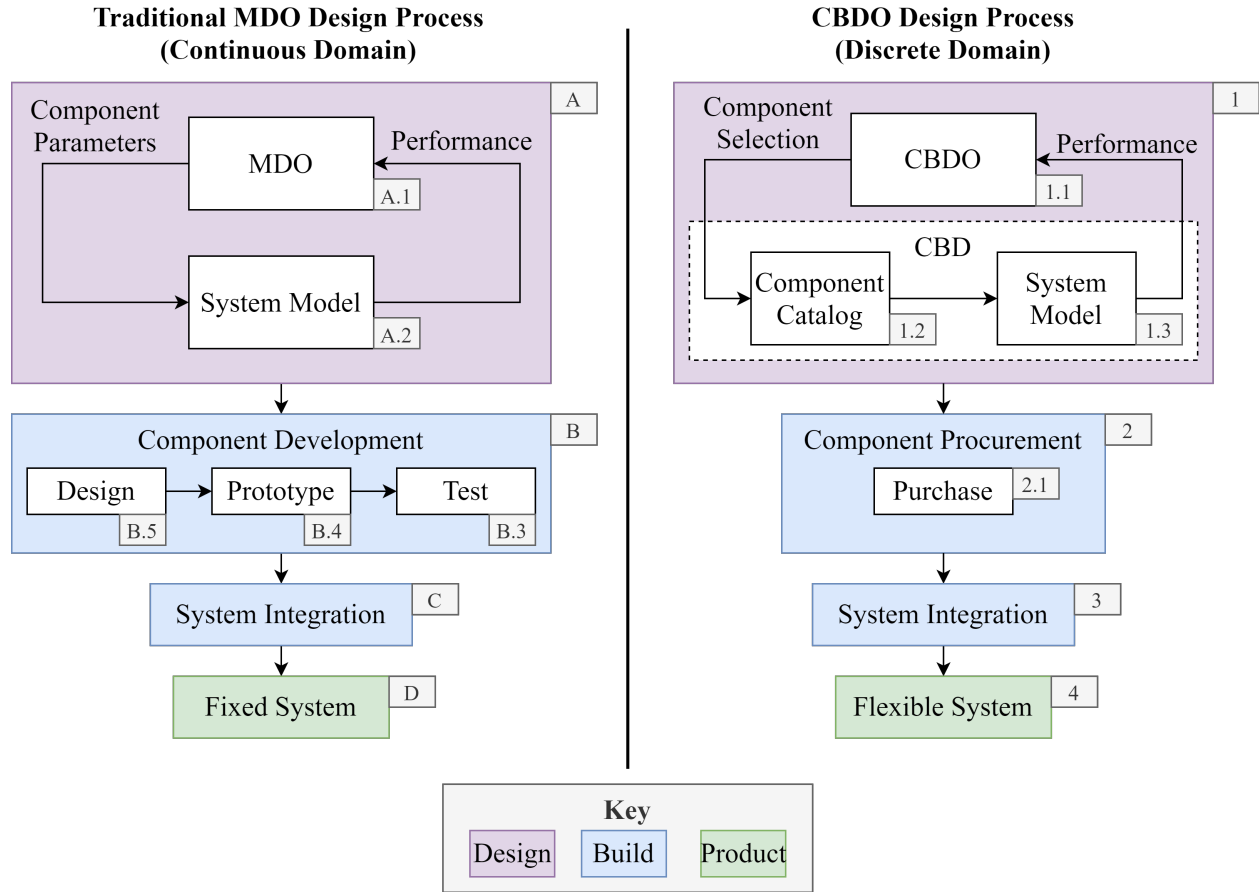


Figure 1.2: MDO and CBDO Design Processes

The production benefits of CBDO are realized in the “Build” stage. In the traditional MDO process, new system components must be designed and produced to match the optimal parameter set resulting from design block A. In CBDO, by contrast, the optimal components suggested by design block 1 are simply purchased from suppliers. Once the components have been made or acquired, both processes require the integration of the components into the final system. The system resulting from the traditional continuous MDO approach is typically fixed, as it was built from the ground up with components designed specifically for the system. The CBDO process typically results in a more flexible, modular system: different components with compatible interfaces can be substituted into the system to achieve different goals.

### 1.3 CBDO State of the Art

Though the majority of academic work on engineering design optimization focuses on continuous-domain problems, there are a few recent examples of CBDO in the literature.

Component-based software design is a large contributor to this body of work, as the approach has been noted to create higher quality software with a significant reduction in development time and costs [18]. In [18], the authors use integer programming techniques to select software components which maximize pliability, “a flexible measure that assesses software quality across different quality attributes in terms of the quality of its components.” Other approaches to optimal software component selection can be found in [19] and [20]. In the mechanical realm, CBDO has been applied in drivetrain design ([21], [22]); industrial robotics ([23], [24], and [25]); electronic devices and systems ([26], [27], and [28]); and electrohydraulic servosystems ([29]).

For the CBDO case studies in this thesis, we chose a specific platform of interest: the multirotor unmanned aerial vehicle (UAV). The multirotor’s rise in popularity and relative simplicity have led to a significant body of work on multirotor component selection. Multirotor CBDO approaches typically fall into two camps: solving the discrete component selection problem directly or parameterizing the components and solving the problem in the continuous domain.

The work of Ng and Leng [30] falls in the former category: they design small-scale quadrotor UAVs using a genetic algorithm (GA) to optimize component selection and layout. Their objective is to minimize size without violating physical constraints. Arellano-Quintana et. al. also employ a GA to optimize a multirotor design [31]. Their optimization includes rod length and diameter as continuous variables and motor, propeller, and battery selection as discrete-integer variables. These parameters are optimized for two cases: maximum thrust-to-weight ratio and maximum flight time. In “Multicopter UAV Design Optimization,” Magnussen formulates a mixed-integer linear program to select components from a set of low-cost, off-the-shelf parts [32]. The design variables include battery, motor, and propeller selection indices, the number of rotors, and the frame size; flight time, dynamic performance, and system cost are considered in his objectives and constraints. See [33], [34], and [35] for further examples of this approach.

The indirect, component parameterization approach is taken by Rothfus in [36]. In this work, the authors use a GA to optimize a set of continuous and discrete parameters that represent the battery, motor, propeller, electronic speed controller (ESC), and number of rotors. The design objective is to achieve user-specified targets on cost, payload, and flight time. Parameteric fitting functions generated from manufacturer data are used to map parameter values to mass and cost. The user is then left to select components which best match the optimized parameter values. In “Electric Multirotor Propulsion System Sizing for Performance Prediction and Design Optimization,” Bershady et. al. attempt to provide more rigorous methods for propulsion-system component selection [37]. The authors draw

from component databases to construct simple parametric relationships to predict mass from several key characteristics of the drive components. These parameterizations allow sensitivity analyses as well as range and endurance optimization in the continuous domain. Again, the user is responsible for translating continuous parameter values into discrete components. Ampatis and Papadopoulos take a similar approach in [38], where they express component functional parameters as a function of ‘equivalent length,’ defined as the cubic root of the components volume. Gradient-based algorithms then tune the equivalent lengths to optimize energy consumption or vehicle diameter with requirements on the payload, flight time, and thrust ratio. Further examples of the continuous parameterization approach can be found in [33] and [39].

Current multirotor CBDO practice is limited in a number of areas. First, authors have not yet drawn from the best of both continuous and discrete optimization techniques to efficiently solve the selection problem. Studies that parameterize components and employ gradient-based optimization fail to take the additional step of selecting real component combinations from the catalog; they leave this step to the user. Unless the catalog contains components that exactly match the continuous solution, a great deal of optimality could be sacrificed in manually translating continuous-domain solutions to discrete-domain configurations. This is especially true if component datasets are limited, the components are tightly coupled, or the fitting models used to parameterize the components are inaccurate. On the other hand, purely discrete strategies, as noted above, are less efficient and transparent than their continuous counterparts.

Another limitation is the size of the component databases used in many of the studies. For example, the dataset in [32] contains 6 propellers, 5 motors, and 5 batteries resulting in 150 discrete combinations. The dataset in [31] contains 7 motors, 8 propellers, 7 batteries, and 3 choices for the number of rotors resulting in 1,176 possibilities. In reality, these datasets could become much larger should a designer wish to incorporate multiple manufacturers or product lines into the catalog. As the discrete decision space grows combinatorially with the number of components in the catalog, it is critical that proposed solutions to the CBDO problem are proven to perform well in realistic situations where brute-force enumeration is infeasible. For parameterization-based approaches, larger datasets also help to create more dependable fitting models. Another limitation is the highly simplified steady-state models used in these optimization studies. Such models only allow for static estimations of endurance, range, or dynamic performance. This dramatically limits a study’s usefulness, especially for systems as intrinsically dynamic as multirotor aircraft. A notable exception is found in [34], where the authors use more advanced models and consider system dynamics. They enforce a controllability constraint when optimizing a multirotor’s configuration and

implement a linear-quadratic regulator (LQR) for feedback control. Tian and Voskuijl further integrate system dynamics by generating multiphysics simulation models and associated analysis functions for each candidate design [35]. Then, for each representation, a dedicated control system is automatically developed using model inversion.

Typical objectives are also highly engineering focused. In industrial applications, the true objective of a system design optimization problem is to maximize profit, a function of cost and performance. Studies occasionally consider system price as an objective or constraint, but none, to the author’s knowledge, have taken the additional step of formulating a business-focused objective like profit or return on investment. Figure 1.3 characterizes several contributions to multirotor CBDO literature along five axes.

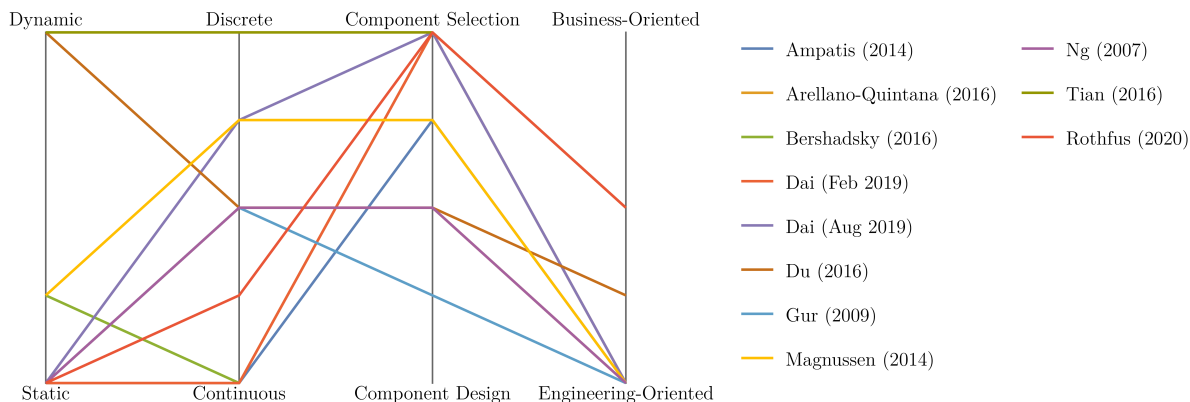


Figure 1.3: Current work in Multirotor Design Optimization

The first axis is the degree to which the author considers system dynamics in the model, objective function, or constraints. As previously mentioned, most studies only consider static problems or use static indicators of dynamic performance. The second axis describes the nature of the optimization problems formulated in the study, ranging from continuous to discrete. Studies that include both discrete and continuous variables lie on the interior of this axis, with the relative proportion of discrete to continuous variables dictating its position along the axis. The studies that are not purely continuous use gradient-free algorithms. The next axis indicates whether the underlying problem of the study is one of component selection or component design. For example, many studies include a continuous variable describing the size of the frame. Since the intent is to design a frame with this optimal length instead of selecting from a discrete set of frames, component design is an aspect of the underlying problem. The nature of the studies’ objectives are mapped to the final axis. The few that



are not purely engineering-oriented consider system cost as one of several objectives.

## 1.4 Scope of Thesis

In this thesis, a CBDO methodology is developed that leverages continuous-domain information to efficiently search the discrete space of possible components and find an optimal configuration. Underpinning the optimization is the use of graph-based modeling techniques, which are well-suited to CBDO for their computational efficiency, their applicability to multiple energy domains, and the ability to build system-level graph models out of component graph models. A generalized CBDO problem is formulated in the discrete domain and then translated into the continuous domain by parameterizing the components with physically meaningful design variables. Then, various solution methods are discussed. These include common purely discrete approaches like the GA and continuous approaches like sequential quadratic programming (SQP). The core of this work is the development of a hybrid strategy that uses the solution to the continuous problem to inform a discrete search of the design space via a distance-sorted search algorithm

The hybrid CBDO process is then applied to select a battery, motor, and propeller for a quadrotor system. The system model is developed using the aforementioned graph-based modeling techniques, and is experimentally validated against a commercial UAV platform. The discrete selection problem and corresponding continuous problems are formulated, and the continuous component parameterization is used to conduct sensitivity analyses and better understand the system. Two cases for the objective are considered: a static case and a dynamic case. In the static case, the optimization seeks to maximize the endurance of the system in steady-state hover per system price. In the dynamic case, the optimization seeks to minimize the time required to complete a dynamic mission. The performance of the algorithm is discussed, along with comparisons of the optimized design to the initial design.

## 1.5 Thesis Organization

Chapter 2 discusses modeling for design optimization and presents an efficient and modular graph-based modeling framework. It also overviews the tools used to automatically compose a system graph model from component graph models. In Chapter 3, general CBDO problem formulations are presented. In Chapter 4, the hybrid optimization approach for solving CBDO problems is developed. It outlines the process of parameterizing components with physically-meaningful design parameters and obtaining a continuous-domain representation of the problem and its solution. It also specifies the distance-sorted search algorithm used to

obtain a discrete solution using continuous-domain information, and it contains a comparison of the hybrid approach with direct approaches such as the GA. In Chapter 5, the hybrid approach is applied in two case studies: optimizing a quadrotor system to maximize endurance per system price, and optimizing a planar quadrotor system for minimal mission duration. Graph-based powertrain models and rigid-body dynamic models are developed for the systems, and the quadrotor steady-state model is validated against an experimental test platform. The component databases and parameterizations used in both of the case studies are provided. Sensitivity analyses and optimization results are presented for both case studies, along with discussions of the optimized design and algorithm performance. Finally, Chapter 6 concludes the thesis with a summary of contributions and identifies potential directions for future research.

# Chapter 2

## Modeling

The purpose of design optimization is to optimize, or at least improve, the performance of a given system of interest. The system to be optimized performs some function and results in some output, and some function of this output can be used to evaluate the system's performance. Various system properties influence the resulting output. Some of these properties are fixed quantities, say gravitational acceleration, while others are directly or indirectly influenced by design decisions, aspects of the system that its designers have direct control of like dimensions or materials. It is these design decisions which the design optimization process seeks to inform, guiding the designer toward a design with good performance. Essential to this process is the ability to predict, with a system model, what the system's performance will be given a set of design choices. A model is an abstract description of the real world that approximately represents a system's complex functions, making it feasible to predict performance before building and testing the system. To begin, the system of interest must be fully defined. All systems that could ever be designed exist within an environment or another larger system. One must choose an appropriate system boundary that aligns with the purpose of the analysis. Anything that crosses the boundary is a link between the system and its environment, representing an input or output of the system. Even after defining a system boundary, one must choose a particular level of fidelity to describe the system. At the most fundamental level, the system is a collection of atomic particles bound by electromagnetic forces. One could also view the system as a collection of components, each playing a specific role, or as the interaction of various energy domains each governed by the conservation of energy. In all cases the system is the same, only the perspective differs. It is from the chosen perspective that the model is constructed, as a photographer flattens a 3-dimensional world into a 2-dimensional, viewpoint-dependent depiction of a scene. The perspective informs the approach used to create the model. These include purely symbolic techniques, graphical techniques like bond graph modeling, or simulation-based techniques like

finite element analysis (FEA) or time-domain simulations.

## 2.1 Modeling for Design Optimization

Though functionally similar, it is important to distinguish between models created for design and those created for analysis. Design models are predictive in nature: we desire to study how a design performs and how we can influence its performance. In design models, design decisions, at least those under consideration, must be treated as input or ‘tunable’ parameters. The mathematical relationships underlying design models must be sufficiently robust to handle wide-ranging values for these input parameters to avoid artificially constraining the design space [16]. Design models must also include the decision-making criteria, or objective, as an output. In models constructed for analysis, it is common for designers to intuitively or implicitly account for system constraints when assigning values to parameters. In design models, requirements and constraints must be explicitly accounted for. For example, consider a system that includes a beam with the cross section shown in Figure 2.1.

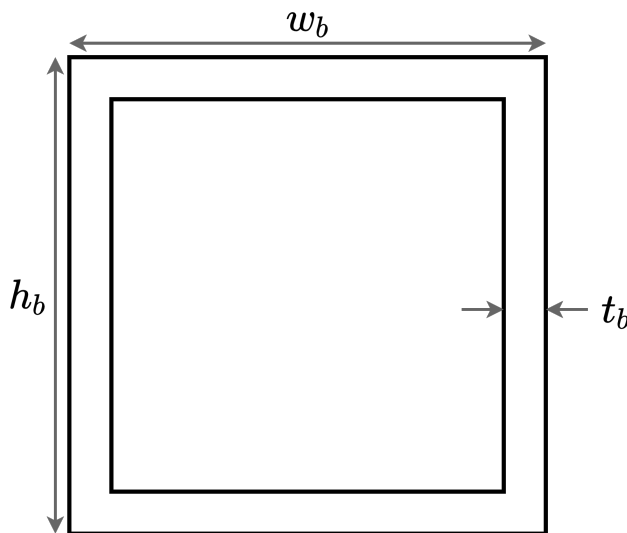


Figure 2.1: Beam cross section

The area of the beam’s cross section is calculated with Equation 2.1.

$$A_b = (w_b h_b - (w_b - 2t_b)(h_b - 2t_b)) \quad (2.1)$$

If either the beam width or beam height is less than twice the beam thickness, then the geometry is infeasible and a negative area could result. A designer understands this intuitively

and would assign dimensions that align with his or her existing notions of rectangular beam profiles. A design optimization algorithm has no such foreknowledge. It would happily choose a beam with a negative cross sectional area should the design result in a lower objective function value. To enforce a feasible geometry, the model must explicitly include the constraints given in 2.2.

$$\begin{aligned} w_b &\geq 2t_b \\ h_b &\geq 2t_b \end{aligned} \tag{2.2}$$

In his *Principles of Optimal Design*, Papalambros notes three desirable properties of mathematical models for use in design optimization [40]: 1) low computational cost, 2) continuity, and 3) differentiability.

### Low Computational Cost

An optimization study usually requires several iterations performed in the computer, where for each iteration the model functions are evaluated once or multiple times. For complicated system models, the computational cost of evaluation may become prohibitive, resulting in early termination of the optimization process. There exists a trade-off between realism and feasibility. On one hand, the designer must be able to trust the results, or at least the relationships, predicted by the model. On the other hand, the model must not be so unwieldy that meaningful results are impractical to obtain. ‘An answer’ is better than ‘no answer,’ and ‘no answer’ is often preferable to a misleading answer. The best approach to develop models for optimization is to start with the simplest meaningful model [40]. Such a model captures interesting trade-offs to be explored by an optimization study, but does so with the simplest possible mathematical relations. Complexity should only be added as required by the study of more complicated or extensive trade-offs.

Furthermore, some modeling strategies are inherently ‘lighter’ than others for a given level of accuracy. General-purpose modeling tools like FEA, so frequently employed in engineering analysis, require a significant amount of computational effort to achieve reasonable results. Though they excel in applications where a high degree of realism is required, they are less suitable for studies that seek only to understand the basic relationships between parametric inputs and performance outputs. To maximize an optimization’s utility and efficiency, one must select appropriate modeling tools for the job, exploiting system structure whenever possible.

It is also desirable to construct a model with variable fidelity or several models with different levels of fidelity. With this capability, low-fidelity implementations can be utilized early in the design process to facilitate rapid exploration of a large design space. As the

design converges, the level of fidelity can be increased to improve the accuracy of the solution.

## Continuity

To use efficient gradient-based optimization algorithms, the model functions must be  $\mathcal{C}^2$  smooth. Algorithms can often tolerate the occasional discontinuity, provided they occur away from an optimal point [16]. In some engineering problems, discontinuities are inherent in the underlying model. Examples are gear backlash or coulomb friction models. So long as such discontinuities are infrequent and exist away from an optimal point, they will not preclude the use of a gradient-based algorithm. A more troublesome cause for discontinuities are truncation and numerical noise. Optimization algorithms generally require tighter tolerances than are used for analysis [16]. Including fewer significant digits in the model functions can limit the type of optimization algorithm that can be used effectively. Noisy outputs can severely mislead or prematurely terminate an optimization. Such noise can result from converging a system of equations within a model function, simulations run within a model function, or loss of precision resulting from the evaluation process. Input and output files are a common culprit, since significant digits are often lost when data is exported to an external file [16]. It is important, then, to maintain a moderately high degree of numerical precision when constructing the model functions. Whenever possible, numerical operations like simulations are replaced with explicit relationships.

## Differentiability

As previously mentioned, gradient-based optimization algorithms require  $\mathcal{C}^2$  model functions. But they also require values for the first, and sometimes second, derivatives of the outputs with respect to the design variables. These inform the step size and direction. The best-case scenario is model functions paired with analytical expressions for the derivatives. Access to analytical derivatives is often taken for granted in optimization pedagogy or the theoretical development of optimization algorithms, but they are often quite difficult or impossible to obtain in practice. The difficulty is that many important engineering problems are based on implicit, numerical models for function evaluation, or the process of obtaining an expression for the derivatives is simply too complicated and time-consuming to justify [40]. When this is the case, more implementable but less efficient alternatives can be employed. One increasingly viable option is automatic differentiation techniques that use the chain rule to construct total derivatives from the sequence of elementary arithmetic operations and functions executed in a computer program. Another is to simply replace a complicated model function with an easily-differentiable surrogate model. Though these models take time to

construct and verify, they can dramatically improve the efficiency and robustness of the optimization algorithm. In addition to providing access to analytical derivatives, they help to smooth noisy functions and reduce the cost of a function evaluation. Care must be taken in the construction of these models, however, to avoid limiting the valid region of the design space. Some implementations build the surrogate model as the optimization progresses, extending the domain of the surrogates as needed. Finally, an expensive but effective method is finite differencing, in which the derivative of function  $f$  taken with respect to variable  $x$  is approximated numerically as

$$\frac{df}{dx} = \frac{f(x + \Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x) \quad (2.3)$$

where  $\Delta x$  is the step size. Error of the approximation scales linearly with the step size, encouraging the use of small steps. Unfortunately, the numerical challenge of subtractive cancellation arises small steps are used [41]. Small step sizes also amplify any numerical noise present in the model function. When a model function permits complex arithmetic, the complex step method (2.4) may be employed to achieve better accuracy.

$$\frac{df}{dx} = \frac{\text{Im}[f(x + ih)]}{h} + \mathcal{O}(h^2) \quad (2.4)$$

In 2.4, the step  $h$  is taken in the complex plane. In addition to quadratic scaling of error with step size, complex step manages to avoid the subtractive cancellation issue so that the step  $h$  can be chosen arbitrarily small. Also, complex arithmetic is roughly 2.5 times more expensive than real arithmetic [41].

Often, a design model sequentially evaluates multiple model functions in computing the outputs. When this is the case, one can leverage partial derivative information of each model function to construct the total derivative. This grants the flexibility to choose different differentiation techniques for each of the model functions, taking advantage of more efficient techniques whenever possible. This ‘derivative synthesis’ is the computational backbone of the popular OpenMDAO software package [42]. For more information on derivative synthesis algorithms, refer to [43].

Again, we see a need for care in selecting and implementing a modeling strategy. Methods that permit the computation of analytical derivatives, automatic differentiation, or complex arithmetic are preferable.

## 2.2 Graph-Based Modeling

In this work, a graph-based framework was selected to model the vehicle’s powertrain, as it exhibits many of an optimization model’s desirable qualities. This framework was initially developed for model-based control, and it is especially suited for hierarchical control, decentralized control, distributed control, and other applications that require system model decomposition [44], [45]. In the graph-based modeling framework, graph models are physics-based models derived from conservation laws, typically those of mass or energy. The models are energy domain independent; they inherently capture the exchange of power among different energy domains. This facilitates system-wide design, analysis, and control.

In graph models, capacitive storage elements are represented as graph vertices, and graph edges represent transfer between storage elements. When the model is based on energy conservation, vertices represent energy storage and edges represent the transfer of power. The majority of system-specific behaviors are captured by the power flow equations corresponding to each edge. Expressions describing these power flows are, most generally, nonlinear functions of the corresponding edge’s head and tail vertex. Therefore, a wide variety of interactions between elements can be captured in a single framework.

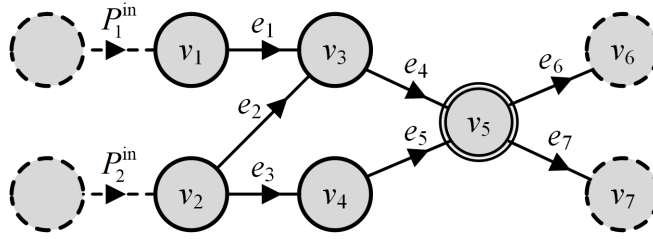


Figure 2.2: Notional system graph depicting source flows, internal edges and vertices, and sink vertices

Figure 2.2 depicts a notional oriented graph  $G = (V, E)$  containing the set of  $N_v$  vertices  $V = \{v_i\}, i \in [1, N_v]$  and the set of  $N_e$  edges  $E = \{e_j\}, j \in [1, N_e]$  [44]. The edge  $e_j$  has an associated flow  $P_j$ , and it represents the direction of positive flow from tail vertex  $v_j^{\text{tail}}$  to head vertex  $v_j^{\text{head}}$ . For each vertex  $v_i$ , the set of edges directed into the vertex is  $E_i^{\text{in}} = \{e_j | v_j^{\text{head}} = v_i\}$  and the set of edges directed from the vertex is  $E_i^{\text{out}} = \{e_j | v_j^{\text{tail}} = v_i\}$ . Each vertex is also associated with a state  $x_i$ . In the framework, flows  $P_j$  are constrained to be function of an associated input  $u_j$  and the states of the head and tail vertices.

$$P_j = f_j(x_j^{\text{tail}}, x_j^{\text{head}}, u_j) \quad (2.5)$$

Applying the conservation law to vertex  $v_i$  yields 2.6.



$$C_i \dot{x}_i = \sum_{e_j \in E_i^{\text{in}}} P_j - \sum_{e_j \in E_i^{\text{out}}} P_j \quad (2.6)$$

In 2.6,  $C_i$  is the storage capacitance of the vertex  $v_i$ . When the capacitance is zero, the state value is determined by an algebraic relationship. These algebraic vertices are depicted with an additional outer ring as shown by vertex  $v_5$  in Figure 2.2. Vertices with a dashed ring, vertex  $v_6$  in Figure 2.2 for example, are external vertices whose states are not part of the system under consideration. Instead, they belong to neighboring systems or the surrounding environment. Edges with a dashed line,  $P_1^{\text{in}}$  in Figure 2.2 for example, represent external flows from neighboring systems or the surrounding environment.

The structure of the system is defined by incidence matrices  $M$  and  $D$ , defined in 2.7 and 2.8 respectively.

$$M = [m_{ij}] = \left\{ \begin{array}{ll} +1 & v_i \text{ is the tail of } e_j \\ -1 & v_i \text{ is the head of } e_j \\ 0 & \text{else} \end{array} \right\} \quad (2.7)$$

$$D = [d_{ij}] = \left\{ \begin{array}{ll} 1 & v_i \text{ is the head of } P_j^{\text{in}} \\ 0 & \text{else} \end{array} \right\} \quad (2.8)$$

In 2.8,  $P_j^{\text{in}}$  is the  $j^{\text{th}}$  element of  $P^{\text{in}}$ , the source power flows entering the system.

With these incidence matrices defined, the dynamic equation 2.6 can be written in the linear form 2.9.

$$C \dot{\mathbf{x}} = -MP + DP^{\text{in}} \quad (2.9)$$

In 2.9,  $C = \text{diag}([C_i])$  is a diagonal matrix of capacitance coefficients,  $\mathbf{x} = [x_i]$  is the state vector, and  $P = [P_j]$  is the vector of flows along the edges in  $G$ . Refer to [44] for a detailed formulation of graph-based models.

### 2.2.1 Graph-Based Models for Design Optimization

Though initially developed for model-based control applications, graph-based models are well-suited for design optimization. Primarily, they are quite computationally efficient, allowing for fast exploration of the design space. This property is demonstrated in [46]. In this work, the authors developed a graph model for an aircraft power system and compared its simulation performance to a nonlinear model of the same system built using individual toolboxes for the electrical, air cycle machine, and single-phase thermal-fluid systems. After executing an 8000-second simulation 25 times, the graph model completed the simulation with an average

time of 25.9s, a full order of magnitude faster than the 277-second average simulation time of the nonlinear model. Another benefit of the graph model framework is its modularity [44]. Alternative system configurations can be evaluated through the rearrangement, addition, or removal of component models. Though not utilized in the present work, this functionality could be leveraged in future work to explore various system topologies alongside the CBDO analysis.

In [47], the authors expand the model framework for topology and component sizing optimization by augmenting the dynamic equation 2.9 with design matrices. These matrices capture how continuous sizing and discrete configuration variables impact the graph’s vertices and edges. They then utilized this augmented model in the optimization of a cooling subsystem and electric vehicle powertrain design, seeking to optimize thermal and electrical component sizes in addition to discrete system topology decisions. The augmented model framework is also utilized in [48]. In this work, a GA is used to optimize plant sizing variables and control gains in a closed-loop battery-ultra-capacitor hybrid energy storage system.

## 2.2.2 Automated System Model Composition and the AGILe Toolbox

Previously, graph models used in design optimization studies were assembled by hand. Though the process is straightforward, it becomes arduous as the complexity of the system increases. Furthermore, errors can easily occur in translating a system graph into its corresponding dynamic equations. The AGILe toolbox, written in MATLAB, was developed to automate the process of generating symbolic models, useful for control and design optimization tasks, for complicated systems comprising multiple components. It takes advantage of the graph-based modeling framework’s modular nature in an automatic system composition algorithm that combines component graph models into a system graph model. An extension of the AGILe toolbox, written in Python, was developed to import the symbolic system models into OpenMDAO and Dymos for optimization. OpenMDAO is an open-source MDO framework that excels in solving problems with coupled models [42]. Key to its success is the ability to efficiently calculate total derivatives across complex model hierarchies. This enables the use of gradient-based, Newton-type algorithms without having to numerically differentiate across the entire model. Built on the OpenMDAO framework, Dymos is a library for optimizing control schedules for dynamic systems [49]. Together, these tools facilitate the design optimization and optimal control of a graph-based model generated in AGILe. Figure 2.3 provides an overview of the AGILe design optimization workflow.

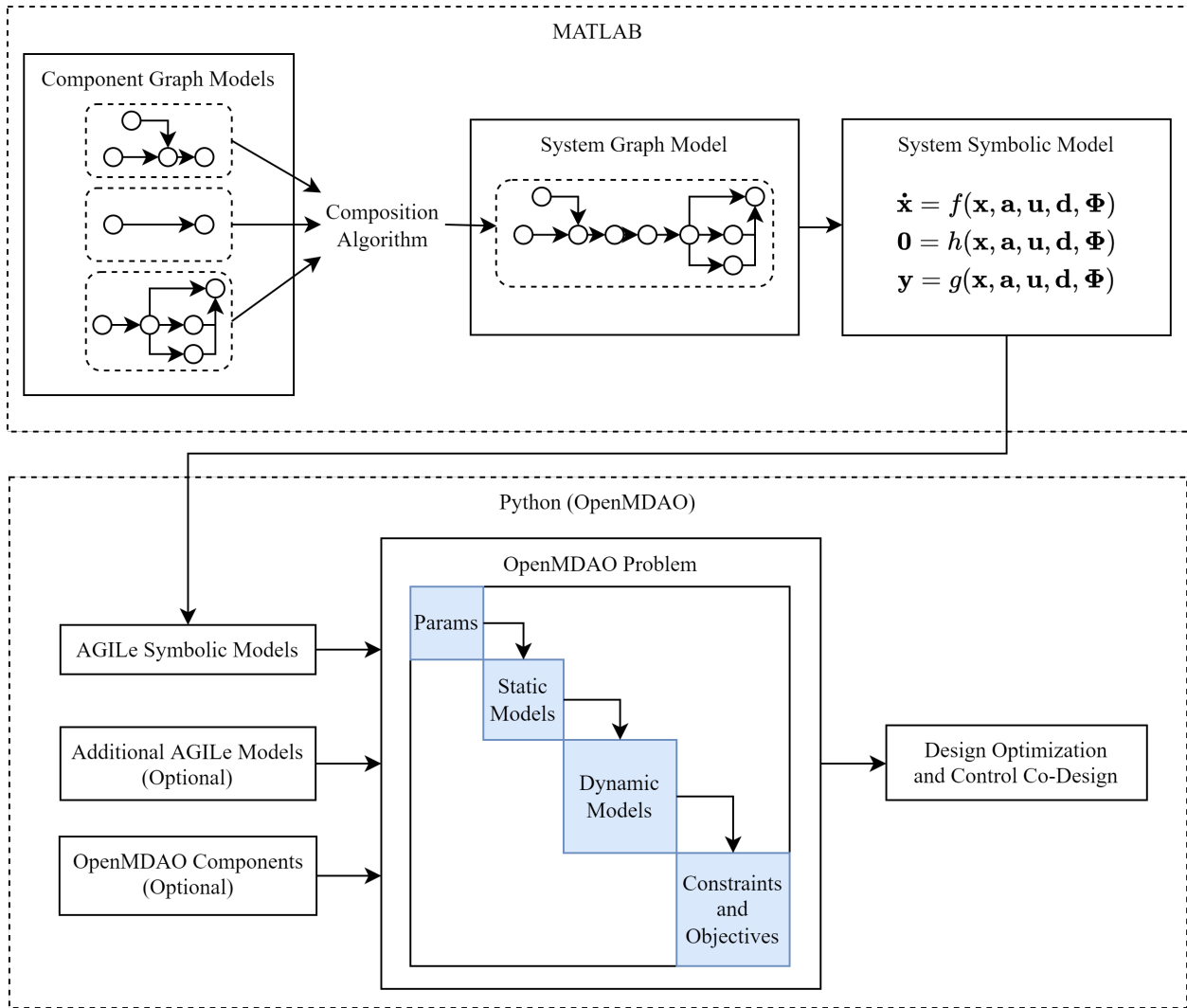


Figure 2.3: AGILe design optimization workflow

In the toolbox, users specify individual component models by defining component parameters; graph vertices and their associated capacitance functions; graph edges and their associated power flow functions, inputs, and head and tail vertices; and ports that expose a graph edge to coincide with the edge of a port of the connecting graph. Once all of the required component models have been defined, system graphs are automatically generated in a ‘Combine’ function. Arguments to this function are 1) a list of components to be connected and 2) a list of connecting ports.

After synthesizing component graphs into a system graph, the user has the option to export it to one of several symbolic model types:

1. An ordinary differential equation (ODE) model of the form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}, \Phi) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{d}, \Phi)\end{aligned}\tag{2.10}$$

2. A differential-algebraic equation (DAE) model of the form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{d}, \Phi) \\ \mathbf{0} &= \mathbf{h}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{d}, \Phi) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{d}, \Phi)\end{aligned}\tag{2.11}$$

3. An algebraic equation (AE) model of the form:

$$\begin{aligned}\mathbf{0} &= \mathbf{h}(\mathbf{a}, \mathbf{u}, \mathbf{d}, \Phi) \\ \mathbf{y} &= \mathbf{g}(\mathbf{a}, \mathbf{u}, \mathbf{d}, \Phi)\end{aligned}\tag{2.12}$$

In the above equations,  $\mathbf{x}$  is the dynamic state vector,  $\mathbf{a}$  is the algebraic state vector,  $\mathbf{u}$  is the input vector,  $\mathbf{d}$  is the disturbance input vector, and  $\Phi$  is a vector of constant parameters. Though all of the states in a graph model are dynamic in general, algebraic states arise when the capacitance associated with a state is zero; that is, there exists an algebraic relationship between inputs and outputs. For the ODE model, if algebraic states are present, then these are solved for explicitly and substituted into the dynamic equations. When exporting to a DAE model, the algebraic states are represented implicitly in the residual function  $\mathbf{h}$ . Finally, if all the states in the graph are algebraic, an AE model is exported. Furthermore, if the user desires, the Jacobians of each model function can be calculated. For evaluation, the model functions and their Jacobians can be exported to memory as an anonymous MATLAB function, to an external MATLAB function, or converted to Python syntax and saved externally as a .py file.

When importing an AGILE model into OpenMDAO, the user can choose to import a static model or a dynamic model. The static model may be used to solve a purely algebraic system or to find the steady-state solution of a dynamic system. In solving a static model, any dynamic states are made algebraic and combined with the model's other algebraic states. In a dynamic model, the system dynamics are handled by the Dymos library. The implementation for an ODE model is shown in [2.4a](#). As mentioned previously, exporting an ODE model that contains algebraic relationships requires the symbolic solution of the algebraic states and substitution of those states into the dynamic equations. One problem with this approach

is that it makes the exported models difficult to interpret, since explicit expressions for the algebraic states can be quite unwieldy. Another issue has to do with solution efficiency: the solve-and-substitute approach can lead to dense Jacobians that slow system convergence. An alternative is the DAE framework shown in 2.4b. In the DAE approach, the algebraic states are numerically solved for by driving the residual function  $\mathbf{h}$  to zero. Though this creates an additional numerical step in the solution process, it results in a much simpler, sparser Jacobian of the dynamic equations  $\dot{\mathbf{x}} = \mathbf{f}(\cdot)$ . Once the algebraic states are converged, they are fed into the dynamic equations to calculate the state derivatives. Appendix C provides an example of a DAE model exported to Python functions.

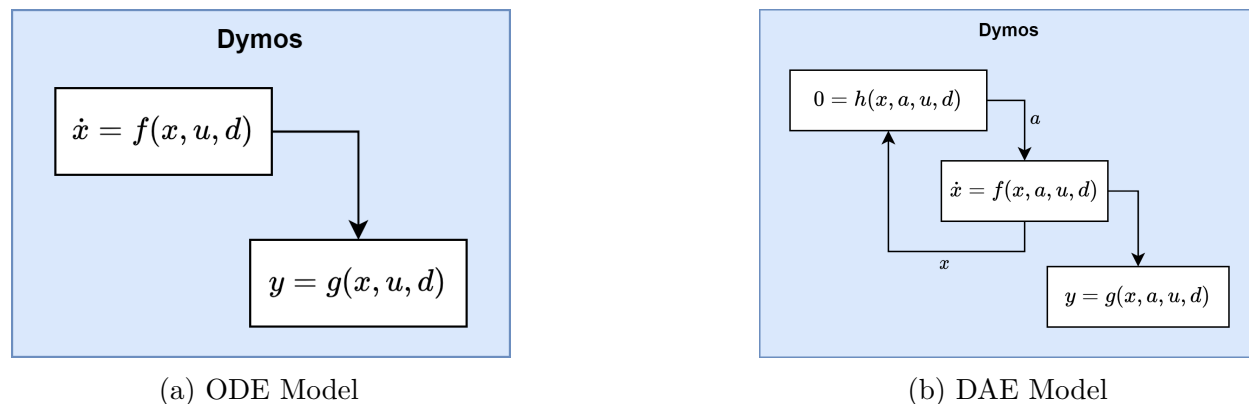


Figure 2.4: DynamicModel architectures

For maximum flexibility, the toolbox allows one to integrate multiple AGILE models and additional OpenMDAO components into the OpenMDAO system model. For example, a simplified static AGILE model and a full dynamic AGILE model can be evaluated simultaneously to provide steady-state values along with system dynamics. When some system dynamics are modeled outside the graph framework, these can be represented by standard OpenMDAO components and coupled to the graph-based model within the OpenMDAO system. For example, a system’s rigid-body dynamics could be modeled by manually defining OpenMDAO components while the system’s powertrain is modeled as a graph-based AGILE model. In the OpenMDAO system, the outputs of the powertrain model could be fed as inputs to the rigid-body model for a complete system representation. Additionally, the ability to integrate OpenMDAO components from outside of AGILE is useful for defining objective or constraint functions of signals within the AGILE model.

In previous design optimization studies of graph-based models, the system was augmented with design matrices to be tuned by the optimizer. In AGILE, design variables are represented symbolically in the system equations and can therefore be modified directly. This is made possible with the ‘Params’ framework. ‘Params’ arranges system parameters in a directed

dependency graph. Each node of the graph is a parameter, and each edge represents a functional dependence of the child parameter at the edge's head on a parent parameter edge's tail. The user specifies the function that maps parent parameter values to child parameter values. When exporting an AGILE model, the user has control over which parameters are 'Tunable' and which are not. Tunable parameters become symbols in the exported model, and their dependency function, parents, and other properties are stored in an external metadata file. Non-tunable parameters are converted to numeric values at export. This functionality allows a user to minimize the complexity of the exported model by only including parameters that will be of interest in future design studies.

The directed dependency graph structure of parameters is maintained when an AGILE model is imported into OpenMDAO. In OpenMDAO, the user has control over which parameters become design variables in an optimization study. They also have control over the parameter's dependency state. In a dependent state, a parameter's value is the output of its dependency function. In an independent state, its value is manually assigned by the user and held constant. A parameter's dependency state can be easily toggled on and off without having to rebuild the model. This feature is critical for the hybrid optimization process described in forthcoming chapters.

# Chapter 3

## CBDO Problem Formulation

### 3.1 General CBDO Formulation

A general CBDO problem formulation is presented in Equation 3.1.

$$\begin{aligned} & \underset{\mathbf{i}}{\text{minimize}} && J(\mathbf{C}(\mathbf{i})) \\ & \text{subject to} && 1 \leq i_t \leq N_C^t \forall t \in T, \\ & && 0 \geq \mathbf{g}_P(\mathbf{C}(\mathbf{i})) \end{aligned} \tag{3.1}$$

In the formulation, the design decisions are selection indices  $\mathbf{i} = [i_1 \ i_2 \ \dots \ i_{N_T}]^\top$ . Each entry in  $\mathbf{i}$  belongs to the set of positive integers and represents a selection for each component type  $t \in T$ , where  $T$  is the set of all component types and contains  $N_T$  elements. A component type can be thought of as an individual decision for which there are several alternatives. For example, if the CBDO problem includes the selection of a battery, then the battery would be a component type. Each type  $t$  is also associated with an ordered component database  $C_t$  containing  $N_C^t$  components.  $C_t$  maps the type's selection index to the parameter vector  $\Phi_c$  corresponding to a specific component. That is,  $C_t : \mathbb{Z}_+ \rightarrow \mathbb{R}^{N_\Phi^t}$ , where  $N_\Phi^t$  is the length of the parameter vector associated with type  $t$ . Component parameters include both design parameters  $\Phi_c^d$ , such as dimensions, as well as performance parameters  $\Phi_c^p$ , such as mass. Thinking of a component as an individual system, design parameters are inputs while performance parameters are outputs. The combined component database  $\mathbf{C}$  maps the selection index vector  $\mathbf{i}$  to a complete parameter vector  $\Phi = [\Phi_{c,1}^\top \ \Phi_{c,2}^\top \ \dots \ \Phi_{c,N_T}^\top]^\top$ .

Specifically,  $\mathbf{C} : \mathbb{Z}_+^{N_T} \rightarrow \mathbb{R}^{N_\Phi}$ , where

$$N_\Phi = \sum_{t=1}^{N_T} N_\Phi^t$$

To evaluate system performance, the combined parameter vector must include all the parameters required by the system model. The component databases, therefore, must specify the parameters' values for each component. Herein lies a key condition for effective CBDO: components must be sourced from suppliers that provide thorough specifications and, when necessary for calculating performance parameters, test data. One typically pays a premium for well-documented components, though superior quality of service and product often justifies the expense.

The cost function  $J$  is the metric by which system performance is evaluated, typically a function of system model outputs. The steps to evaluate  $J$  as a function of selection indices includes 1) calculating the stacked parameter vector with the component databases, 2) evaluating the system model for the resulting parameter values, and 3) calculating the cost function from the model outputs.

Each selection index  $i_t$  can be no larger than the number of components in the corresponding database, as required by the first constraint in 3.1. The physical constraint function  $\mathbf{g}_P(\cdot)$  is a vector-valued function that captures any engineering constraints or physical limitations of the system and is typically a function of model outputs.

## 3.2 Dynamic CBDO Formulation

Many modern engineering systems are inherently dynamic, and their dynamic behavior constitutes a large part of their value [50]. Such systems can be divided into two categories: passive systems and active systems. Passive systems require no external energy and rely solely on intrinsic dynamical behavior. A classic example of a passive system is a vehicle suspension that uses only well-tuned springs and dampers to provide a comfortable ride. Active systems, on the other hand, include a control mechanism that actively influences the system's dynamics. A building's HVAC system is active because it includes a thermostat that controls the level of heating or cooling. When designing a dynamic system, a common approach is to optimize a proxy objective like mass or thrust ratio that doesn't require the evaluation of system dynamics within the optimization model. Though practical, this approach can only yield an approximate solution. To optimize a system's performance for a given mission, the system dynamics must be integrated into the formulation. In 3.2, the



CBDO formulation in 3.1 is extended to include system dynamics.

$$\begin{aligned}
& \underset{\mathbf{i}, \mathbf{u}(t), \xi(t)}{\text{minimize}} && J_d(\mathbf{C}(\mathbf{i}), \mathbf{u}(t), \xi(t)) \\
& \text{subject to} && \dot{\xi}(t) = \mathbf{f}(\xi(t), \mathbf{u}(t), \mathbf{C}(\mathbf{i})), \\
& && i_t \leq N_C^t \forall t \in T, \\
& && 0 \geq \mathbf{g}_P(\mathbf{C}(\mathbf{i})), \\
& && 0 \geq \mathbf{g}_D(\mathbf{u}(t), \xi(t), \mathbf{C}(\mathbf{i}))
\end{aligned} \tag{3.2}$$

The dynamic cost function  $J_d$  replaces the static cost function  $J$  in 3.1. Two additional design variables are added in 3.2: the control input trajectory  $\mathbf{u}(t)$  and the system state trajectories  $\xi(t)$ . When solving an open-loop dynamic CBDO problem, the time-varying input trajectory  $\mathbf{u}(t)$  is solved for directly. If the system includes a parameterized feedback controller, then the controller parameters would replace the input trajectory as optimization variables. The state trajectories  $\xi(t)$  capture the evolution of the dynamic system over time and are governed by the dynamic equation  $\dot{\xi} = \mathbf{f}(\cdot)$ . The dynamic equation is presented in its most general form in 3.2, depending on the system state, control decisions, and system parameters. Constraints not dependent on time comprise the static constraint function  $\mathbf{g}_P$ , while constraints on time-varying signals comprise the dynamic constraint function  $\mathbf{g}_D$ .

# Chapter 4

## Hybrid Optimization

Formulations 3.1 and 3.2 are nonlinear integer and nonlinear mixed-integer programs, respectively. As mentioned in Section 1.2.1, solving these discrete formulations is computationally challenging and limits the transparency of the solution process. The fundamental contribution of this work is a hybrid approach to solve CBDO problems; an overview of the solution process is given in Figure 4.1.

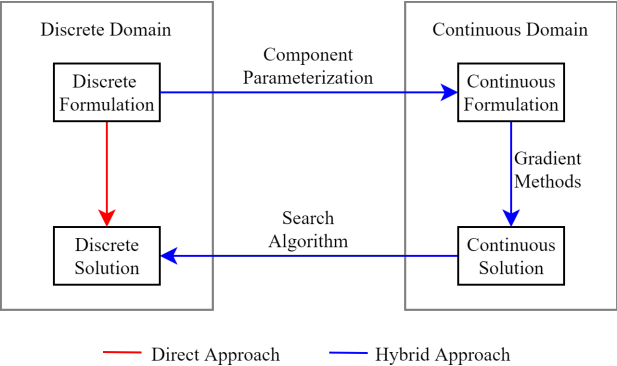


Figure 4.1: Overview of hybrid optimization process

The hybrid approach is an indirect solution strategy: the problem is first converted to a continuous-domain nonlinear program (NLP) via component parameterization and regression models. A solution to the continuous-domain representation of the problem is then found using gradient-based optimization methods. Finally, a search algorithm is employed to obtain a discrete solution, a solution of the original problem, from the continuous solution. As noted in 1.3, obtaining a continuous representation with parameterization and regression models is common practice in optimization. In our hybrid approach, we go a step further and search for the true discrete solution instead of returning the continuous solution.

## 4.1 Component Parameterization

### 4.1.1 Continuous Parameterization

In the discrete formulations, components are identified by selection indices that locate a component within the component databases. To convert the problem to the continuous domain, the mapping from discrete selection index to a component must be replaced with a mapping from a continuous variable, or variables, to a component. That is, the components must be *parameterized* by continuous variables. A component’s design parameters, those which the component designer directly controls, provide a natural parameterization. Unlike selection indices, which contain no information about the component itself, design parameters provide a physically meaningful identity. In the hybrid approach, performance parameters are modeled as functions of design parameters. Therefore, the design parameters must provide a one-to-one mapping to a component; each component must have a unique design. When several components in a supplier’s catalog have the same design parameter values, it is likely that the distinguishing parameter(s) were unidentified or unspecified. This often occurs when manufacturers differentiate components with product lines, encapsulating groups of design parameters and concealing their numerical values. Some design parameters may also be discrete, for example, the number of cells in a battery pack. In the hybrid optimization approach, discrete parameters must be relaxed to allow for continuous values and categorical parameters like product lines must be eliminated or replaced with numerical parameters.

### 4.1.2 Parameter Surrogate Models

In the hybrid approach, a parameter surrogate model is used to predict performance parameter values as a function of design parameter values. The component database serves as the foundation for the surrogate models, and any regression modeling technique can be used to generate them. When the relationship between design parameters and performance parameters is governed by a physical law, it is often advantageous to use parametric regression to fit the data to a specific functional form. For a concrete example, suppose a motor’s design parameters include height  $h$  and radius  $r$ . To predict the mass  $m$ , a performance parameter, one might reasonably fit the data to

$$m = \kappa h(\pi r^2) \tag{4.1}$$

where  $\kappa$  is a parameter of the regression model. This model assumes the motor is a solid cylinder and uses  $\kappa$  as an effective density to be determined by the data. When an underlying

relationship between design and performance is unknown, nonparametric regression techniques may be employed. Locally estimated scatterplot smoothing (LOESS), for instance, constructs a smooth model by fitting simple functions to local subsets of the data [51].

Selecting an appropriate surrogate model is a core responsibility of the designer in the hybrid approach. The model must be smooth for compatibility with gradient-based optimization, and analytic partial derivatives are desirable to improve computational efficiency. Also, the designer must balance smoothness with accuracy in selecting a surrogate model. Smoothing improves problem convexity, helping to prevent the solver from converging to local minima, but overly smooth models may fail to align with component data. When the data is nonconvex, it may be appropriate to select a model with controllable smoothness. The continuous optimization formation can then be solved by starting with very smooth surrogate models to capture general trends and iteratively decreasing model smoothness to improve solution accuracy as in [52].

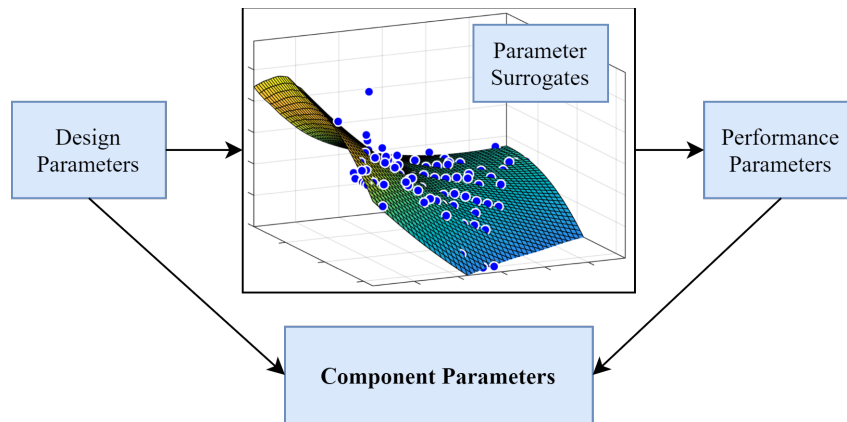


Figure 4.2: Parameter surrogate models used to calculate performance parameters

### 4.1.3 Boundary Constraint Functions

In an optimization analysis with the hybrid approach, it is important that the surrogate models do not extrapolate into regions of the design space where components do not exist. This is because we seek only to use the surrogate models as a continuous approximation of the component database. Moving too far away from the discrete components in a continuous analysis will make the search for the discrete solution more difficult. Also, because the surrogate models are generated from data in the component databases, extrapolating beyond the component data can lead to invalid results. Drawing a bounding box about all the components in a component database would be a natural choice, but, depending on the shape of the data, it can still contain large areas without components. To constrain the design parameter values further, boundary constraint functions are developed. Let  $P_t$  be the set

of points in the design variable space from data in the component database  $C_t$ . That is, elements of  $P_t$  are the design parameters  $\Phi_c^d$  of each component  $c$  in  $C_t$ . Construction of the boundary constraint function starts by taking the convex hull  $B$  of  $P_t$ . That is,  $B$  is the minimum convex set containing  $P_t$ , and it serves as a tighter boundary than a bounding box. The design variable space is then shifted so that the centroid of  $P_t$  lies at the origin and scaled by the range of  $P_t$ . That is, for any point  $\Phi_c^d$  in  $P_t$ , the point  $\tilde{\Phi}_c^d \in \tilde{P}_t$  under transformation  $T_t$  is

$$\tilde{\Phi}_c^d = T_t(\Phi_c^d) = K_t (\Phi_c^d - \mathbf{p}_C) \quad (4.2)$$

where  $\mathbf{p}_C$  is the centroid of the set  $P_t$  and  $K_t$  is a scaling matrix defined below.

$$K_t = \text{diag} \left[ \frac{1}{p_1^{\max} - p_1^{\min}} \quad \frac{1}{p_2^{\max} - p_2^{\min}} \quad \cdots \quad \frac{1}{p_M^{\max} - p_M^{\min}} \right] \quad (4.3)$$

Above,  $p_k^{\min}$  and  $p_k^{\max}$  is the minimum and maximum, respectively, of  $P_t$  along dimension  $k$ , and  $M = N_t^d$  is the number of design parameters associated with type  $t$ . This scaling of the design space maintains good problem conditioning when design variables have dissimilar magnitudes. Without scaling, the boundary constraint function is dominated by the variable with the larger magnitude. Let  $\tilde{B}$  be the result of applying transformation  $T$  to the convex hull  $B$ . Next, the scaled convex hull  $\tilde{B}$  is smoothed to avoid discontinuities in the boundary constraint function. This smoothing is achieved using the buffer function in Python's Shapely package [53], which returns an approximate representation of the set of points a given distance from a polygon.  $\tilde{B}$  is shrunk using the buffer function, resulting in  $\tilde{B}'$ . Then, the Hausdorff metric between  $\tilde{B}$  and  $\tilde{B}'$  is calculated. The Hausdorff distance is the maximum of all the distances from a point in one set to the closest point in another set as defined in [54]. Expanding  $\tilde{B}'$  by the Hausdorff distance results in a smooth boundary  $B_s$  that tightly includes each vertex of  $B$ . The scaled boundary constraint function  $\tilde{g}_B^t(\cdot)$ , corresponding to type  $t$ , is defined as the minimum distance from any point in the scaled design space to  $B_s$ , with points inside  $B_s$  taking a negative value. Because the formulation works with unscaled design parameters, it is useful to define the boundary constraint function as the composition of the transformation  $T_t$  and the scaled boundary constraint function 4.4.

$$g_B^t(\cdot) := \tilde{g}_B^t(T_t(\cdot)) \quad (4.4)$$

Therefore, any point  $\Phi_t^d$ , which represents the design variable values in the continuous domain for a component of type  $t$ , is valid if it satisfies 4.5.

$$g_B^t(\Phi_t^d) \leq 0 \quad (4.5)$$

## 4.2 Continuous-Domain Representations and Solution

Substituting the surrogate models for the component databases, replacing the selection indices with the continuous design parameters, and adding the boundary constraint functions results in a continuous-domain representation of the problem.

### 4.2.1 CBDO Formulation

The continuous-domain representation for the CBDO problem is given in 4.6.

$$\begin{aligned}
 & \underset{\Phi^d}{\text{minimize}} && J(\Phi^d, \mathbf{S}(\Phi^d)) \\
 & \text{subject to} && 0 \geq g_B^t(\Phi_t^d) \forall t \in T, \\
 & && 0 \geq \mathbf{g}_P(\Phi^d, \mathbf{S}(\Phi^d))
 \end{aligned} \tag{4.6}$$

In 4.6,  $\Phi^d$  is a vector combining the design parameters of each component type, and  $\Phi_t^d$  is a vector of type  $t$ 's design parameters. The parameter surrogate function  $\mathbf{S}$  uses the parameter surrogates to calculate the performance parameters from the design parameters. That is,  $\Phi^p = \mathbf{S}(\Phi^d)$ . The boundary constraint functions  $g_B^t$ , defined in the previous section, ensure the design parameters remain in the valid region of the design space. Cost function  $J$  and physical constraints  $\mathbf{g}_P$  are identical to that in 3.1.

Because 4.6 has continuous design variables and smooth objective and constraint functions, it can be solved using efficient, gradient-based optimization algorithms. These include SQP and related algorithms, which are generally considered the most efficient class of general-purpose nonlinear programming solvers [40]. Solution of the continuous-domain representation with SQP is a key contributor to the efficiency gains realized by the hybrid approach.

### 4.2.2 Dynamic CBDO Formulation

The continuous-domain representation for the dynamic CBDO problem is given in 4.7.

$$\begin{aligned}
 & \underset{\Phi^d, \mathbf{u}(t), \xi(t)}{\text{minimize}} && J_d(\Phi^d, \mathbf{S}(\Phi^d), \mathbf{u}(t), \xi(t)) \\
 & \text{subject to} && \dot{\xi}(t) = \mathbf{f}(\xi(t), \mathbf{u}(t), \Phi^d, \mathbf{S}(\Phi^d)), \\
 & && 0 \geq g_B^t(\Phi_t^d) \forall t \in T, \\
 & && 0 \geq \mathbf{g}_P(\Phi^d, \mathbf{S}(\Phi^d)), \\
 & && 0 \geq \mathbf{g}_D(\mathbf{u}(t), \xi(t), \Phi^d, \mathbf{S}(\Phi^d))
 \end{aligned} \tag{4.7}$$

In 4.7, the dynamic cost function  $J_d$ , control trajectory  $\mathbf{u}(t)$ , static constraint functions

$\mathbf{g}_P$ , dynamic constraint function  $\mathbf{g}_D$ , state vector  $\xi(t)$ , and dynamics  $\mathbf{f}$  are identical to those in 3.2. Other notation is defined above for 4.6. Solving 4.7 is somewhat more difficult because the addition of system dynamics results in an infinite-dimensional optimization problem. One approach is to use direct transcription (DT) to discretize the system dynamics, resulting in a finite-dimensional NLP [50]. In DT, a numerical integration method is used to convert the dynamic equations into a system of algebraic equations. The resulting equations, known as defect constraints, are posed as equality constraints in the optimization problem. The discretized state trajectory and any control variables are treated as optimization variables. Because DT results in a standard NLP, it can be combined directly with the system design problem. The optimization algorithm searches for the optimal system design and control values while simultaneously converging the dynamic equations.

Applying a DT approach to 4.7 results in the following formulation 4.8.

$$\begin{aligned}
& \underset{\Phi^d, \mathbf{U}, \Xi, \mathbf{t}}{\text{minimize}} && J_d(\Phi^d, \mathbf{S}(\Phi^d), \mathbf{U}, \Xi, \mathbf{t}) \\
& \text{subject to} && 0 = \zeta(\Xi, \mathbf{U}, \mathbf{t}, \Phi^d, \mathbf{S}(\Phi^d)), \\
& && 0 \geq g_B^t(\Phi_t^d) \forall t \in T, \\
& && 0 \geq \mathbf{g}_P(\Phi^d, \mathbf{S}(\Phi^d)), \\
& && 0 \geq \mathbf{g}_D(\mathbf{U}, \Xi, \mathbf{t}, \Phi^d, \mathbf{S}(\Phi^d))
\end{aligned} \tag{4.8}$$

In 4.8,  $\Xi$  is the discretized state trajectory,  $\mathbf{U}$  is the discretized control trajectory, and  $\mathbf{t}$  is the time vector of the discretization. Additionally,  $\zeta$  is a vector-valued defect constraint function that ensures the system dynamic equations are satisfied.

The DT approach presents a number of benefits over alternative formulations. First, the system dynamics are represented directly in the optimization formulation; dynamics are an integral part of the formulation. This eliminates the need for any forward simulation within the analysis, as would be the case in a single or multiple-shooting approach. Another benefit is flexibility. DT imposes no assumption on the controller architecture, which is especially helpful during early stage design. Solving the open-loop optimal control problem provides insights into the performance limitations of the system; then, once a control architecture is specified, the controller and plant can be tuned with minimal modifications to the formulation. Additionally, DT provides the ability to impose both inequality and equality constraints on trajectories, which is not generally possible with classical optimal control methods such as Pontryagin’s Maximum Principle [50]. Finally, DT promotes efficient computation. Because the optimization variables appear explicitly in the defect constraint functions, it is straightforward to obtain analytic derivatives. If electing not to use analytical

derivatives, the Jacobian sparsity pattern enables the efficient application of finite differences [50]. Finally, for linear systems, DT formulations often reduce to quadratic or linear programs [50]. Historically, DT has been challenging to implement for system design problems due to a lack of software packages that accommodate plant design variables. Most software DT packages are specific to optimal control. Further, because it is a fully-integrated approach requiring implementation at the equation level, DT does not easily mesh with popular modeling environments like Simulink. The aforementioned Dymos software package, however, flexibly accommodates system design variables via Dymos parameters. In Dymos, the user specifies the state variables, control input variables, and parameters; any of which can be considered optimization variables. The continuous-domain system dynamic equations, along with their Jacobians, must be specified as an explicit OpenMDAO component. Path and boundary constraints can be added to any state or timeseries output. Finally, a dynamic objective such as the final time of the trajectory must be specified. Once the problem is fully defined, Dymos transcribes the system dynamics via implicit collocation, also known as pseudospectral methods, to produce the defect constraints and their Jacobians. The final result is a standard OpenMDAO optimization problem that can be converged with any OpenMDAO-compatible solver. If additional OpenMDAO components are contained within the system model, their design variables and constraints are considered as well.

### 4.3 Discrete Search Algorithms

Solving problems 4.6 or 4.7 results in an optimal continuous-domain value for the system design parameters,  $\Phi^{d*}$ . Most parameterization-based approaches to the CBDO problem treat these parameter values as the solution, and require the designer to select components with similar parameter values. Two situations limit the accuracy of a solution obtained by manual selection: sparse component datasets and imperfect parameter surrogate models. If the continuous solution falls in a sparse region, the designer is forced to select a component some distance away. When components are coupled, as in most realistic CBDO problems, then one component's deviation from the continuous solution will influence the optimal design of the other components. Put another way, the answer to the problem can change when translating the solution back to the discrete domain. A similar effect occurs when the parameter surrogate model contains imperfections. Misalignment between the surrogate models' predictions and the components' true parameter values lead to some deviation between the continuous and discrete solution. A more systematic approach to obtain a discrete solution is necessary under these conditions.



### 4.3.1 Sorted Search Algorithms

The hybrid approach utilizes a class of algorithms referred to as ‘sorted search algorithms’ to obtain a discrete solution to the component selection problem. These algorithms iteratively narrow the discrete design space while increasing the fidelity of the objective function calculation. Let  $S$  be a set containing all possible designs. The algorithm first uses a low-fidelity predictor  $\tilde{J}_1$  of the true objective function  $J$  to sort  $S$ . The sorted set  $S_1^*$  is in ascending order of predictor value such that, if the predictor is reasonable, good designs appear early in the list and bad designs late in the list. At this point, the sorting process can be repeated with predictors of increasing fidelity up to  $\tilde{J}_K$ , resulting in a final sorted set  $S_K^*$ . As the predictor  $\tilde{J}_k$  increases in computational expense, it evaluates fewer designs in the previous set  $S_{k-1}^*$ . That is, the sorted set gets shorter with each iteration as poor designs relegated to the end of the list are ignored. This results in a narrowing of the design space such that the most expensive analyses are only applied to good designs. Also, at any iteration, infeasible designs may be removed from the set by predicting or evaluating constraint function values. After the iterative sorting procedure, the true objective  $J$  is evaluated sequentially for each element in  $S_K^*$  until the search terminates. Termination criteria include 1) the algorithm evaluates each configuration in  $S_K^*$  or reaches a maximum number of search iterations 2) the algorithm finds a configuration with an objective function value lower than some specified threshold value, or 3) the algorithm ‘stalls,’ that is, the best objective function value stops changing over a specified number of stall iterations. Like any discrete optimization algorithm, sorted-search approaches cannot guarantee the optimal design is found unless each of the configurations in  $S$  are evaluated. No predictor is perfect, so it is indeed possible that some good designs are missed in the filtering process. The user has a good deal of flexibility, however, in determining an appropriate balance between computational effort and solution accuracy by specifying how many configurations to evaluate at each sort iteration.

### 4.3.2 Distance-Sorted Search (DSS) Algorithm

The sorted search algorithm we developed for use in this thesis is the distance-sorted search (DSS) algorithm. In this method, the predictor  $\tilde{J}_1$  is the ‘distance’ in the design space between a configuration and a target design. In this application, the target design is taken to be the solution to the continuous representation,  $\Phi^{d*}$ . An approximation of the solution is found by solving the continuous problem, and the distance-sorted search algorithm searches for designs near that point. A flowchart of the distance-sorted search algorithm is presented in Figure 4.3.

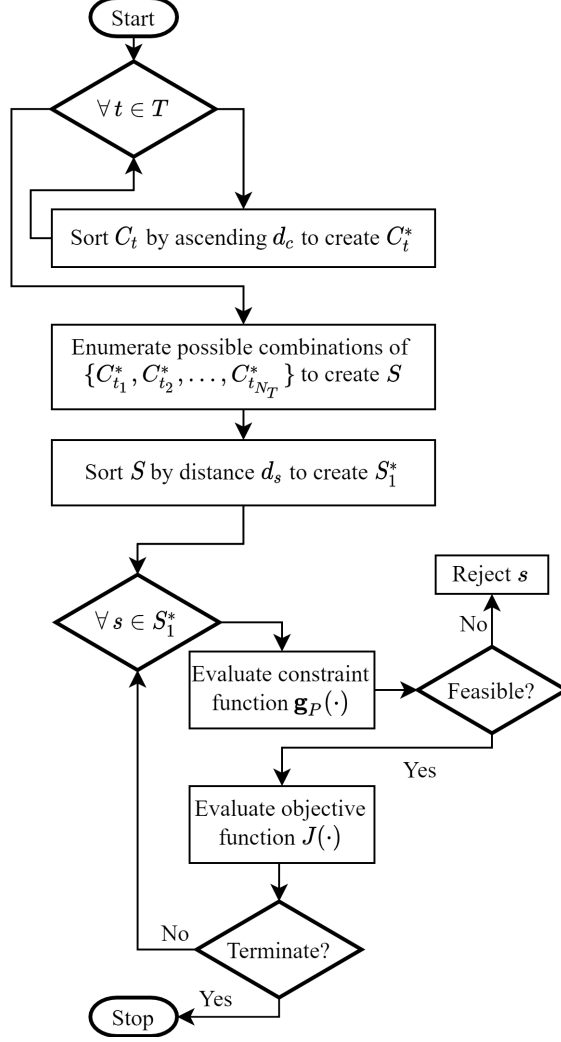


Figure 4.3: Distance-sorted search algorithm flowchart

First, a component distance metric  $d_c$ , defined in 4.9, is evaluated for each of the components in the component databases. To prevent parameters with a larger magnitude from being prioritized, the distance is scaled by the corresponding value of the target.

$$d_c = \sqrt{\sum_{i=1}^M \left( \frac{\Phi_{c,i}^d}{\Phi_{t,i}^{d^*}} - 1 \right)^2} \quad (4.9)$$

In 4.9,  $M$  is the number of design parameters associated with the component's type,  $\Phi_{c,i}^d$  is the  $i^{\text{th}}$  element of the component's design parameter vector, and  $\Phi_{t,i}^{d^*}$  is the corresponding element of the target parameters vector. Each component database is then sorted in ascending  $d_c$ , and components with large  $d_c$  may be rejected to reduce the size of the design space. Then, all possible combinations are enumerated to create the set  $S$ . Each element in  $S$  is a

configuration  $s$  that contains one component of each type. From the multiplication principle in combinatorics, the length of  $S$  is the product of the number of components in each catalog, or

$$N_S = \prod_{t=1}^{N_T} N_C^t \quad (4.10)$$

For each configuration in  $S$ , a configuration distance  $d_s$  is calculated as defined in 4.11. It is simply the norm of the component distances associated with the configuration. This metric serves as the predictor for the distance-sorted search algorithm.

$$d_s = \sqrt{(d_{c,1}^2 + d_{c,2}^2 + \dots + d_{c,N_T}^2)} \quad (4.11)$$

Then, the sorted configuration set  $S_1^*$  is obtained by sorting  $S$  in ascending order of  $d_s$ . For each configuration  $s$  in  $S_1^*$ , the parameters of each of the components in  $s$  are substituted into the system model. The constraint function  $\mathbf{g}_P$  is then evaluated, and  $s$  is rejected if the constraints are not satisfied. If  $s$  is feasible, then the objective function is evaluated. This process continues until any of the sorted-search termination criteria are met.

Note that there is some flexibility in selecting appropriate component and configuration distance metrics. For example, if some parameters are known to have a higher priority than others, then the component distance metric can be weighted as in 4.12.

$$d_c = \sqrt{\sum_{i=1}^M \left( w_i \left( \frac{\Phi_{c,i}^d}{\Phi_{t,i}^{d*}} - 1 \right) \right)^2} \quad (4.12)$$

Above,  $w_i$  is a weight associated with the  $i^{\text{th}}$  parameter, possibly determined by the gradient of the objective function with respect to that parameter at the target point. If certain component types are prioritized, then the configuration distance metric can be weighted similarly.

## 4.4 Comparison to Direct Approaches

As previously mentioned, approaches to the CBDO problem typically fall into two camps: solving the discrete selection problem directly or by parameterizing the components and solving the continuous-domain representation of the problem. In multirotor CBDO literature, the most commonly employed algorithm used to solve discrete and mixed-integer problems is the GA. Popularity of the GA is largely attributable to its global search properties, compatibility with discrete variables, and ease of implementation [16]. The GA is a natural, ‘plug and play’ solution to the CBDO problem, but it has a few fundamental drawbacks. First,

it scales poorly as the complexity of the problem increases. In the hybrid approach, the highly scalable gradient-based optimization in the continuous domain helps to mitigate the poor scalability of the discrete search. The user also has a high degree of control over the objective function predictors and iteration limits used to narrow the design space. The GA is stochastic: it randomly selects an initial population and makes random mutations for each generation. These random decisions make it difficult to intuitively understand the optimization process, to repeat or test an analysis, and to locate software bugs. To benchmark a GA's typical performance, the problem must be solved repeatedly until a statistically significant sample is obtained. The hybrid approach, on the other hand, is deterministic. If the model and algorithm parameters remain fixed, the outcome of the analysis will always be the same. Performance differences between the hybrid approach and the GA are discussed further in the following chapter.

In practice, it is important to consider an algorithm's ease of implementation in addition to its performance. The GA is quite easy to implement, requiring only the specification of objective and constraint functions to get up and running. However, should the user wish to tune the algorithm to modify its performance, they face a plethora of decisions. In MATLAB's GA implementation, custom functions can be specified for a population's "creation," "crossover," "distance measure," "fitness scaling," "mutation," and "selection" [55]. Further decisions include the population and generation size along with numerous tolerances. Appropriate values for these settings are typically determined by trial and error. The hybrid approach requires significant up-front effort to parameterize the components and, if desired, to calculate analytical gradients for the objective and constraint functions. The regression models used to generate the parameter surrogates are selected and tuned by the user. This process is transparent, however. The development of fitting models is well understood, and a fit can be visually and quantitatively verified. Tuning of the hybrid approach occurs in two independent steps: 1) selecting and tuning the gradient-based optimization algorithm and 2) tuning the DSS algorithm. The field of gradient-based NLP algorithms is mature, and textbooks such as Papalambros' *Principles of Optimal Design* [40] or Martins and Ning's *Engineering Design Optimization* [16] provide background on these algorithms as well as their practical implementation. Parameters for the DSS algorithm are straightforward. The user must specify termination parameters like iterations limits, and may specify custom component or configuration distance metrics if they desire.

# Chapter 5

## Case Study: Multirotor Design Optimization

### 5.1 Background

To demonstrate the CBDO process and other concepts discussed above, a design optimization case study is conducted. Two priorities guided the selection of application area and candidate system. First, the application needed to be realistic and align with potential applications of CBDO in the real world. Second, to perform physical model validation tests and verify design improvements under time and budget constraints, the system needed to be relatively simple and inexpensive. Electric vertical take-off and landing (eVTOL) and multirotor aircraft provide an exciting application area for CBDO. These systems have a vast design space afforded by distributed-electric powertrains. This design space includes component sizing and/or selection as well as system configuration decisions. Where the design of large jet aircraft has largely converged to the familiar airplane shape, no one eVTOL design has emerged as the most desirable. This is partly due to the strong trade-offs among range, performance, and carrying capacity present when designing aircraft. No single configuration accommodates every need; the best designs are mission specific. Mission-specific CBDO tools, therefore, could play a large role in the design of such systems.

Though a full-scale eVTOL aircraft is the ideal application for CBDO, it was not feasible to build and test such a vehicle given the scope and resources required relative to the time frame for this thesis. Quadrotor UAVs share many powertrain components with today's eVTOL aircraft, but on a far smaller scale. Their ubiquity has made them inexpensive and easily accessible. Though the quadrotor has an established configuration, the selection of powertrain components for these vehicles is largely based on heuristics, designer experience,

and low-fidelity tools. For these reasons, the quadrotor UAV was selected as the system of interest for the CBDO case study.

## 5.2 Quadrotor System

Specifically, the candidate quadrotor UAV is the HolyBro S500 V2 platform shown in Figure 5.1 [56]. The system comes as a kit which includes a frame, Pixhawk flight controller with GPS



Figure 5.1: HolyBro S500 V2 quadrotor platform

sensor, power management board, and telemetry along with easily interchangeable electronic speed controllers (ESCs), motors, and propellers. Though there are many commercially-available multirotor aircraft, the S500's low cost, open-source Pixhawk flight controller, and modular design made it an ideal test platform for component-based design. The Pixhawk flight controller and autopilot is supported by a large community of hobbyists and researchers. Though the software is largely plug-and-play, it is highly customizable and can flexibly accommodate various system configurations. It is compatible with a wide variety of sensors for state estimation and safety, as well as power monitors for accurate measurement of battery

voltage, current, and state of charge. The included GPS module and autopilot features allow the operator to pre-program a desired mission, which is useful for running the same mission in multiple experiments or trials.

The S500's use of industry-standard component interfaces makes it easy to integrate other off-the-shelf multirotor components. Increasing interest in customized multirotors among hobbyist, semi-professional, and research communities have created a large and varied supply of multirotor parts. In this CBDO case study, different batteries, motors, and propellers from third-party suppliers will be integrated with the S500's chassis, flight controller, power management, and sensors.

### 5.2.1 Component Compatibility

In CBD, the decision space is limited not only by component availability but by compatibility with the system of interest. Each component must satisfy compatibility constraints of two types: physical compatibility and operational compatibility. Physically compatible components have interfaces that are designed to fit together. For example, if a power distribution board contains a female XT-60 connector, then a physically compatible battery must include a male XT-60 connector. Operational compatibility, on the other hand, ensures that the sets of safe operating conditions for each component overlap. For example, a battery with six series cells is not operationally compatible with a speed controller that is rated for an input voltage of two to four series cells. Compatibility with the fixed system, the aspects of the design that are constant throughout the optimization, can be used to trim the component catalogs so that only feasible components are considered. Compatibility with mutable aspects of the system, however, must be enforced as optimization constraints. For example, in the following work, the system's steady-state bus current is limited by the battery, a mutable component. This operational compatibility constraint is enforced by the optimizer since the maximum current depends on the battery selection. In this section, the fixed compatibility constraints for the S500 quadrotor platform are defined. The fixed chassis and power management components are quite accommodating, but they do narrow the space of feasible batteries, motors, and propellers.

#### Battery and Electronic Speed Controller

The choice of battery that can be integrated into the system is limited by the S500's power module and the configuration's electronic speed controller (ESC). The system ships with a HolyBro PM02 power module [57]. Its current sensor can handle a maximum input voltage of 60V, limiting the battery to 12 series cells, and a maximum current of 120A. The system

also ships with BLHeli ESCs rated for a continuous current of 20A and burst current of 30A [58]. These ESCs are only compatible with batteries having no more than 4 series cells. To accommodate batteries with 2 to 6 series cells, the ESCs were later upgraded to the KDEXF-UAS20LV from KDE Direct [59]. In addition to the higher peak voltage, these ESCs have a faster refresh rate and higher peak current of 35A.

Specification	Compatibility Requirement
Series Cells	2-6
Connector	XT-60

Table 5.1: Battery compatibility requirements

## Motor

The choice of motor is constrained by the mounting pattern on the S500’s chassis, maximum system voltage, and maximum system current. The mounting hold pattern on the S500 chassis’ arm is shown in Figure 5.2. It allows for motors with 18mm or 25mm-diameter hole patterns. System voltage and current depend on the battery and ESC selection, which requires enforcement of compatibility within the design optimization process.

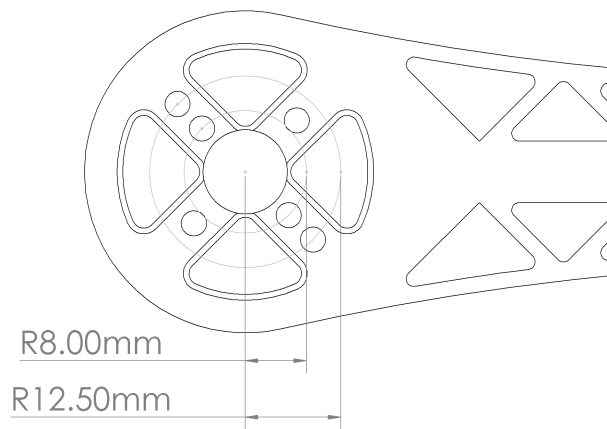


Figure 5.2: Motor mounting pattern on S500 chassis arm

Specification	Compatibility Requirement
Hole Pattern	16mm or 25mm

Table 5.2: Motor compatibility requirements



## Propeller

Lastly, the choice of propeller is limited by the chassis geometry and the motor's propeller coupling. Figure 5.3 provides a top view of the chassis and the maximum feasible propeller diameter. Any propeller with a diameter larger than 0.356m will mechanically interfere with the other propellers.

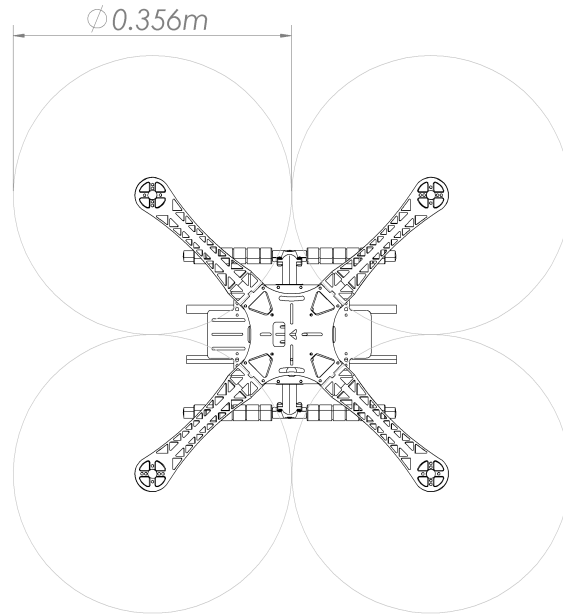


Figure 5.3: Top view of S500 chassis

Specification	Compatibility Requirement
Maximum Diameter	0.356m

Table 5.3: Propeller compatibility requirements

## 5.3 System Model

To create a design model of the quadrotor system, the system was first split into two subsystems: the powertrain and the rigid-body dynamics. The powertrain was modeled using graph-based techniques and translated to a symbolic model using the AGILe toolbox (See Appendix C). The rigid-body dynamic model was developed from Newton's second law and Euler's rotation equation. Figure 5.4 provides an overview of the model architecture.

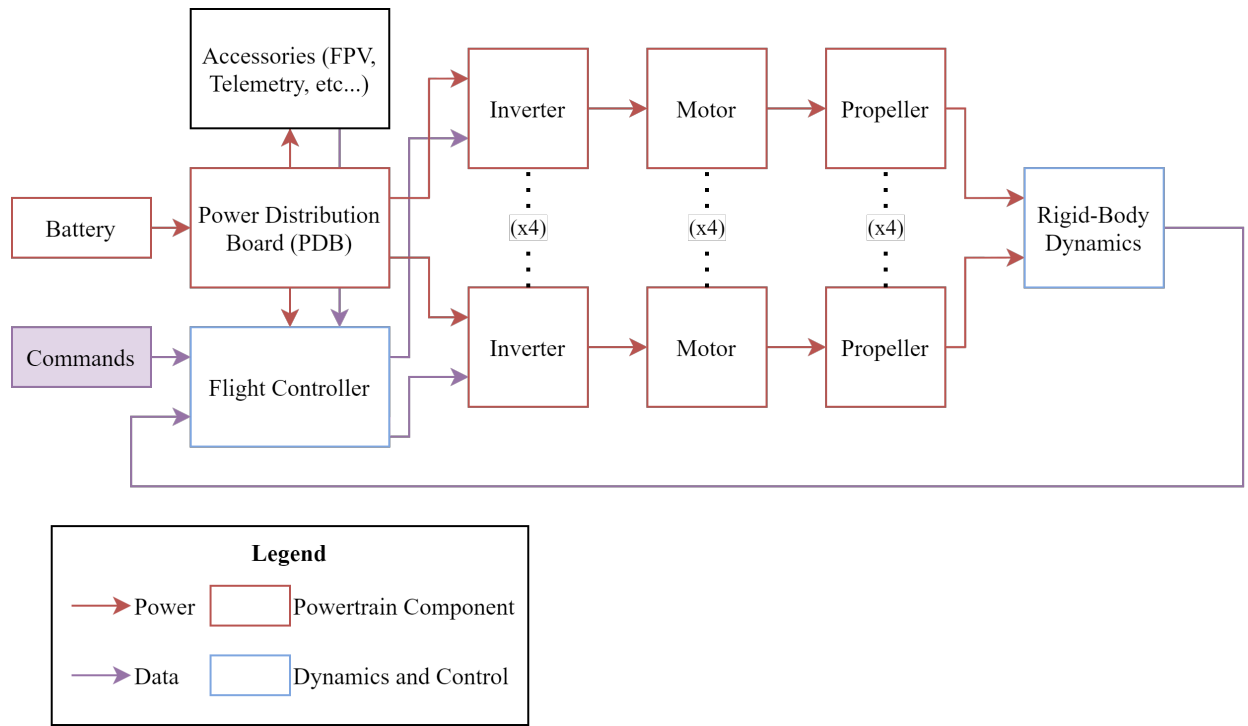


Figure 5.4: Quadrotor model architecture

### 5.3.1 Powertrain Model

#### Battery

The powertrain's battery is a lithium polymer (LiPo) battery pack with  $N_s$  series cells and  $N_p$  parallel cells. Each cell is modeled as a second-order electrical circuit, shown in Figure 5.5 [60]. In the figure,  $v_{OCV}(q)$  is the open-circuit cell voltage,  $i$  is the battery current,  $R_s$  is the cell series resistance, and  $R_{1,2}$ ,  $C_{1,2}$ ,  $v_{1,2}$  are the resistance, capacitance, and voltage across the first and second RC pair, respectively.

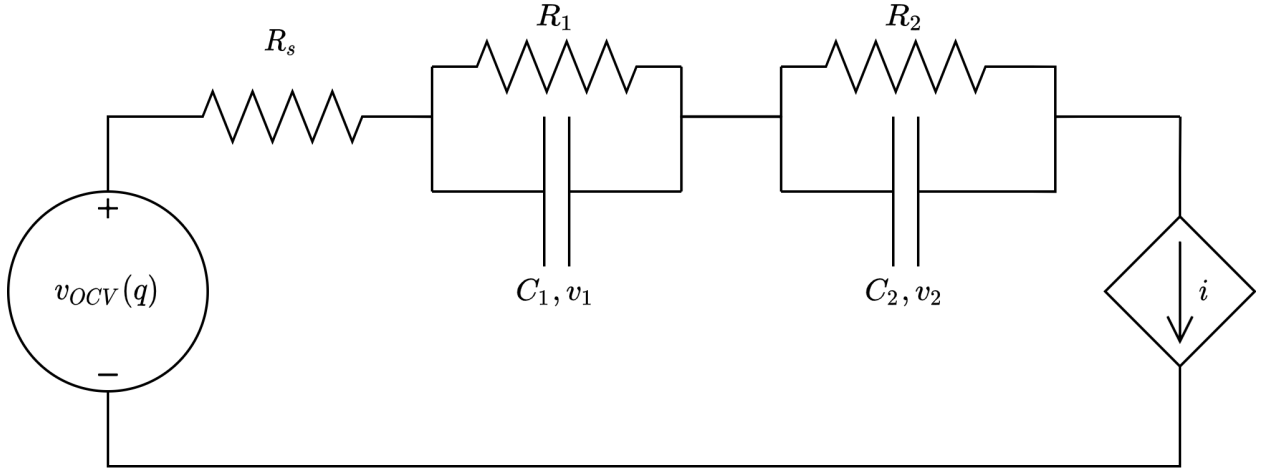


Figure 5.5: Battery second-order equivalent circuit model

Individual cell parameters can be converted to effective pack parameters using the familiar formulas for series and parallel connections of resistors and capacitors. The effective pack resistance is  $R_s^P = \frac{N_s}{N_p} R_s$ , effective RC pair capacitance is  $C_{1,2}^P = \frac{N_p}{N_s} C_{1,2}$ , and effective RC pair resistance is  $R_{1,2}^P = \frac{N_s}{N_p} R_{1,2}$ .

Applying Kirchoff's Voltage Law around the loop gives,

$$N_s v_{OCV}(q) = \frac{N_s}{N_p} R_s i + v_1 + v_2 \quad (5.1)$$

The voltage dynamics for the two RC pairs are given in 5.2

$$\frac{N_p}{N_s} C_{1,2} \dot{v}_{1,2} = i - \frac{N_p}{N_s} \frac{v_1}{R_1} \quad (5.2)$$

Finally, the state of charge dynamics are given in 5.3, where  $q$  is the battery state of charge (SOC) and  $Q$  is the battery cell capacity.

$$N_p Q \dot{q} = -i \quad (5.3)$$

Note that, in 5.1, the open-circuit voltage  $v_{OCV}$  is a function of  $q$ . This relationship was captured by fitting an eighth-order polynomial to experimental data from [61]. The relationship is shown in Figure 5.6.

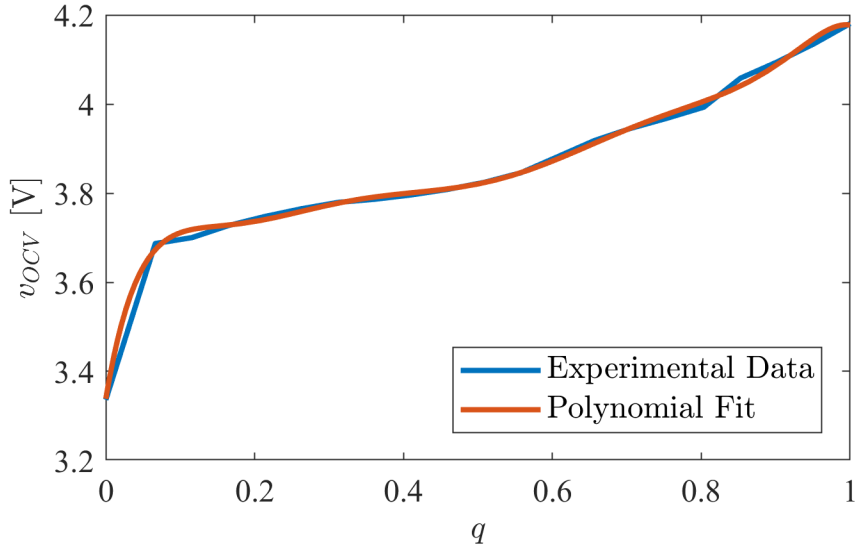


Figure 5.6: Relationship between battery cell SOC and open-circuit voltage

Batteries are frequently specified with a C-rating that is used to calculate the maximum allowable discharge current  $i^{\max}$ . This relationship is given in Equation 5.4.

$$i \leq i^{\max} = \frac{CN_pQ}{1000} \quad (5.4)$$

Above,  $C$  is the C-Rating in units of A/(Ah) and  $Q$  is specified in units of mAh. The battery current saturation is posed as a constraint in the optimization formulation.

Equations 5.1 - 5.3 lead to the graph model representation shown in Figure 5.7.

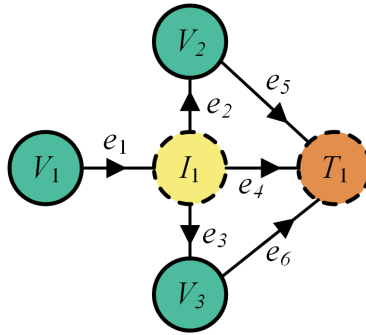


Figure 5.7: Battery graph model

The battery's RC dynamics were later omitted in the optimization study, as dynamic response data was not available for batteries in the database and the effects of the second-order dynamics were negligible.

## Electrical Bus

The model of the direct current (DC) electrical bus is similar to that given in [62]. It includes a single input for the battery and an output for each of the four ESCs. An electrical component diagram is shown in Figure 5.8.

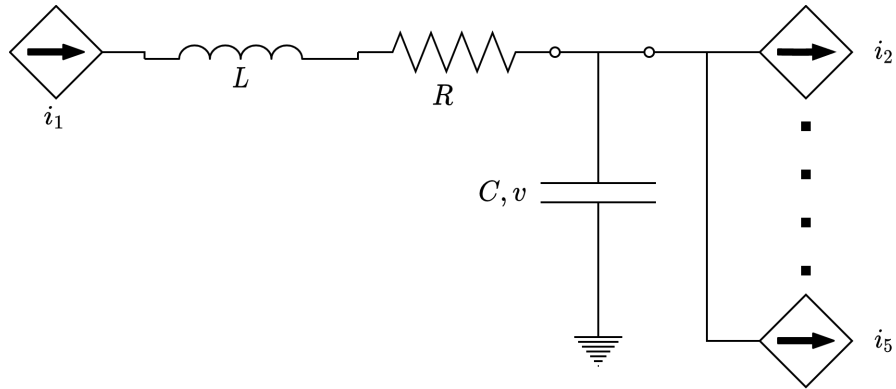


Figure 5.8: DC electrical bus component model

Applying Kirchoff's voltage law, the voltage across the inductor is given in 5.5.

$$L\dot{i}_1 = -Ri_1 - v \quad (5.5)$$

Summing the current into the capacitor gives 5.6.

$$C\dot{v} = i_1 - \sum_{k=2}^5 i_k \quad (5.6)$$

The graph representation of the electrical bus is presented in Figure 5.9.

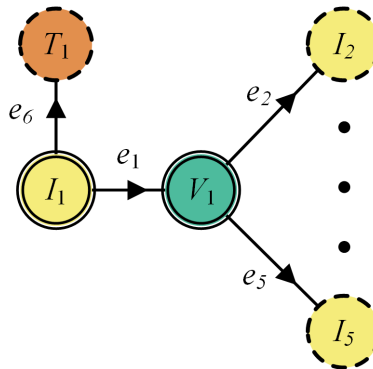


Figure 5.9: DC electrical bus graph model

## Inverter (ESC)

In this work, a simplified model of an inverter is utilized. It assumes the inverter losses are captured through an effective resistance, and that the maximum amplitude of the inverter's three-phase output voltage is equivalent to the DC input voltage. The power-invariant Park transformation, detailed in Appendix A, is used to represent the output voltage and current waveforms in a rotating frame. Under this transformation, A.10 leads to the relationship between peak  $q$ -axis voltage and DC bus voltage given in 5.7.

$$v_{DC} = \sqrt{\frac{2}{3}} \bar{v}_q \quad (5.7)$$

Above,  $v_{DC}$  is the DC input voltage and  $\bar{v}_q$  is the peak  $q$ -axis output voltage. Let  $u$  be the inverter's control input such that  $u = 1$  corresponds to maximum voltage output and  $u = 0$  corresponds to zero voltage output. The resulting relationship between input and output voltage is given in 5.8.

$$uv_{dc} = \sqrt{\frac{2}{3}} v_q + Ri_{DC} \quad (5.8)$$

In 5.8,  $v_q$  is the  $q$ -axis output voltage,  $R$  is an effective resistance used to capture inverter losses, and  $i_{DC}$  is the DC input current. Inverters are typically rated for a maximum current. That is,  $i_{DC} \leq i_{DC}^{\max}$ , where  $i_{DC}^{\max}$  is the maximum allowable current. This current saturation is posed as a constraint in the optimization formulation. The power balance across the inverter is given in 5.9

$$uv_{DC}i_{DC} = i_{DC}^2 R + v_q i_q \quad (5.9)$$

where  $i_q$  is the  $q$ -axis output current. The power contribution from the  $d$ -axis is omitted because the motor is assumed to be controlled using a field-oriented control (FOC) scheme that drives the  $d$ -axis current to zero [63].

The inverter model has the graph representation given in Figure 5.10.

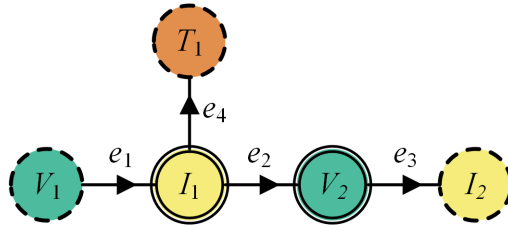


Figure 5.10: Inverter graph model

## Motor

The system's brushless DC motor is modeled as a permanent-magnet synchronous machine (PMSM) with the following assumptions:

- Stator currents and voltages are sinusoidal and balanced with the same angular velocity as the rotor speed [64].
- Stator windings are balanced and sinusoidally distributed [64].
- The number of rotor and stator poles are equal [64].
- Cogging torque between permanent magnet and stator teeth is neglected (smooth air gap model) [65].
- A linear magnetics model is used that neglects magnetic saturation [65].
- Core losses in magnetic materials of the machine are neglected [65].
- Frequency and temperature dependence of stator resistance is neglected [65].

Under these assumptions Chapman [66] and Kraus [64] give the  $qd0$  phase voltages in Equation 5.10, flux linkages in 5.11, and electrical torque in 5.12.

$$\begin{aligned} v_q &= r i_q + \omega_r \lambda_d + \frac{d}{dt} \lambda_q \\ v_d &= r i_d - \omega_r \lambda_q + \frac{d}{dt} \lambda_d \\ v_0 &= r i_0 + \frac{d}{dt} \lambda_0 \end{aligned} \quad (5.10)$$

$$\begin{aligned} \lambda_q &= L_q i_q \\ \lambda_d &= L_d i_d + \lambda_m \\ \lambda_0 &= L_0 i_0 \end{aligned} \quad (5.11)$$

$$\tau_e = \frac{3P}{2} \lambda_m i_q + (L_d - L_q) i_d i_q \quad (5.12)$$

In the above equations, subscripts  $q$ ,  $d$ , and  $0$  represent the  $q$ ,  $d$ , and  $0$ -axis of the variable transformed by the Park transformation given in A.1.  $L_d$ ,  $L_q$ , and  $L_0$  are constant inductances derived from motor geometry,  $\lambda_m$  is the peak strength of the flux linkage due to the magnets [66],  $P$  is the number of poles,  $\tau_e$  is the torque due to the magnetic field acting on the rotor, and  $\omega_r$  is the angular velocity of the rotating electrical frame. The model can be further simplified by making the following additional assumptions:

- The machine is surface-mounted so that stator inductances are independent of rotor position and  $L_q = L_d = L_0 = L$  [66].

- Motor phases are wye-connected such that  $i_0 = \lambda_0 = v_0 = 0$  [66].
- Field-oriented control (FOC) techniques are used so that  $i_d \approx 0$  [63].

With FOC, torque control is achieved by controlling  $i_q$  and driving  $i_d$  to zero. The simplified equations are given in 5.13-5.15. Note that, with FOC control, the model dynamics are similar to that of a DC motor.

$$\begin{aligned} v_q &= r i_q + \omega_r \lambda_d + \frac{d}{dt} \lambda_q \\ v_d &= -\omega_r \lambda_q \end{aligned} \quad (5.13)$$

$$\begin{aligned} \lambda_q &= L i_q \\ \lambda_d &= \lambda_m \end{aligned} \quad (5.14)$$

$$\tau_e = \frac{3}{2} \frac{P}{2} \lambda_m i_q \quad (5.15)$$

The relationship between electrical and mechanical angular velocity is given in 5.16.

$$\omega_r = \frac{P}{2} \omega_m \quad (5.16)$$

To derive a graph model representation from the above equations, a change of variables to the power-invariant form of the Park transformation A.7 is required. This transformation is given in A.11. Making this change of variables and replacing electrical angular velocity with mechanical angular velocity using 5.16 gives 5.17 and 5.18.

$$\begin{aligned} \tilde{v}_q &= r \tilde{i}_q + \frac{P}{2} \sqrt{\frac{3}{2}} \lambda_m \omega_m + \frac{d}{dt} (L \tilde{i}_q) \\ \tilde{v}_d &= -\frac{P}{2} \omega_m (L \tilde{i}_q) \end{aligned} \quad (5.17)$$

$$\tilde{\tau}_e = \frac{P}{2} \sqrt{\frac{3}{2}} \lambda_m \tilde{i}_q \quad (5.18)$$

Above, the  $\tilde{\phantom{x}}$  overscript denotes the variables under the power-invariant Park transformation.

The mechanical dynamics are given in 5.19.

$$J \dot{\omega}_m = \tau_e - \tau_l - B_v \omega_m - \text{sgn}(\omega_m) \tau_c \quad (5.19)$$

Where  $J$  is motor inertia,  $\tau_l$  is a load torque with an orientation opposing  $\tau_e$ ,  $B_v$  is a viscous friction factor,  $\tau_c$  is a constant opposing torque from Coulomb or “dry” friction, and  $\omega_m$  is the mechanical speed of the rotor. Note that the sign function  $\text{sgn}(\cdot)$  is discontinuous, which reduces the model’s compatibility with gradient-based optimization solvers [16]. To ensure a



smooth derivative, the sign function can be replaced with the sigmoid function given in 5.20 as suggested in [62].

$$\text{sig}(\omega) = \frac{2}{1 + \exp(-\alpha\omega)} - 1 \quad (5.20)$$

where  $\alpha$  is a smoothing constant. In the specific application of the quadrotor system, the motor's angular velocity has a constant sign. This allows us to omit the step or sigmoid function in the present formulation.

Suppliers often characterize motors with speed and torque constants. Equations 5.21 relate the motor's torque constant to its flux linkage and number of poles.

$$K_\tau = \frac{P}{2} \lambda_m \quad (5.21)$$

where  $K_\tau$  is the torque constant. Equation 5.22 relates the torque constant to the speed constant.

$$kV = \frac{30}{\pi} \frac{1}{K_\tau} \quad (5.22)$$

where  $kV$  is the speed constant in RPM/V and  $K_\tau$  is the torque constant in Nm/A.

The graph model of the motor is presented in Figure 5.11.

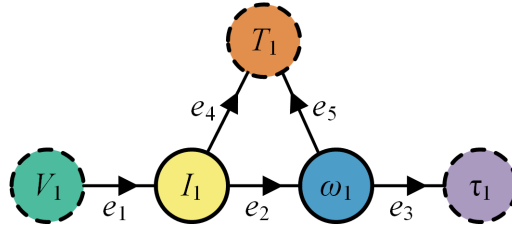


Figure 5.11: Motor graph model

## Propeller

The propeller model assumes a constant thrust and torque coefficient as defined in [67]. These coefficients are generally functions of the propeller design, Reynolds number, and advance ratio [68]. The thrust generated by the propeller is presented in 5.23, and the shaft torque generated by the propeller is given in 5.24.

$$T = k_T \rho D^4 \omega^2 \quad (5.23)$$

$$\tau = k_Q \rho D^5 \omega^2 \quad (5.24)$$

Above,  $T$  is propeller thrust,  $k_T$  is the thrust coefficient,  $\rho$  is air density,  $D$  is propeller diameter,  $\omega$  is the propeller's rotational speed,  $\tau$  is shaft torque, and  $k_Q$  is the torque

coefficient. Often, a power coefficient is specified instead of the torque coefficient. These are related by 5.25 [69].

$$k_Q = \frac{k_P}{2\pi} \quad (5.25)$$

Applying Newton's Second Law for rotation, we have

$$J\dot{\omega} = \tau_m - \tau \quad (5.26)$$

where  $J$  is rotor inertia and  $\tau_m$  is shaft input torque from the motor. Re-writing 5.26 in terms of power and substituting the torque value, we have

$$J\omega\dot{\omega} = \tau_m\omega - k_Q\rho D^5\omega^3 \quad (5.27)$$

Figure 5.12 shows the graph form of this equation.

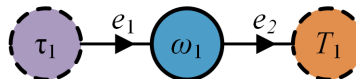


Figure 5.12: Propeller graph model

### Powertrain System Model

The component models defined above were combined into a powertrain system graph model using the AGILe toolbox. Figure 5.13 shows the complete system graph, and model details are provided in Appendix B.

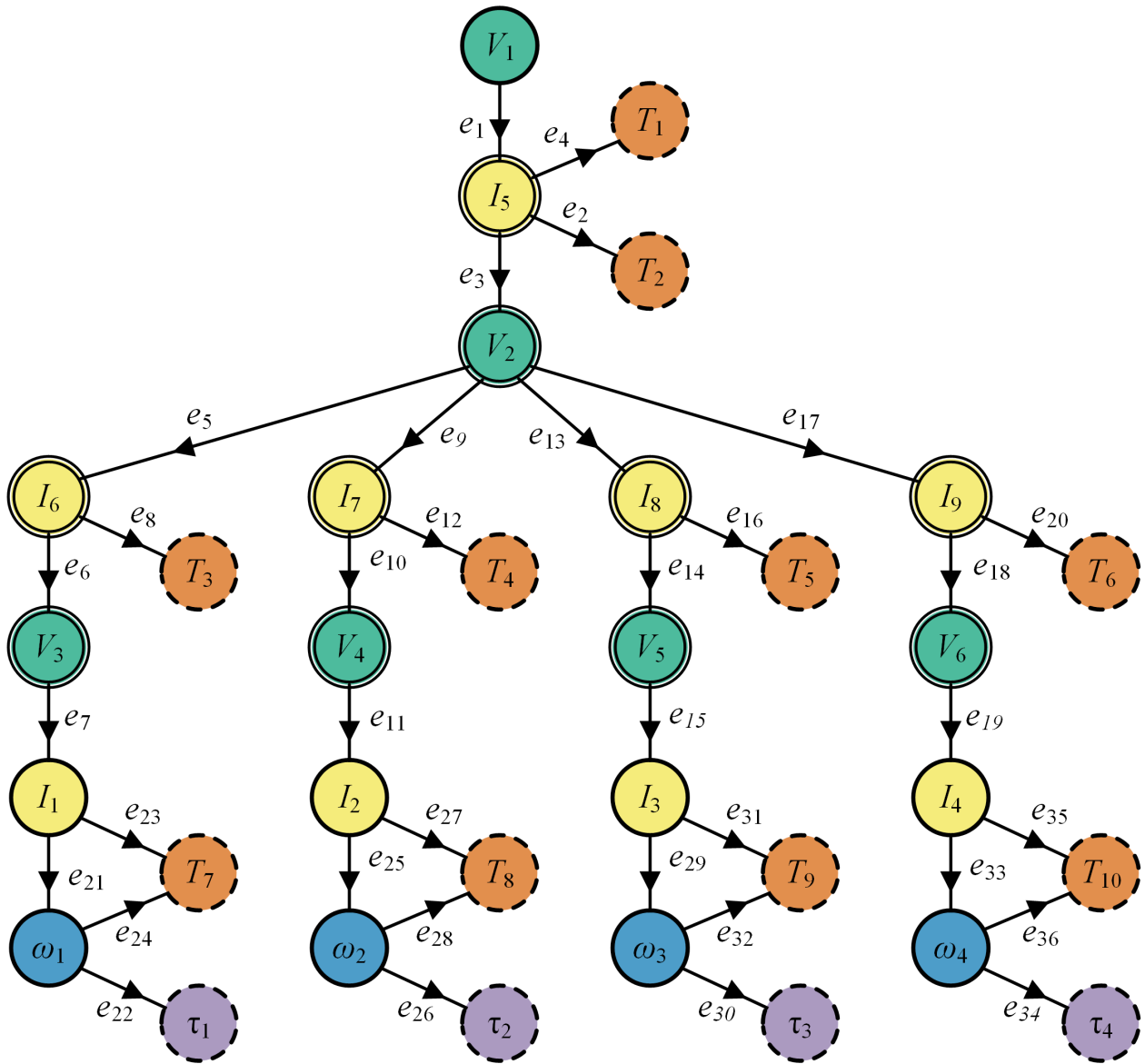


Figure 5.13: Powertrain system graph model

### Simplified Powertrain System Model

The powertrain system model can be simplified by assuming that the system is balanced, that is, corresponding states and inputs in each of the rotors are equivalent. Equation 5.28

states this explicitly.

$$\begin{aligned}
I_{2\dots4} &= I_1 \\
\omega_{2\dots4} &= \omega_1 \\
V_{4\dots6} &= V_3 \\
I_{7\dots9} &= I_6 \\
u_{2\dots4} &= u_1
\end{aligned} \tag{5.28}$$

This assumption is useful for evaluating purely vertical dynamics or when calculating the steady-state hover condition. Applying these assumptions yields the dynamic state equations 5.29 and the algebraic state equations 5.30.

$$\dot{\mathbf{x}} = \begin{bmatrix} \left(-1 / \left(\phi_{N_p}^B \phi_Q^B\right)\right) I_5 \\ \left(1 / \left(\phi_L^M\right)\right) \left(V_3 - \phi_R^M I_1 - \phi_{K_t}^M \sqrt{3 / 2} \omega_1\right) \\ \left(1 / \left(\phi_J^M + \phi_J^P\right)\right) \left(\phi_{K_t}^M \sqrt{3 / 2} I_1 - \tau_1\right) \end{bmatrix} \tag{5.29}$$

$$\mathbf{0} = \begin{bmatrix} \phi_{N_s}^B \phi_{v_{OCV}}^B \left(V_1\right) - \left(\phi_{N_s}^B / \phi_{N_p}^B\right) \phi_{R_s}^B I_5 - V_2 \\ I_5 - u_1 4 I_6 \\ u_1 V_2 - \sqrt{2 / 3} V_3 - \phi_R^I I_6 \\ \sqrt{2 / 3} I_6 - I_1 \\ \tau_1 - \phi_{k_d}^P \phi_\rho^P \left(\phi_D^P\right)^5 \omega_1^2 \end{bmatrix} \tag{5.30}$$

Above, the notation  $\phi_x^t$  is used to denote parameter  $x$  corresponding to component type  $t$ , where  $B$  is the battery,  $I$  is the inverter,  $M$  is the motor, and  $P$  is the propeller.

### 5.3.2 Quadrotor Body Dynamic Model

The following derivation of the quadrotor's rigid-body dynamic model follows that of [70]. It includes the development of the coordinate systems and attitude representation used to describe the vehicle's pose, a rigid-body kinematic model to relate angular velocity in the body frame to attitude rates, and a dynamic model to determine the vehicle's acceleration.

#### Coordinate System and Attitude Representation

Two coordinate frames are employed to describe the state of the quadrotor:

- Earth Coordinate Frame ( $o_e x_e y_e z_e$ ): The initial position of the quadrotor defines the origin  $o_e$  of this coordinate frame. The  $o_e x_e$  axis points in a given fixed direction in the horizontal plane, the  $o_e z_e$  axis points downward toward the earth's center, and the  $o_e y_e$

axis is determined according to the right hand rule.

- **Body Coordinate Frame ( $o_b x_b y_b z_b$ ):** This coordinate frame is attached to the quadrotor, and the quadrotor's center of gravity is chosen as its origin  $o_b$ . The  $o_b x_b$  axis points in the nose direction as indicated in Figure 5.15. The  $o_b z_b$  axis points downward perpendicular to the  $o_b x_b$  axis. The  $o_b y_b$  axis is determined from the right hand rule.

The relationship between the two coordinate systems is depicted in Figure 5.14. A left superscript is used to indicate the frame to which a vector is relative, i.e.  ${}^e \mathbf{x}$  is a vector relative to the earth frame and  ${}^b \mathbf{x}$  is a vector relative to the body frame. We must also define a standard basis with unit vectors  $\mathbf{e}_1 := [1, 0, 0]^\top$ ,  $\mathbf{e}_2 := [0, 1, 0]^\top$ , and  $\mathbf{e}_3 := [0, 0, 1]^\top$ . In the earth frame, the axes  $o_e x_e$ ,  $o_e y_e$ , and  $o_e z_e$  are expressed with  $\mathbf{e}_1$ ,  $\mathbf{e}_2$ , and  $\mathbf{e}_3$  respectively. Body unit vectors  $\mathbf{b}_1 := o_b x_b$ ,  $\mathbf{b}_2 := o_b y_b$ , and  $\mathbf{b}_3 := o_b z_b$  can be expressed relative to the body frame as  ${}^b \mathbf{b}_i = \mathbf{e}_i$  and relative to the earth frame as  ${}^e \mathbf{b}_i$  for all  $i \in \{1, 2, 3\}$ .

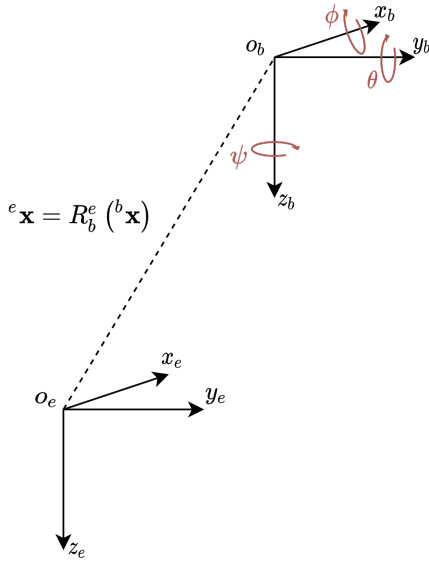


Figure 5.14: Coordinate system representation

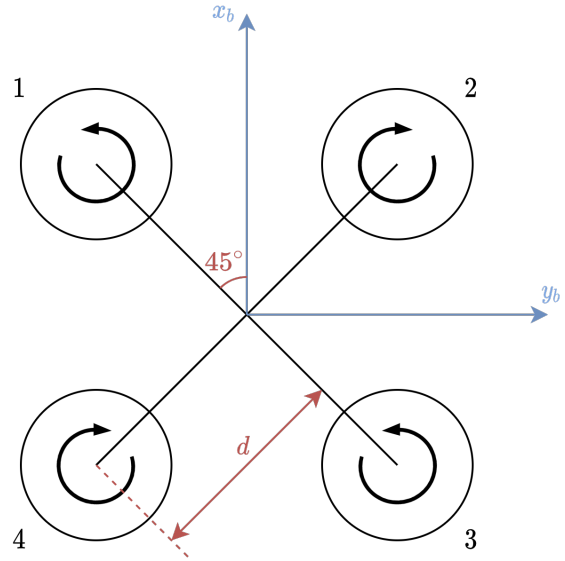


Figure 5.15: Body coordinate system, top-down view

Euler angles  $\psi$  (yaw angle),  $\theta$  (pitch angle), and  $\phi$  (roll angle) express the aircraft's attitude. The body orientation is achieved by three successive rotations about the  $z$ ,  $y$ , and  $x$  axes around a fixed point. Let the frame resulting from the first elemental rotation be  $k$  with unit vectors  $\mathbf{k}_i$  and that of the second elemental rotation be  $n$  with unit vectors  $\mathbf{n}_i$ . Frame  $k$  is achieved by a yaw rotation about the  $\mathbf{e}_3$  axis by  $\psi$ . Frame  $n$  is achieved by a pitch rotation about the  $\mathbf{k}_2$  axis by  $\theta$ . The final body frame is achieved by a roll rotation about the  $\mathbf{n}_1$  axis by  $\phi$ . The combined rotation can be expressed as the product of three rotation

matrices. Denote a rotation matrix that performs a change of basis from frame  $a$  to frame  $b$  (rotation of frame  $a$  relative to fixed frame  $b$ ) as  $\mathbf{R}_a^b$ . The elemental rotation matrices are,

$$\begin{aligned}\mathbf{R}_z(\psi) &= \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{R}_y(\theta) &= \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \\ \mathbf{R}_x(\phi) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix}\end{aligned}\tag{5.31}$$

The combined rotation matrix that transforms a vector  $\mathbf{x}$  expressed in the earth frame  ${}^e\mathbf{x}$  into a vector expressed in the body frame  ${}^b\mathbf{x}$  is then,

$${}^b\mathbf{x} = \mathbf{R}_n^b \mathbf{R}_k^n \mathbf{R}_e^k ({}^e\mathbf{x}) = \mathbf{R}_e^b ({}^e\mathbf{x})\tag{5.32}$$

Where  $\mathbf{R}_n^b = \mathbf{R}_x(\phi)$ ,  $\mathbf{R}_k^n = \mathbf{R}_y(\theta)$ , and  $\mathbf{R}_e^k = \mathbf{R}_z(\psi)$ . Performing the matrix multiplication gives,

$$\mathbf{R}_e^b = \begin{pmatrix} c(\theta)c(\psi) & c(\theta)s(\psi) & -s(\theta) \\ s(\theta)c(\psi)s(\phi) - s(\psi)c(\phi) & s(\theta)s(\psi)s(\phi) + c(\psi)c(\phi) & c(\theta)s(\phi) \\ s(\theta)c(\psi)c(\phi) + s(\psi)s(\phi) & s(\theta)s(\psi)c(\phi) - c(\psi)s(\phi) & c(\theta)c(\phi) \end{pmatrix}\tag{5.33}$$

Where  $c(\cdot)$  is an abbreviation of  $\cos(\cdot)$  and  $s(\cdot)$  is an abbreviation of  $\sin(\cdot)$ .

## Rigid-body Kinematic Model

In order to analyze the kinetics of the aircraft, we must first understand the relationship between the angular velocity of the body  ${}^b\boldsymbol{\omega} = [p, q, r]^\top$  and the attitude rate  $\dot{\boldsymbol{\Theta}} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^\top$ . To do so, we employ the theorem involving the time derivative of rotation matrices from Zhao in [71] given in Equation 5.34,

$$\frac{d}{dt}\mathbf{R}_b^e = \mathbf{R}_b^e [{}^b\boldsymbol{\omega}]_\times\tag{5.34}$$

where  $[\cdot]_{\times}$  is the skew symmetric operator used to convert a cross product of two vectors into a matrix-vector product. Rearranging 5.34 gives,

$$[{}^b\omega]_{\times} = \mathbf{R}_b^{e\top} \frac{d}{dt} \mathbf{R}_b^e = \mathbf{R}_e^b \frac{d}{dt} \mathbf{R}_b^e \quad (5.35)$$

Employing the chain rule to expand the derivative yields,

$$[{}^b\omega]_{\times} = \mathbf{R}_b^{e\top} \frac{d}{dt} \mathbf{R}_b^e = \mathbf{R}_e^b \left( \frac{d\mathbf{R}_b^e}{d\phi} \dot{\phi} + \frac{d\mathbf{R}_b^e}{d\theta} \dot{\theta} + \frac{d\mathbf{R}_b^e}{d\psi} \dot{\psi} \right) \quad (5.36)$$

Simplifying the expression achieves a linear relationship between  ${}^b\omega$  and  $\dot{\Theta}$ .

$${}^b\omega = \begin{pmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta) \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta) \cos(\phi) \end{pmatrix} \dot{\Theta} =: \mathbf{W}^{-1} \dot{\Theta} \quad (5.37)$$

Taking the inverse provides an explicit expression for the attitude rates.

$$\dot{\Theta} = \mathbf{W} \cdot {}^b\omega = \begin{pmatrix} 1 & \tan(\theta) \sin(\phi) & \tan(\theta) \cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin \phi / \cos \theta & \cos(\phi) / \cos(\theta) \end{pmatrix} {}^b\omega \quad (5.38)$$

Kinematic equations for the quadrotor body can finally be written. Let  $\mathbf{p}$  represent the position of the quadrotor's center of gravity (i.e.  $o_b$ ) and  $\mathbf{v}$  represent the linear velocity.

$$\begin{aligned} {}^e\dot{\mathbf{p}} &= {}^e\mathbf{v} \\ \dot{\Theta} &= \mathbf{W} \cdot {}^b\omega \end{aligned} \quad (5.39)$$

## Rigid-body Dynamic Model

*Position Dynamic Model:* With level propellers producing thrust parallel to  $\mathbf{b}_3$ , the position dynamics are described by Newton's second law,

$${}^e\dot{\mathbf{v}} = g\mathbf{e}_3 - \frac{T}{m} {}^e\mathbf{b}_3 \quad (5.40)$$

where  $g$  is the gravitational acceleration,  $T$  is the combined thrust produced by the four propellers, and  $m$  is the mass of the aircraft. The rotation matrix  $\mathbf{R}_b^e$  is used to write Equation 5.40 in the body frame. This yields,

$${}^e\dot{\mathbf{v}} = g\mathbf{e}_3 + \frac{1}{m} \mathbf{R}_b^e \cdot {}^b\mathbf{T}_3 \quad (5.41)$$

where  ${}^b\mathbf{T}_3 := -T\mathbf{e}_3$ .

*Attitude Dynamic Model:* The attitude dynamic model begins with Euler's equation describing the rotation of a rigid body [72],

$$\mathbf{J} \cdot {}^b\dot{\boldsymbol{\omega}} + {}^b\boldsymbol{\omega} \times \mathbf{J} \cdot {}^b\boldsymbol{\omega} = \mathbf{M} \quad (5.42)$$

where  $\mathbf{J}$  is the inertia tensor relative to the body frame,  ${}^b\boldsymbol{\omega}$  is the angular velocity defined above, and  $\mathbf{M}$  is the sum of the moments acting on the craft in the body frame. For a simple quadrotor model,  $\mathbf{M}$  comes from two primary sources: moments generated by the propellers  $\boldsymbol{\tau} = [\tau_x, \tau_y, \tau_z]^\top$  and the gyroscopic torques associated with the rotors  $\mathbf{G}_a = [G_{a,x}, G_{a,y}, G_{a,z}]$ .

$$\mathbf{M} = \boldsymbol{\tau} + \mathbf{G}_a \quad (5.43)$$

The gyroscopic torque of the  $k_{th}$  rotor is found by looking at the torque acting on the rotor in the rotating body frame,

$$\mathbf{G}_{a,k} = - \left( \frac{d}{dt} ({}^b\mathbf{L}_k) + {}^b\boldsymbol{\omega} \times {}^b\mathbf{L}_k \right) \quad (5.44)$$

where  ${}^b\mathbf{L}_k$  is the angular momentum contributed by rotor  $k$  relative to the body frame. The quantity in parenthesis in the right-hand side of Equation 5.44 represents the external moments in the body frame acting on rotor  $k$  given the body's angular velocity  ${}^b\boldsymbol{\omega}$ .  $\mathbf{G}_{a,k}$  is therefore the torque exerted on the body by rotor  $k$ .  ${}^b\mathbf{L}_k$  is the rotor's inertia about its central axis  $J_r$  times its angular velocity vector in the body frame  ${}^b\boldsymbol{\omega}_k$ ,

$${}^b\mathbf{L}_k = J_r \cdot {}^b\boldsymbol{\omega}_k = J_r (-1^k) |\omega_k| \mathbf{e}_3 \quad (5.45)$$

where  $|\omega_k|$  is the unsigned angular speed of rotor  $k$ . Note that  $\frac{d}{dt} ({}^b\mathbf{L}_k)$  is zero if we assume  $|\omega_k|$  is constant. This allows us to write,

$$\mathbf{G}_{a,k} = - ({}^b\boldsymbol{\omega} \times {}^b\mathbf{L}_k) = {}^b\mathbf{L}_k \times {}^b\boldsymbol{\omega} \quad (5.46)$$

Substituting 5.45 into 5.46 gives,

$$\mathbf{G}_{a,k} = J_r (-1^k) |\omega_k| \mathbf{e}_3 \times {}^b\boldsymbol{\omega} \quad (5.47)$$



The total gyroscopic torque is then,

$$\mathbf{G}_a = \sum_{k=1}^{N_r} \mathbf{G}_{a,k} = J_r (\mathbf{e}_3 \times {}^b\boldsymbol{\omega}) \sum_{k=1}^{N_r} (-1^k) |\omega_k| = J_r \left( \sum_{k=1}^{N_r} (-1^k) |\omega_k| \right) [\mathbf{e}_3]_{\times} {}^b\boldsymbol{\omega} \quad (5.48)$$

## Parametric Inertia Tensor

A dynamic model for design optimization must be parametrized by the design variables. Because the inertial properties of the quadrotor will change depending on the motor, propeller, and battery design, the inertial tensor must be expressed as a function of these elements. Most examples of quadrotor dynamics and simulation in the literature determine the inertia tensor by 1) conducting physical experiments such as a bifilar pendulum [70] or 2) building a detailed CAD model of the system. Though accurate, neither of these approaches can account for changes in the physical design and therefore are not compatible with design optimization.

We can, however, use a combined numerical and analytical approach. First, the inertial tensor  $\mathbf{J}_f$  of the ‘fixed’ system (i.e. excluding the battery, motors, and propellers) is calculated via CAD or physical experiments. Then, the inertia matrix for each of the optimization components  $\mathbf{J}'_{c,i}$  is calculated analytically about the component’s center of mass. Then, the generalized parallel axis theorem is used to calculate each components’ inertia about the system’s center of mass  $\mathbf{J}_{c,i}$ . The formula for the generalized parallel axis theorem is given in [73] as Equation 5.49.

$$\mathbf{J}_{c,i} = \mathbf{J}'_{c,i} + m_{c,i} (\|\mathbf{r}_{c,i}\|^2 I_3 - \mathbf{r}_{c,i} \otimes \mathbf{r}_{c,i}) \quad (5.49)$$

Above,  $m_{c,i}$  is the component mass and  $\mathbf{r}_{c,i}$  is the displacement vector from the system center of mass to the component center of mass. Finally, the system inertia tensor  $\mathbf{J}$  is found by summing each individual inertia tensor:

$$\mathbf{J} = \mathbf{J}_f + \sum_{c,i} \mathbf{J}_{c,i} \quad (5.50)$$

Equations 5.49 and 5.50 are simplified by assuming the motor and propeller are point masses at the end of each arm and the battery is a point mass at the system’s center of gravity. Though these assumptions make for a very rough approximation, they capture the general trend of how design changes impact the real system’s inertial properties. Applying Equation 5.49 to the rotors  $r$  gives

$$\mathbf{J}_{r,k} = (m_P + m_M) (\|\mathbf{r}_{r,k}\|^2 I_3 - \mathbf{r}_{r,k} \otimes \mathbf{r}_{r,k}) \quad (5.51)$$

where  $m_P$  is the mass of the propeller and  $m_M$  is the mass of the motor. The displacement

from the system's center of mass to the rotors' center of mass  $\mathbf{r}_{r,k}$  is approximately

$$\begin{aligned}
\mathbf{r}_{r,1} &= d \frac{\sqrt{2}}{2} \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^\top \\
\mathbf{r}_{r,2} &= d \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top \\
\mathbf{r}_{r,3} &= d \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}^\top \\
\mathbf{r}_{r,4} &= d \frac{\sqrt{2}}{2} \begin{bmatrix} -1 & -1 & 0 \end{bmatrix}^\top
\end{aligned} \tag{5.52}$$

where  $d$  is the distance from the system's center of mass to that of the rotor as shown in Figure 5.15. Applying Equation 5.51 and summing over all four rotors gives the moments of inertia contributed by the rotors.

$$\mathbf{J}_r = \sum_{k=1}^4 \mathbf{J}_{r,k} = (m_P + m_M) d^2 \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \tag{5.53}$$

By assuming the battery is a point mass at the system's center of mass, the battery's contribution to inertia is zero. Therefore, we can write a simplified version of the system's inertia tensor in Equation 5.50 as,

$$\mathbf{J} = \mathbf{J}_f + \mathbf{J}_r \tag{5.54}$$

### 5.3.3 S500 CAD Model and Inertia Estimate

A CAD model was constructed to estimate the inertial tensor of the 'fixed' system  $\mathbf{J}_f$ , the inertia contributed by the frame and other static components. The model, shown in Figure 5.16, was constructed using Velimir's 'S500 Frame' model on GrabCad [74], HolyBro's reference designs [75], and Harvick Tang's propeller model [76]. To improve the accuracy of the inertia estimate, material properties were assigned to each of the components and component masses were manually adjusted to reflect measured values.



Figure 5.16: HolyBro S500 CAD Model

Using the CAD model, the inertia of the frame  $\mathbf{J}_f$  was calculated to be:

$$\mathbf{J}_f = \begin{bmatrix} 4.70e-3 & 6.00e-8 & 3.20e-5 \\ 6.00e-8 & 4.50e-3 & -5.30e-7 \\ 3.20e-5 & -5.30e-7 & 4.70e-3 \end{bmatrix} \text{ kg m}^2 \quad (5.55)$$

Evaluating Equations 5.54 and 5.53 gives,

$$\mathbf{J} = \begin{bmatrix} 1.30e-2 & 6.00e-8 & 3.20e-5 \\ 6.00e-8 & 1.27e-2 & -5.30e-7 \\ 3.20e-5 & -5.30e-7 & 2.12e-2 \end{bmatrix} \text{ kg m}^2 \quad (5.56)$$

The more accurate system inertia matrix calculated with the CAD Model is:

$$\mathbf{J}_{\text{CAD,Battery}} = \begin{bmatrix} 1.62e-2 & 7.00e-8 & 3.62e-5 \\ 7.00e-8 & 1.66e-2 & -3.30e-7 \\ 3.62e-5 & -3.30e-7 & 2.25e-2 \end{bmatrix} \text{ kg m}^2 \quad (5.57)$$

including the battery and

$$\mathbf{J}_{\text{CAD,No Battery}} = \begin{bmatrix} 1.47e-2 & 6.00e-8 & 4.41e-5 \\ 6.00e-8 & 1.45e-2 & 4.00e-8 \\ 4.41e-5 & 4.00e-8 & 2.16e-2 \end{bmatrix} \text{ kg m}^2 \quad (5.58)$$

excluding the battery. The parametric inertia estimate (Equation 5.56) underestimates the

moments of inertia, but is sufficiently accurate for design optimization purposes.

### Combined Dynamic Model

Equations 5.39, 5.41, and 5.42 can be combined to give a nonlinear model for the system's dynamics:

$$\begin{bmatrix} {}^e\dot{\mathbf{p}} \\ {}^e\dot{\mathbf{v}} \\ \dot{\Theta} \\ {}^b\dot{\omega} \end{bmatrix} = \begin{bmatrix} {}^e\mathbf{v} \\ g\mathbf{e}_3 + \frac{1}{m}\mathbf{R}_b^e \cdot {}^b\mathbf{T}_3 \\ \mathbf{W} \cdot {}^b\omega \\ \mathbf{J}^{-1}(-{}^b\omega \times \mathbf{J} \cdot {}^b\omega + \tau + \mathbf{G}_a) \end{bmatrix} \quad (5.59)$$

Recall that the inputs to the dynamic model are  $T$  and  $\tau$ .

### Control Effectiveness Model (Thrusts/Torques)

It remains to calculate  $T$  and  $\tau$  as a function of the propeller speeds. Let  $C_T = k_T \rho D^4$  be the lumped thrust coefficient and  $C_Q = k_Q \rho D^5$  be the lumped drag coefficient such that, for each propeller  $k$ , the thrust  $T_k$  and torque  $\tau_k$  produced by each propeller is

$$\begin{aligned} T_k &= C_T \omega_k^2 \\ \tau_k &= C_Q \omega_k^2 \end{aligned} \quad (5.60)$$

The combined thrust  $T$  is simply the sum of the thrusts generated by each propeller

$$T = C_T \sum_{k=1}^{N_r} \omega_k^2 \quad (5.61)$$

Similarly, the combined moment  $\tau_z$ , taken about the  ${}^b\mathbf{b}_3$  axis, is the sum of the torques generated by each propeller.

$$\tau_z = C_Q \sum_{k=1}^{N_r} (-1)^{k-1} \omega_k^2 \quad (5.62)$$

Where the alternating sign comes from the rotors' alternating directions. To calculate  $\tau_x$  and  $\tau_y$ , note from Figure 5.15 that the moment arm of each propeller about the  ${}^b\mathbf{b}_1$  and  ${}^b\mathbf{b}_2$  axis is,

$$l = d \sin 45^\circ = \frac{\sqrt{2}}{2} d \quad (5.63)$$

Therefore,

$$\begin{aligned}\tau_x &= lC_T (\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \\ \tau_y &= lC_T (\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2)\end{aligned}\tag{5.64}$$

Equations 5.61 - 5.64 can be written in matrix form as,

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ lC_T & -lC_T & -lC_T & lC_T \\ lC_T & lC_T & -lC_T & -lC_T \\ C_Q & -C_Q & C_Q & -C_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}\tag{5.65}$$

### Simplified Body Dynamic Model

When the system is assumed to be balanced (see Equation 5.28), then only the body's vertical dynamics need to be considered. This is achieved with the force balance given in Equation 5.66.

$$m\dot{v} = T - mg\tag{5.66}$$

where  $v$  is vertical velocity,  $T$  is the combined thrust of the four rotors,  $m$  is system mass, and  $g$  is gravitational acceleration.

### 5.3.4 The Planar Quadrotor

Evaluating the full 3D dynamic model of the quadrotor system is quite computationally intensive. To feasibly analyze dynamic optimization objectives, the full 3D quadrotor model was replaced with a 2D “planar quadrotor” model. This system captures the nonlinearities, differential flatness, and underactuated properties of the full system without the complications of 3D rigid-body dynamics. A diagram of the planar quadrotor system is given in Figure 5.17.

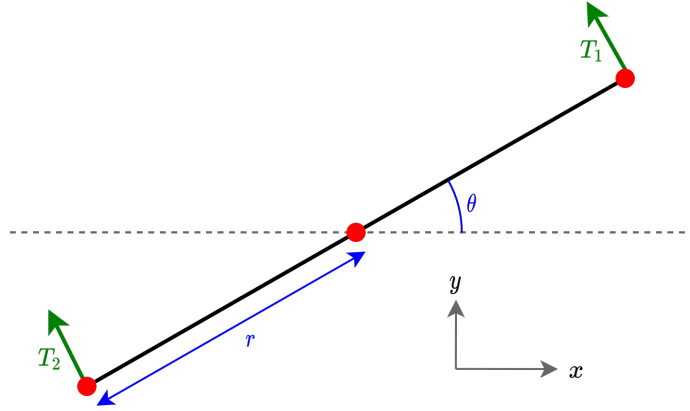


Figure 5.17: Planar quadrotor system

The equations of motion are given in 5.67 [77].

$$\begin{aligned}
 m\ddot{x} &= -(T_1 + T_2) \sin(\theta) \\
 m\ddot{y} &= (T_1 + T_2) \cos(\theta) - mg \\
 I\ddot{\theta} &= r(T_1 - T_2)
 \end{aligned}
 \tag{5.67}$$

Above,  $m$  is the system mass,  $I$  is the moment of inertia about the center of mass, and  $r$  is the distance from the center of mass to the base of the rotor. Inputs  $T_1$  and  $T_2$  are the thrusts generated from the right and left rotors, respectively. The moment of inertia, as before, is calculated by combining the inertia contribution from the frame with the inertia contribution from the rotor.

$$I = I_f + I_r \tag{5.68}$$

where  $I_f$  is the frame's inertia and  $I_r$  is the inertia of the rotors, both taken about the system's center of mass. The frame's contribution is found by assuming the frame is a rod of mass  $m_f$  with length  $2r$ .

$$I_f = \frac{1}{12} m_f (2r)^2 \tag{5.69}$$

The rotor's contribution is calculated with the parallel axis theorem

$$I_r = 2(m_P + m_M) r^2 \tag{5.70}$$

where  $m_P$  and  $m_M$  are propeller and motor mass, respectively. The powertrain dynamics for the planar quadrotor system are identical to that of the full quadrotor system, though with two rotors instead of four. Also, for dynamic objectives, the battery open-circuit voltage was assumed constant since the change in state of charge is negligible over a dynamic mission.

## 5.4 Model Tuning and Validation

### 5.4.1 Experimental Setup

The quadrotor system model, simplified using the ‘balanced’ assumptions in 5.28, was experimentally validated against a test platform. During the tests, the craft was launched and made to hover until the battery reached approximately 50% SOC. The flight controller recorded various signals including battery voltage, bus current, and inverter input. For accurate measurement of the vehicle’s bus voltage and current, the supplied power meter was replaced with a Mauch HS-200-LV [78]. Later, the recorded inverter input signal was fed into the dynamic model, and the resulting simulation data was compared to the data captured by the flight controller. Three configurations were selected to test the model at different points in the design space.

### 5.4.2 Configuration 1

The system was first configured with the 880kV motors and 10x4.5in propellers included with the S500 platform, as well as a Turnigy Graphene 4S 4000mAh battery purchased separately. After conducting the hover experiment, the final SOC was measured to be 54%. After feeding the recorded input signal into the simulation model, the simulated final SOC was 66.4%. Two primary source attribute to this error: 1) external disturbances from wind and noisy sensor readings and 2) model error. To determine the possible contribution of disturbances to the error, a disturbance model was created and integrated into the simulation model. Disturbances were found to have a negligible effect on final state-of-charge, as shown in Figure 5.18. Note that, in the figures, the measured SOC is calculated as a function of bus voltage. Because the bus voltage drops when the system is active, SOC is accurately measured when the system is at rest.

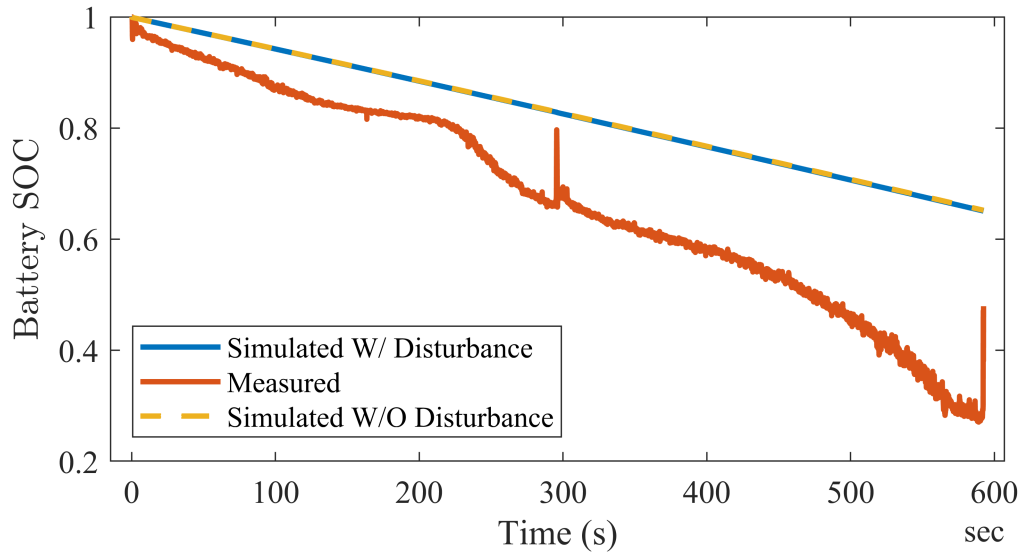


Figure 5.18: Measured vs simulated battery SOC, with and without disturbance model

Because disturbances were not a significant contributor to the discrepancy between the experiment and the simulation, the model parameters were tuned to align with the recorded data. The primary source of model error was underestimated powertrain losses, though the motor speed constant also required a small adjustment. After the necessary tuning, the experimental and simulation were in agreement. Figure 5.19 shows the measured and simulated battery SOC after the tuning process. The final SOC value of the tuned model agrees with the measured value.

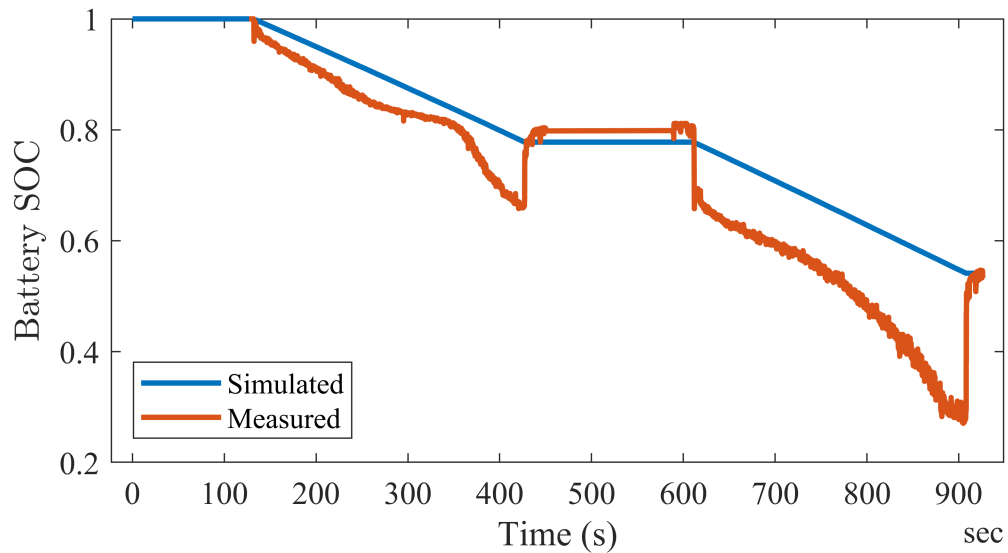


Figure 5.19: Measured vs. simulated battery SOC after model tuning process



Figure 5.20a compares measured and simulated bus voltage, and Figure 5.20b compares measured and simulated bus current.

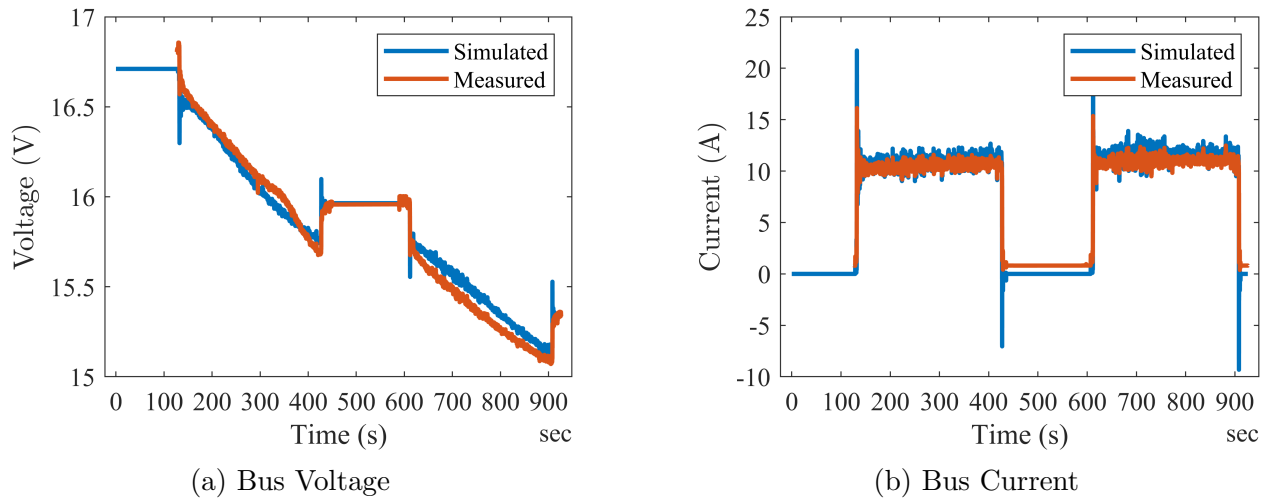


Figure 5.20: Measured vs. simulated signals, Configuration 1

### 5.4.3 Configuration 2

The second configuration consisted of KDE2814XF-515 515kV motors and KDEXF-UAS20LV 20A speed controllers, both sourced from KDE Direct, the LP12038SF 12x3.8 propeller from APC, and the 4s 4000mAh LiPo battery pack used in the previous configuration. Tests of Configuration 2 revealed a discrepancy between predicted and actual thrust and torque coefficients, likely attributable to the dependence of these values on rotor speed and aerodynamic interference between propellers [79]. Figure 5.21a compares measured and simulated bus voltage, and Figure 5.21b compares measured and simulated bus current, after tuning the propeller aerodynamic coefficients in the model. First, the power coefficient was tuned so that the current vs. throttle relationship aligned. Then, the thrust coefficient was tuned so that the measured and simulated throttle values aligned.

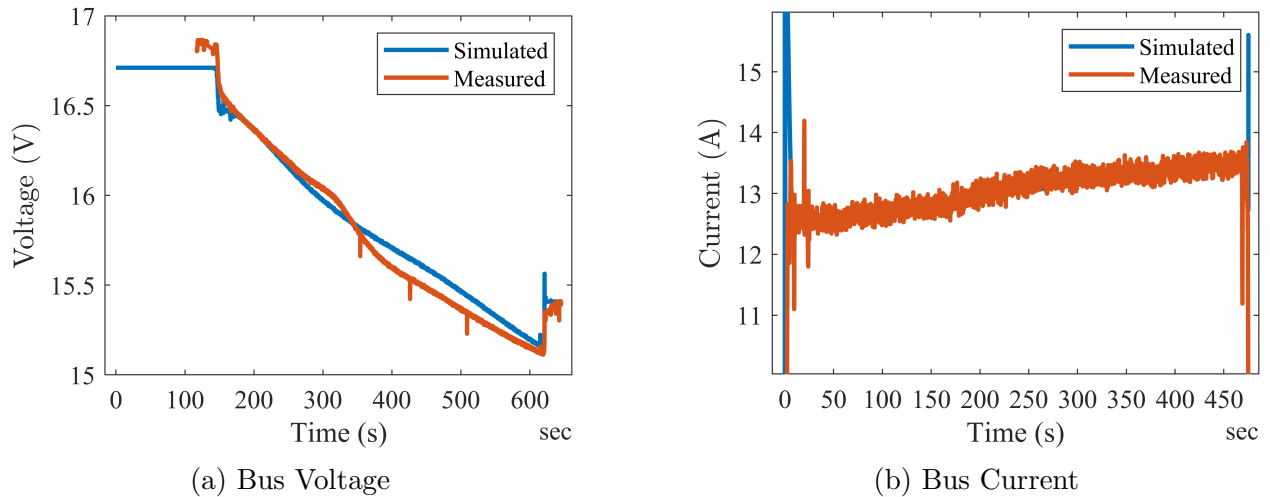


Figure 5.21: Measured vs. simulated signals, Configuration 2

### 5.4.4 Configuration 3

The third configuration substituted a 6s 5000mAh Turnigy Graphene battery into Configuration 2. The tuned results from this test are given in Figure 5.22.

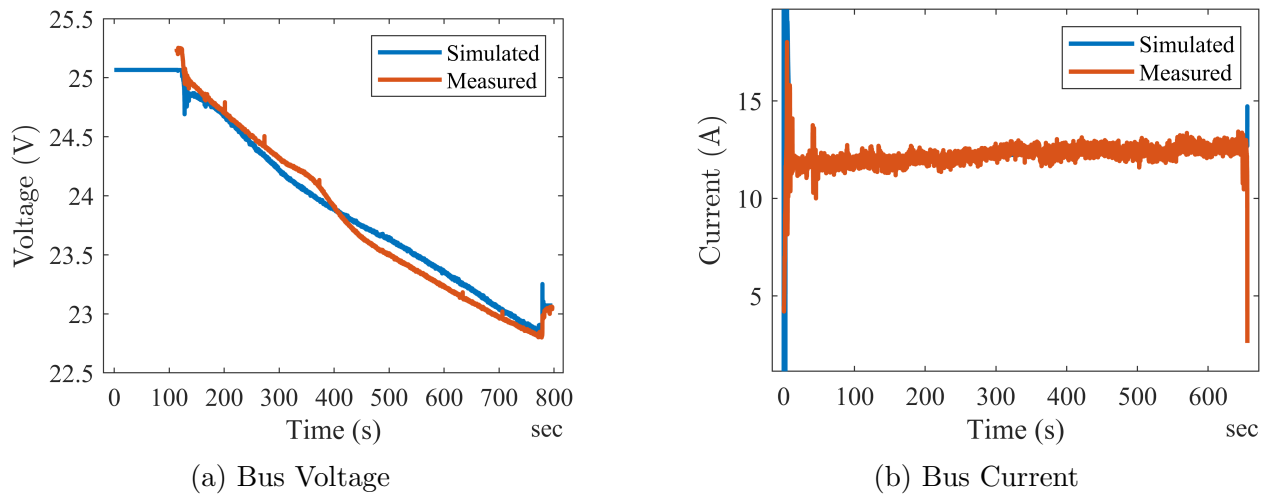


Figure 5.22: Measured vs. simulated signals, Configuration 3

Considerable manual tuning of manufacturer aerodynamic coefficients was required to achieve good alignment between the model and experimental data in the above tests. To account for this discrepancy in the design optimization studies, the aerodynamic coefficients reported by the manufacturer were modified with correction factors specified in Table 5.4. These values were found by computing the average of the correction factors used to tune the coefficients in each of the tests. Note that these corrections did not impact the design

optimized for the endurance-per-price objective, detailed below. The optimal configuration found with the correction factors was the same as that found without them.

Variable	Description	Correction Factor
$k_P$	Power Coefficient	1.25
$k_T$	Thrust Coefficient	0.85

Table 5.4: Aerodynamic coefficient correction factors

## 5.5 Optimization Preliminaries

In this work, we desire to select a battery, motor, and propeller for the quadrotor system in two optimization studies. In the first ‘static’ study, the quadrotor’s endurance per system price is maximized. In the second ‘dynamic’ study, the planar quadrotor’s time to complete a dynamic mission is minimized. Both of these studies draw from the same component databases, use the same parameter surrogate models, and use the same boundary constraint functions detailed in this section. Also, each of the studies are initialized from a common initial configuration.

### 5.5.1 Component Databases

Components are selected from three component databases: the battery is selected from a database of 33 Turnigy Graphene battery packs [80], the motor is selected from a database of 27 brushless motors from KDE Direct [81], and the propeller is selected from a database of 90 propellers from APC’s “Electric” line. Tabular data for each of the component databases can be found in Appendix D. Using Equation 4.10, the design space includes 80,190 possible system configurations. The aforementioned suppliers were selected for having ample catalogs, high-quality products, and well-documented specifications and test data. It is worth noting that, originally, the entire APC catalog was considered. This includes their “Electric” (E), “Folding” (F), “Multi-Rotor” (MR), and “Slow Flyer” (SF) product lines. Unfortunately, these product lines are categorical design parameters that conceal any quantifiable difference in the physical design. Propellers with similar pitch and diameter, their design variable values, have significantly different performance parameter values across product lines. This violates the one-to-one mapping requirement for effective component parameterization. Figure 5.23 illustrates this by plotting the torque and power coefficients as a function of pitch and diameter for each of the product lines.

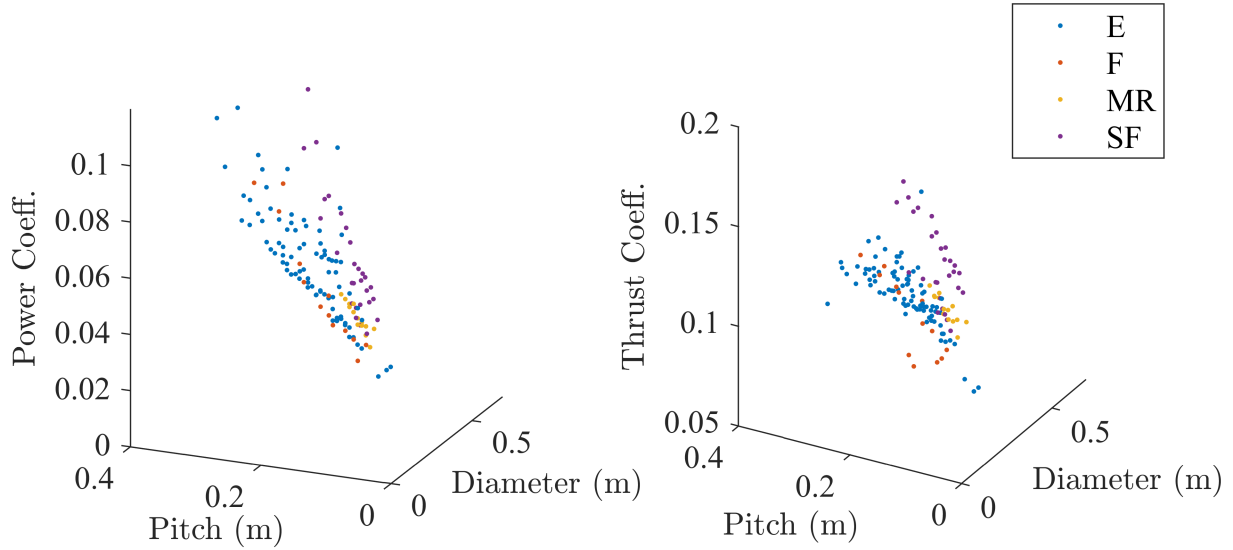


Figure 5.23: APC propeller database consisting of “Electric” (E), “Folding” (F), “Multi-Rotor” (MR), and “Slow Flyer” (SF) product lines

The SF line has significantly higher torque and power constants for a given pitch and diameter, as they are designed to reduce required rotor speed. The MR line occupies a region between the SF line and the remaining data. There is no clear distinction between the E and F lines, which together constitute most of the data. To eliminate the categorical variable, the propeller database was limited to the E line since it contains the most components. Also note that APC specified experimentally determined thrust and power coefficients over a range of RPM values. The effective values were taken to be the mean of the data between 1,000 RPM and 10,000 RPM. The quadratic relationship between thrust/power and propeller speed began to break down for speeds larger than 10,000 RPM, which rendered the aerodynamic coefficients unusable at high speeds.

## 5.5.2 Component Parameterization and Surrogate Models

### Battery

The battery’s design and performance parameters are given in Table 5.5

Design Parameters	
$N_s$	Series Cells
$Q$	Pack Capacity (mAh)
Performance Parameters	
$R_s$	Cell Resistance ( $\Omega$ )
$m$	Mass (kg)
$p$	Price (USD)

Table 5.5: Battery design and performance parameters

### Cell Resistance Surrogate

The parameter surrogate for estimating the cell resistance was created using nonlinear least-squares regression to fit the parametric model 5.71 to the component data. This model was chosen to reflect features observed in the data, including the inverse relationship between  $Q$  and  $R_s$  and the approximately linear relationship between  $N_s$  and  $R_s$ .

$$S_{R_s}^B(N_s, Q) = \frac{a}{Q + b} + cN_s^{d/Q} + k \quad (5.71)$$

The employed model parameter values are given in Table 5.6.

Parameter	Value	Confidence Bounds (95%)
$a$	7.67	(5.42, 9.93)
$b$	-1.07e2	(-2.43e2, 2.98e1)
$c$	-8.12	(-6.30e3, 6.28e3)
$d$	5.97e-1	(-4.62e2, 4.63e2)
$k$	8.12	(-6.28e3, 6.30e3)

Table 5.6: Regression model parameters for battery cell resistance surrogate

Figure 5.24 provides a plot of the surrogate model, and 5.25 plots the surrogate model’s relative error for each component in the database. The relative error is defined in Equation 5.72.

$$e_{\phi,c}^S = \frac{S_{\phi}(\Phi_c^d) - \phi_c}{\phi_c} \quad (5.72)$$

Above,  $e_{\phi,c}^S$  is the relative error of the surrogate for parameter  $\phi$  evaluated at component  $c$ .  $S_{\phi}$  is the surrogate model that estimates  $\phi$  as a function of the component’s design parameters  $\Phi_c^d$ . The true parameter value obtained from the component database is  $\phi_c$ .

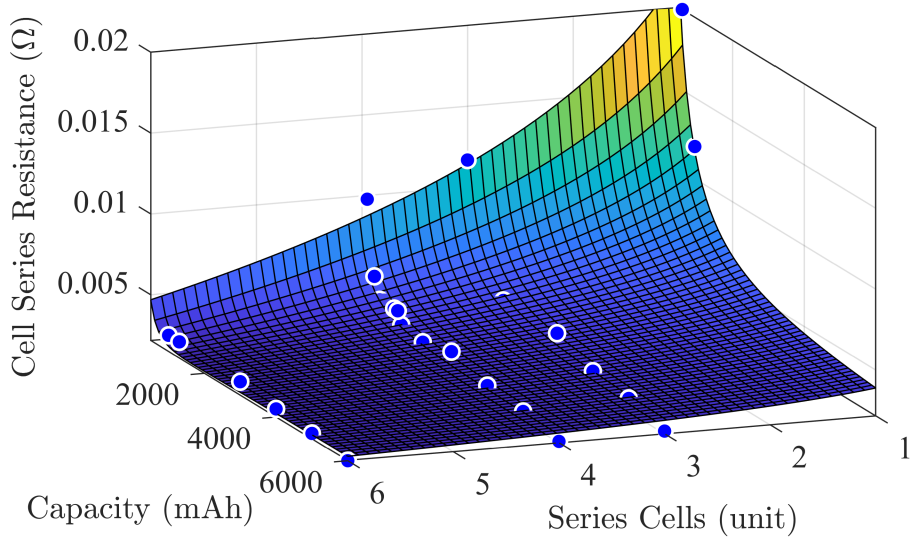


Figure 5.24: Battery cell series resistance surrogate

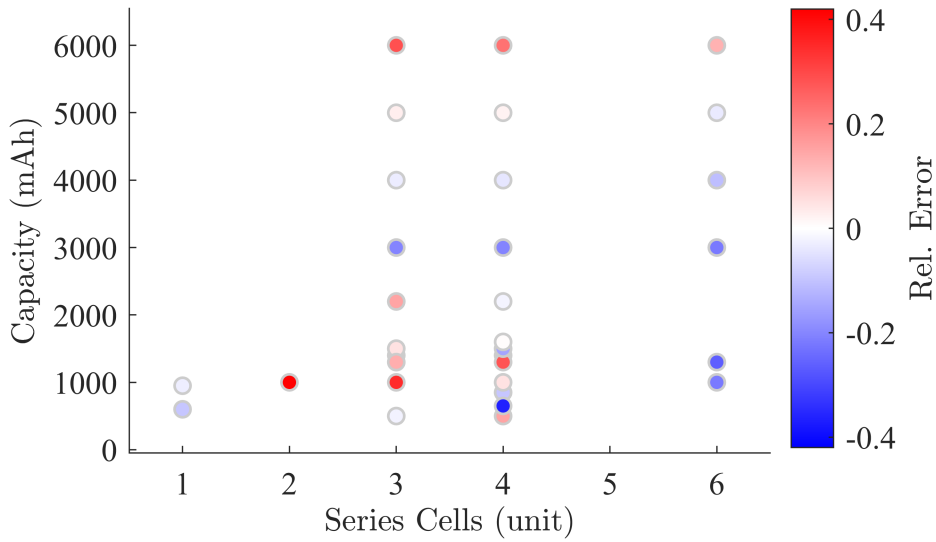


Figure 5.25: Relative error of surrogate model compared to component data

### Mass Surrogate

The parameter surrogate for estimating battery pack mass was created using nonlinear least-squares regression to fit the parametric model given in 5.73 to the component data. This model was chosen to reflect the positive relationship between pack size and mass, accounting for a small offset from packaging materials.

$$S_m^B(N_s, Q) = aN_sQ + b \quad (5.73)$$

The employed model parameter values are given in Table 5.7.

Parameter	Value	Confidence Bounds (95%)
$a$	$3.13e-5$	$(3.08e-5, 3.18e-5)$
$b$	$1.94e-2$	$(1.30e-2, 2.57e-2)$

Table 5.7: Regression model parameters for battery pack mass surrogate

Figure 5.26 provides a plot of the surrogate model, and 5.27 plots the surrogate model’s relative error for each component in the database.

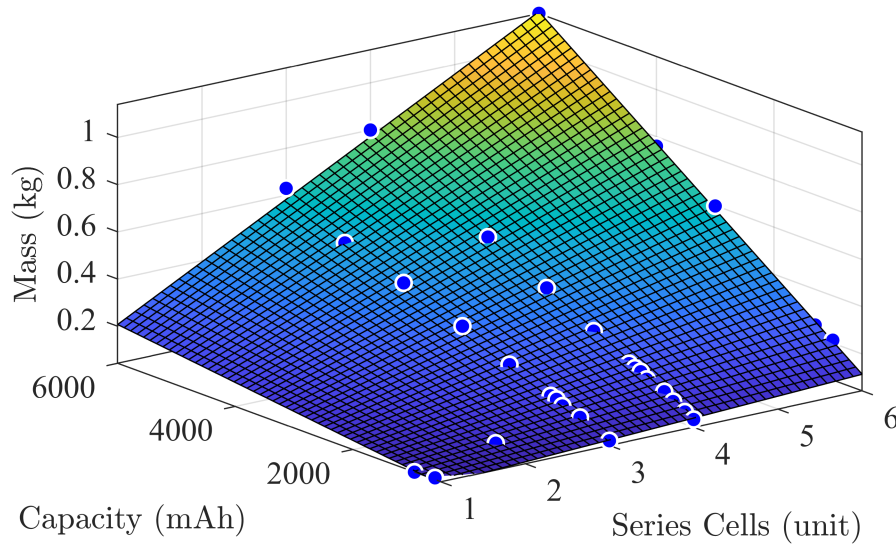


Figure 5.26: Battery pack mass surrogate

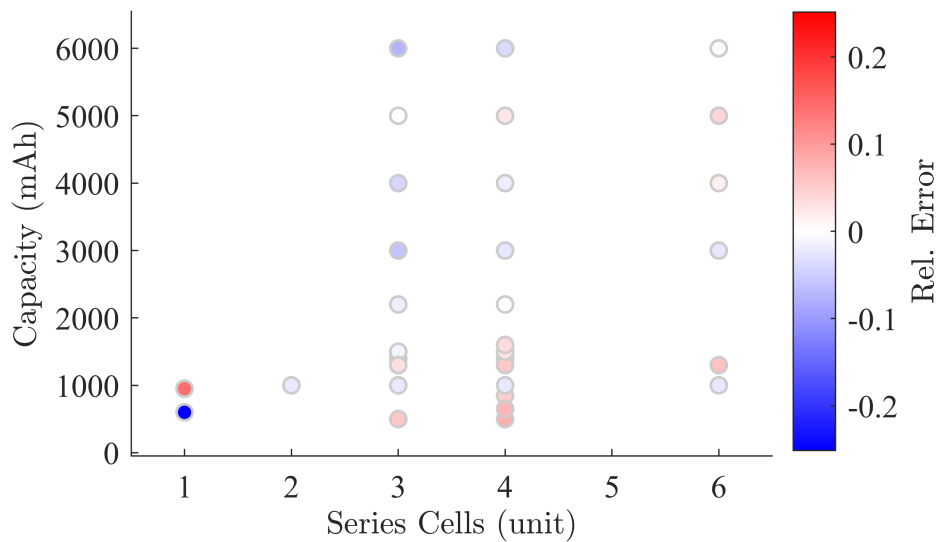


Figure 5.27: Relative error of surrogate model compared to component data

## Price Surrogate

The parameter surrogate for estimating battery pack price was created using linear least-squares regression to fit the third-order polynomial surface given in 5.74 to the component data.

$$S_p^B(N_s, Q) = p_{00} + p_{10}N_s + p_{01}Q + p_{20}N_s^2 + p_{11}N_sQ + p_{02}Q^2 + p_{30}N_s^3 + p_{21}N_s^2Q + p_{12}N_sQ^2 + p_{03}Q^3 \quad (5.74)$$

The employed model parameter values are given in Table 5.8.

Parameter	Value	Confidence Bounds (95%)
$p_{00}$	4.25e1	(4.01e1, 4.49e1)
$p_{10}$	1.47e1	(1.09e1, 1.85e1)
$p_{01}$	2.96e1	(2.65e1, 3.26e1)
$p_{20}$	-7.49e-1	(-2.36, 8.61e-1)
$p_{11}$	9.93	(6.78, 1.31e1)
$p_{02}$	1.38	(-1.73, 4.50)
$p_{30}$	-1.02e-1	(-1.51, 1.30)
$p_{21}$	3.57e-1	(-1.78, 2.49)
$p_{12}$	-2.79	(-4.75, -8.32e-1)
$p_{03}$	-1.93	(-3.84, -1.12e-2)

Table 5.8: Regression model parameters for battery pack price surrogate

Figure 5.28 provides a plot of the surrogate model, and 5.29 plots the surrogate model's relative error for each component in the database.

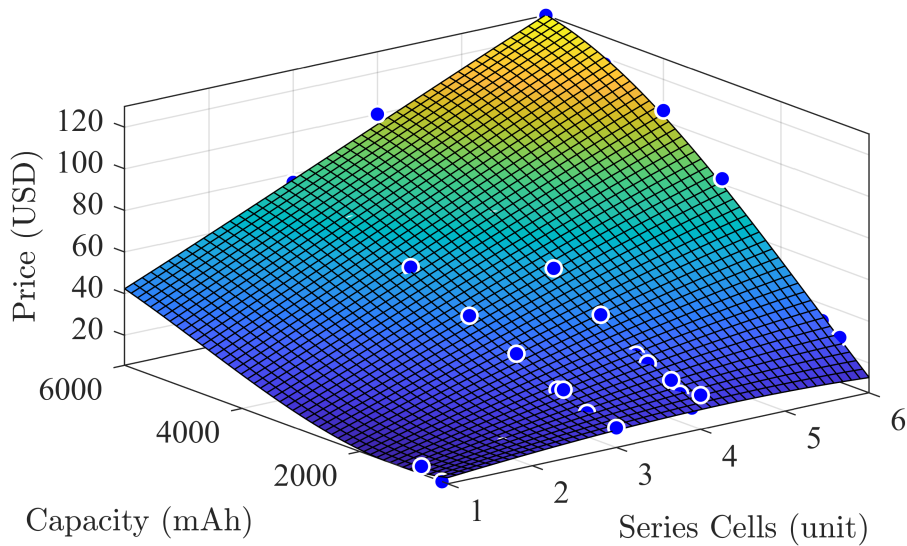


Figure 5.28: Battery pack price surrogate



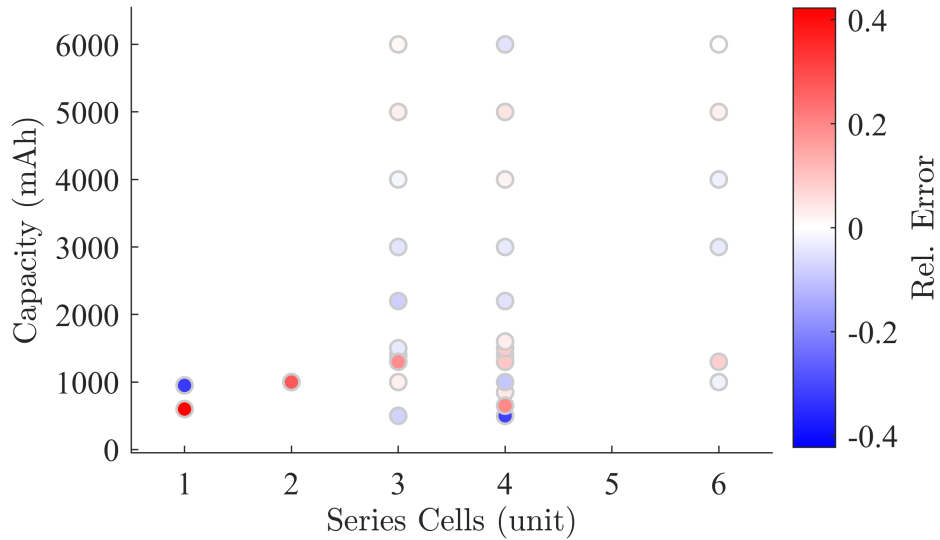


Figure 5.29: Relative error of surrogate model compared to component data

## Motor

The motor’s design and performance parameters are given in Table 5.9. Note that the design parameters chosen below, the speed constant and winding resistance, are not design parameters in the typical sense; the designer does not have direct control over them. However, these are the numeric quantities that most clearly differentiate components in the catalog, and they serve as good predictors of the other motor parameters.

Design Parameters	
$kV$	Speed Constant (RPM/V)
$Rm$	Winding Resistance ( $\Omega$ )
Performance Parameters	
$D$	Diameter (m)
$m$	Mass (kg)
$p$	Price (USD)

Table 5.9: Motor design and performance parameters

### Diameter Surrogate

The parameter surrogate for estimating motor diameter was created using nonlinear least-squares regression to fit the parametric model given in 5.75 to the component data. This model reflects the inverse relationship between a motor’s speed constant and winding resistance and its size.

$$S_D^M(kV, Rm) = \left( \frac{a}{kV + f} \right)^d + \left( \frac{b}{Rm + g} \right)^e + c \quad (5.75)$$

Model parameter values are given in Table 5.10.

Parameter	Value	Confidence Bounds (95%)
$a$	1.22e2	(-5.01e2, 7.44e2)
$b$	1.25e-3	(-1.01e-2, 1.26e-2)
$c$	1.65e-2	(-4.35e-3, 3.73e-2)
$d$	2.02	(-2.11, 6.15)
$e$	1.22	(-1.92, 4.37)
$f$	3.72e2	(-7.82e2, 1.53e3)
$g$	5.28e-5	(-4.19e-2, 4.21e-2)

Table 5.10: Regression model parameters for motor diameter surrogate

Figure 5.30 provides a plot of the surrogate model, and 5.31 plots the surrogate model's relative error for each component in the database.

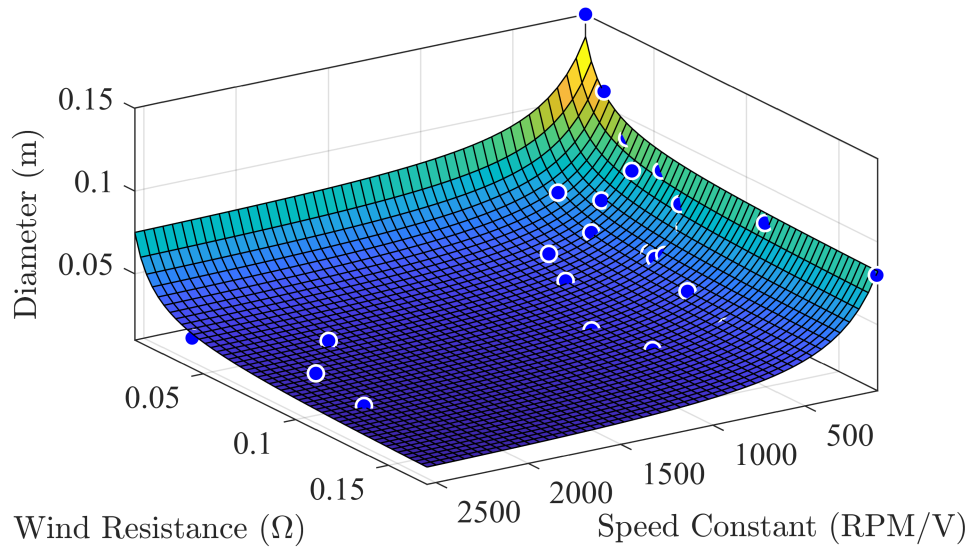


Figure 5.30: Motor diameter surrogate

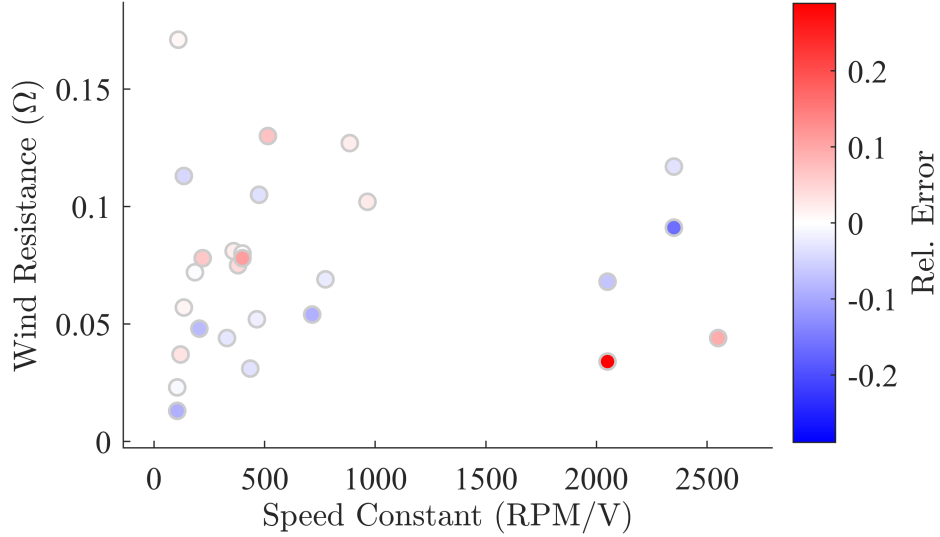


Figure 5.31: Relative error of surrogate model compared to component data

### Mass Surrogate

The parameter surrogate for estimating motor mass was created using nonlinear least-squares regression to fit the parametric model given in 5.76 to the component data. This model reflects the inverse relationship between a motor’s speed constant and winding resistance and its size.

$$S_m^M(kV, Rm) = \left( \frac{a}{kV + f} \right)^d + \left( \frac{b}{Rm + g} \right)^e + c \quad (5.76)$$

The employed model parameter values are given in Table 5.11.

Parameter	Value	Confidence Bounds (95%)
$a$	7.33e1	(-1.69e2, 3.16e2)
$b$	2.25e-2	(-5.16e-2, 9.66e-2)
$c$	6.62e-6	(-1.56e-1, 1.56e-1)
$d$	1.16	(-1.09, 3.40)
$e$	3.00	(-5.26, 1.13e1)
$f$	9.99e-2	(-3.05e2, 3.05e2)
$g$	7.35e-3	(-6.51e-2, 7.98e-2)

Table 5.11: Regression model parameters for motor mass surrogate

Figure 5.32 provides a plot of the surrogate model, and 5.33 plots the surrogate model’s relative error for each component in the database.

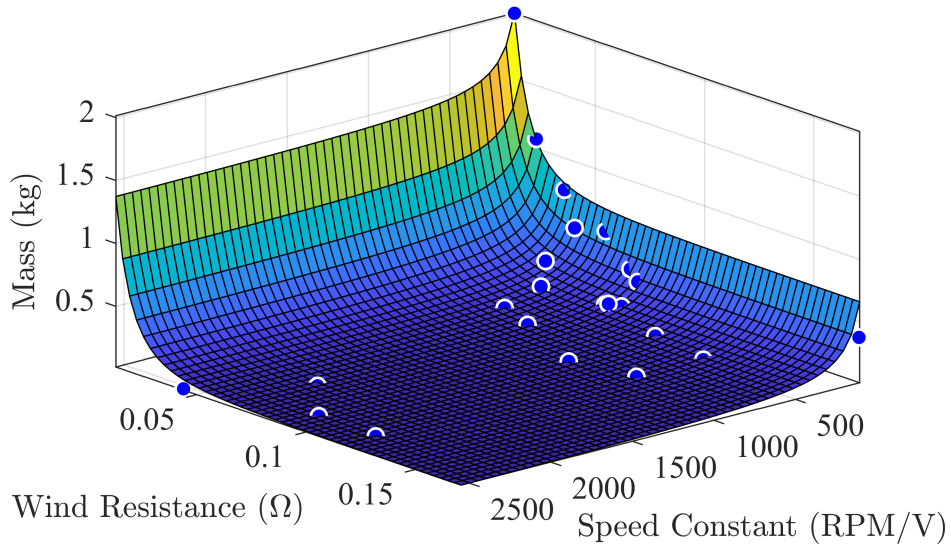


Figure 5.32: Motor mass surrogate

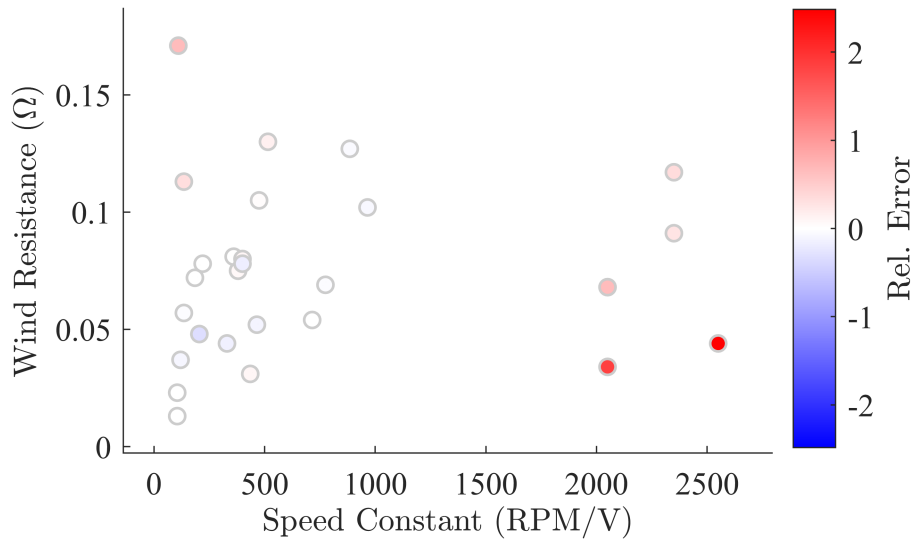


Figure 5.33: Relative error of surrogate model compared to component data

### Price Surrogate

The parameter surrogate for estimating motor price was created using nonlinear least-squares regression to fit the parametric model given in 5.77 to the component data. This model reflects the inverse relationship between a motor's speed constant and winding resistance and its price.

$$S_p^M(kV, Rm) = \left( \frac{a}{kV + f} \right)^d + \left( \frac{b}{Rm + g} \right)^e + c \quad (5.77)$$

The employed model parameter values are given in Table 5.12.

Parameter	Value	Confidence Bounds (95%)
$a$	4.85e3	(-1.71e4, 2.67e4)
$b$	5.01e-1	(-2.36, 3.37)
$c$	1.02e-5	(-1.22e2, 1.22e2)
$d$	1.94	(-2.67, 6.54)
$e$	1.87	(-2.81, 6.55)
$f$	9.82e1	(-5.59e2, 7.55e2)
$g$	3.65e-4	(-4.51e-2, 4.58e-2)

Table 5.12: Regression model parameters for motor price surrogate

Figure 5.34 provides a plot of the surrogate model, and 5.35 plots the surrogate model's relative error for each component in the database.

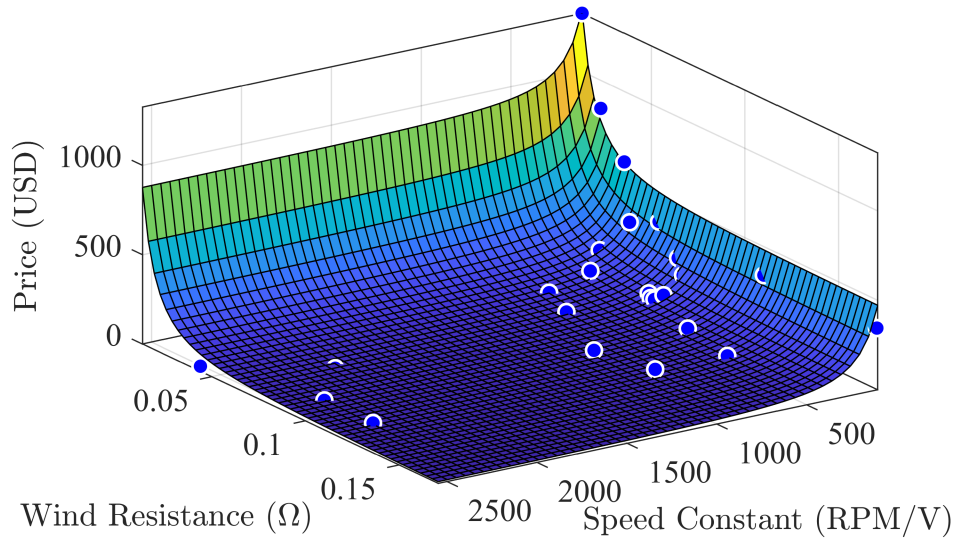


Figure 5.34: Motor price surrogate

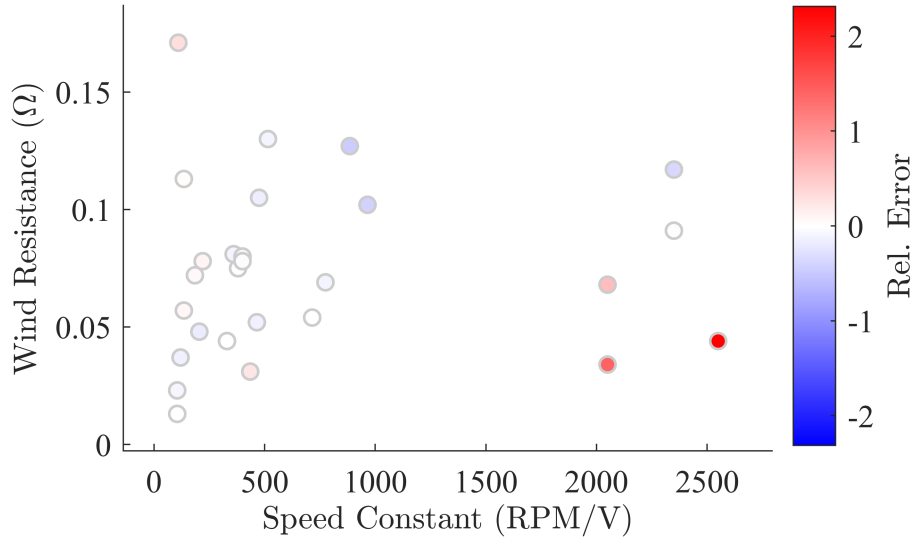


Figure 5.35: Relative error of surrogate model compared to component data

## Propeller

The propeller’s design and performance parameters are given in Table 5.13.

Design Parameters	
$D$	Diameter (m)
$P$	Pitch (m)
Performance Parameters	
$k_P$	Power Coefficient
$k_T$	Thrust Coefficient
$m$	Mass (kg)
$p$	Price (USD)

Table 5.13: Propeller design and performance parameters

Each of the propeller’s parameter surrogates were created using the “Locally Weighted Smoothing Quadratic Regression” model in MATLAB’s Curve Fitting Toolbox [82]. The ‘Normalize’ option was set to ‘on,’ the ‘Method’ option was set to ‘LowessFit,’ the ‘Robust’ option was set to ‘Bisquare,’ and the ‘Span’ option was set to 0.8.

### Power Coefficient Surrogate

Figure 5.36 provides a plot of the power coefficient surrogate model, and 5.37 plots the surrogate model’s relative error for each component in the database.

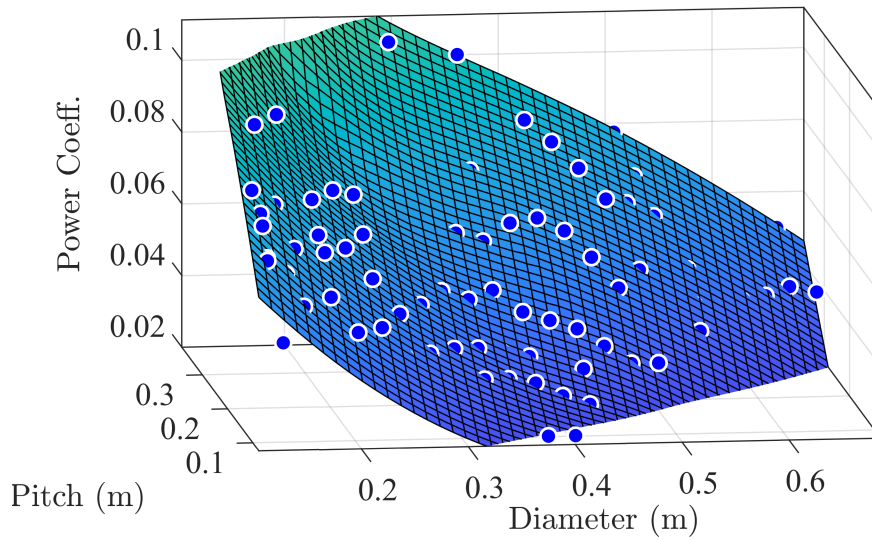


Figure 5.36: Propeller power coefficient surrogate

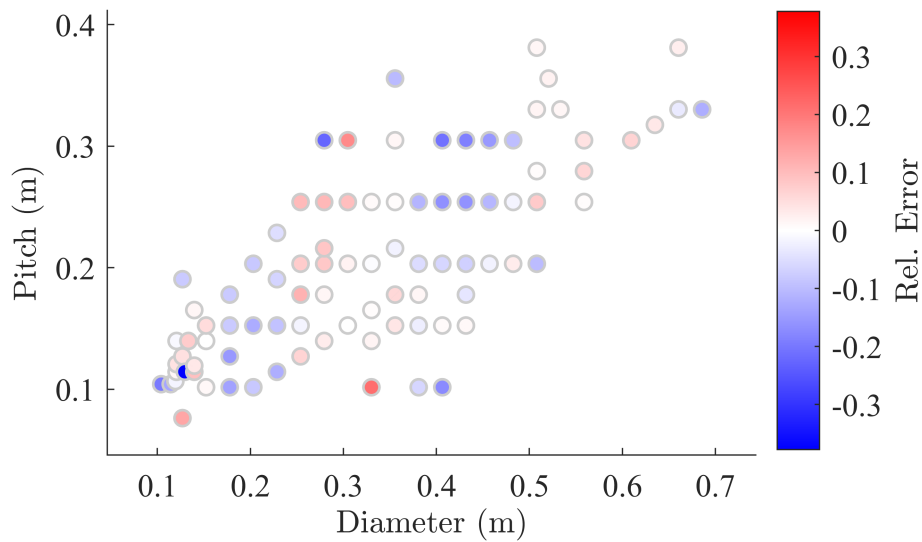


Figure 5.37: Relative error of surrogate model compared to component data

### Thrust Coefficient Surrogate

Figure 5.38 provides a plot of the thrust coefficient surrogate model, and 5.39 plots the surrogate model's relative error for each component in the database.

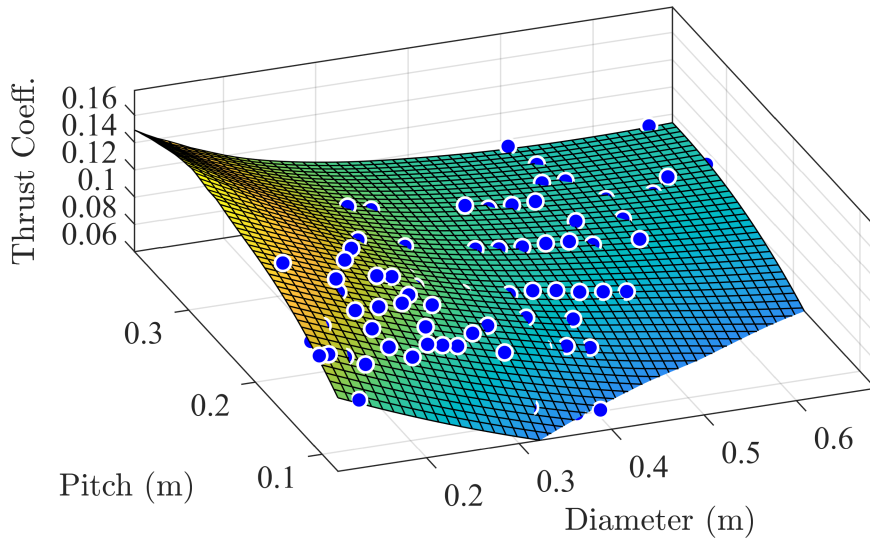


Figure 5.38: Propeller thrust coefficient surrogate

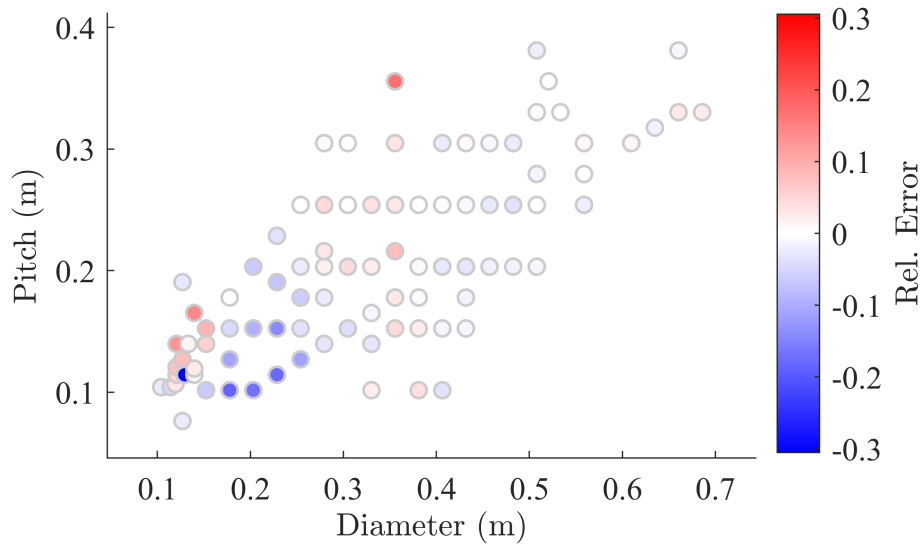


Figure 5.39: Relative error of surrogate model compared to component data

### Mass Surrogate

Figure 5.40 provides a plot of the mass surrogate model, and 5.41 plots the surrogate model's relative error for each component in the database.



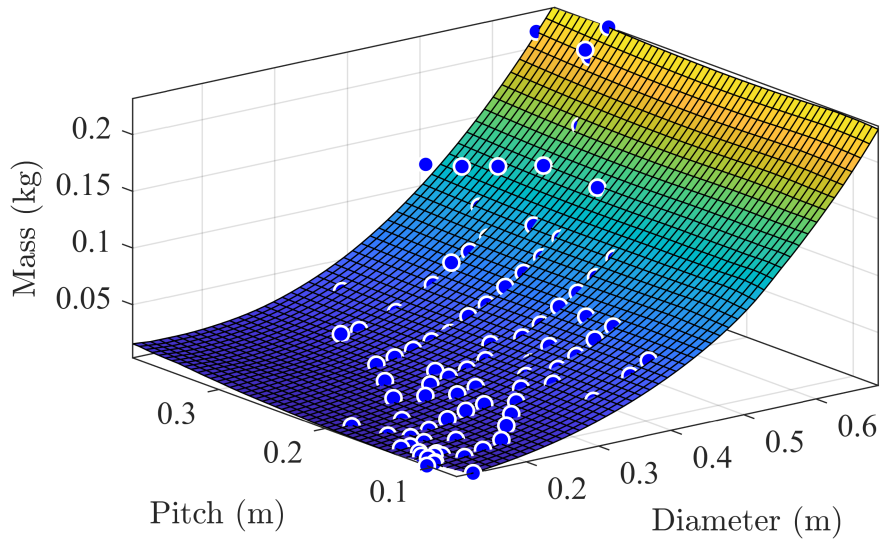


Figure 5.40: Propeller mass surrogate

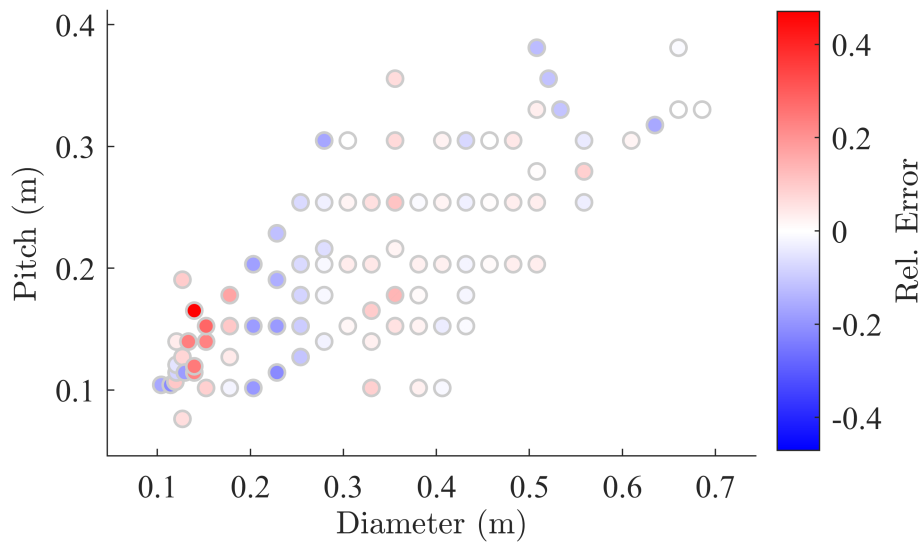


Figure 5.41: Relative error of surrogate model compared to component data

### Price Surrogate

Figure 5.42 provides a plot of the price surrogate model, and 5.43 plots the surrogate model's relative error for each component in the database.

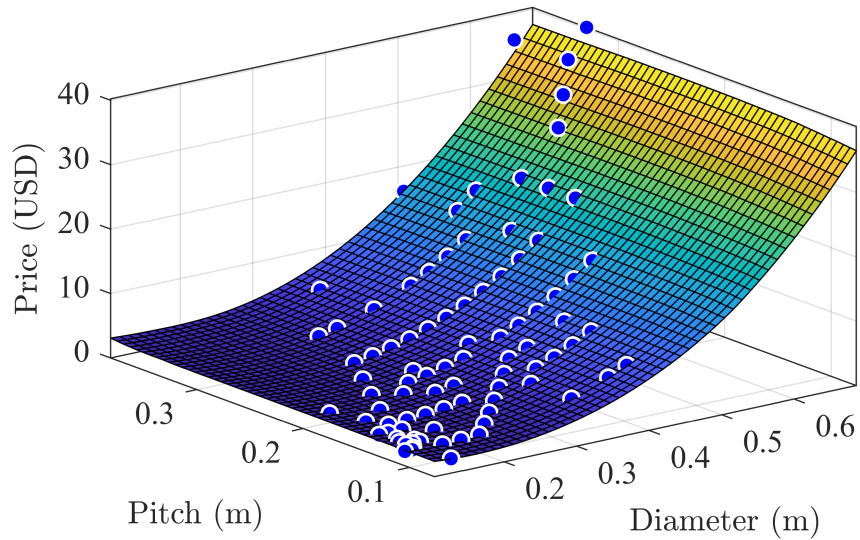


Figure 5.42: Propeller price surrogate

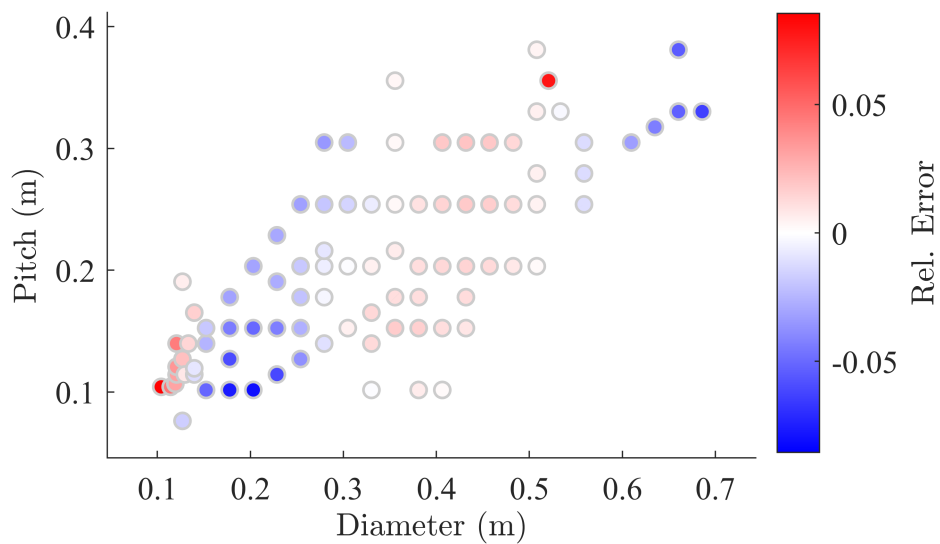


Figure 5.43: Relative error of surrogate model compared to component data

Figure 5.44 plots the distribution of relative errors for each component, and Table 5.14 provides the minimum, maximum, and standard deviation of relative errors for each of the surrogate models. The largest errors are seen in the motor mass and price surrogates, where two motors with a high speed constant and low winding resistance are significantly less heavy and costly than predicted by the surrogates.

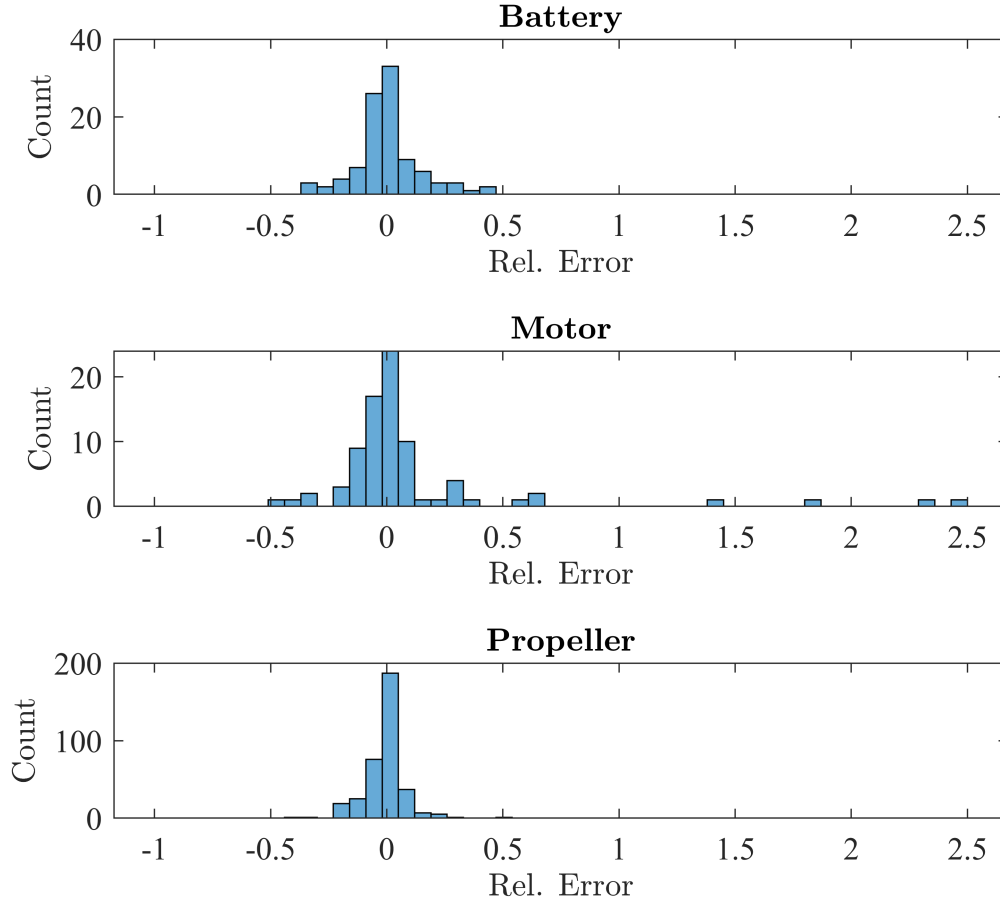


Figure 5.44: Distribution of relative errors

Surrogate	Min Rel. Error	Max Rel. Error	Standard Deviation
Battery Cell Resistance $\phi_{R_s}^B$	$-3.65e-1$	$4.20e-1$	$1.81e-1$
Battery Mass $\phi_m^B$	$-2.52e-1$	$1.43e-1$	$6.39e-2$
Battery Price $\phi_p^B$	$-3.27e-1$	$4.23e-1$	$1.43e-1$
Motor Diameter $\phi_d^M$	$-1.64e-1$	$2.89e-1$	$8.27e-2$
Motor Mass $\phi_m^M$	$-3.23e-1$	2.49	$6.10e-1$
Motor Price $\phi_p^M$	$-4.55e-1$	2.32	$5.61e-1$
Propeller Power Coefficient $\phi_{k_P}^P$	$-3.78e-1$	$2.14e-1$	$9.50e-2$
Propeller Thrust Coefficient $\phi_{k_T}^P$	$-3.06e-1$	$1.72e-1$	$6.57e-2$
Propeller Mass $\phi_m^P$	$-2.15e-1$	$4.71e-1$	$1.14e-1$
Propeller Price $\phi_p^P$	$-8.03e-2$	$8.56e-2$	$2.94e-2$

Table 5.14: Minimum, maximum, and standard deviation of relative errors for each surrogate model

When the optimization objective is sensitive to a parameter, then error in the parameter

surrogate can significantly impact the solution to the problem's continuous-representation. The designer must keep this in mind when tuning the surrogate model to balance smoothness and accuracy. When significant inaccuracies cannot be avoided, then the scope of the discrete search should be increased to compensate for less reliable sorting.

### 5.5.3 Boundary Constraint Functions

The process presented in Section 4.1.3 was used to generate boundary constraint functions for the battery, motor, and propeller. These are shown in Figures 5.45-5.47. Any point in the design space where the component's boundary constraint function evaluates to zero or a negative value is valid.

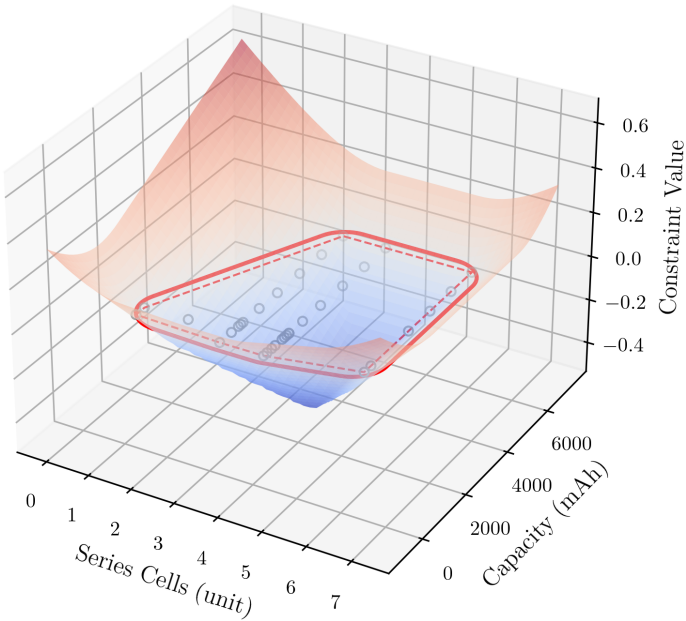


Figure 5.45: Battery boundary constraint function

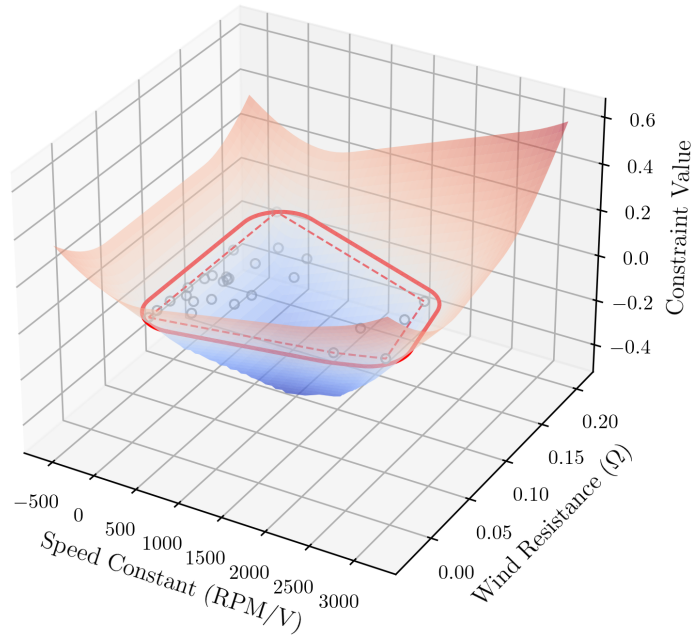


Figure 5.46: Motor boundary constraint function

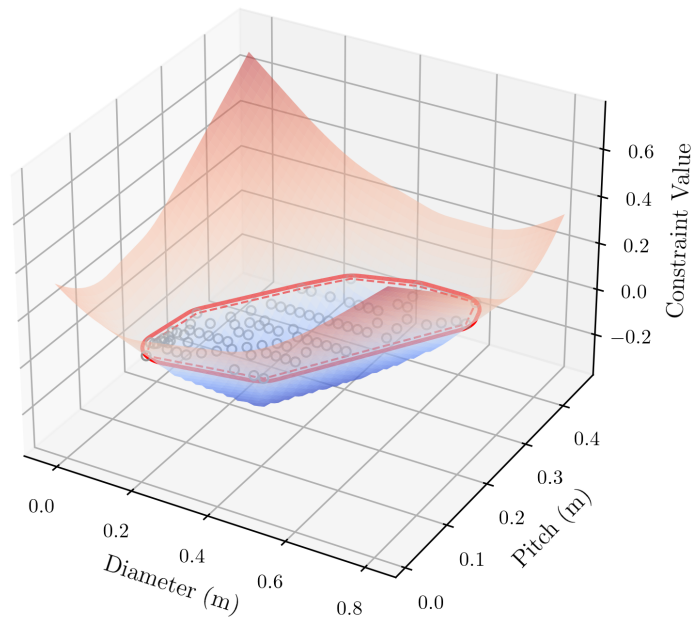


Figure 5.47: Propeller boundary constraint function

In the following optimization study, the boundary constraint function for the motor and the

propeller have a significant impact on the continuous-representation solution, helping to keep the design variables close to regions of the design space populated by discrete components.

### 5.5.4 Initial Configuration

Both optimization studies are initiated from the same point in the design space. Components in the initial configuration were manually selected from the component databases, and they have design variable values that are similar to those that ship with the HolyBro S500 V2 platform. The components in the initial configuration are specified in Table 5.15

Component	Make	Model
Battery	Turnigy	Graphene Panther 4S 4000mAh
Motor	KDE	KDE2315XF-965 (965kV)
Propeller	APC	9x4.5E

Table 5.15: Components in initial configuration

## 5.6 Optimization Study 1: Endurance per System Price

### 5.6.1 Discrete-Domain Formulation

The CBDO problem formulation for the first study is given in Problem 5.78

$$\begin{aligned}
 & \underset{\mathbf{i} = [i_B, i_M, i_P]}{\text{minimize}} && J_{E/P}(\mathbf{C}(\mathbf{i})) \\
 & \text{subject to} && 1 \leq i_B \leq 33, \\
 & && 1 \leq i_M \leq 27, \\
 & && 1 \leq i_P \leq 90, \\
 & && \phi_D^P(i_P) \leq \phi_{D,\max}^P, \\
 & && 0 \leq u_{ss}(\mathbf{C}(\mathbf{i})) \leq 1, \\
 & && i_{ss}^B(\mathbf{C}(\mathbf{i})) \leq i_{\max}^B(i_B), \\
 & && i_{ss}^I(\mathbf{C}(\mathbf{i})) \leq i_{\max}^I
 \end{aligned} \tag{5.78}$$

In 5.78,  $i_B$ ,  $i_M$ , and  $i_P$  refer to the battery, motor, and propeller indices, respectively, and the combined component database  $\mathbf{C}$  maps the selection indices to the corresponding parameter values. The cost function  $J_{E/P}$  calculates the endurance, or maximum flight time in steady-state hover, divided by the system price. It is defined in Equation 5.79. Note that, to pose

problem as a minimization, the endurance-per-price calculation is negated in the objective.

$$J_{E/P} = - \left( \frac{\phi_Q^B(i_B)}{i_{ss}(\mathbf{i})} \right) \frac{1}{\phi_p^S(\mathbf{i})} \quad (5.79)$$

Above, the notation  $\phi_x^t$  is used to refer to the  $x$  parameter of component type  $t$ , and  $\phi_x^S$  refers to the system parameter  $x$ . The system price  $\phi_p^S$  is the sum of the battery price, motor price, propeller price, and price of the fixed system. Any additional components not selected in the optimization are included in the fixed-system price, including the power management system, flight controller, GPS, telemetry radio, and nominal ESC. The steady-state bus current  $i_{ss}$ , steady-state inverter input  $u_{ss}$  and other states are calculated by numerically solving the simplified powertrain model (Equations 5.29 and 5.30) and the simplified body dynamic model (Equation 5.66) for steady-state. The quadrotor frame dimensions restrict the propeller diameter  $\phi_D^P$  to a maximum propeller diameter  $\phi_{D,\max}^P$ . Because the inverter input is a PWM signal, it is constrained to range between zero and one. The battery and inverter have maximum current ratings, requiring additional constraints. The battery's steady-state current  $i_{ss}^B$  must remain below its maximum current  $i_{\max}^B$ , calculated with Equation 5.4. Similarly, the inverter steady-state current  $i_{ss}^I$  must not exceed its limiting value  $i_{\max}^I$ . In these studies, the maximum inverter current is taken to be 80A, since it is assumed that an inverter up to this size can be purchased depending on the power requirements of the optimized system.

## 5.6.2 Continuous-Domain Representation

The continuous-domain representation of Problem 5.78 is presented in Problem 5.80.

$$\begin{aligned} \Phi^d &= [\phi_{N_s}^B, \phi_Q^B, \phi_{kV}^M, \phi_R^M, \phi_P^P, \phi_D^P] \\ &\text{minimize} && J_{E/P}(\Phi^d, \mathbf{S}(\Phi^d)) \\ &\text{subject to} && 0 \geq g_B^B(\phi_{N_s}^B, \phi_Q^B), \\ & && 0 \geq g_B^M(\phi_{kV}^M, \phi_R^M), \\ & && 0 \geq g_B^P(\phi_P^P, \phi_D^P), \\ & && 0 \leq u_{ss}(\Phi^d, \mathbf{S}(\Phi^d)) \leq 1, \\ & && i_{ss}^B(\Phi^d, \mathbf{S}(\Phi^d)) \leq i_{\max}^B(\phi_{N_s}^B, \phi_Q^B), \\ & && i_{ss}^I(\Phi^d, \mathbf{S}(\Phi^d)) \leq i_{\max}^I, \\ & && \phi_D^P \leq \phi_{D,\max}^P \end{aligned} \quad (5.80)$$

In 5.80, the design parameter vector  $\Phi^d$  contains the design parameters for the battery, motor, and propeller respectively. The parameter surrogate function  $\mathbf{S}$  uses the surrogates

specified in 5.5.2 to calculate each component’s performance parameters. Functions  $g_B^B$ ,  $g_B^M$ , and  $g_B^P$  are the boundary constraint functions for the battery, motor, and propeller, respectively.

### 5.6.3 Continuous-Domain Analysis and Solution

The continuous-domain representation of the problem permits further analysis to gain deeper insight into the problem. A sensitivity analysis of the objective function to the design parameters was conducted about the initial design point. First, finite differencing, as defined in Equation 2.3, was used to determine the gradient of the objective function with respect to each design variable. The gradient was then scaled to determine the objective’s relative sensitivity to each parameter. A scaled gradient value  $y$  of function  $f$  taken with respect to parameter  $x$  can be interpreted as follows: “Increasing  $x$  by 1% would increase  $f$  by  $y\%$  assuming  $f$  stays sufficiently linear.” This scaling is given in Equation 5.81.

$$\frac{\partial \tilde{J}}{\partial \tilde{\Phi}^d} = \frac{\partial J}{\partial \Phi^d} \left| \frac{K}{J(\Phi_0^d)} \right| \quad (5.81)$$

Above,  $J$  is the cost function,  $\partial \tilde{J} / \partial \tilde{\Phi}^d$  is the scaled gradient,  $\partial J / \partial \Phi^d$  is the unscaled gradient,  $\Phi_0^d$  are the design variable values at the analysis point, and  $K$  is a scaling matrix defined as:

$$K = \text{diag}(\Phi_0^d) \quad (5.82)$$

The results of the parameter sensitivity analysis are presented in Table 5.16.

Parameter	Value (units)	Gradient	Scaled Gradient
Propeller Diameter $\phi_D^P$	2.29e−1 (m)	−4.00	−7.35e−1
Propeller Pitch $\phi_P^P$	1.14e−1 (m)	1.27	1.20e−1
Battery Series Cells $\phi_{N_s}^B$	4	−1.08e−1	−3.48e−1
Battery Capacity $\phi_Q^B$	4.00e3 (mAh)	−1.00e−4	−3.25e−1
Motor Speed Constant $\phi_{k_V}^M$	9.65e2 (RPM/V)	−2.00e−4	−1.90e−1
Motor Winding Resistance $\phi_{R_m}^M$	1.02e−1 ( $\Omega$ )	−4.25	−3.48e−1

Table 5.16: Objective’s sensitivity to parameters

To understand the sensitivity of the objective to a component type, the sum of the absolute value of the scaled gradient, taken with respect to each of the type’s design parameters, was



considered. That is,

$$\Sigma_t^J = \sum_{i=1}^M \left| \frac{\partial J}{\partial \Phi_{t,i}^d} \right| \quad (5.83)$$

where  $\Sigma_t^J$  is the sensitivity of  $J$  to component type  $t$ , and  $\Phi_{t,i}^d$  is the  $i^{\text{th}}$  design variable of  $t$ . These values are given in Table 5.17.

Component	Sensitivity
Propeller	8.51e-1
Battery	6.74e-1
Motor	5.39e-1

Table 5.17: Objective’s sensitivity to components

For this objective, the problem is most sensitive to the propeller design, followed by the battery and motor. The scaled sensitivities indicate that the objective favors large, efficient propellers with a relatively low pitch.

### Takeaway

The endurance-per-price objective is most sensitive to the propeller, and it favors large, efficient propellers with low pitch.

Problem 5.80 was then solved with the SQP algorithm in MATLAB’s ‘fmincon’ function. The gradient information required by the algorithm was calculated with finite differencing, though analytical derivatives could have been specified to improve solution efficiency. The algorithm required 15 iterations to converge to the location in the design space indicated in Figure 5.48.

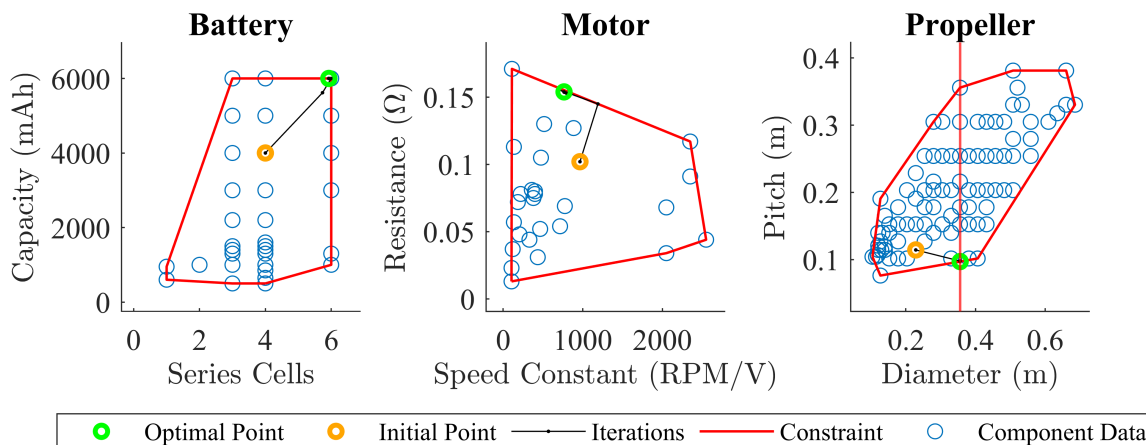


Figure 5.48: Continuous solution in the design variable space

Figure 5.49 plots the objective function value and first-order optimality at each of the solver iterations. For constrained optimization problems, first-order optimality measures how well a point satisfies the Karush-Kuhn-Tucker (KKT) conditions; the reader is referred to [83] for more details. At the final iteration, the first order optimality is near zero with a value of  $1.66\text{e-}6$ , indicating that the solver successfully converged to a stationary point.

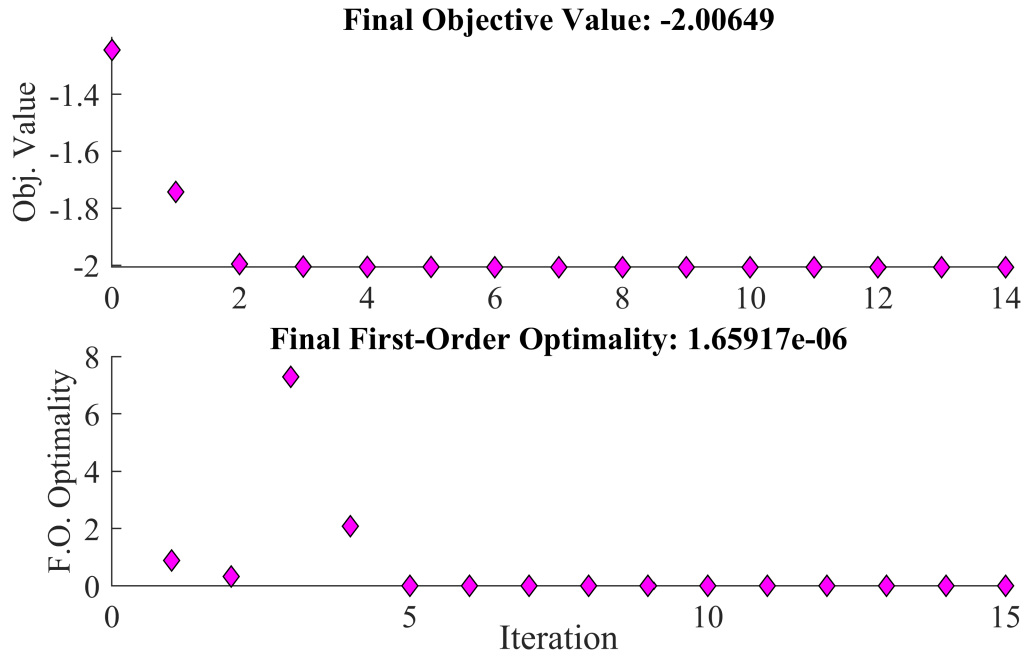


Figure 5.49: Objective function value and first-order optimality measure vs. solver iterations.

The final Lagrange multiplier values indicate which constraints are active and quantify how much the corresponding constraints influence the solution [16]. Three constraints are active at the solution: the battery capacity constraint with a Lagrange multiplier of 0.14, the propeller boundary constraint with a Lagrange multiplier value of 1.40, and the upper bound on propeller diameter with a Lagrange multiplier value of 0.45.

Finally, Table 5.18 specifies the optimal design parameter values and their deviation from the initial design. These values serve as the target in the next phase of the hybrid approach, the distance-sorted search algorithm.

Parameter	Unit	Optimal Value	Initial Value	% Change
Propeller Diameter $\phi_D^P$	m	3.56e-1	2.29e-1	55.73 %
Propeller Pitch $\phi_P^P$	m	9.70e-2	1.14e-1	-15.12 %
Battery Series Cells $\phi_{N_s}^B$	unit	5.93	4	48.19 %
Battery Capacity $\phi_Q^B$	mAh	6.00e3	4.00e3	50.00 %
Motor Speed Constant $\phi_{kV}^M$	RPM/V	7.65e2	9.65e2	-20.75 %
Motor Winding Resistance $\phi_{Rm}^M$	$\Omega$	1.54e-1	1.02e-1	50.88 %

Table 5.18: Optimal design parameter values

### 5.6.4 Discrete Problem Solution

The DSS algorithm was run using the solution to the continuous-domain representation as the target. First, the propeller database was filtered to remove all components exceeding the maximum propeller diameter. The set of configurations was then enumerated and sorted by the configuration distance metric. For each configuration evaluated, the configuration's parameters were substituted into the simplified quadrotor model, and the model was solved for its steady-state condition. Constraint functions were calculated to determine the feasibility of the configuration before calculating the objective function. Five-hundred configurations were evaluated in the search, as shown in Figure 5.50. In the figure, each iteration represents a unique configuration. The lower axes show the configuration distance metric  $d_s$  increasing monotonically, as this was the metric used to sort the list of configurations, along with the component distance metric values. The upper axes show the objective function value corresponding to the configuration. Infeasible configurations are not assigned an objective function value and create a gap in the plot. This data follows an increasing trend, indicating that the configuration's distance from the continuous-domain solution effectively predicts the objective. If the predictor were perfect, however, no noise would exist and the objective function values would rise monotonically. One source of this noise is the error in the parameter surrogate models. Another source is the topography of the objective function, as nonconvexities and ill-conditioning of the Hessian matrix near the optimal point muddle the relationship between distance and objective function value.

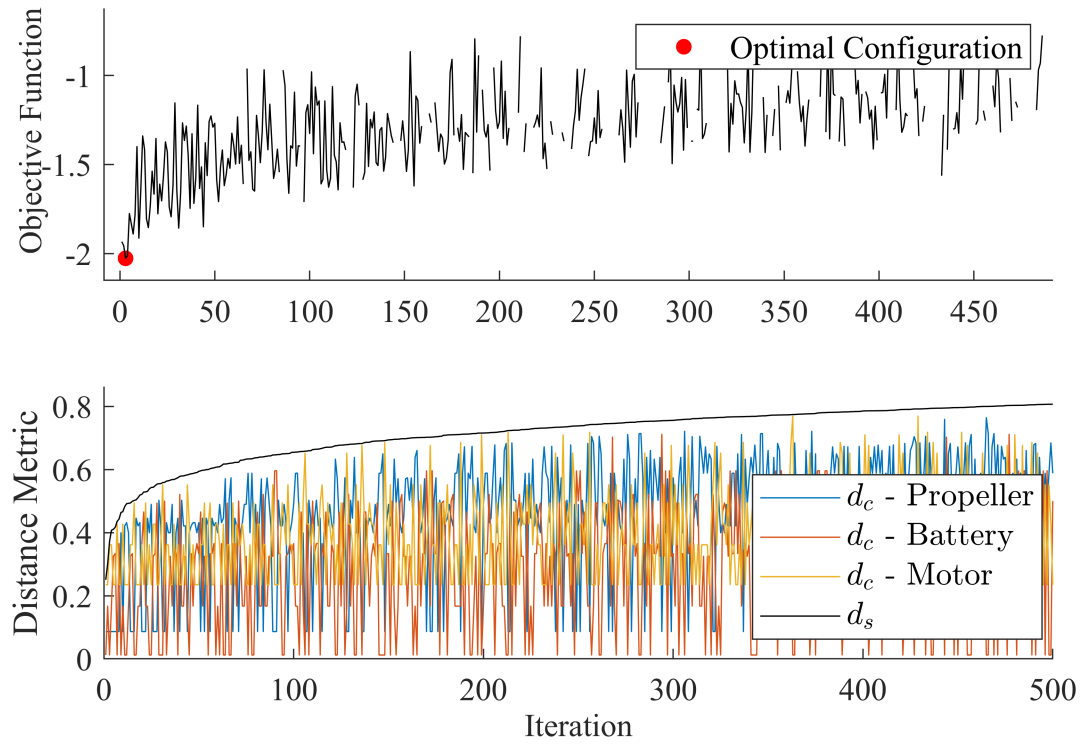


Figure 5.50: First 500 iterations of the DSS algorithm

The best component configuration was identified on the third search iteration, and is provided below in Table 5.19.

Make	Model	SKU	Mass (kg)	Price (USD)
Turnigy	Graphene Panther	9067000420-0	1.14	\$129.99
KDE	KDE2814XF-515	N/A	9.50e-2	\$74.95
APC	13x4E	LP13040E	3.01e-2	\$5.06

Table 5.19: Discrete solution to Problem 5.78

Table 5.20 provides the design parameter values of the discrete solution in comparison with those of the initial configuration.

Parameter	Unit	Continuous Sol.	Discrete Sol.	Initial	% Change
Propeller Diameter $\phi_D^P$	m	3.56e-1	3.30e-1	2.29e-1	44.44 %
Propeller Pitch $\phi_P^P$	m	9.70e-2	1.02e-1	1.14e-1	-11.11 %
Battery Series Cells $\phi_{N_s}^B$	unit	5.93	6	4	50.00 %
Battery Capacity $\phi_Q^B$	mAh	6.00e3	6.00e3	4.00e3	50.00 %
Motor Speed Constant $\phi_{kV}^M$	RPM/V	7.65e2	5.15e2	9.65e2	-46.63 %
Motor Winding Resistance $\phi_{R_m}^M$	$\Omega$	1.54e-1	1.30e-1	1.02e-1	27.45 %

Table 5.20: Optimal design parameter values

The optimized design has a larger propeller with a less aggressive pitch than the initial configuration. The optimal battery is the largest available in the catalog, and the motor has a lower speed constant and larger winding resistance.

### 5.6.5 Discussion of Optimized Design

The optimal design improves the objective by 76.19% from 1.15 seconds per dollar to 2.03 seconds per dollar, as presented in Table 5.21.

Mission Duration	
Initial	1.15
Final	2.03
Change	76.19%

Table 5.21: Endurance per price objective for initial and final designs

The estimated system endurance increased from 575.16 seconds to 1,372.1 seconds, while the price increased from \$563.65 to \$676.53. The larger components increase the mass of the system substantially, from 1.54 kg to 2.32 kg. The powertrain is more efficient, largely attributable to increased system voltage. The combined efficiency of the powertrain components increased from 81.26% to 87.51%. Furthermore, the large-diameter propeller leads to a lower steady-state rotor speed. The initial configuration operated at 724.7 rad/s with 0.0877 Newton-meters of torque, while the optimized configuration operates at 513.6 rad/s with 0.126 Newton-meters of torque. Despite the increase in mass, the optimized system has a larger thrust ratio, the ratio of maximum thrust to weight. The thrust ratio increased from 2.590 to 3.424.

### 5.6.6 Algorithm Performance

In this study, the metric used to quantify a CBDO algorithm’s performance is the number of objective function evaluations required to find the global optimum. Because this objective function is computationally inexpensive, it was feasible to evaluate each of the configurations to identify the true globally optimal solution. The hybrid algorithm required a total of 123 function evaluations to identify this solution: 120 evaluations in optimizing the continuous representation and an additional three in the DSS algorithm.

Two other direct approaches were employed as performance comparisons: the DSS algorithm with the initial design as the target and a genetic algorithm. In the former approach, the DSS algorithm evaluates configurations closest to the initial point in the design space. This direct approach required a total of 8,116 function evaluations to identify the optimal configuration, illustrating the value in guiding the discrete search with continuous-domain information.

The second contending algorithm, the genetic algorithm, is the most popular algorithm used to solve discrete and mixed-integer optimization problems in multirotor literature. It therefore provides a relevant performance baseline. In this study, MATLAB’s “ga” function with default settings was used to solve Problem 5.78 for 50 trials. Because the GA is stochastic, each trial exhibited different performance. Of the 50 trials, 29 successfully found the globally optimal configuration within 300 generations, as shown in Figure 5.51.

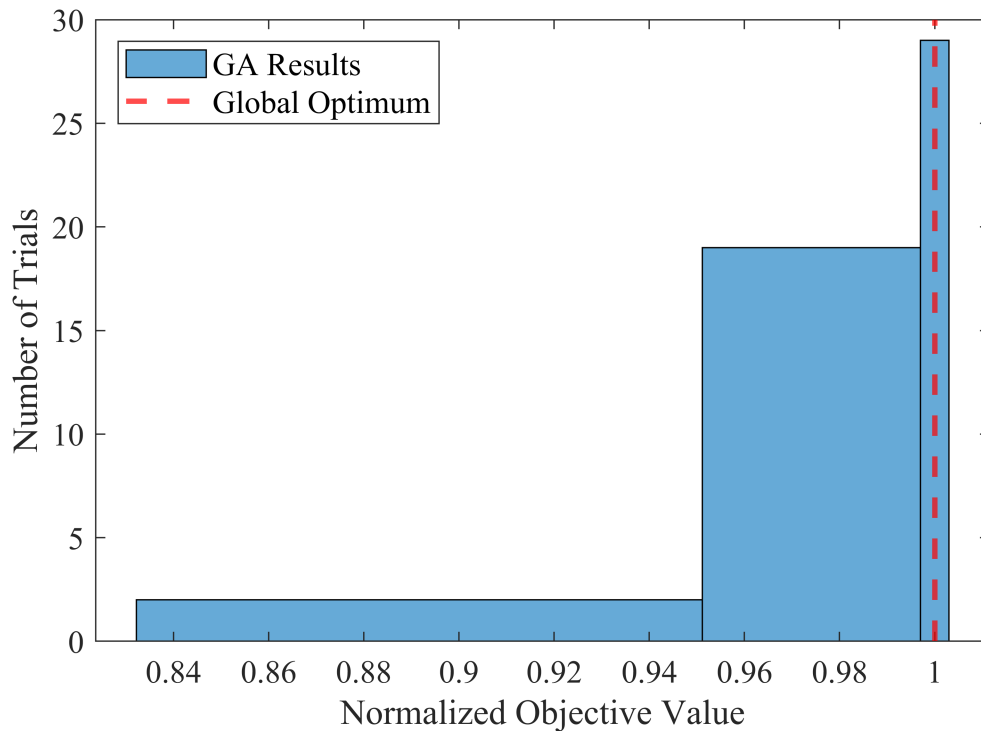


Figure 5.51: Histogram of best objective function values found by GA within 300 generations

Of the successful trials, the median number of function evaluations required to find the optimal configuration was 651, the minimum was 272, and the maximum was 1,259. Figure 5.52 provides this distribution in comparison with the number of function evaluations required in the hybrid approach.

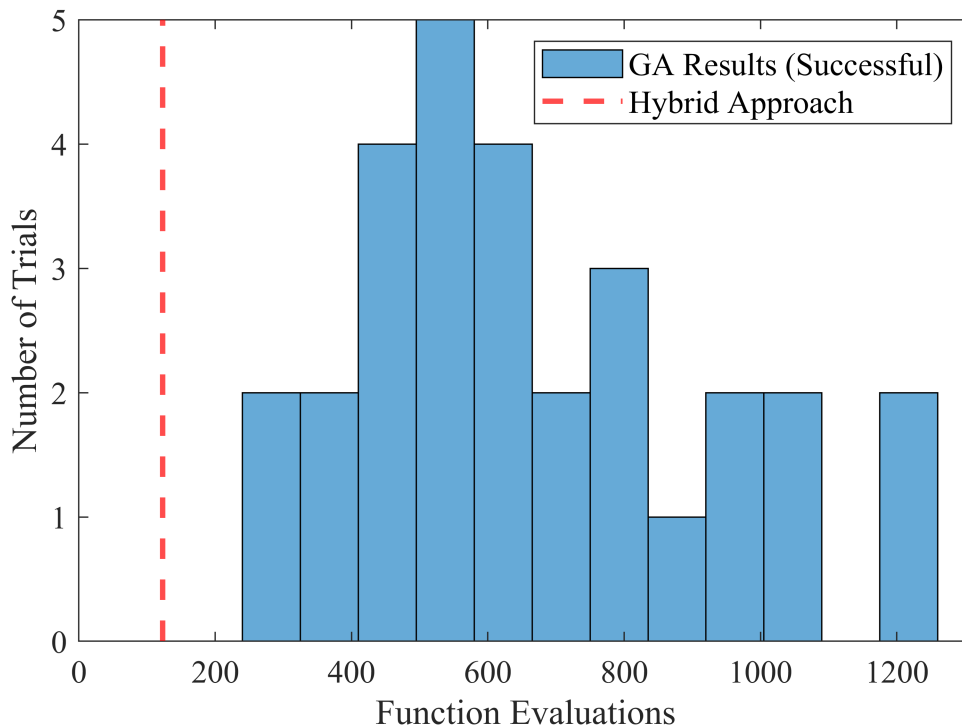


Figure 5.52: Distribution of function evaluations required to find the optimal configuration in each successful GA trial

These results indicate that the hybrid approach is a computationally-efficient approach for solving CBDO problems: the process reduced the number of function evaluations by 81.1% over the GA and 98.5% over the purely discrete DSS approach. Work remains, however, to verify that the hybrid approach’s performance benefits generalize beyond this case study.

## 5.7 Optimization Study 2: Minimum Mission Time

In the second optimization study, the planar quadrotor’s components are selected to minimize the time to complete a dynamic mission. An overview of the mission is provided in Figure 5.53. It consists of five phases A through E. In the first phase, the planar quadrotor moves from rest at point 0 to the opening at point 1. In phase B, it continues through the opening to come to a complete stop at point 2. In phase C, the planar quadrotor moves from point 2 to pass through point 3. In phase D, it continues through the opening at point 4. Finally, in phase E, the planar quadrotor continues through the opening and comes to a complete stop at point 5.



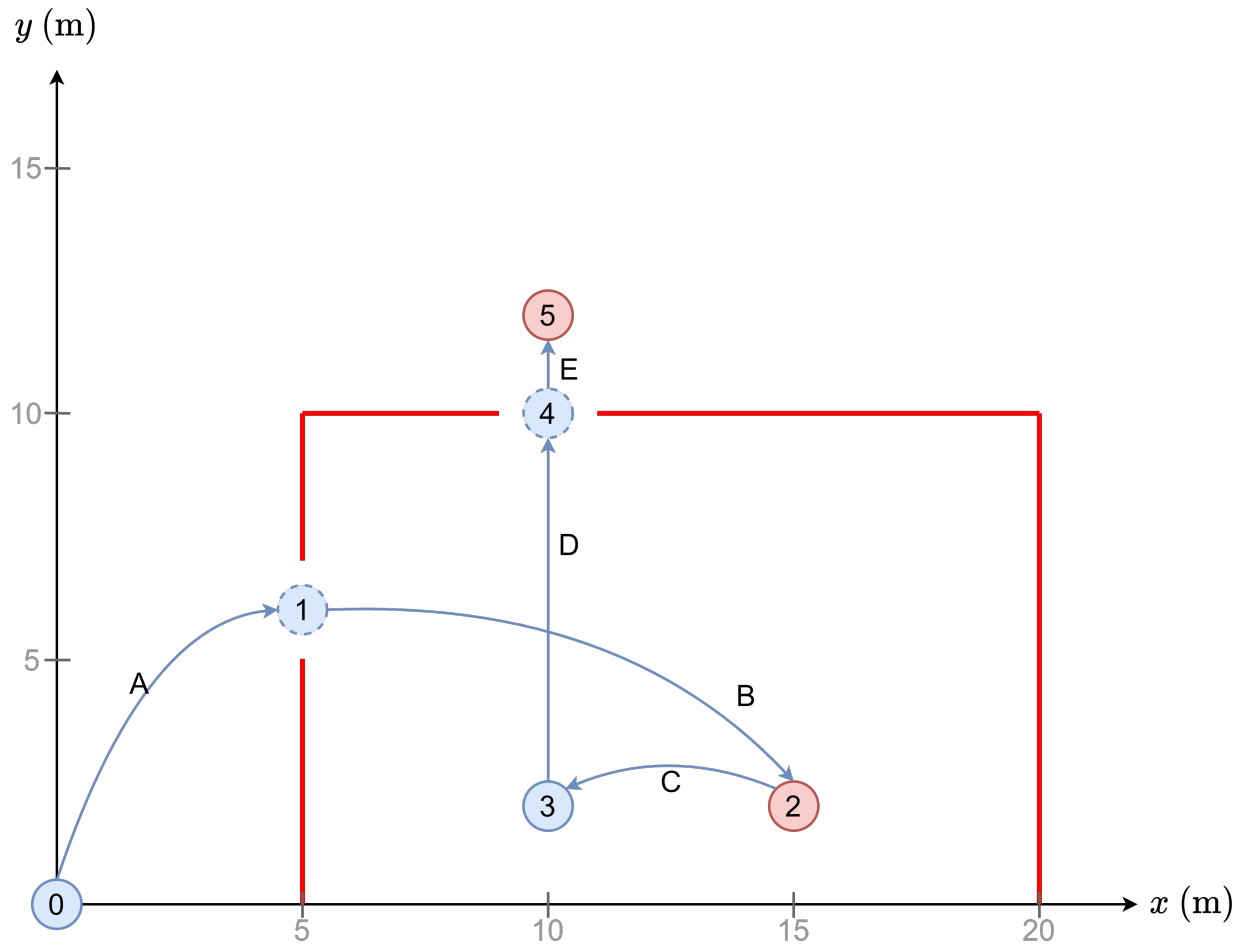


Figure 5.53: Planar quadrotor dynamic mission

This mission is defined by applying mission or continuity constraints to the beginning or end of each phase. Mission constraints ensure the planar quadrotor's states satisfy the mission criteria, while continuity constraints ensure that transitions between phases remain smooth. Table 5.22 specifies the constraints applied to the beginning and end of each phase.

Phase	Initial	Final
A	$\xi_{A,i}^B = 0$	$\xi_{A,f}^B = \xi_{B,i}^B$
B	$x_{B,i} = 5$ $5 \leq y_{B,i} \leq 7$	$\xi_{B,f}^B = \xi_{C,i}^B$
C	$x_{C,i} = 15$ $y_{C,i} = 2$ $\theta_{C,i} = \dot{x}_{C,i} = \dot{y}_{C,i} = \dot{\theta}_{C,i} = 0$	$\xi_{C,f}^B = \xi_{D,i}^B$
D	$x_{D,i} = 10$ $y_{D,i} = 2$	$\xi_{D,f}^B = \xi_{E,i}^B$
E	$9 \leq x_{E,i} \leq 11$ $y_{E,i} = 10$	$x_{E,i} = 10$ $y_{E,i} = 12$ $\theta_{C,i} = \dot{x}_{C,i} = \dot{y}_{C,i} = \dot{\theta}_{C,i} = 0$

Table 5.22: Planar quadrotor mission constraints

In Table 5.22,  $\xi^B = [x \ y \ \theta \ \dot{x} \ \dot{y} \ \dot{\theta}]^\top$  is a vector of the planar quadrotor's body states. States  $x$  and  $\dot{x}$  are displacement and velocity, respectively, along the  $x$ -axis;  $y$  and  $\dot{y}$  are displacement and velocity, respectively, along the  $y$ -axis; and  $\theta$  and  $\dot{\theta}$  are angular displacement and velocity, respectively. Subscripts denote the phase followed by  $i$  for the beginning of the phase and  $f$  for the end of the phase.

### 5.7.1 Discrete-Domain Formulation

The discrete formulation of this dynamic CBDO problem is given in Problem 5.84.

$$\begin{aligned}
 & \underset{\mathbf{i} = [i_B, i_M, i_P], \mathbf{u}(t), \xi(t)}{\text{minimize}} && t_{E,f} \\
 & \text{subject to} && 1 \leq i_B \leq 33, \\
 & && 1 \leq i_M \leq 27, \\
 & && 1 \leq i_P \leq 90, \\
 & && \phi_D^P(i_P) \leq \phi_{D,\max}^P, \\
 & && 0 \leq \mathbf{u}(t) \leq 1 \forall t, \\
 & && i^I(t) \leq i_{\max}^I \forall t, \\
 & && -\pi/2 \leq \theta(t) \leq \pi/2, \\
 & && 0 \geq \mathbf{g}_M(\xi(t)), \\
 & && 0 = \mathbf{h}_M(\xi(t))
 \end{aligned} \tag{5.84}$$

The inverter input  $\mathbf{u}(t)$  is optimized directly in this study, representing a system in the early design stages before a control architecture has been specified. The state vector  $\xi(t)$  contains the dynamic states of the planar quadrotor's powertrain and body model. The objective is to minimize  $t_{E,f}$ , the final time of phase E and the time at which the mission is complete. Static constraints on the selection indices and propeller diameter are retained from Problem 5.78, but the constraints on the inverter input and inverter current now apply to time-varying signals. The planar quadrotor's angular displacement is constrained such that  $|\theta| \leq \pi/2$  rad. Finally,  $\mathbf{g}_M$  and  $\mathbf{h}_M$  contain the mission and continuity constraints defined in Table 5.22. Note that the battery current constraint is omitted in this formulation. In a dynamic mission, quick accelerations produce large current spikes over short time durations. A battery's continuous current rating and maximum current rating can differ substantially. The Turnigy Graphene series, for instance, has a constant discharge rating of 75C and a peak discharge rating of 150C for up to three seconds. If the battery current trajectory contains spikes that exceed the continuous current limit, it may well remain within safe operating conditions. There is, however, a well-known trade-off between discharge rate and battery reliability. Frequent discharges at high rates can reduce the battery's lifetime. Instead of attempting to formulate these soft limits and trade-offs as constraints, we leave it to the designer to verify that the battery current profile falls within appropriate limits to balance safety, reliability, and performance.

## 5.7.2 Continuous-Domain Analysis and Solution

Problem 5.84 was first translated into a continuous-domain representation as in Optimization Study 1. The same design parameters, parameter surrogates, and boundary functions were employed. The continuous-domain representation was then implemented in the Dymos/OpenMDAO framework for analysis. Dymos’ ‘GaussLobatto’ transcription method with 25 segments per phase was used to transcribe the infinite-dimensional problem into an NLP. The optimization algorithm chosen for this study was Scipy’s Sequential Least Squares Programming (SLSQP) algorithm, originally implemented by Dieter Kraft [84]. To improve solution efficiency, analytical derivatives were specified via Dymos and OpenMDAO’s derivative framework.

A sensitivity analysis was performed for the minimum mission time objective using finite differencing. In this analysis, the continuous representation was first solved with design parameter values fixed at the initial design point. With fixed design parameters, the problem becomes an optimal control problem. Solution of the optimal control problem at the initial design point resulted in the nominal optimal control signal, state trajectories, and objective function value. Then, for each element in the design parameter vector, the parameter’s value was increased by 1% and the optimal control problem was solved to obtain the perturbed objective function value. Equation 2.3 was applied to estimate the derivative, and Equation 5.81 was applied to determine the scaled gradient values. These results are given in Table 5.23.

Parameter	Value (units)	Gradient	Scaled Gradient
Propeller Diameter $\phi_D^P$	2.29e-1 (m)	-2.78e1	-1.02
Propeller Pitch $\phi_P^P$	1.14e-1 (m)	-5.85	-1.08e-1
Battery Series Cells $\phi_{N_s}^B$	4	-9.76e-1	-6.28e-1
Battery Capacity $\phi_Q^B$	4.00e3 (mAh)	4.93e-4	3.17e-1
Motor Speed Constant $\phi_{kV}^M$	9.65e2 (RPM/V)	-3.33e-3	-5.18e-1
Motor Winding Resistance $\phi_{R_m}^M$	1.02e-1 ( $\Omega$ )	3.20	5.25e-2

Table 5.23: Sensitivity analysis for minimum mission time objective

As before, the objective is most sensitive to propeller diameter. The analysis indicates that a larger, more aggressive propeller with a higher pitch will decrease the minimum mission time. Also, a higher voltage but lower capacity battery is favored to achieve high thrust with minimal additional mass. At this point in the design space, increasing the motor speed constant while decreasing the winding resistance would better match the motor to the propeller and reduce winding losses.

Equation 5.83 was then used to calculate the component-level sensitivities, provided in Table 5.24. As in the previous optimization study, the objective is most sensitive to the propeller, followed by the battery and the motor.

Component	Sensitivity
Propeller	1.13
Battery	$9.45e-1$
Motor	$5.70e-1$

Table 5.24: Objective’s sensitivity to components

Following the sensitivity analysis, the continuous representation was solved with all six of the design parameters included as optimization variables. For computational efficiency, state and control trajectories were initialized to the values found from solving the optimal control problem. Over 437 iterations, the SLSQP algorithm converged to the point indicated by the green marker in Figures 5.54 - 5.56. In the figures, the blue marker represents the continuous solution to the endurance-per-price objective, for reference.

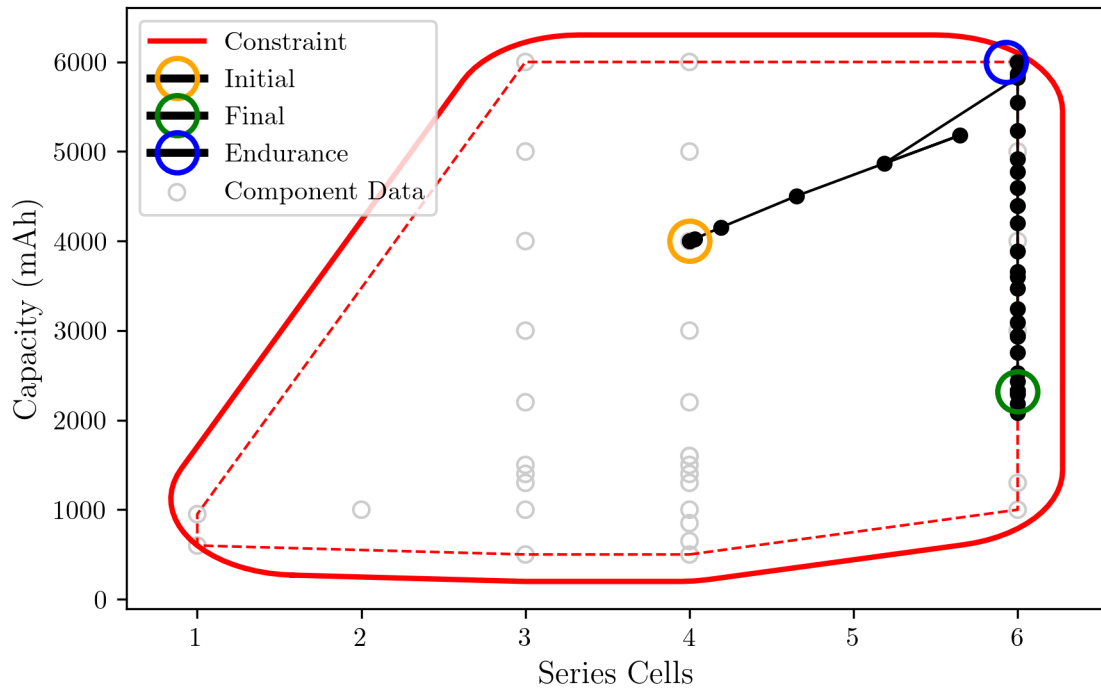


Figure 5.54: Solution of continuous-representation - battery design space

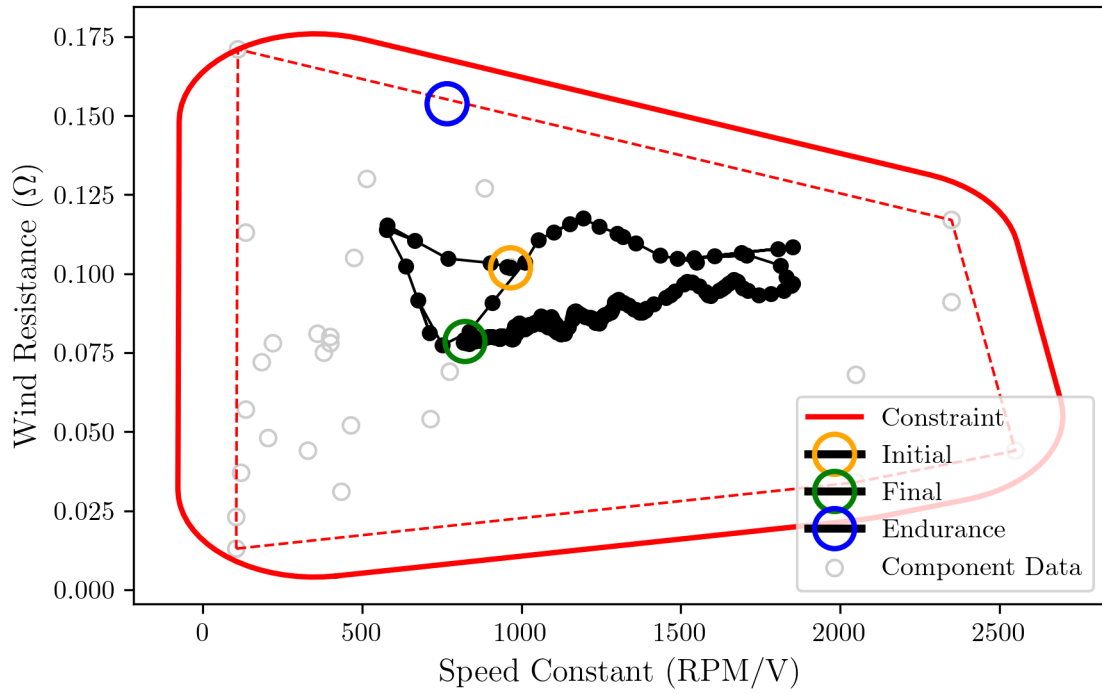


Figure 5.55: Solution of continuous-representation - motor design space

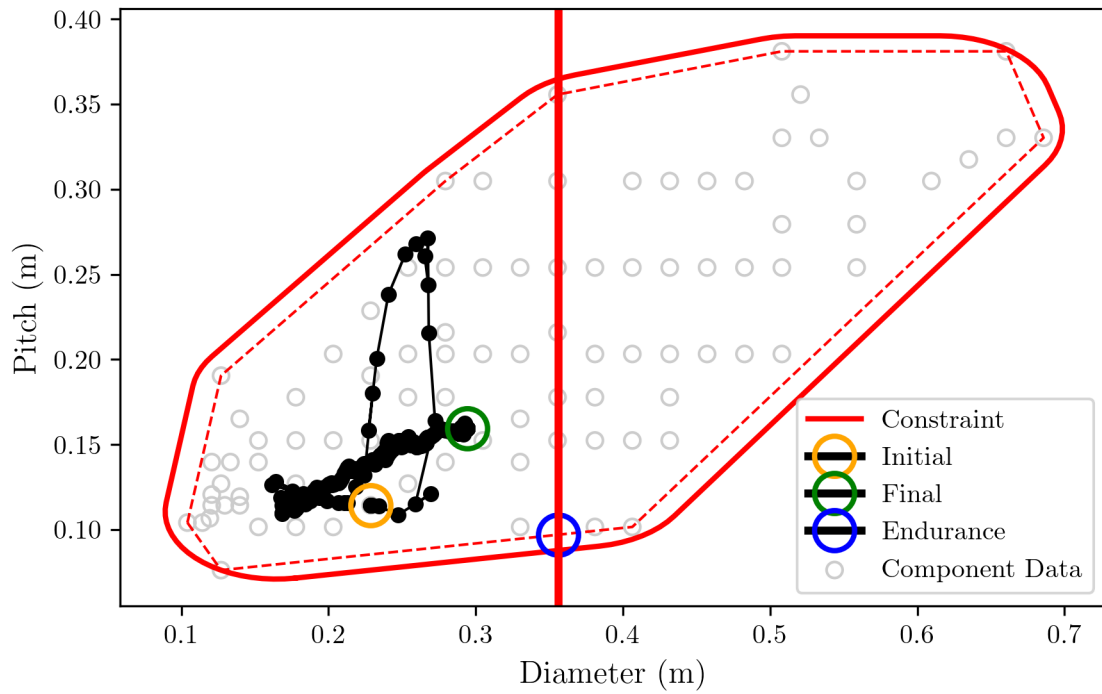


Figure 5.56: Solution of continuous-representation - propeller design space

In this optimization, the route from the initial design to final design is more circuitous, reflecting the increased complexity of the problem and the impact of component coupling. This coupling is made clear in the motor and propeller design space plots. In practice, high-torque motors with a low speed constant are paired with large, aggressive propellers, and motors with a high speed constant are paired with smaller propellers. This relationship can be seen in their paths through the design space. When the optimization evaluates a propeller with low diameter and pitch, the motor’s speed constant increases.

Though the SLSQP algorithm does not return a first-order optimality metric, the termination message indicated that it converged successfully. Specifically, it met the function tolerance value of  $1.00\text{e-}6$  [85]. The upper limit on the number of battery series cells is active at the solution, though Lagrange multiplier values are not returned by this algorithm. Table 5.25 provides the optimal design parameter values in comparison with the initial design.

Parameter	Unit	Optimal Value	Initial Value	% Change
Propeller Diameter $\phi_D^P$	m	$2.94\text{e-}1$	$2.29\text{e-}1$	28.79 %
Propeller Pitch $\phi_P^P$	m	$1.60\text{e-}1$	$1.14\text{e-}1$	39.56 %
Battery Series Cells $\phi_{N_s}^B$	unit	6	4	50.00 %
Battery Capacity $\phi_Q^B$	mAh	$2.32\text{e}3$	$4.00\text{e}3$	-42.06 %
Motor Speed Constant $\phi_{kV}^M$	RPM/V	$8.21\text{e}2$	$9.65\text{e}2$	-14.90 %
Motor Winding Resistance $\phi_{R_m}^M$	$\Omega$	$7.86\text{e-}2$	$1.02\text{e-}1$	-22.97 %

Table 5.25: Design parameter values that minimize the continuous representation of Problem 5.84

### 5.7.3 Discrete Problem Solution

Using the design parameter values in Table 5.25 as the target, the DSS algorithm was run for 100 iterations. Compared to the previous case study, the fewer search iterations reflect the increased computational expense of evaluating the dynamic objective. Though there is no guarantee of locating the globally optimal design in the first 100 configurations, the hybrid approach focuses the search on a region of the design space with high-potential configurations. The process of evaluating a configuration for the dynamic objective was as follows. First, parameters from the configuration were substituted into the system model, and dynamic trajectories were initialized to the solution of the optimal control problem. Then, the model’s steady-state condition was evaluated to calculate the thrust ratio. If the calculated thrust ratio was less than one, then the configuration was rejected as infeasible. Otherwise, then the optimal control problem was solved with the new parameter values and

Component	Make	Model	SKU	Mass (kg)	Price (USD)
Battery	Turnigy	Graphene Panther	9067000513-0	2.49e-1	\$30.99
PMSMMotor	KDE	KDE2814XF-775	N/A	9.50e-2	\$74.95
Propeller	APC	10x7E	LP10070E	2.00e-2	\$3.21

Table 5.26: Best configuration found for mission duration objective

the objective function value corresponding to the configuration was recorded. Figure 5.57 plots the objective function value and distance metrics for the 100 configurations evaluated. An increasing trend in the objective function value is less apparent than in the previous case study, indicating that distance from the continuous solution is a less effective predictor for this objective. The variation in objective function values, however, increases considerably as worse designs are found further from the target.

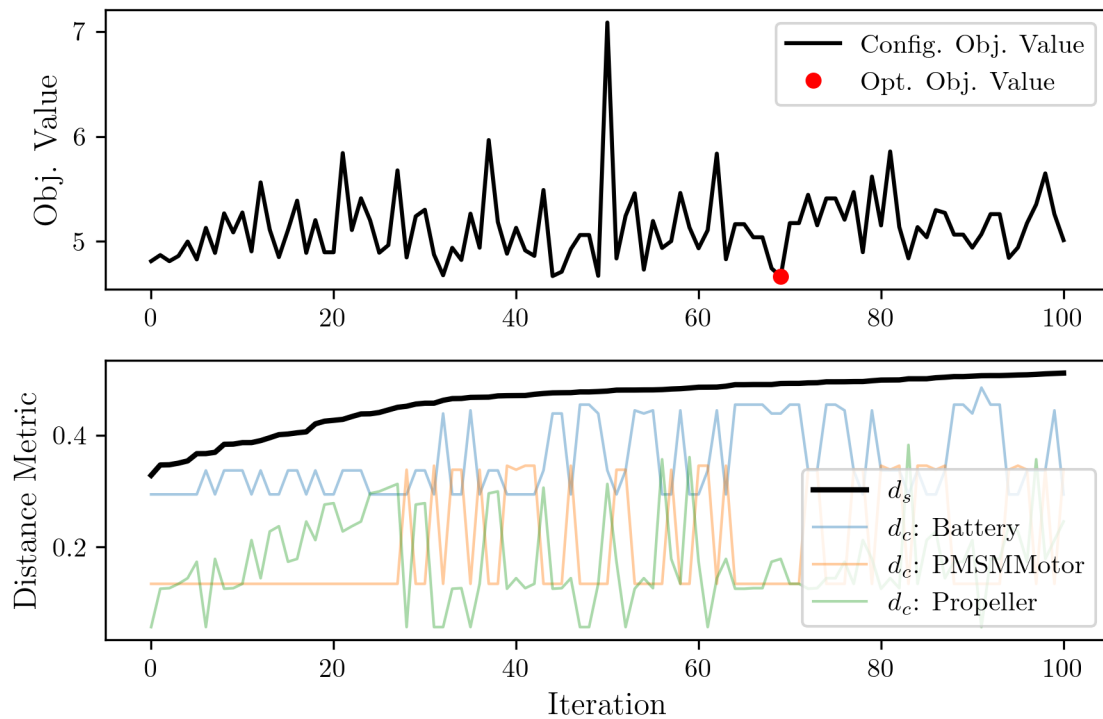


Figure 5.57: First 100 iterations of DSS algorithm

The best configuration was found on the 69<sup>th</sup> search iteration. It is specified in Table 5.26, along with corresponding design parameter values in Table 5.27.



Parameter	Unit	Continuous Sol.	Discrete Sol.	Initial	% Change
Propeller Diameter $\phi_D^P$	m	2.94e-1	2.54e-1	2.29e-1	11.11 %
Propeller Pitch $\phi_P^P$	m	1.60e-1	1.78e-1	1.14e-1	55.56 %
Battery Series Cells $\phi_{N_s}^B$	unit	6	6	4	50.00 %
Battery Capacity $\phi_Q^B$	mAh	2.32e3	1.30e3	4.00e3	-67.50 %
Motor Speed Constant $\phi_{kV}^M$	RPM/V	8.21e2	7.75e2	9.65e2	-19.69 %
Motor Winding Resistance $\phi_{R_m}^M$	$\Omega$	7.86e-2	6.90e-2	1.02e-1	-32.35 %

Table 5.27: Design parameter values of best configuration for mission duration objective

Figures 5.58-5.60 plot the best configuration and target design in the design space. The green ‘Opt. Config’ marker indicates the optimal configuration for the mission time objective, while the blue ‘Endurance’ marker indicates the optimal configuration for the endurance-per-price objective. Finally, the fill color of the ‘Components’ markers indicates the component distance metric, with dark values corresponding to a small distance. Note the design differences between the two objectives. Compared with the static objective, the dynamic objective benefits from a battery with a lower capacity, a motor with a higher speed constant and a lower winding resistance, and a propeller with a smaller diameter and a larger pitch.

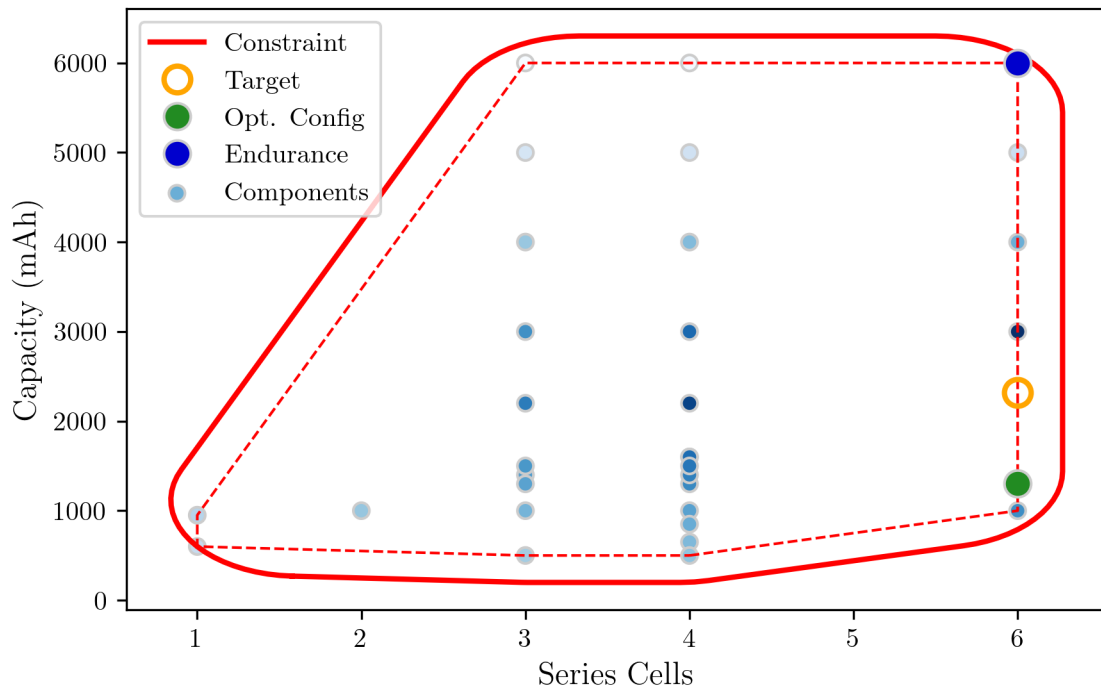


Figure 5.58: Optimal battery for mission duration objective

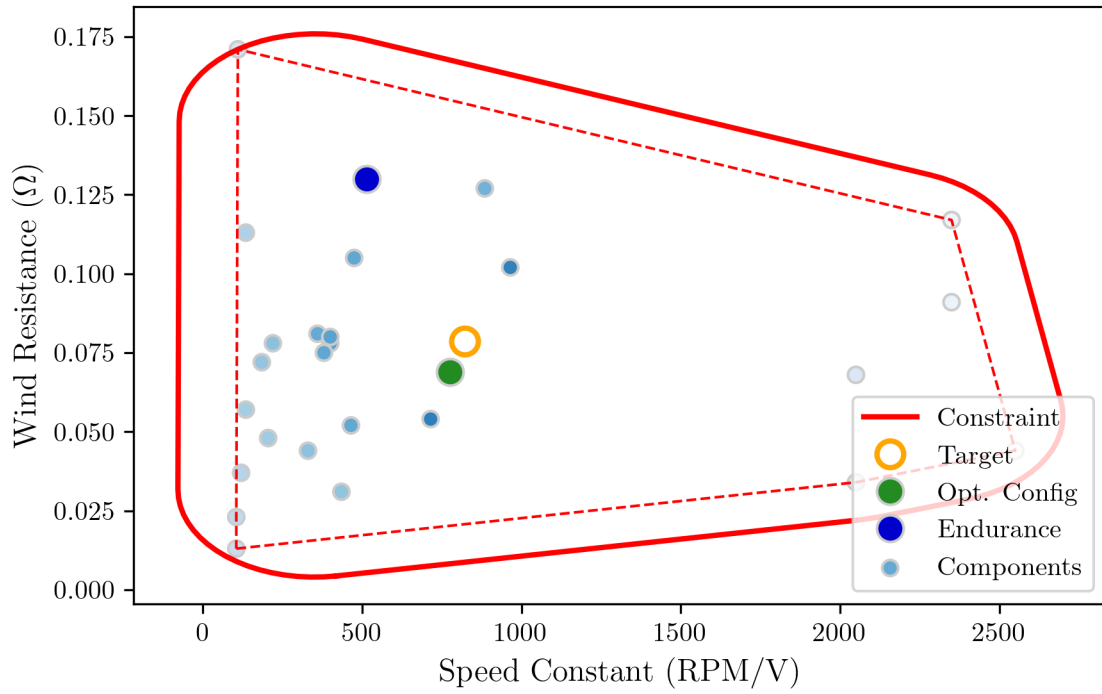


Figure 5.59: Optimal motor for mission time objective

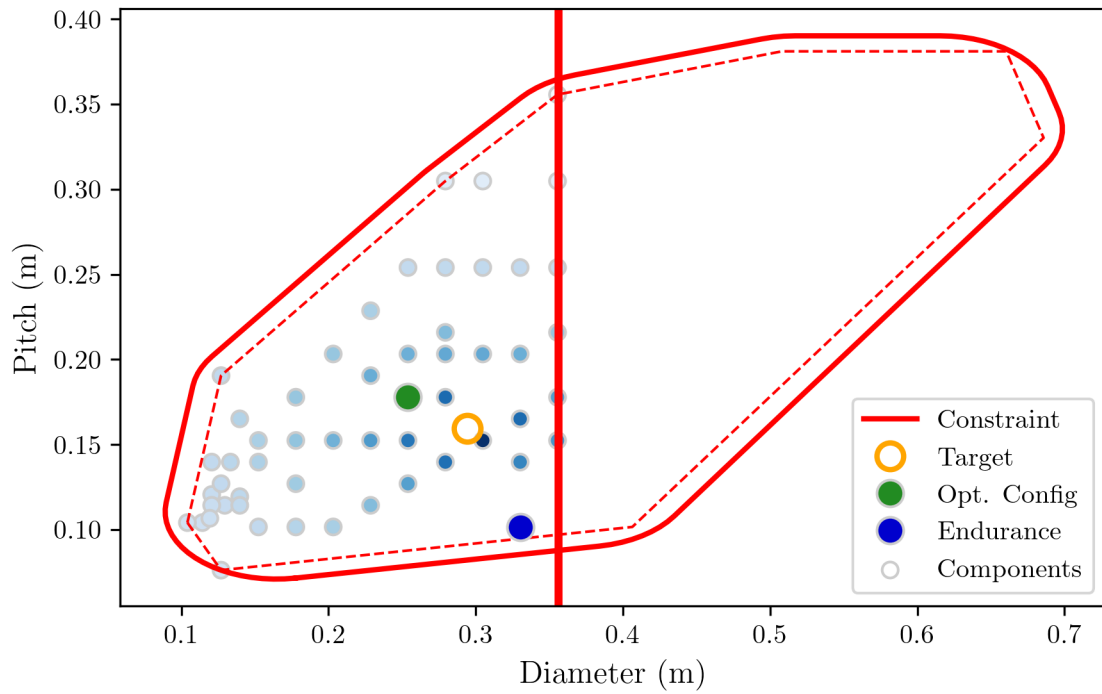


Figure 5.60: Optimal propeller for mission time objective

### 5.7.4 Discussion of Optimized Design

The selected configuration reduces the mission duration by 25.64%, as summarized in Table 5.28.

Mission Duration	
Initial	6.275
Final	4.666
Change	-25.64%

Table 5.28: Mission duration objective for initial and final designs

Figure 5.61 compares the horizontal displacement  $x$ , vertical displacement  $y$ , and angular displacement  $\theta$  trajectories of the initial design with that of the optimized design. In the figure, the vertical dashed lines represent phase boundaries. The optimized design is capable of far more aggressive accelerations, which are especially apparent in the initial takeoff phase.

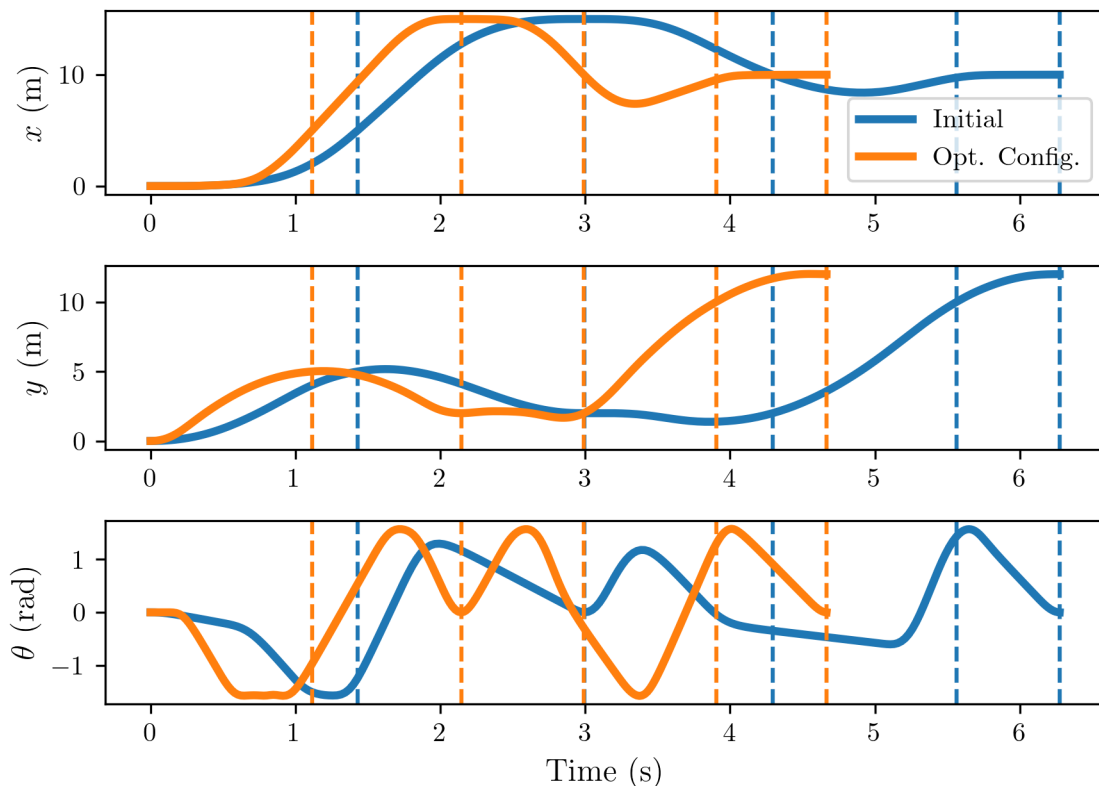


Figure 5.61: Body state trajectories of initial and optimal configurations

Figure 5.62 compares the battery SOC, DC bus voltage, and DC bus current trajectories

of the initial and optimized designs. Increasing the number of battery series cells from four to six considerably increases the bus voltage. This increased voltage, along with the more aggressive maneuvers, create larger spikes in the current profile of the optimized design. Many of these spikes exceed the battery’s continuous discharge limit of 97.5A, though they are safely within the peak discharge limit of 195A sustainable for three seconds. Increased power consumption and less battery pack energy results in a steeper SOC decline.

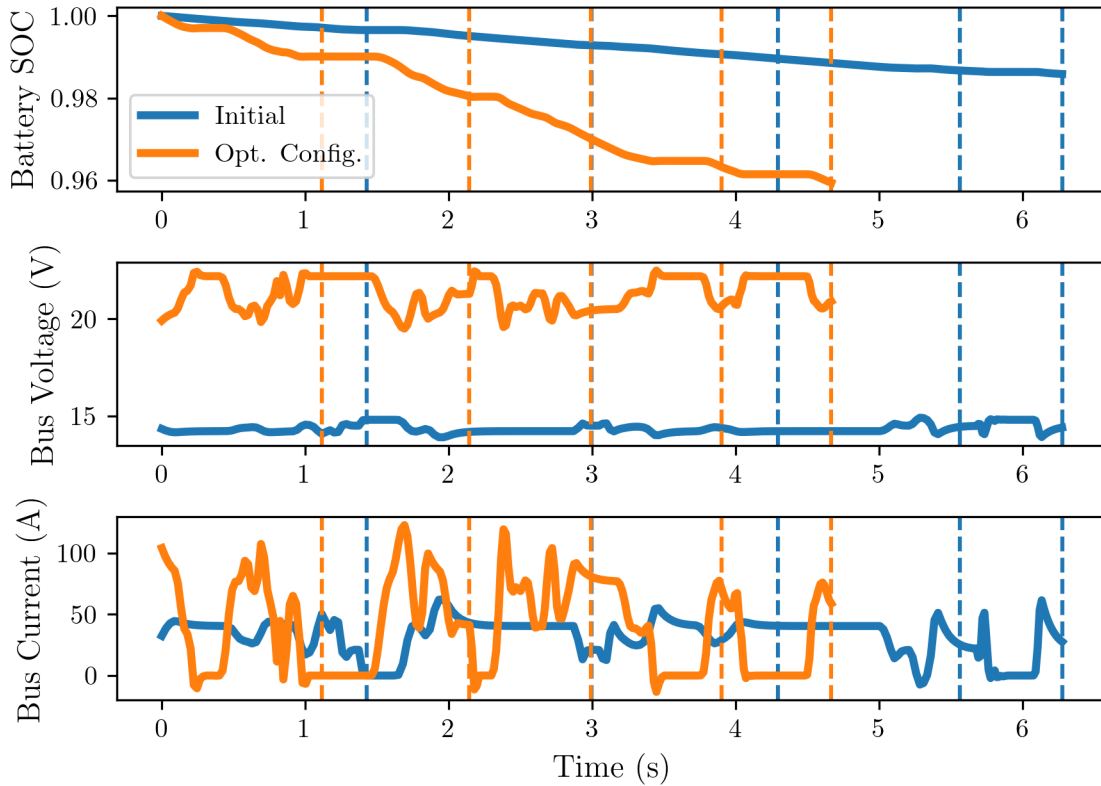


Figure 5.62: Powertrain state trajectories of initial and optimal configurations

The performance differences are largely attributable to an increase in thrust ratio from 1.85 to 5.31. The mass and maximum thrust of the initial design is 0.92 kg and 16.73 N, respectively, while that of the optimized design is 0.73 kg and 38.06 N, respectively. Key to achieving a high thrust ratio is a high effective system voltage. First, a battery with the maximum of six series cells was selected. Battery capacity was selected to balance mass with series resistance. Low-capacity packs have higher series resistance, leading to a decrease in battery terminal voltage under load. Additional mass was also incurred in selecting a motor with low winding resistance, which allows more current through the motor windings. Overall, the minimization of mass is less critical than intuition would suggest.

### Takeaway

For dynamic performance, minimization of mass was less critical than increasing powertrain voltage to achieve a high thrust ratio.

The reader may wonder why, if the best motor is near the target, the best propeller is relatively far from the target. This can be explained by errors in the propeller surrogate model, shown in Figures 5.37 and 5.39. At the point selected in the discrete search, the parameter surrogates over-predict the power coefficient by 11.3% and under-predict the torque coefficient by 6.1%. Effectively, the search discovered a propeller that required less torque and produces more thrust for a given speed. This highlights the value of the discrete search step in the hybrid approach, since the configuration closest to the target is sub-optimal.

### 5.7.5 Algorithm Performance

The solution of the optimal control problem, used in the sensitivity analysis and to initialize subsequent optimizations, required 551 function evaluations. The continuous representation of Problem 5.84 required 477 function evaluations. Finally, the discrete search required a total of 29,519 function evaluations to evaluate 100 configurations. The best configuration was found on the 69<sup>th</sup> iteration, with 20,980 function evaluations accumulated up to that point. Therefore, the total cost of this hybrid optimization process was 30,547 function evaluations.

# Chapter 6

## Conclusion

### 6.1 Summary of Contributions

Engineered systems have become far more complex in the pursuit of performance and efficiency. Companies and their engineers have responded to increased system complexity by becoming more specialized, with many different actors contributing small parts to the design of the larger system. A high degree of coordination is required to ensure each design decision is made in service of the high-level system objective. Simultaneously, engineering companies have embraced component-based design practices to efficiently bring more flexible systems to market. These trends create a need for component-based design optimization tools that assist designers in selecting appropriate components for integration into systems and subsystems. When the system's dynamic performance is of value, then the design optimization tool must also be able to accommodate system dynamics within the formulation.

Two solution paths for CBDO problems currently exist in the literature: direct approaches and parameterization-based approaches. Direct approaches solve the discrete selection problem using discrete optimization algorithms such as the genetic algorithm. Though this is a natural solution path, general-purpose discrete algorithms scale poorly and are less efficient than gradient-based algorithms. Furthermore, discrete approaches limit design insight and intuition that can be gained via sensitivity studies and other continuous-domain analyses. The alternative is parameterization-based approaches that transform the discrete problem into a continuous problem. Practitioners of this approach, however, often stop once they have obtained a solution to the continuous-representation of the problem. Due to component coupling, sparsity in the component databases, and errors in the fitting models, manually selecting components based on the continuous solution may produce a sub-optimal configuration.

This thesis presents an indirect hybrid approach that integrates ideas from both paths. It

transforms the discrete problem into a continuous representation via component parameterization and regression models, solves the continuous representation using efficient, gradient-based algorithms, and uses the continuous-domain information to search for a discrete solution to the original problem. To facilitate the final step, a distance-sorted search algorithm was developed that focuses the search on components most similar to the continuous solution.

Two case studies were conducted to demonstrate the hybrid approach. The first was a ‘static’ study of a quadrotor system, and the second was a ‘dynamic’ study of a planar quadrotor system. Component models were developed using graph-based techniques, which are especially well-suited for optimization applications, and assembled into the system models using automated algorithms. Accuracy of the quadrotor’s steady-state model was verified with an experimental test platform. In both case studies, batteries, motors, and propellers were selected from a database of commercially available components. To facilitate the hybrid approach, components in the databases were parameterized and parameter surrogates were created with regression analysis.

The objective of the ‘static’ study was to maximize endurance per system price. A sensitivity analysis of the objective determined that the propeller had the largest impact and indicated ways in which the design could be improved. The hybrid approach was then applied to the problem. It identified a design with a large-diameter, small-pitch propeller; a high-capacity, high-voltage battery; and a motor with a lower speed constant and higher winding resistance than that of the initial design. These design changes improved the endurance-per-price objective by 76.5%. To obtain a baseline for algorithm performance, the CBDO problem was solved directly using a genetic algorithm. The hybrid approach was found to outperform the GA for this case study, requiring 81.1% fewer function evaluations to find the globally optimal configuration.

The objective of the second ‘dynamic’ study was to minimize the time required to complete a dynamic mission. Like in the previous case study, a sensitivity analysis indicated that the dynamic objective was most sensitive to propeller design. The hybrid optimization process identified a propeller with a smaller pitch; a low-capacity, high-voltage battery; and a motor with a lower speed constant and lower winding resistance than the initial design. The selected configuration improved the mission duration objective by 25.64%. This case study demonstrated the significance of the discrete search step, as the best configuration was somewhat far from the solution to the continuous representation.

## 6.2 Future Work

Much work remains to improve the performance of the hybrid approach and verify that it scales to more complex systems and optimization problems. First, additional component and configuration distance metrics should be explored to best leverage the results of the continuous analysis. One possibility is to weight the component-distance metric with sensitivity information obtained at the target design. In this approach, parameters to which the objective is most sensitive would be prioritized. Entirely different sorted-search algorithms should be considered as well. In an objective-sorted search, for instance, components could be evaluated by substituting its parameters into the target design and evaluating the objective function. Configurations would then be sorted by the average of the objective function values associated with its components. Therefore, the first configuration in the list would contain the individually optimal components. If no coupling between components existed, then it would be the optimal configuration.

In addition to improving the algorithms themselves, future work could focus on optimizing their implementation for better performance. All of the software tools developed for this thesis are single threaded. Significant performance benefits could be realized with parallel processing, especially when evaluating the list of configurations in the discrete search. Different NLP optimization algorithms could also be explored, especially if the structure of the continuous representation can be exploited.

Another direction for future work is dynamic CBDO problems that incorporate closed-loop controller design. Though open-loop optimal control provides insight in early design stages, it is rarely used directly in real-world systems. Most dynamic systems incorporate feedback controllers that are tuned to balance performance with robustness. When the physical elements and control system are designed in parallel, a process referred to as Control Co-Design (CCD), synergies between the two can be exploited to achieve superior dynamics and controllability. This often leads to lower system cost and improved reliability [86].

Finally, additional case studies are required to demonstrate that the hybrid approach can scale to meet the real-world design challenges faced by modern engineers. Analyses of larger eVTOL systems with 3D dynamics and more decision variables should be conducted.



# References

- [1] Hagerty Drivers Foundation, director, *The Castle Duesenberg: Luxury Legend - A History of the First Passenger Duesenberg*, Nov. 11, 2020. [Online]. Available: [https://www.youtube.com/watch?v=5KH\\_V0FJkPQ](https://www.youtube.com/watch?v=5KH_V0FJkPQ) (visited on 01/24/2022).
- [2] K. V. Shaw. “1921 Duesenberg Model A Belonged to the Same Family for Almost a Century,” *The Drive*. (Feb. 13, 2021), [Online]. Available: <https://www.thedrive.com/news/39252/1921-duesenberg-model-a-belonged-to-the-same-family-for-almost-a-century> (visited on 01/26/2022).
- [3] J. Peek. “The Castle Duesenberg: One historic car that elevated two family legacies,” *Hagerty Media*. (Nov. 11, 2020), [Online]. Available: <https://www.hagerty.com/media/car-profiles/the-castle-duesenberg-one-historic-car-that-elevated-two-family-legacies/> (visited on 01/26/2022).
- [4] History.com Editors. “Ford’s assembly line starts rolling,” *HISTORY*. (Nov. 30, 2021), [Online]. Available: <https://www.history.com/this-day-in-history/fords-assembly-line-starts-rolling> (visited on 02/03/2022).
- [5] M. International. “A History of the Automotive Assembly Line,” *Mayco International - Automotive tier 1 supplier*. (Apr. 23, 2021), [Online]. Available: <https://maycointernational.com/blog/a-history-of-the-automotive-assembly-line/> (visited on 01/25/2022).
- [6] E. B. White, “Farewell, My Lovely!” *The New Yorker*, May 8, 1936, ISSN: 0028-792X. [Online]. Available: <https://www.newyorker.com/magazine/1936/05/16/farewell-my-lovely> (visited on 01/26/2022).
- [7] *1922 Model T Ford Car Parts Accessories*, 1922. [Online]. Available: <https://www.ebay.com/itm/153317884228> (visited on 02/15/2022).
- [8] A. J. Baime, “Grandson Restores the 1918 Ford Model T His Grandfather Assembled by Hand a Century Ago,” *Wall Street JournalLife*, Sep. 11, 2021, ISSN: 0099-9660. [Online]. Available: <https://www.wsj.com/articles/grandson-restores-the-1918-ford->

- [model-t-his-grandfather-assembled-by-hand-a-century-ago-11631368802](#) (visited on 01/26/2022).
- [9] Boy Scouts of America, *Boys' Life*. Boy Scouts of America, Inc., Mar. 1956, 80 pp. Google Books: [2RWgSgxSs34C](#).
- [10] N. Wakelin. "How Many Parts are in a Car?" NAPA Know How Blog. (Jul. 2, 2021), [Online]. Available: <https://knowhow.napaonline.com/how-many-parts-are-in-a-car/> (visited on 02/04/2022).
- [11] D. Silver. "The Automotive Supply Chain, Explained," Self-Driving Cars. (May 31, 2016), [Online]. Available: <https://medium.com/self-driving-cars/the-automotive-supply-chain-explained-d4e74250106f> (visited on 02/03/2022).
- [12] E. A. Lee and A. L. Sangiovanni-Vincentelli, "Component-based design for the future," in *2011 Design, Automation Test in Europe*, Mar. 2011, pp. 1–5. DOI: [10.1109/DATE.2011.5763168](https://doi.org/10.1109/DATE.2011.5763168).
- [13] K. Fowler, "Build versus Buy," *IEEE Instrumentation Measurement Magazine*, vol. 7, no. 3, pp. 67–73, Sep. 2004, ISSN: 1941-0123. DOI: [10.1109/MIM.2004.1337916](https://doi.org/10.1109/MIM.2004.1337916).
- [14] Y. Sered and Y. Reich, "Standardization and modularization driven by minimizing overall process effort," *Computer-Aided Design*, vol. 38, no. 5, pp. 405–416, May 1, 2006, ISSN: 0010-4485. DOI: [10.1016/j.cad.2005.11.005](https://doi.org/10.1016/j.cad.2005.11.005). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010448505001910> (visited on 09/07/2021).
- [15] D.-Y. Kim, J.-W. Park, S. Baek, *et al.*, "A modular factory testbed for the rapid reconfiguration of manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 661–680, Mar. 1, 2020, ISSN: 1572-8145. DOI: [10.1007/s10845-019-01471-2](https://doi.org/10.1007/s10845-019-01471-2). [Online]. Available: <https://doi.org/10.1007/s10845-019-01471-2> (visited on 02/08/2022).
- [16] J. R. R. A. Martins and S. A. Ning, *Engineering Design Optimization*. Cambridge ; New York, NY: Cambridge University Press, 2021, 1 p., ISBN: 978-1-108-83341-7.
- [17] V. Belle and I. Papantonis, "Principles and Practice of Explainable Machine Learning," *Frontiers in Big Data*, vol. 4, 2021, ISSN: 2624-909X. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fdata.2021.688969> (visited on 02/12/2022).

- [18] J. Pande, C. J. Garcia, and D. Pant, "Optimal component selection for component based software development using pliability metric," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 1, pp. 1–6, Jan. 23, 2013, ISSN: 0163-5948. DOI: [10.1145/2413038.2413044](https://doi.org/10.1145/2413038.2413044). [Online]. Available: <https://doi.org/10.1145/2413038.2413044> (visited on 09/06/2021).
- [19] L. Gesellensetter and S. Glesner, "Only the Best Can Make It: Optimal Component Selection," *Electronic Notes in Theoretical Computer Science*, Proceedings of the Workshop on Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA 2006), vol. 176, no. 2, pp. 105–124, May 31, 2007, ISSN: 1571-0661. DOI: [10.1016/j.entcs.2006.02.034](https://doi.org/10.1016/j.entcs.2006.02.034). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1571066107002071> (visited on 09/06/2021).
- [20] P. C. Jha, V. Bali, S. Narula, and M. Kalra, "Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme," *Journal of Computational Science*, Empowering Science through Computing + BioInspired Computing, vol. 5, no. 2, pp. 233–242, Mar. 1, 2014, ISSN: 1877-7503. DOI: [10.1016/j.jocs.2013.07.003](https://doi.org/10.1016/j.jocs.2013.07.003). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877750313000835> (visited on 09/06/2021).
- [21] W. Pawlus, G. Hovland, M. Choux, D. Frick, and M. Morari, "Drivetrain design optimization for electrically actuated systems via mixed integer programming," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2015, pp. 001 465–001 470. DOI: [10.1109/IECON.2015.7392307](https://doi.org/10.1109/IECON.2015.7392307).
- [22] H. van de Straete, P. Degezelle, J. De Schutter, and R. Belmans, "Servo motor selection criterion for mechatronic applications," *IEEE/ASME Transactions on Mechatronics*, vol. 3, no. 1, pp. 43–50, Mar. 1998, ISSN: 1941-014X. DOI: [10.1109/3516.662867](https://doi.org/10.1109/3516.662867).
- [23] P. Chedmail and M. Gautier, "Optimum Choice of Robot Actuators," *Journal of Engineering for Industry*, vol. 112, no. 4, pp. 361–367, Nov. 1, 1990, ISSN: 0022-0817. DOI: [10.1115/1.2899600](https://doi.org/10.1115/1.2899600). [Online]. Available: <https://doi.org/10.1115/1.2899600> (visited on 09/06/2021).
- [24] M. Pettersson, "Design Optimization in Industrial Robotics," p. 81,
- [25] M. Tarkian, J. Persson, J. O'lvander, and X. Feng, "Multidisciplinary Design Optimization of Modular Industrial Robots," presented at the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection,

- Jun. 12, 2012, pp. 867–876. DOI: [10.1115/DETC2011-48196](https://doi.org/10.1115/DETC2011-48196). [Online]. Available: <https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2011/54822/867/353947> (visited on 09/03/2021).
- [26] E. P. Zafiropoulos and E. N. Dyalynas, “Methodology for the optimal component selection of electronic devices under reliability and cost constraints,” *Quality and Reliability Engineering International*, vol. 23, no. 8, pp. 885–897, 2007, ISSN: 1099-1638. DOI: [10.1002/qre.850](https://doi.org/10.1002/qre.850). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.850> (visited on 09/03/2021).
- [27] M. Sauppe, T. Horn, E. Markert, U. Heinkel, H.-W. Sahm, and K.-H. Otto, “Optimal component selection for energy-efficient systems,” in *Proceedings of the 2013 Forum on Specification and Design Languages (FDL)*, Sep. 2013, pp. 1–8.
- [28] A. M. Patel and S. K. Singal, “Optimal component selection of integrated renewable energy system for power generation in stand-alone applications,” *Energy*, vol. 175, pp. 481–504, May 15, 2019, ISSN: 0360-5442. DOI: [10.1016/j.energy.2019.03.055](https://doi.org/10.1016/j.energy.2019.03.055). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360544219304591> (visited on 09/06/2021).
- [29] E. Papadopoulos and L. Davliakos, “A Systematic Methodology for Optimal Component Selection of Electrohydraulic Servosystems,” *International Journal of Fluid Power*, vol. 5, no. 3, pp. 15–24, Jan. 1, 2004, ISSN: 1439-9776. DOI: [10.1080/14399776.2004.10781198](https://doi.org/10.1080/14399776.2004.10781198). [Online]. Available: <https://doi.org/10.1080/14399776.2004.10781198> (visited on 09/03/2021).
- [30] T. T. H. Ng and G. S. B. Leng, “Design of small-scale quadrotor unmanned air vehicles using genetic algorithms,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 221, no. 5, pp. 893–905, May 1, 2007, ISSN: 0954-4100. DOI: [10.1243/09544100JAER0113](https://doi.org/10.1243/09544100JAER0113). [Online]. Available: <https://doi.org/10.1243/09544100JAER0113> (visited on 07/17/2021).
- [31] V. M. Arellano-Quintana, E. A. Portilla-Flores, E. A. Merchan-Cruz, and P. A. Niño-Suarez, “Multirotor design optimization using a genetic algorithm,” in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2016, pp. 1313–1318. DOI: [10.1109/ICUAS.2016.7502564](https://doi.org/10.1109/ICUAS.2016.7502564).
- [32] Ø. Magnussen, G. Hovland, and M. Ottestad, “Multicopter UAV design optimization,” in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Sep. 2014, pp. 1–6. DOI: [10.1109/MESA.2014.6935598](https://doi.org/10.1109/MESA.2014.6935598).

- [33] X. Dai, Q. Quan, J. Ren, and K. Cai, “An Analytical Design-Optimization Method for Electric Propulsion Systems of Multicopter UAVs With Desired Hovering Endurance,” *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 1, pp. 228–239, Feb. 2019, ISSN: 1941-014X. DOI: [10.1109/TMECH.2019.2890901](https://doi.org/10.1109/TMECH.2019.2890901).
- [34] T. Du, A. Schulz, B. Zhu, B. Bickel, and W. Matusik, “Computational multicopter design,” *MIT Web Domain*, Nov. 2016, ISSN: 0730-0301. [Online]. Available: <https://dspace.mit.edu/handle/1721.1/111061> (visited on 02/17/2021).
- [35] F. Tian and M. Voskuijl, “Mechatronic Design and Optimization Using Knowledge-Based Engineering Applied to an Inherently Unstable and Unmanned Aerial Vehicle,” *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 542–554, Feb. 2016, ISSN: 1941-014X. DOI: [10.1109/TMECH.2015.2441832](https://doi.org/10.1109/TMECH.2015.2441832).
- [36] K. Rothfus, “Multirotor Aircraft Genetic Design Algorithm (MAGDA),” M.S. Northern Arizona University, United States – Arizona, May 2020, 153 pp., ISBN: 9798672104089. [Online]. Available: <https://www.proquest.com/docview/2448297018/abstract/E928DF9B116D4688PQ/1> (visited on 07/13/2021).
- [37] D. Bershadsky, S. Haviland, and E. N. Johnson, “Electric Multirotor UAV Propulsion System Sizing for Performance Prediction and Design Optimization,” in *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, ser. AIAA SciTech Forum, 0 vols., American Institute of Aeronautics and Astronautics, Jan. 1, 2016. DOI: [10.2514/6.2016-0581](https://doi.org/10.2514/6.2016-0581). [Online]. Available: <http://arc.aiaa.org/doi/10.2514/6.2016-0581> (visited on 01/23/2021).
- [38] C. Ampatis and E. Papadopoulos, “Parametric design and optimization of multi-rotor aerial vehicles,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6266–6271. DOI: [10.1109/ICRA.2014.6907783](https://doi.org/10.1109/ICRA.2014.6907783).
- [39] O. Gur and A. Rosen, “Optimizing Electric Propulsion Systems for Unmanned Aerial Vehicles,” *Journal of Aircraft*, vol. 46, no. 4, pp. 1340–1353, Jul. 2009, ISSN: 0021-8669, 1533-3868. DOI: [10.2514/1.41027](https://doi.org/10.2514/1.41027). [Online]. Available: <https://arc.aiaa.org/doi/10.2514/1.41027> (visited on 07/13/2021).
- [40] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design: Modeling and Computation*, 3rd. Cambridge: Cambridge University Press, Jan. 8, 2017, ISBN: 978-1-316-45103-8. DOI: [10.1017/9781316451038](https://doi.org/10.1017/9781316451038).
- [41] O. D. Team. “How Total Derivatives are Computed,” OpenMDAO Documentation. (2021), [Online]. Available: [https://openmdao.org/newdocs/versions/latest/theory\\_manual/total\\_derivs\\_theory.html](https://openmdao.org/newdocs/versions/latest/theory_manual/total_derivs_theory.html) (visited on 04/15/2022).

- [42] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, “OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization,” *Structural and Multidisciplinary Optimization*, vol. 59, no. 4, pp. 1075–1104, Apr. 1, 2019, ISSN: 1615-1488. DOI: [10.1007/s00158-019-02211-z](https://doi.org/10.1007/s00158-019-02211-z). [Online]. Available: <https://doi.org/10.1007/s00158-019-02211-z> (visited on 05/24/2022).
- [43] J. T. Hwang and J. R. Martins, “A Computational Architecture for Coupling Heterogeneous Numerical Models and Computing Coupled Derivatives,” *ACM Transactions on Mathematical Software*, vol. 44, no. 4, pp. 1–39, Aug. 9, 2018, ISSN: 0098-3500, 1557-7295. DOI: [10.1145/3182393](https://doi.org/10.1145/3182393). [Online]. Available: <https://dl.acm.org/doi/10.1145/3182393> (visited on 04/15/2022).
- [44] J. P. Koeln, M. A. Williams, H. C. Pangborn, and A. G. Alleyne, “Experimental Validation of Graph-Based Modeling for Thermal Fluid Power Flow Systems,” presented at the ASME 2016 Dynamic Systems and Control Conference, American Society of Mechanical Engineers Digital Collection, Feb. 15, 2017. DOI: [10.1115/DSCC2016-9782](https://doi.org/10.1115/DSCC2016-9782). [Online]. Available: <http://asmedigitalcollection.asme.org/DSCC/proceedings/DSCC2016/50701/V002T21A008/231046> (visited on 10/25/2020).
- [45] J. P. Koeln, M. A. Williams, and A. G. Alleyne, “Hierarchical Control of Multi-Domain Power Flow in Mobile Systems: Part I — Framework Development and Demonstration,” presented at the ASME 2015 Dynamic Systems and Control Conference, American Society of Mechanical Engineers Digital Collection, Jan. 12, 2016. DOI: [10.1115/DSCC2015-9908](https://doi.org/10.1115/DSCC2015-9908). [Online]. Available: <http://asmedigitalcollection.asme.org/DSCC/proceedings/DSCC2015/57243/V001T08A006/228090> (visited on 10/25/2020).
- [46] M. A. Williams, J. P. Koeln, H. C. Pangborn, and A. G. Alleyne, “Dynamical Graph Models of Aircraft Electrical, Thermal, and Turbomachinery Components,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 4, Apr. 1, 2018, ISSN: 0022-0434. DOI: [10.1115/1.4038341](https://doi.org/10.1115/1.4038341). [Online]. Available: <https://asmedigitalcollection.asme.org/dynamicsystems/article/140/4/041013/384614/Dynamical-Graph-Models-of-Aircraft-Electrical> (visited on 10/25/2020).
- [47] 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,” *Journal of Mechanical Design*, vol. 142, no. 9, Sep. 1, 2020, ISSN: 1050-0472. DOI: [10.1115/1.4046066](https://doi.org/10.1115/1.4046066). [Online]. Available: <https://asmedigitalcollection-asme-org.proxy2.library.illinois.edu/mechanicaldesign/article/142/9/091701/1072722/A-Novel-Framework-for-Simultaneous-Topology-and> (visited on 08/26/2020).

- [48] C. Laird, D. Docimo, C. T. Aksland, and A. G. Alleyne, “GRAPH-BASED DESIGN AND CONTROL OPTIMIZATION OF A HYBRID ELECTRICAL ENERGY STORAGE SYSTEM,” presented at the ASME 2020 Dynamic Systems and Control Conference, Pittsburgh, PA, Oct. 4, 2020, p. 9.
- [49] R. Falck, J. Gray, K. Ponnappalli, and T. Wright, “Dymos: A Python package for optimal control of multidisciplinary systems,” *Journal of Open Source Software*, vol. 6, no. 59, p. 2809, Mar. 31, 2021, ISSN: 2475-9066. DOI: [10.21105/joss.02809](https://doi.org/10.21105/joss.02809). [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.02809> (visited on 10/29/2021).
- [50] J. Allison and D. R. Herber, “Multidisciplinary Design Optimization of Dynamic Engineering Systems,” in *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Boston, Massachusetts: American Institute of Aeronautics and Astronautics, Apr. 8, 2013, ISBN: 978-1-62410-223-3. DOI: [10.2514/6.2013-1462](https://doi.org/10.2514/6.2013-1462). [Online]. Available: <http://arc.aiaa.org/doi/10.2514/6.2013-1462> (visited on 09/24/2020).
- [51] W. Guthrie, J. Filliben, and A. Heckert, “Process Modeling,” in *E-Handbook of Statistical Methods*, NIST/SEMATECH, Apr. 2012. [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmd/section1/pmd144.htm> (visited on 06/25/2022).
- [52] N. Dirkx, M. Bosselaar, and T. Oomen, “A Fast Smoothing-Based Algorithm to Generate l-infinity-Norm Constrained Signals for Multivariable Experiment Design,” *IEEE Control Systems Letters*, vol. 6, pp. 1784–1789, 2022, ISSN: 2475-1456. DOI: [10.1109/LCSYS.2021.3133655](https://doi.org/10.1109/LCSYS.2021.3133655).
- [53] S. Gillies, *Shapely*, shapely, Jun. 26, 2022. [Online]. Available: <https://github.com/shapely/shapely> (visited on 06/28/2022).
- [54] N. Gregoire and M. Bouillot. “Hausdorff distance between convex polygons,” CS 507 Computational Geometry. (1998), [Online]. Available: <http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html> (visited on 06/28/2022).
- [55] The MathWorks, Inc. “Find minimum of function using genetic algorithm - MATLAB ga,” MathWorks Help Center. (2022), [Online]. Available: [https://www.mathworks.com/help/gads/ga.html#mw\\_4a8bfdb9-7c4c-4302-8f47-d260b7a43e26](https://www.mathworks.com/help/gads/ga.html#mw_4a8bfdb9-7c4c-4302-8f47-d260b7a43e26) (visited on 06/30/2022).
- [56] HolyBro. “S500 V2 Kit,” Holybro. (), [Online]. Available: <http://www.holybro.com/product/pixhawk4-s500-v2-kit/> (visited on 09/10/2021).

- [57] HolyBro. “PM02 Power Module,” Holybro Store. (), [Online]. Available: <http://www.holybro.com/product/power-modulepm02-v3/> (visited on 05/30/2022).
- [58] HolyBro. “Spare Parts-S500 V2 Kit,” HolyBro Store. (), [Online]. Available: [https://shop.holybro.com/spare-parts-s500-kit\\_p1251.html](https://shop.holybro.com/spare-parts-s500-kit_p1251.html) (visited on 05/30/2022).
- [59] KDE Direct. “KDEXF-UAS20LV 20A+ Electronic Speed Controller (ESC) for Electric Multi-Rotor (UAS) Series,” KDE Direct. (), [Online]. Available: <https://www.kdedirect.com/products/kdexf-uas20lv> (visited on 05/30/2022).
- [60] X. Lin, H. E. Perez, S. Mohan, *et al.*, “A lumped-parameter electro-thermal model for cylindrical batteries,” *Journal of Power Sources*, vol. 257, pp. 1–11, Jul. 1, 2014, ISSN: 0378-7753. DOI: [10.1016/j.jpowsour.2014.01.097](https://doi.org/10.1016/j.jpowsour.2014.01.097). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0378775314001244> (visited on 09/04/2020).
- [61] 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 *2019 American Control Conference (ACC)*, Jul. 2019, pp. 4253–4259. DOI: [10.23919/ACC.2019.8814930](https://doi.org/10.23919/ACC.2019.8814930).
- [62] C. T. Aksland, “Modular Modeling and Control of a Hybrid Unmanned Aerial Vehicle’s Powertrain,” M.S. University of Illinois at Urbana-Champaign, 2019. [Online]. Available: <http://hdl.handle.net/2142/106283>.
- [63] V. M. Bida, D. V. Samokhvalov, and F. S. Al-Mahturi, “PMSM vector control techniques — A survey,” in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2018, pp. 577–581. DOI: [10.1109/EIConRus.2018.8317164](https://doi.org/10.1109/EIConRus.2018.8317164).
- [64] Paul Krause, Oleg Wasynczuk, Scott Sudhoff, and Steven Pekarek, *Analysis of Electric Machinery and Drive Systems*, 1st ed. John Wiley & Sons, Ltd, 2013. DOI: [10.1002/9781118524336](https://doi.org/10.1002/9781118524336). [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/9781118524336> (visited on 09/17/2020).
- [65] Brad Hieb and Heath Hofmann. “Parameterizing and Verifying a Permanent Magnet Synchronous Motor Model,” MathWorks Videos. (2012), [Online]. Available: <https://www.mathworks.com/videos/parameterizing-and-verifying-a-permanent-magnet-synchronous-motor-model-92982.html> (visited on 09/22/2020).
- [66] Patrick L. Chapman, “Permanent-Magnet Synchronous Machine Drives,” in *The Power Electronics Handbook*, 1st ed., CRC Press, 2001, ISBN: 978-1-315-22064-2.



- [67] A. F. El-Sayed, “Piston Engines and Propellers,” in *Fundamentals of Aircraft and Rocket Propulsion*, A. F. El-Sayed, Ed., London: Springer, 2016, pp. 219–314, ISBN: 978-1-4471-6796-9. DOI: [10.1007/978-1-4471-6796-9\\_4](https://doi.org/10.1007/978-1-4471-6796-9_4). [Online]. Available: [https://doi.org/10.1007/978-1-4471-6796-9\\_4](https://doi.org/10.1007/978-1-4471-6796-9_4) (visited on 10/26/2020).
- [68] Z. S. Spakovszky. “Performance of Propellers,” *Thermodynamics and Propulsion*. (2002), [Online]. Available: <https://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html> (visited on 09/09/2020).
- [69] J.B. Brandt, R.W. Deters, G.K. Ananda, O.D. Dantsker, and M.S. Selig. “UIUC Propeller Database.” (), [Online]. Available: <https://m-selig.ae.illinois.edu/props/propDB.html> (visited on 01/29/2021).
- [70] Q. Quan, *Introduction to Multicopter Design and Control*. Singapore: Springer Singapore, 2017, ISBN: 978-981-10-3381-0 978-981-10-3382-7. DOI: [10.1007/978-981-10-3382-7](https://doi.org/10.1007/978-981-10-3382-7). [Online]. Available: <http://link.springer.com/10.1007/978-981-10-3382-7> (visited on 02/17/2021).
- [71] S. Zhao, “Time Derivative of Rotation Matrices: A Tutorial,” Sep. 20, 2016. arXiv: [1609.06088 \[cs\]](https://arxiv.org/abs/1609.06088). [Online]. Available: <http://arxiv.org/abs/1609.06088> (visited on 06/10/2021).
- [72] Jerry B. Marion, *Classical Dynamics of Particles and Systems*. Elsevier, 1965, ISBN: 978-1-4832-5676-4. DOI: [10.1016/C2013-0-12598-6](https://doi.org/10.1016/C2013-0-12598-6). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/C20130125986> (visited on 06/09/2021).
- [73] Wikipedia contributors, *Parallel axis theorem*, in *Wikipedia*, Wikipedia, The Free Encyclopedia., Jun. 9, 2021. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Parallel\\_axis\\_theorem&oldid=1027622731](https://en.wikipedia.org/w/index.php?title=Parallel_axis_theorem&oldid=1027622731) (visited on 06/14/2021).
- [74] Velimir. “S500 Frame,” 3D CAD Model Library — GrabCAD. (Jul. 7, 2020), [Online]. Available: <https://grabcad.com/library/s500-frame-1> (visited on 06/16/2021).
- [75] Holybro. “Reference Frames,” Wikifactory. (Feb. 8, 2021), [Online]. Available: <https://wikifactory.com/+holybro/reference-frame-1> (visited on 06/16/2021).
- [76] H. Tang. “APC 11x8 E Prop,” 3D CAD Model Library — GrabCAD. (May 29, 2017), [Online]. Available: <https://grabcad.com/library/apc-11x8-e-prop-1> (visited on 06/16/2021).
- [77] R. Tedrake. “Ch. 3 - Acrobots, Cart-Poles, and Quadrotors,” *Underactuated Robotics - Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. (Oct. 19, 2021), [Online]. Available: <http://underactuated.mit.edu/acrobot.html#section3> (visited on 10/29/2021).

- [78] Mauch Technology. “Mauch HS-200-LV,” Mauch Electronic. (2019), [Online]. Available: <https://www.mauch-electronic.com/> (visited on 06/06/2022).
- [79] J. Y. Hwang, M. K. Jung, and O. J. Kwon, “Numerical Study of Aerodynamic Performance of a Multirotor Unmanned-Aerial-Vehicle Configuration,” *Journal of Aircraft*, vol. 52, no. 3, pp. 839–846, 2015. DOI: [10.2514/1.C032828](https://doi.org/10.2514/1.C032828). [Online]. Available: <https://doi.org/10.2514/1.C032828> (visited on 08/30/2021).
- [80] HobbyKing. “Turnigy Graphene Batteries,” Hobbyking. (), [Online]. Available: [https://hobbyking.com/en\\_us/batteries-chargers/batteries/lipo.html](https://hobbyking.com/en_us/batteries-chargers/batteries/lipo.html) (visited on 09/09/2021).
- [81] KDE Direct. “UAS Multi-Rotor Brushless Motors,” KDE Direct. (), [Online]. Available: <https://www.kdedirect.com/collections/uas-multi-rotor-brushless-motors> (visited on 09/09/2021).
- [82] The MathWorks, Inc. “Curve Fitting Toolbox,” Mathworks Products. (2022), [Online]. Available: <https://www.mathworks.com/products/curvefitting.html> (visited on 07/02/2022).
- [83] MathWorks. “First-Order Optimality Measure,” Help Center. (2022), [Online]. Available: <https://www.mathworks.com/help/optim/ug/first-order-optimality-measure.html> (visited on 09/10/2021).
- [84] D. Kraft, *A Software Package for Sequential Quadratic Programming* (Forschungsbericht / Deutsche Forschungs- Und Versuchsanstalt Für Luft- Und Raumfahrt, DFVLR ; 1988:28). Köln, 1988.
- [85] The SciPy community. “SciPy Minimize SLSQP Method,” SciPy API Reference. (2022), [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html> (visited on 07/06/2022).
- [86] M. Garcia-Sanz, “Control Co-Design: An engineering game changer,” *Advanced Control for Applications*, vol. 1, no. 1, e18, 2019, ISSN: 2578-0727. DOI: [10.1002/adc2.18](https://doi.org/10.1002/adc2.18). [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/adc2.18> (visited on 08/26/2020).
- [87] MathWorks. “Implement abc to dq0 transform,” MATLAB Documentation. (), [Online]. Available: <https://www.mathworks.com/help/physmod/sps/ref/parktransform.html> (visited on 04/13/2021).
- [88] APC Propellers. “APC Propellers - Electric Motors,” APC Propellers. (), [Online]. Available: <https://www.apcprop.com/product-category/electric-motors/> (visited on 09/09/2021).

# Appendix A

## Park Transform

A Park transform with initial a-phase to q-axis alignment, is shown in Figure A.1 [64]. Three-phase variables of stationary circuit elements are transformed to the  $dq$  reference frame as expressed in A.1.

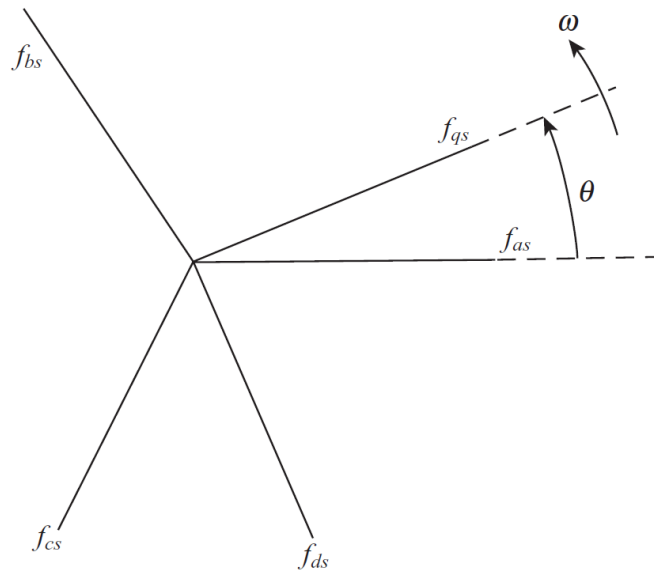


Figure A.1: DQ transformation with initial a-phase to q-axis alignment [64]

$$\mathbf{k}_{qd0} = K_s(\theta) \mathbf{k}_{abc} \quad (\text{A.1})$$

Above,  $\mathbf{k}$  can represent either voltage, current, flux linkage, or electric charge.  $K_s$  is a transformation matrix defined in A.2.

$$K_s = \frac{2}{3} \begin{pmatrix} \cos \theta & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ \sin \theta & \sin(\theta - 2\pi/3) & \sin(\theta + 2\pi/3) \\ 1/2 & 1/2 & 1/2 \end{pmatrix} \quad (\text{A.2})$$

In A.2,  $\theta$  is the electrical angular displacement of the rotating frame. Under this transformation, the total power expressed in the  $qd0$  variables must equal the total power in represented in the  $abc$  variables.

$$\begin{aligned} P_{qd0} &= P_{abc} \\ &= v_{abc}^\top i_{abc} \\ &= (K_s^{-1}(\theta) v_{qd0})^\top (K_s^{-1}(\theta) i_{qd0}) \\ &= v_{qd0}^\top (K_s^{-1}(\theta))^\top K_s^{-1}(\theta) i_{qd0} \\ &= v_{qd0}^\top \begin{bmatrix} \frac{3}{2} & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & 3 \end{bmatrix} i_{qd0} \\ &= \frac{3}{2} (v_q i_q + v_d i_d + 2v_0 i_0) \end{aligned} \quad (\text{A.3})$$

The  $\frac{3}{2}$  factor comes from the choice of constant used in the transformation. With this choice of constant, the transformation is invariant to waveform amplitude. That is, letting  $A$  be the amplitude of the three-phase signal  $\mathbf{k}_{abc}$ ,

$$\mathbf{k}_{abc} = A \begin{bmatrix} \cos(\theta) \\ \cos(\theta - \frac{2\pi}{3}) \\ \cos(\frac{2\pi}{3} + \theta) \end{bmatrix} \quad (\text{A.4})$$

Taking the transformation gives,

$$\begin{aligned} \mathbf{k}_{qd0} &= K_s \mathbf{k}_{abc} \\ &= AK_s \begin{bmatrix} \cos(\theta) \\ \cos(\theta - \frac{2\pi}{3}) \\ \cos(\frac{2\pi}{3} + \theta) \end{bmatrix} \\ &= \begin{bmatrix} A \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (\text{A.5})$$

It is sometimes desirable to use a matrix  $\tilde{K}_s$  such that the transformation is power  $P$

invariant. That is,

$$P = \mathbf{v}_{abc}^\top \mathbf{i}_{abc} = \tilde{\mathbf{v}}_{qd0}^\top \tilde{\mathbf{i}}_{qd0} \quad (\text{A.6})$$

where,

$$\tilde{\mathbf{k}}_{qd0} = \tilde{K}_s \mathbf{k}_{abc} \quad (\text{A.7})$$

Equation A.6 is satisfied if we choose  $\tilde{K}_s$  to be orthogonal, that is,  $\tilde{K}_s^\top \tilde{K}_s = I$ .

$$\begin{aligned} \tilde{\mathbf{v}}_{qd0}^\top \tilde{\mathbf{i}}_{qd0} &= \mathbf{v}_{abc}^\top \mathbf{i}_{abc} \\ \mathbf{v}_{abc}^\top \tilde{K}_s^\top \tilde{K}_s \mathbf{i}_{abc} &= \mathbf{v}_{abc}^\top \mathbf{i}_{abc} \\ \mathbf{v}_{abc}^\top \mathbf{i}_{abc} &= \mathbf{v}_{abc}^\top \mathbf{i}_{abc} \end{aligned} \quad (\text{A.8})$$

Such an orthogonal transformation matrix is presented in [87].

$$\tilde{K}_s = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\frac{2\pi}{3} + \theta) \\ \sin(\theta) & \sin(\theta - \frac{2\pi}{3}) & \sin(\frac{2\pi}{3} + \theta) \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix} \quad (\text{A.9})$$

When this power-invariant Park transform A.7 is applied to the three-phase waveform  $\mathbf{k}_{abc}$  with amplitude  $A$  defined in A.4, then the transformed variables  $\tilde{\mathbf{k}}_{qd0}$  are,

$$\tilde{\mathbf{k}}_{qd0} = \tilde{K}_s \mathbf{k}_{abc} = \begin{bmatrix} \sqrt{\frac{3}{2}} A \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{3} k_{RMS} \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.10})$$

where  $k_{RMS}$  is the root-mean-square value of  $\mathbf{k}_{abc}$ .

The transformation from standard Park transformation variables to power-invariant Park transformation variables is given in A.11.

$$\begin{aligned} \tilde{\mathbf{k}}_{qd0} &= \tilde{K}_s K_s^{-1} \mathbf{k}_{qd0} \\ &= \begin{bmatrix} \sqrt{\frac{3}{2}} & 0 & 0 \\ 0 & \sqrt{\frac{3}{2}} & 0 \\ 0 & 0 & \sqrt{3} \end{bmatrix} \mathbf{k}_{qd0} \end{aligned} \quad (\text{A.11})$$

# Appendix B

## Powertrain System Model Details

Tables [B.1](#) and [B.2](#) provide a description of the vertices and edges, respectively, in the powertrain system graph shown in [Figure 5.13](#). In the tables and system model equations, the notation  $\phi_x^t$  is used to denote parameter  $x$  corresponding to component type  $t$ , where  $B$  is the battery,  $\text{Bus}$  is the electrical bus,  $I$  is the inverter,  $M$  is the motor,  $P$  is the propeller, and  $S$  is the quadrotor system.

Number	Parent	Description	Symbol	Capacitance
1	Battery	State of Charge	$V_1$	$\phi_{N_p}^B \phi_{N_s}^B \phi_Q^B \phi_{v_{OCV}}^B (V_1)$
2	Motor 1	Current ( $i_q$ )	$I_1$	$\phi_L^M I_1$
3	Propeller 1	Angular Velocity ( $\omega_m$ )	$\omega_1$	$(\phi_J^M + \phi_J^P)\omega_1$
4	Motor 2	Current ( $i_q$ )	$I_2$	$\phi_L^M I_2$
5	Propeller 2	Angular Velocity ( $\omega_m$ )	$\omega_2$	$(\phi_J^M + \phi_J^P)\omega_2$
6	Motor 3	Current ( $i_q$ )	$I_3$	$\phi_L^M I_3$
7	Propeller 3	Angular Velocity ( $\omega_m$ )	$\omega_3$	$(\phi_J^M + \phi_J^P)\omega_3$
8	Motor 4	Current ( $i_q$ )	$I_4$	$\phi_L^M I_4$
9	Propeller 4	Angular Velocity ( $\omega_m$ )	$\omega_4$	$(\phi_J^M + \phi_J^P)\omega_4$
10	DC Electrical Bus	Voltage	$V_2$	0
11	DC Electrical Bus	Current	$I_5$	0
12	Inverter 1	Current ( $i_{DC}$ )	$I_6$	0
13	Inverter 1	Voltage ( $v_q$ )	$V_3$	0
14	Inverter 2	Current ( $i_{DC}$ )	$I_7$	0
15	Inverter 2	Voltage ( $v_q$ )	$V_4$	0
16	Inverter 3	Current ( $i_{DC}$ )	$I_8$	0
17	Inverter 3	Voltage ( $v_q$ )	$V_5$	0
18	Inverter 4	Current ( $i_{DC}$ )	$I_9$	0
19	Inverter 4	Voltage ( $v_q$ )	$V_6$	0
20	Battery	Temperature Sink	$T_1$	—
21	DC Electrical Bus	Temperature Sink	$T_2$	—
22	Inverter 1	Temperature Sink	$T_3$	—
23	Inverter 2	Temperature Sink	$T_4$	—
24	Inverter 3	Temperature Sink	$T_5$	—
25	Inverter 4	Temperature Sink	$T_6$	—
26	Propeller 1	Torque Sink	$\tau_1$	—
27	Motor 1	Temperature Sink	$T_7$	—
28	Propeller 2	Torque Sink	$\tau_2$	—
29	Motor 1	Temperature Sink	$T_8$	—
30	Propeller 3	Torque Sink	$\tau_3$	—
31	Motor 1	Temperature Sink	$T_9$	—
32	Propeller 4	Torque Sink	$\tau_4$	—
33	Motor 1	Temperature Sink	$T_{10}$	—

Table B.1: Vertices of powertrain system graph model

Number	Parent	Power Flow	Input
1	Battery	$\phi_{N_s}^B \phi_{v_{OCV}}^B(x_t)$	—
2	Battery	$(\phi_{N_s}^B / \phi_{N_p}^B) \phi_{R_s}^B x_t^2$	—
3	DC Electrical Bus	$x_h x_t$	—
4	DC Electrical Bus	$\phi_R^{\text{Bus}} x_t^2$	—
5	Inverter 1	$u_1 x_h x_t$	$u_1$
6	Inverter 1	$\sqrt{2/3} x_h x_t$	—
7	Inverter 1	$x_h x_t$	—
8	Inverter 1	$\phi_R^I x_t^2$	—
9	Inverter 2	$u_2 x_h x_t$	$u_2$
10	Inverter 2	$\sqrt{2/3} x_h x_t$	—
11	Inverter 2	$x_h x_t$	—
12	Inverter 2	$\phi_R^I x_t^2$	—
13	Inverter 3	$u_3 x_h x_t$	$u_3$
14	Inverter 3	$\sqrt{2/3} x_h x_t$	—
15	Inverter 3	$x_h x_t$	—
16	Inverter 3	$\phi_R^I x_t^2$	—
17	Inverter 4	$u_4 x_h x_t$	$u_4$
18	Inverter 4	$\sqrt{2/3} x_h x_t$	—
19	Inverter 4	$x_h x_t$	—
20	Inverter 4	$\phi_R^I x_t^2$	—
21	Motor 1	$\sqrt{3/2} (30 / (\phi_{kV}^M \pi)) x_h x_t$	—
22	Propeller 1	$(\phi_{kP}^P / (2\pi)) \rho D^5 x_t^3$	—
23	Motor 1	$\phi_{Rm}^M x_t^2$	—
24	Motor 1	0	—
25	Motor 2	$\sqrt{3/2} (30 / (\phi_{kV}^M \pi)) x_h x_t$	—
26	Propeller 2	$(\phi_{kP}^P / (2\pi)) \rho D^5 x_t^3$	—
27	Motor 2	$\phi_{Rm}^M x_t^2$	—
28	Motor 2	0	—
29	Motor 3	$\sqrt{3/2} (30 / (\phi_{kV}^M \pi)) x_h x_t$	—
30	Propeller 3	$(\phi_{kP}^P / (2\pi)) \rho D^5 x_t^3$	—
31	Motor 3	$\phi_{Rm}^M x_t^2$	—
32	Motor 3	0	—
33	Motor 4	$\sqrt{3/2} (30 / (\phi_{kV}^M \pi)) x_h x_t$	—
34	Propeller 4	$(\phi_{kP}^P / (2\pi)) \rho D^5 x_t^3$	—
35	Motor 4	$\phi_{Rm}^M x_t^2$	—
36	Motor 4	0	—

Table B.2: Edges of powertrain system graph model



# Appendix C

## AGILe Exported Model: Planar Quadrotor Powertrain

The Python functions  $f(x, a, u, d, \theta)$ ,  $g(x, a, u, d, \theta)$ , and  $h(x, a, u, d, \theta)$  defined below are the state derivatives, model outputs, and algebraic-state residuals of the planar quadrotor powertrain. They were automatically generated by the AGILe toolbox.

```
def f(x,a,u,d,theta):
# auto-generated function from matlab

    x2=x[1]
    x3=x[2]
    a2=a[1]
    a7=a[6]
    a8=a[7]
    Capacity__Battery=theta[0]
    J_r__MotorProp=theta[6]
    K_Q__Propeller=theta[7]
    K_t__Motor=theta[9]

    out1 = -a2/Capacity__Battery
    out2 = -(2*K_Q__Propeller*x2**2
             - 2**(1/2)*3**(1/2)*K_t__Motor*a7)/(2*J_r__MotorProp)
    out3 = -(2*K_Q__Propeller*x3**2
             - 2**(1/2)*3**(1/2)*K_t__Motor*a8)/(2*J_r__MotorProp)
```

```

        return out1, out2, out3

def g(x,a,u,d,theta):
# auto-generated function from matlab

    x2=x[1]
    x3=x[2]
    K_Q__Propeller=theta[7]
    K_T__Propeller=theta[8]

    out1 = K_T__Propeller*x2**2
    out2 = K_Q__Propeller*x2**2
    out3 = K_T__Propeller*x3**2
    out4 = K_Q__Propeller*x3**2

    return out1, out2, out3, out4

def h(x,a,u,d,theta):
# auto-generated function from matlab

    x2=x[1]
    x3=x[2]
    a1=a[0]
    a2=a[1]
    a3=a[2]
    a4=a[3]
    a5=a[4]
    a6=a[5]
    a7=a[6]
    a8=a[7]
    u1=u[0]
    u2=u[1]
    K_t__Motor=theta[9]
    N_s__Battery=theta[17]
    R__PMSMInverter_2=theta[25]
    R_p__Battery=theta[26]
    Rm__Motor=theta[28]

```

```

out1 = a2 - a3*u1 - a5*u2
out2 = (37*N_s__Battery)/10 - a1 - (3*a2)/1000 - R_p__Battery*a2
out3 = a1*u1 - R__PMSMInverter_2*a3 - (2**(1/2)*3**(1/2)*a4)/3
out4 = (2**(1/2)*3**(1/2)*a3)/3 - a7
out5 = a1*u2 - R__PMSMInverter_2*a5 - (2**(1/2)*3**(1/2)*a6)/3
out6 = (2**(1/2)*3**(1/2)*a5)/3 - a8
out7 = a4 - Rm__Motor*a7 - (2**(1/2)*3**(1/2)*K_t__Motor*x2)/2
out8 = a6 - Rm__Motor*a8 - (2**(1/2)*3**(1/2)*K_t__Motor*x3)/2

return out1, out2, out3, out4, out5, out6, out7, out8

```

# Appendix D

## Component Databases

### D.1 Battery Component Database [\[80\]](#)

In the following table,  $N_p$  is the number of parallel cells in the pack,  $N_s$  is the number of series cells in the pack,  $Q$  is the pack capacity, and  $R_s$  is the cell series resistance.

Make	Model	SKU	C	Mass (kg)	$N_p$	$N_s$	Price (USD)	$Q$ (mAh)	$R_s$ ( $\Omega$ )
Turnigy	Graphene Panther	9067000422-0	75	0.405	1	4	56.99	3000	0.00375
Turnigy	Graphene Panther	9067000421-0	75	0.32	1	3	42.97	3000	0.0043333
Turnigy	Graphene Panther	9067000418-0	75	0.076	1	4	21.97	500	0.0075
Turnigy	Graphene Panther	9067000417-0	75	0.196	1	4	27.87	1400	0.00425
Turnigy	Graphene Panther	9067000416-0	75	0.156	1	3	23.99	1400	0.0043333
Turnigy	Graphene Panther	9067000415-0	75	0.162	1	3	22.97	1500	0.0043333
Turnigy	Graphene Panther	9067000412-0	75	0.529	1	4	69.99	4000	0.003
Turnigy	Graphene Panther	9067000411-0	75	0.412	1	3	55.95	4000	0.0033333
Turnigy	Graphene Panther	9067000410-0	75	0.212	1	6	26.99	1000	0.0031667
Turnigy	Graphene Panther	9067000409-0	75	0.137	1	3	16.99	1300	0.0043333
Turnigy	Graphene Panther	9067000408-0	75	0.12	1	4	18.99	850	0.0055
Turnigy	Graphene Panther	9067000407-0	75	0.084	1	2	9.54	1000	0.0055
Turnigy	Graphene Panther	9067000406-0	75	0.043	1	1	9.32	950	0.012
Turnigy	Graphene Panther	9067000376-0	75	0.598	1	6	82.57	3000	0.003
Turnigy	Graphene Panther	9067000375-0	75	0.92	1	6	116.95	5000	0.0025
Turnigy	Graphene Panther	9067000374-0	75	0.49	1	3	65.97	5000	0.003
Turnigy	Graphene Panther	9067000373-0	75	0.76	1	6	104.97	4000	0.0026667
Turnigy	Graphene Panther	9067000366-0	75	0.212	1	4	24.99	1600	0.0035
Turnigy	Graphene Panther	9067000363-0	75	0.173	1	4	23.99	1300	0.003
Turnigy	Graphene Panther	9067000360-0	75	0.148	1	4	23.97	1000	0.00425
Turnigy	Graphene Panther	9067000515-0	75	0.202	1	4	26.97	1500	0.00425
Turnigy	Graphene Panther	9067000513-0	75	0.249	1	6	31.99	1300	0.0031667
Turnigy	Graphene Panther	9067000420-0	75	1.14	1	6	129.99	6000	0.0021667
Turnigy	Graphene Panther	9067000419-0	75	0.63	1	4	81.99	5000	0.00275
Turnigy	Graphene Panther	9067000414-0	75	0.8	1	4	99.99	6000	0.00225
Turnigy	Graphene Panther	9067000372-0	75	0.295	1	4	42.97	2200	0.00325
Turnigy	Graphene Panther	9067000371-0	75	0.23	1	3	32.99	2200	0.0033333
Turnigy	Graphene Panther	9067000370-0	75	0.094	1	4	14.19	650	0.01
Turnigy	Graphene Panther	9067000369-0	75	0.051	1	1	5.57	600	0.02
Turnigy	Graphene Panther	9067000368-0	75	0.063	1	3	14.99	500	0.011667
Turnigy	Graphene Panther	9067000365-0	75	0.212	1	4	29.99	1600	0.0035
Turnigy	Graphene Panther	9067000361-0	75	0.116	1	3	16.99	1000	0.0043333
Turnigy	Graphene Panther	9067000413-0	75	0.63	1	3	75.97	6000	0.0023333

Table D.1: Battery component database

## D.2 Motor Component Database [81]

In the following table,  $D$  is motor diameter,  $I_{\max}$  is the motor's maximum current,  $I_o$  is the no-load current at 10V,  $Rm$  is the winding resistance, and  $kV$  is the speed constant.

Make	Model	SKU	$D$ (m)	$I_{\max}$ (A)	$I_o$ (A)	Mass (kg)	Price (USD)	$R_m$ ( $\Omega$ )	kV (RPM/V)
KDE	KDE13218XF-105		0.1502	150	3.1	2.015	1325.95	0.013	105
KDE	KDE10218XF-105		0.1091	142	1	1.075	842.95	0.023	105
KDE	KDE8218XF-120		0.089	110	0.8	0.76	615.95	0.037	120
KDE	KDE7215XF-135		0.0808	85	0.5	0.555	383.95	0.057	135
KDE	KDE7208XF-110		0.08	50	0.4	0.38	345.95	0.171	110
KDE	KDE7208XF-135		0.08	58	0.4	0.38	360.95	0.113	135
KDE	KDE6815XF-205		0.0772	57	1.9	0.55	350.95	0.048	205
KDE	KDE6213XF-185		0.07	50	0.6	0.36	264.95	0.072	185
KDE	KDE5215XF-220		0.06	44	0.5	0.305	205.95	0.078	220
KDE	KDE5215XF-330		0.06	62	0.7	0.305	201.95	0.044	330
KDE	KDE5215XF-435		0.06	72	1.4	0.305	201.95	0.031	435
KDE	KDE4215XF-465		0.0482	62	0.7	0.195	153.95	0.052	465
KDE	KDE4213XF-360		0.0482	38	0.4	0.175	138.95	0.081	360
KDE	KDE4014XF-380		0.0465	36	0.5	0.16	123.95	0.075	380
KDE	KDE4012XF-400		0.0465	32	0.5	0.145	118.95	0.08	400
KDE	KDE3520XF-400		0.0422	45	0.3	0.19	116.95	0.078	400
KDE	KDE3510XF-475		0.0422	30	0.2	0.12	95.95	0.105	475
KDE	KDE3510XF-715		0.0422	45	0.5	0.12	95.95	0.054	715
KDE	KDE2814XF-515		0.0355	24	0.3	0.095	74.95	0.13	515
KDE	KDE2814XF-775		0.0355	36	0.5	0.095	74.95	0.069	775
KDE	KDE2315XF-885		0.0283	24	0.5	0.064	63.95	0.127	885
KDE	KDE2315XF-965		0.0283	26	0.5	0.064	63.95	0.102	965
KDE	KDE2315XF-2050		0.0283	44	1.3	0.064	63.95	0.034	2050
KDE	KDE2306XF-2050		0.0283	24	0.6	0.029	28.95	0.068	2050
KDE	KDE2306XF-2550		0.0283	34	1.2	0.029	28.95	0.044	2550
KDE	KDE2304XF-2350		0.0283	20	0.7	0.024	26.95	0.091	2350
KDE	KDE1806XF-2350		0.023	18	0.4	0.018	28.95	0.117	2350

Table D.2: Motor component database

### D.3 Propeller Component Database [88]

In the following table,  $D$  is propeller diameter,  $P$  is propeller pitch,  $k_P$  is the propeller’s power coefficient, and  $k_T$  is the propeller’s torque coefficient.

Make	Model	SKU	$D$ (m)	Mass (kg)	$P$ (m)	Price (USD)	$k_P$	$k_T$
APC	4.1x4.1E	LP04141E	0.10414	0.0031184	0.10414	2.25	0.09	0.13
APC	4.5x4.1E	LP04541E	0.1143	0.0039689	0.10414	2.25	0.08	0.13
APC	4.7x4.2E	LP04742E	0.11938	0.0031184	0.10668	2.25	0.07	0.12
APC	4.75x4.5E	LP04745E	0.12065	0.0039689	0.1143	2.25	0.07	0.12
APC	4.75x4.75E	LP04747E	0.12065	0.0039689	0.12065	2.25	0.07	0.12
APC	4.75x5.5E	LP04755E	0.12065	0.0039689	0.1397	2.25	0.08	0.12
APC	5x3E	LP05030E	0.127	0.0031184	0.0762	2.25	0.05	0.11
APC	5x5E	LP05050E	0.127	0.0039689	0.127	2.25	0.07	0.12

APC	5x7.5E	LP05075E	0.127	0.0051029	0.1905	2.35	0.1	0.15
APC	5.1x4.5E	LP05145E	0.12954	0.0051029	0.1143	2.25	0.11	0.17833
APC	5.25x5.5E	LP05355E	0.13335	0.0039689	0.1397	2.25	0.07	0.13
APC	5.5x4.5E	LP05545E	0.1397	0.0039689	0.1143	2.25	0.06	0.12
APC	5.5x4.7E	LP05547E	0.1397	0.0039689	0.11938	2.25	0.065	0.12
APC	5.5x6.5E	LP05565E	0.1397	0.0039689	0.1651	2.25	0.08	0.12
APC	6x4E	LP06040E	0.1524	0.0051029	0.1016	2.29	0.0575	0.12
APC	6x5.5E	LP06055E	0.1524	0.0051029	0.1397	2.29	0.07	0.12
APC	6x6E	LP06060E	0.1524	0.0051029	0.1524	2.29	0.07	0.12
APC	7x4E	LP07040E	0.1778	0.0079379	0.1016	2.39	0.06	0.13
APC	7x5E	LP07050E	0.1778	0.0079379	0.127	2.39	0.07	0.13
APC	7x6E	LP07060E	0.1778	0.0079379	0.1524	2.39	0.0725	0.13
APC	7x7E	LP07070E	0.1778	0.0079379	0.1778	2.39	0.08	0.13
APC	8x4E	LP08040E	0.2032	0.013041	0.1016	2.57	0.05	0.12
APC	8x6E	LP08060E	0.2032	0.013891	0.1524	2.57	0.06875	0.13
APC	8x8E	LP08080E	0.2032	0.015025	0.2032	2.57	0.08	0.1375
APC	9x4.5E	LP09045E	0.2286	0.01786	0.1143	2.84	0.05	0.12
APC	9x6E	LP09060E	0.2286	0.01786	0.1524	2.84	0.06	0.13
APC	9x7.5E	LP09075E	0.2286	0.01786	0.1905	2.84	0.06875	0.13
APC	9x9E	LP09090E	0.2286	0.01786	0.2286	2.84	0.07625	0.13
APC	10x10E	LP10010E	0.254	0.020128	0.254	3.21	0.06625	0.12
APC	10x5E	LP10050E	0.254	0.020128	0.127	3.21	0.04	0.11
APC	10x6E	LP10060E	0.254	0.020128	0.1524	3.21	0.05	0.11
APC	10x7E	LP10070E	0.254	0.020128	0.1778	3.21	0.05	0.12
APC	10x8E	LP10080E	0.254	0.020128	0.2032	3.21	0.0575	0.12
APC	11x10E	LP11010E	0.2794	0.022963	0.254	3.68	0.0625	0.11
APC	11x12E	LP11012E	0.2794	0.026082	0.3048	3.68	0.11125	0.115
APC	11x5.5E	LP11055E	0.2794	0.022963	0.1397	3.68	0.04	0.1
APC	11x7E	LP11070E	0.2794	0.022963	0.1778	3.68	0.05	0.11
APC	11x8E	LP11080E	0.2794	0.022963	0.2032	3.68	0.0525	0.11
APC	11x8.5E	LP11085E	0.2794	0.024097	0.2159	3.68	0.055	0.11
APC	12x10E	LP12010E	0.3048	0.026082	0.254	4.3	0.06	0.11
APC	12x12E	LP12012E	0.3048	0.026082	0.3048	4.3	0.07	0.11
APC	12x6E	LP12060E	0.3048	0.026932	0.1524	4.3	0.04	0.1
APC	12x8E	LP12080E	0.3048	0.026082	0.2032	4.3	0.05125	0.1025
APC	13x10E	LP13010E	0.3302	0.03005	0.254	5.06	0.0625	0.1025
APC	13x4E	LP13040E	0.3302	0.03005	0.1016	5.06	0.02	0.07
APC	13x5.5E	LP13055E	0.3302	0.032035	0.1397	5.06	0.0325	0.09
APC	13x6.5E	LP13065E	0.3302	0.03005	0.1651	5.06	0.03875	0.09625
APC	13x8E	LP13080E	0.3302	0.030901	0.2032	5.06	0.04875	0.1
APC	14x10E	LP14010E	0.3556	0.034019	0.254	5.99	0.06	0.1
APC	14x12E	LP14012E	0.3556	0.03487	0.3048	5.99	0.075	0.10125
APC	14x14E	LP14014E	0.3556	0.036004	0.3556	5.99	0.1025	0.0875

APC	14x6E	LP14060E	0.3556	0.037138	0.1524	5.99	0.03125	0.08375
APC	14x7E	LP14070E	0.3556	0.034019	0.1778	5.99	0.03625	0.09125
APC	14x8.5E	LP14085E	0.3556	0.037138	0.2159	5.99	0.05	0.0925
APC	15x10E	LP15010E	0.381	0.045076	0.254	7.1	0.065	0.1
APC	15x4E	LP15040E	0.381	0.045076	0.1016	7.1	0.02	0.06
APC	15x6E	LP15060E	0.381	0.045076	0.1524	7.1	0.03	0.08125
APC	15x7E	LP15070E	0.381	0.045076	0.1778	7.1	0.035	0.09
APC	15x8E	LP15080E	0.381	0.043942	0.2032	7.1	0.045	0.09375
APC	16x10E	LP16010E	0.4064	0.05188	0.254	8.42	0.06625	0.0975
APC	16x12E	LP16012E	0.4064	0.05188	0.3048	8.42	0.08875	0.1025
APC	16x4E	LP16040E	0.4064	0.054998	0.1016	8.42	0.02	0.06
APC	16x6E	LP16060E	0.4064	0.056132	0.1524	8.42	0.02625	0.08
APC	16x8E	LP16080E	0.4064	0.05188	0.2032	8.42	0.0425	0.09375
APC	17x10E	LP17010E	0.4318	0.06407	0.254	9.95	0.0625	0.09625
APC	17x12E	LP17012E	0.4318	0.068039	0.3048	9.95	0.0825	0.0975
APC	17x6E	LP17060E	0.4318	0.06407	0.1524	9.95	0.02375	0.07625
APC	17x7E	LP17070E	0.4318	0.06407	0.1778	9.95	0.03125	0.08375
APC	17x8E	LP17080E	0.4318	0.06407	0.2032	9.95	0.04	0.09125
APC	18x10E	LP18010E	0.4572	0.072858	0.254	11.72	0.055	0.09625
APC	18x12E	LP18012E	0.4572	0.073992	0.3048	11.72	0.075	0.0975
APC	18x8E	LP18080E	0.4572	0.072858	0.2032	11.72	0.035	0.0875
APC	19x10E	LP19010E	0.4826	0.083064	0.254	13.75	0.04625	0.095
APC	19x12E	LP19012E	0.4826	0.083064	0.3048	13.75	0.06625	0.0975
APC	19x8E	LP19080E	0.4826	0.083064	0.2032	13.75	0.03	0.085
APC	20x10E	LP20010E	0.508	0.096105	0.254	16.05	0.03875	0.09
APC	20x11E	LP20011E	0.508	0.09894	0.2794	16.05	0.04875	0.09375
APC	20x13E	LP20013E	0.508	0.098089	0.3302	16.05	0.0625	0.095
APC	20x15E	LP20015E	0.508	0.11793	0.381	16.05	0.0775	0.095
APC	20x8E	LP20080E	0.508	0.096105	0.2032	16.05	0.03	0.0825
APC	20.5x14E	LP20514E	0.5207	0.12304	0.3556	16.05	0.0675	0.09375
APC	21x13E	LP21013E	0.5334	0.13012	0.3302	18.6	0.05875	0.09375
APC	22x10E	LP22010E	0.5588	0.13409	0.254	21.45	0.03375	0.08875
APC	22x11E	LP22011E	0.5588	0.11992	0.2794	21.45	0.03875	0.09
APC	22x12E	LP22012E	0.5588	0.13608	0.3048	21.45	0.04625	0.09125
APC	24x12E	LP24012E	0.6096	0.16386	0.3048	28.05	0.037143	0.09
APC	25x12.5E	LP25125E	0.635	0.22311	0.3175	31.8	0.037143	0.092857
APC	26x13E	LP26013E	0.6604	0.20894	0.3302	35.85	0.038333	0.09
APC	26x15E	LP26015E	0.6604	0.21404	0.381	35.85	0.05	0.094
APC	27x13E	LP27013E	0.6858	0.2319	0.3302	40.25	0.036667	0.09

Table D.3: Propeller component database



## D.4 Animation of Optimized Planar Quadrotor System

Included with this thesis is a video file ‘planar\_quadrotor\_animation.avi’ that plots the trajectory of the planar quadrotor as it completes the dynamic mission. In the video, the red line represents the trajectory of the initial design and the green line represents the trajectory of the optimized design. The animation is played at one-quarter speed.