The Pennsylvania State University

The Graduate School

Department of Mechanical and Nuclear Engineering

OPTIMIZATION OF HYBRID POWER SOURCES FOR MOBILE ROBOTICS THROUGH THE USE OF ALLOMETRIC DESIGN PRINCIPLES AND DYNAMIC PROGRAMMING

A Thesis in

Mechanical Engineering

by

Drew Gregory Logan

© 2010 Drew Gregory Logan

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2010

The thesis of Drew Gregory Logan was reviewed and approved* by the following:

Sean N. Brennan
Assistant Professor, Department of Mechanical Engineering
Thesis Advisor

Timothy W. Simpson
Professor, Department of Mechanical & Industrial Engineering

Karl M. Reichard
Professor, Department of Mechanical Engineering

Karen A. Thole
Professor, Department of Mechanical Engineering
Head of the Department of Mechanical Engineering

*Signatures are on file in the Graduate School

Abstract

This work presents design rules and methods used to optimize mobile ground robots early in the design process. The focus ranges from the general geometric considerations for the overall size of a robot, to the optimization of size and control of three sources of hybrid power for specific ground robots. The geometric analysis includes study of the platform's performance requirements for climbing, traversal and speed, and this work demonstrates that one can accurately calculate the necessary bulk properties of the robot including physical size, mass and power.

Once the bulk properties are calculated, a system-level model can be designed for the robot platform using user-specified performance criteria. This system-level view decomposes the robot as a whole into its subsystems and the powertrain components used for locomotion. Such decomposition is used to accurately predict the necessary power, performance and allometry (size dependence) of each component. Once components have thus been correctly sized, the overall system-level performance is calculated including operational time and cruising distance. Comparisons to experiments on existing robot platforms show the fidelity of this approach. Comparisons between conceptual robot models at both system and subsystem-levels allow the user to examine tradeoffs between different performance requirements.

In many cases, a specific sequence of tasks is needed for a robot to complete a given mission. This mission, for a given sized robot, translates into a power profile representing the power draw required to complete the task sequence. Dynamic Programming is used to optimize the control strategy and size of each component within the hybrid power source (battery, ultracapacitor and generator) for a number of missions. This work shows that, based on a given mission, the optimal power topology of a robot varies with the characteristics of its mission.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This work was supported by the NAVSEA Contract Number N00024-D-02-D-6604, Delivery Order Number 0602. The content of the information does not necessarily reflect the position or policy of NAVSEA, and no official endorsement should be inferred.

Chapter 1

Introduction

Robot design is a very new discipline, and hence has fewer formalized and published design practices than the automotive and defense vehicle industries which share many of the same technical and design challenges [1]-[5]. Bekker's contribution to this filed primarily dealt with predicting large scale tracked and wheeled vehicle capabilities including soil terrain modeling over varied terrains [1]-[3]. Taborek developed wheeled vehicle models to predict capabilities and measure performance [4]. Wong extends both works on large scale vehicles to include higher order modeling of vehicles behaviors and dynamics [5]. Gillespie, et al. performed extensive work into automotive dynamic modeling and performance predictions [6]. But while many have researched aspects of robot design, the ground robot design process has historically been dominated by the "build, test, evaluate, and rebuild" procedure wherein iterations on the design, construction, and testing of a robot are repeated until an acceptable design is developed.

This strategy is very costly and is difficult to advance without knowledge of how component behavior affects overall robot system behavior. A particular area exemplifying the need for careful design of a robot platform is in Explosive Ordinance Disposal (EOD) robots, an example that is used throughout this thesis. While much study and evaluation has been performed on existing mobile ground EOD robots, little research has been performed to predict their capabilities during the design process [12], [13]. Frost, et al. developed metrics and performed evaluation testing on the PackBot [12]. Ahlvin, et al. has performed extensive work measuring the capabilities of large scale ground vehicles [13]. But to the largest extent, robotics research is dominated by research on navigation and control, for example the work by Thrun and others [7]-[11]. Thus, while a robot's physical capabilities and component design are critical to its performance, little focus is placed on the ability to predict these capabilities.

Prior to this study, some have recognized the need for comprehensive robot system models. However, the available models either required too much detail to be applied to a new generalized robot model, or are not conclusive enough to describe the full range of capabilities a robot may encounter [13]-[16]. Ahlvin, et al. developed heuristic models experimentally to describe both capabilities and power consumptions for large scale wheeled and tracked vehicles [13]. To apply models developed by this work requires extensive knowledge of each component on the vehicle. Because the experimentation performed was on automotive and larger scaled vehicles, the models developed are inaccurate for ground robots more than an order of magnitude smaller. Hetherington's predictive model for soil penetration, for example, requires extensive knowledge for the vehicles track configuration and overall designs [14]. To use these models, one would have to have completed a design to the level of specifying tread configuration and sprocket geometries, aspects far too detailed to guide the gross physical architecture and layout of a robot.

Others have recognized the need for more simplistic models for components of ground vehicles, but unfortunately omit the size ranges typically occupied by ground robots. McBride, et al. developed mathematical models to predict some relevant ground mobile robot capabilities [15]. The models developed in this work are simplified and do not cover the full range of capabilities a ground robot faces during operation. Marden accurately predicts how one can scaling the net force out of an engine with its mass but does not specify enough characteristics to apply these rules to the development of a conceptual robot design [16]. There is thus a gap in the ability to predict and validate the full breadth of mobile ground robot capabilities that could be encountered during typical operation.

The scaling of powertrain components for robots is perhaps most mature among all robot components, as this area has been long studied within the automotive and defense industries [17], [18]. Ehsani, et al. and Rahman, et al. both sought to use electric drive motors to replace internal combustion engines. Their analysis focuses on the performance evaluations of electric motors to produce the same power as conventional engines [17], [18]. This research has gained even more focus in recent years with the advent of hybrid automotive drivetrains [19]-[21]. Peng, et al. have developed hybrid system models and optimization techniques for a variety of automotive

applications [19]-[21], and some of these techniques are borrowed in this thesis. Internal combustion engines, electric motors and hydraulic drive systems are used for hybrid locomotion in both series and parallel configurations. Batteries, fuels cells and generators, and gasoline are used to power these systems.

Parallel research within the robotic industry has sought to generate generalized equations that describe the scaling of various robotic mechanisms and some isolated drivetrain components [22]-[24] . Much focus has been placed on determining the scaling limitations of robotic drive components [22], [23]. Other research has sought to determine scaling principles for general mechanic devices used by a variety of robots [24]. The focus is to show that mechanical components form beams to actuators scale with the forces being applied to or from the system.

This thesis recreates scaling principles and extends automotive research into the domain of robotic platforms, describing each of the components a mobile ground robot uses for locomotion. These components are then built into a system-level model that can be used to describe both component-level performance and performance of the powertrain system as a whole. The robot's desired capabilities, such as climbing and traversal, define the platform's generalized size and mass. Drawing from this information and the subsystem component scaling models, each of the critical component specifications for the powertrain can be determined. These components are then combined to produce a system-level view to determine the robot's cruising distance and operating capabilities. As observed in the previous literature, a challenge in producing these mathematical models is to ensure that they are computationally efficient but precise enough to capture key behaviors of interest at a level that agrees with experimental measurements.

The generation of these models allows designers to evaluate designs early in the design process before evaluating the physical models of prototypes, reducing the time and resources required to generate an acceptable design. These design rules allow for basic feasibility studies, thus permitting designers to examine the practicality of contradictory performance requirements such as maneuverability in tight spaces and climbing steep inclines. Because it is possible to generate numerous virtual designs, each with unique performance capabilities, one can compare small perturbations to find locally optimal designs, or a pareto-front in tradeoffs. Both will be

illustrated later in the thesis using software known as the ARL trade space visualization (ATSV) tool [25]. Additionally, ATSV is used to examine gross performance tradeoffs between different robot sizes [26]. For example, Simpson, et al. have shown the advantages of using visual steering to facilitate design by shopping to optimize a design or family of designs [25]-[29].

Gasoline and internal combustion engines, while having energy densities several orders of magnitude greater than batteries, are not typically utilized in mobile ground robots [30]. And ultracapacitors have the ability to source a large amount of power with virtually no efficiency loss or wear, are also not commonly used in robots because of their limited energy storage versus batteries [31], [32]. Similar to work within the hybrid automotive industry, this thesis explores the use of a hybrid system that can potentially leverage each source's strengths to achieve a more effective power supply [33], [34]. This thesis examines the use of intelligent control strategies to enable and explore the use of multiple sources of power, to determine if combined systems can outperform classical battery-only designs.

If the specific mission of a robot is known, its design can be further optimized at the powertrain level using variations on hybrid powertrain configurations mentioned earlier. A "mission" is defined as the summation of tasks which a robot needs to complete over a given duration of time. Tasks include climbing stairs, traveling a specified distance or dragging an object. With the mission defined, a power profile is generated to describe the amount of power required to complete each task. This is then used to optimize the power generation architecture for a specific mission, and to perform this optimization, methods based on Dynamic Programming are used [35]-[37].

Dynamic Programming (DP), pioneered by Richard Bellman, is a numerical technique often used in the automotive industry to optimize the control of a variety of hybrid systems [19]-[21]. In the context of this work, Dynamic Programming optimizes the allocation of power from each source and determines when to both charge and discharge power sources. It is a numerical optimization technique which calculates the desired trajectory and power allocation. The algorithm achieves the optimum by working from the final time step backwards to the beginning [35]. Unlike feedback controllers that can be applied in real time or online, DP optimizes backwards in time and it therefore yields the best solution offline given the known profile [37].

For this same reason, DP cannot be used to control a hybrid system without the power profile (robot mission) known in advance.

This work utilizes DP to determine the feasibility of varied hybrid topologies through the generation of optimal utilization profiles for batteries, capacitors, and generators. These are then compared to the relative performances of suboptimal controllers that are implemented in real time on a hybrid robot [38], [39]. The results from DP are also offer insight to improvement of these controllers. These controllers will ultimately be used to demonstrate the optimal hybrid system composition for different scenarios, given a robot design and fixed power source mass. Peng, Grizzle, Kolmanovsky, et al. have used DP in all of these ways to optimize both the control and composition of automotive hybrid systems [19]-[21], [38], [39], thus proving the capability of this approach on systems that are similar to ground robots.

The remainder of this thesis explores the above topics in detail and attempts to derive models and methods that facilitate robot design. Chapter 2 discusses the methods used to predict robot performance. Chapter 3 examines how these performance criteria can be used to calculate the necessary size of each powertrain component necessary to robot locomotion. The system-level modeling and cruising distance prediction method is also included within Chapter 3. Chapter 4 furthers this system-level discussion by showing how multiple designs with small variations can be generated and viewed with ATSV. ATSV is then used to determine optimal designs for a given capability and to examine the tradeoffs between multiple performance requirements. Chapter 5 presents the control and optimization of hybrid power components and how they a robot's mission. Chapter 6 discusses the overall results of the work, the shortcomings of some of the modeling efforts and further research needed to improve the physical robot and power system models.

References:

[1] Bekker, M. (1969). *Introduction to Terrain-Vehicle Systems*. Ann Arbour: The University of Michigan Press.

[2] Bekker, M. (1960). *Off-the-Road Locomotion*. Ann Arbour: The University of Michigan Press.

[3] Bekker, M. (1969). *Theory of Land Locomotion*. Ann Armbour: The University of Michigan Press.

[4] Taborek, J. (1957) Mechanics of Vehicles. Penton Publishing Co., Cleveland, Ohio.

[5] Wong, J. (2001). *Theory of Ground Vehicles Third Edition.* New York: John Wiley & Sons, Inc.

[6] Gillespie, T. (1992). *Fundaments of Vehicle Dynamics*. Warrendale, Pennsylvania: Society of Automotive Engineers International.

[7] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., and Thur, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, Massachusetts: MIT Press.

[8] Bekey, G.A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control*, Cambridge, Massachusetts: MIT Press.

[9] Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robot*, Cambridge University Press, Cambridge, UK.

[10] "DARPA Urban Challenge." Internet: http://www.darpa.mil/grandchallenge/index.asp, 2007 [Mar. 24 2010].

[11] "IGVC." Internet: http://www.igvc.org/, 2009 [Mar. 26 2010].

[12] Frost. T., Norman, C., Pratt, S., Yamauchi, B., McBride, B., and Peri, G. "Derived Performance Metrics and Measurements Compared to Field Experience for the PackBot," in the *Proceedings of the Workshop of the PERMIS 2002, National Institute of Standards and Technology*, Gaithersburg, MD, August 2002.

[13] Ahlvin, R and Haley, P. (1992). NATO Reference Mobility Model Edition II, NRMM User's Guide, Technical Report. GL-92-19.

[14] Hetherington, J. (2001). The application of the MMP concept in specifying off-road mobility for wheeled and tracked vehicles. *Journal of Terramechanics*, 38:63-70.

[15] B. McBride, R. Longoria, and E. Krotkov, "Measurement and Prediction of the Off-Road Mobility of Small Robotic Ground Vehicles," in the *Proceedings of the Performance Metrics for Intelligent Systems (PerMIS)*, 2003.

[16] J. Marden. "Scaling of maximum net force output by motors used for locomotion," *Journal of Experimental Biology*, vol. 208. pp. 1653-1664, Dec. 2004.

[17] Ehsani, M. and Rahman, K. (1996). "Performance Analysis of Electric Motor Drives for Electric and Hybrid Electric Vehicle Applications." *IEEE Power Electronics in Transportation,* 49-56.

[18] Z. Rahman, M. Ehsani, and K.L Butler. "An Investigation of Electric Motor Drive Characteristics for EV and HEV Propulsion Systems." 2000 SAE Future Transportation Technology Conf., Costa Mesa, USA, Aug 21-23, 2000. SAE2000-01-3062

[19] M. Kim, H. Peng. "Power management and design optimization of fuel cell/battery hybrid vehicles." *Journal of Power Sources,* vol. 165. pp. 819-832, Dec. 2006.

[20] C. Lin, J. Kang, J. Grizzle, and H. Peng. "Energy Management Strategy for a Parallel Hybrid Electric Truck" 2001 American Control Conf., Arlington, USA, June 25-27, 2001.

[21] C. Lin, Z. Filipi, L. Louca, H. Peng, D. Assanis, J. Stein. "Modeling and control of medium-duty hybrid electric truck" *Int J. of Vehicle Design*, vol. 11. Pp.349-370, Nos. ¾, 2004.

[22] J-D. Nicoud, "Microengineering: when is small too small? Nanoengineering: when is large too large?" Int. Symposium on Micro Machine and Human Science, pp 1-6, 1995.

[23] G. Caprari, T. Estier, and R. Siegwart. "Fascination of Down Scaling – Alice the Sugar Cube Robot," Journal of Micromechatronics, vol. 1, pp.177-189, 2001.

[24] Waldron, K. and Hubert, C. (2000). Scaling Robotic Mechanisms. *IEEE International Conference on Robotics & Automation*, San Francisco, CA.

[25] G.M. Stump, M.A. Yukish, J.D. Martin, and T.W. Simpson. "The ARL Trade Space Visualizer: An Engineering Decision-Making Tool," in the *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA-2004-4568.

[26] G.M. Stump, M. Yukish, T.W. Simpson, and L. Bennett. "Multidimensional Visualization and Its Application to a Design by Shopping Paradigm," in the *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, AIAA, AIAA-2002-5622.

[27] G.M. Stump, M. Yukish, T.W. Simpson and E.N. Harris. "Design Space Visualization and Its Application to a Design by Shopping Paradigm," in the *ASME Design Engineering Technical Conference – Design Automation Conference*, Chicago, IL, ASME, 2003, Paper No. DETC2003/DAC-48785.

[28] R. Balling, "Design by Shopping: A New Paradigm?" in the *Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO-3)*, Buffalo, NY, University of Buffalo, 1999, pp 295-297.

[29] E.H. Winer, and C.L. Bloebaum, "Development of Visual Design Steering as an Aid in Large-Scale Multidiciplinary Design Optimization. Part I: Method Development," *Structural and Multidiciplinary Optimization*, Vol. 23, No. 6, 2002, pp. 412-424.

[30] M. Fischer, M. Werber, and P. Schwartz. "Batteries: Higher energy density than gasoline?," *Energy Policy*, vol. 37. pp. 2639-2641, Feb. 2009.

[31] Maxwell Technologies Boostcap Ultracapacitor (2009) Retrieved August 4 2009 from http://maxwell.interconnectnet.com/pdf/uc/Maxwell_UC_comparison.pdf.

[32] Peukert Number Derivation (2009) Retrieved January 20 2010 from http://www.smartgauge.co.uk/peukert2 .html.

[33] V. Pop, H.J. Bergveld, D. Danilov, P. Regtien, and P. Notten. *Battery Measurement Systems*. Eindhoven, The Netherlands: Springer, 2008, pp. 14-17.

[34] P. Rodatz, O. Guzzella, F. Buchi, M. Bartchi, A. Tsukada, P, Dietrich, R. Kotz, G. Scherer, and A. Wokaun, "Performance and operational characteristics of a hybrid vehicle powered by fuel cells and supercapacitors," SAE International, Warrendale, PA, Rep. Ro. 2002-01-0418, 2003.

[35] R.E. Bellman, *Eye of the Hurricane: An Autobiography*. World Scientific, Singapore, 1984.

[36] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, *Algorithms*. McGraw-Hill, 2006.

[37] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms (2nd edition)*. MIT Press & McGaw-Hill 2001.

[38] I. Kolmanovsky, I. Siverguina, and B. Lygoe, "Optimization of Powertrain Operating Policy for Feasibility Assessment and Calibration: Stochastic Dynamic Programming Approach." in the *Proceedings of the American Controls Conference*, 2002, pp. 1425-1430.

[39] A. Brahma, Y. Guezennec, and G. Rizzoni, "Optimal Energy Management in Series Hybrid Electric Vehicles." in the *Proceedings of the American Controls Conference*, 2000, pp. 60-64.

Chapter 2

Geometric Consideration for Ground Mobile Robots

The purpose of this chapter is to derive and validate computationally efficient quasi-static models in order to predict maneuverability limits of ground robots. These models predict the robot's capabilities of climbing (i.e. stairs), ditch traversal, indoor maneuverability, and skid-steer turning. The robot's physical characteristics (i.e. chassis dimensions, track size and Center of Gravity location) and powertrain requirements (i.e available torque) are considered as variables in this analysis. The resulting model predictions are validated by currently fielded Explosive Ordnance Disposal (EOD) robots and a custom skid-steer tank-like robot. The validated quasi-static models are then used in later chapters to populate numerous designs rapidly and to iteratively solve for optimal chassis and powertrain designs based upon a predetermined set of maneuverability requirements and/or mission profile.

2.1 Introduction

Ground robots today are regularly fielded for explosive ordnance disposal, surveillance, inspection, and search and rescue. These robots are built to complete a mission or built for a set of specific capabilities, yet few design tools exist at present that translate a mission and operational environment into robot design constraints and requirements.

In the 20th century, significant research went into predicting how large-scale tracked and wheeled vehicles interact with their environment in an effort to predict vehicle capabilities and performance. Bekker's work is most notable and he studied automobiles, tracked vehicles, large scale construction equipment and military transports and even lunar rovers in an effort to describe how a vehicle's performance scales as a function of its size [1]-[3]. The majority of Bekker's contributions to the field are taken from measuring the properties of fielded vehicles. He looked into a generalized method based on geometric features to predict off-road failure modes to predict a vehicles ability to traverse uneven surfaces. Wong's research furthers

Bekker's on large scale vehicles with a primary emphasis on more detailed modeling of vehicle behaviors and dynamics [4]. The NATO Reference Mobility Model (NRMM) and the Army Reference Mobility model add to the experimental side of off-road research by characterizing soil terrain interaction for fielded military vehicles [5]. The focus of this NATO work was to describe and parameterize large scale off-road vehicle performance as a function of characteristics such as: vehicle mass, vehicle speed, terrain, moisture composition, and obstacles being traversed.

More recently Dr. J.G. Hetherington, *et al*, at Cranfield University developed similar work in the field of terramechanics which described modeling techniques for the soil terrain interaction of tanks and other large scale vehicles [6]-[7]. Modeling of these interactions even extends to describing soil-terrain interaction and performance of small scale electrically driven skid steer robots [8]. However, these modeling techniques are highly specialized which require nearly every parameter of the robot's drivetrain to be defined for the model to be useful. Thus these results are of limited use for generalized design of ground robots.

While the above studies have developed relationships for large scale vehicles, they are far less accurate in predicting capabilities of small mobile robots. Because these relationships are primarily derived by the fitting of experimental data, it is unclear whether they extend to small robots. Due to these shortcomings the US Army Corps of Engineers, government agencies, and independent entities, have put considerable effort into running performance evaluation testing of commercially available Explosive Ordnance Disposal (EOD) robots [9]. McBride, et al, have taken performance testing of mobile robots a step further to include models that predict the step, slope, and ditch traversal of a tracked mobile robot [10]. Commercially available computational dynamic models, which required detailed CAD drawings, have been used to compute the relative capabilities of a given robot [8]. The work presented in this chapter discuses computationally efficient methods to predict a mobile robot's performance without specifying extraneous details of a given robot's design.

Developing a formalized method to predict mobile robot capabilities is a useful tool in the design process. Historically, robot design has been a process which involves building a robot, testing its capabilities (climbing, maneuvering, endurance, etc.), evaluating its relative

performance and iteratively redesigning the robot until an acceptable prototype is achieved. With the use of mathematical simulation tools, designers can predict a robot's performance prior to testing a prototype which in turn reduces costs associated with labor, materials, and assembly and shortens the design process.

This chapter demonstrates a formalized method to calculate several performance capabilities and validates these models against experimental data from a number of robot platforms. The models present comparisons of design capabilities such as climbing and maneuvering which give designers the ability to examine tradeoffs. A sensitivity analysis for each design can be performed to determine if small alterations in the physical design of a robot can yield substantial tradeoffs in performance.

To make these predictions, Section 2.2 outlines the parameters which must be defined to predict a robot's performance. Section 2.3 describes the formulation of each model, the assumptions made and the capabilities which are predicted. The mobile robots used to validate these models are described in Section 2.4 with the corresponding results in Section 2.5. The models are compared with the experimental data in Section 2.6 with a discussion of the results, implications and conclusions drawn.

2.2 Physical Considerations

There are, in general, two approaches to predict robot capabilities: using first principles of physics and experimentation. The robot's characteristics which must be defined in order to predict the robot's capabilities are as follows: vehicle mass, overall size, Center of Gravity (CG) location, drive motor torque, and track geometry. In addition to defining the characteristics of the robot, the static coefficient of friction between the treads and ground must be defined. This constant is readily available because the contact patch between the tracks and ground is assumed to be ideal and equivalent to the surface area of the tracks. With these inputs defined, the model predicts the peak traversal performance and the smallest environment through which the robot can successfully maneuver.

Four robot platforms are used to generate experimental data that is used to validate and augment rule sets derived in this chapter. For example a rule to predict the necessary width of a

hallway is derived using both geometric information and measured path variability information taken experimentally for one platform. The rule is used to predict the minimum hallway required for other robots and validated against experimental data on these new platforms. The Talon, Bombot and RONS are three currently fielded EOD platforms used to measure experimental data as well as the Tankbot, a custom skid-steered tracked robot. Certain models, such as hallway maneuverability, take the averaged experimental results across all of the platforms. Other rule sets, such as the zero radius turn capability, use the four platforms to generate a linear correlation between mass and required motor torque. Experimental data taken from the four robots are also used to validate the analytic model predicting the robot's climbing capabilities.

2.3 Capabilities

2.3.1 Slope Climbing

One of the most basic motions a ground robot must achieve is to climb up a slope with an incline, θ_{climb} (Figure 2-1.A), or traverse the side of a sloped plane with an incline, $\theta_{traverse}$, (Figure 2-1.B).

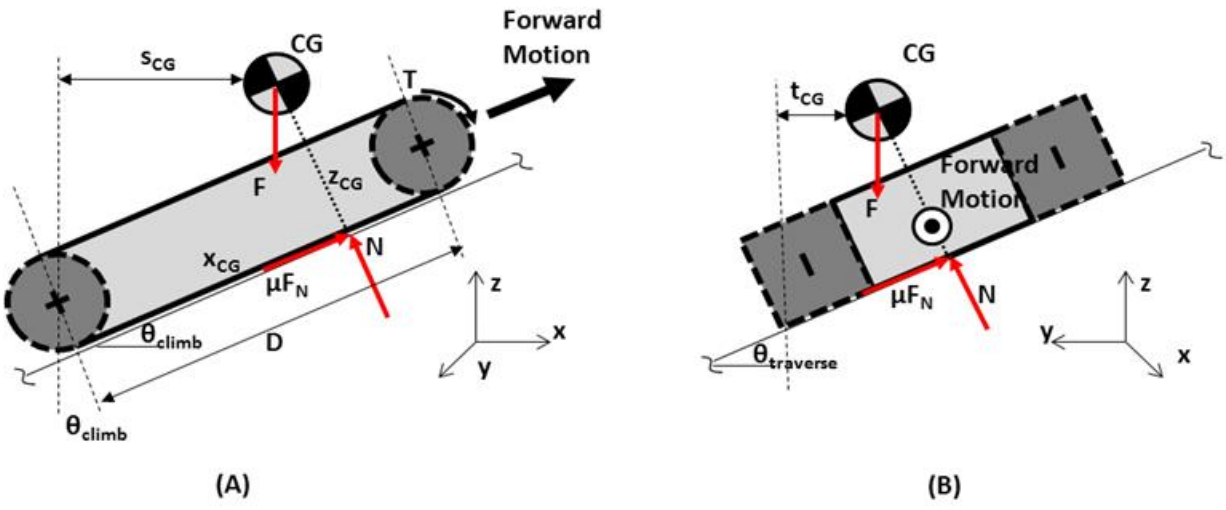

Figure 2-1: Slope Climbing and Traversal

There are three failure modes which dictate a robot's ability to climb up a given hill: the robot can flip backwards due to a poor center of gravity location, the robot can stall due to insufficient torque to climb, and the robot can skid in place due to the climbing surface being too slick to climb. These failure modes define three angles of critical concern. Equation [2-1][2-1] describes the angle, $\theta_{max,climb\_CG}$, at which a robot's center of gravity is directly over its aft axle. [2-1]

$$\theta_{\max,climb\_CG} = \tan^{-1}\left(\frac{x_{CG}}{z_{CG}}\right) \quad \textbf{[2-1]}$$

For any incline greater than $\theta_{max,climb\_CG}$, the robot's CG moves beyond this equilibrium point causing the robot to pivot about its aft axle and tip backwards. For the hill climbing condition, the CG limited case is the inverse tangent of the ratio of the center of gravity location in the x-axis, denoted x_{CG} (m), and the z-axis, denoted z_{CG}. For an incline greater than $\theta_{max,climb\_torque}$, the robot's drive motors are then unable to supply enough torque, T_{motor} (N-m), to overcome the resistive forces from its mass and it will not be able to traverse the incline due to a torque limited condition. This is calculated in Equation [2-2].

$$\theta_{\max,climb\_torque} = \sin^{-1}\left(\frac{n_{motor} \cdot T_{motor}}{r \cdot m_{robot} \cdot g}\right) \quad \textbf{[2-2]}$$

The torque-limited hill climbing condition is shown to also be a function of the number of drive motors, n_{motor}, the radius of the drive sprocket, r (m), and the mass of the robot, m_{robot} (kg). Finally, the angle at which the surface and track friction limit the robot's ability to climb a hill is defined as $\theta_{max,climb\_friction}$ and derived to be a geometric function of the static friction between the surfaces, μ, in Equation [2-3].

All three limiting angles – due to CG, torque and friction – are not dependent upon one another. For example, the equation to predict the CG limited case is not affected by the amount of torque or friction on the ground. The smallest angle of the three conditions, ($\theta_{max,climb\_CG}$, $\theta_{max,climb\_torque}$ or $\theta_{max,climb\_friction}$), is considered to be the limiting factor and therefore the steepest hill climbable for the robot as presented in Equation [2-4].

$$\theta_{\max,climb\_friction} = \sin^{-1}(\mu) \quad \textbf{[2-3]}$$

$$\theta_{\max,climb} = \min(\theta_{\max,climb\_CG}, \theta_{\max,climb\_torque}, \theta_{\max,climb\_friction}) \quad \textbf{[2-4]}$$

2.3.2 Slope Traversal

For a robot with quasi-static dynamics and no (or very stiff) suspension, the ability of a robot to drive longitudinally along a sloped plain, slope traversal, is limited by two factors: the robot's CG location and the friction between the robot and the ground. Again, these two factors define two critical traversal angles. The maximum angle of an incline a robot can traverse, $\theta_{max,traverse\_CG}$, occurs when $t_{CG}<0$ (Figure 2-1.B). The CG of the robot pivots at the edge of the tracks and topples over at any angle greater than $\theta_{max,traverse\_CG}$ (Equation [2-5]. The angle, $\theta_{max,traverse\_friction}$, at which the robot will begin to slide down the hill is computed to be a static coefficient of friction, μ, between the tracks and the terrain (Equation [2-6]). The lesser of these two slopes is the limiting factor and therefore the steepest incline a mobile robot can traverse as shown in Equation [2-7].

$$\theta_{\max,traverse\_CG} = \tan^{-1}\left(\frac{\frac{1}{2}W_{robot} - y_{CG}}{z_{CG}}\right) \quad \textbf{[2-5]}$$

$$\theta_{\max,traverse\_friction} = \sin^{-1}(\mu) \quad \textbf{[2-6]}$$

$$\theta_{\max,traverse} = \min(\theta_{\max,traverse\_CG}, \theta_{\max,traverse\_friction}) \quad \textbf{[2-7]}$$

2.3.3 Step Climbing

There are two situations which are modeled to determine a robot's ability to climb up and over a step. The first condition (Figure 2-2, Diagram A) shows the robot climbing up the face of a step by driving forward into the step. The second condition, (Figure 2-2, Diagram B) occurs

once the robot has climbed up the face of the step where the front axle is past the step edge. The robot completes the step climb when it drives forward and its CG passes over the edge of the step (x-axis) causing the robot to tip onto the top of the step. In each of the two step climbing situations, the robot can be limited by its CG location, available torque, and the friction between the robot and the ground. The robot is considered to be unable to complete a step climb if it is unable to either climb up onto the step or over the step.

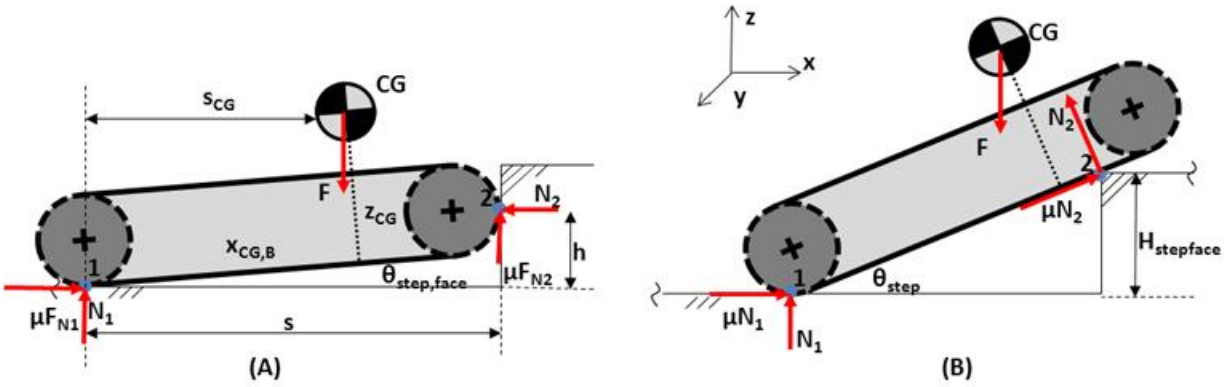

Figure 2-2: Step Climbing Conditions

2.3.3.1 Climbing Step Face

The largest step face angle a robot can climb without tipping backwards, $\theta_{max,stepface\_CG}$, is calculated in Equation [2-8]. It is a function of the robot's CG location and track geometry. A static force analysis is used in Equation [2-9] to relate the angle at which the torque of the motors are no longer able to move the robot up the step face. This analysis assumes that the robot does not tip backwards and that friction does not limit the robot's ability to climb.

$$\theta_{\max,stepface\_CG} = \tan^{-1}\left(\frac{x_{CG}}{z_{CG} - r}\right) \quad \textbf{[2-8]}$$

$$\theta_{\max,stepface\_torque} = \sin^{-1}\left(\frac{n_{motor} \cdot T_{motor}}{r \cdot m_{robot} \cdot g}\right) \quad \textbf{[2-9]}$$

A static force and moment analysis is used to compute a robot's maximum climbable angle up a step face when limited only by friction. Figure 2-2.A shows the robot in contact with a step at two points. While climbing the step face, the robot will be in contact at the step face (Point 2) which is located at the height of the front axle and on the ground in front of the step (Point 1). The static force and moment analysis was conducted by summing the forces in both the x and z axis and by summing the moments at the two points where the robot contacts the ground (points 1 and 2). To compute the moments about the contact points, the distance from the aft axle to the CG (x-axis), the distance from the aft axle to the step and the height of contact point (s_{CG}, s and h respectively) must be known. Figure 2-2 shows how each distance on a free body diagram, and Equation [2-10]-[2-12] presents the formulas for these distances. Each distance is completely dependent upon the geometry of the robot and the angle the robot has climbed up the step face. In addition to the CG location and the radius of the drive sprockets, the distance from the rear to aft axle, D, is also required.

$$s_{CG} = r \cdot \sin(\theta) + x_{CG} \cdot \cos(\theta) - z_{CG} \cdot \sin(\theta) \quad \textbf{[2-10]}$$

$$s = (D + r) \cdot \cos(\theta) \quad \textbf{[2-11]}$$

$$h = D \cdot \sin(\theta) + r \quad \textbf{[2-12]}$$

The four equations from the static force analysis can be used to relate the following four unknowns to one another: the normal forces at point 1 and 2, the friction at each point, and the angle the robot. Through algebraic manipulation of these four equations, the maximum angle a robot can climb up the face of a step can be calculated, is the angle at which the robot is limited by the friction. This angle is related to the geometry of the robot (x_{CG}, z_{CG}, r, and D) and friction. This represents the physical condition where the robot is limited by the friction between the step and the track, where this friction is no longer great enough to overcome the force exerted by the weight of the vehicle. The maximum climbable angle limited by friction, $\theta_{max,stepface\_friction}$ in Equation [2-13], occurs when the friction cannot support the mass of the robot at that given incline. Three constants ($A_{stepface}$, $B_{stepface}$ and $C_{stepface}$) are used to simplify this equation and computed in Equations [2-14]-[2-16]. These three constants are calculated from the geometry of the robot and the static friction between the tracks and the step.

$$\theta_{\max,stepface\_friction} = \sin^{-1}\left(\frac{-C_{stepface}}{\sqrt{A_{stepface}{}^2 + B_{stepface}{}^2}}\right) - \sin^{-1}\left(\frac{B_{stepface}}{\sqrt{A_{stepface}{}^2 + B_{stepface}{}^2}}\right) \quad \textbf{[2-13]}$$

$$A_{stepface} = \left[D \cdot \mu - r \cdot \mu^2 + z_{CG} \cdot \mu^2 - r + z_{CG}\right] \quad \textbf{[2-14]}$$

$$B_{stepface} = \left[D \cdot \mu^2 + r \cdot \mu^2 - x_{CG} \cdot \mu^2 - x_{CG}\right] \quad \textbf{[2-15]}$$

$$C_{stepface} = \left[r \cdot \mu\right] \quad \textbf{[2-16]}$$

2.3.3.2 Climbing Over a Step

Once the robot is partially on the step (Figure 2-3), it will drive forward bringing its aft axle closer to the step face. During this operation the angle of the robot, θ_{step}, will increase until one of two conditions occurs. Either a successful step climb occurs when the robot's CG will move forward (to the right in Figure 2-3) and past the edge of the step and the robot comes to rest on top of the step. Or the robot's CG can move aft during the climb, past the aft axle in the x-axis (move left in Figure 2-3) before the robot passes the edge of the step. When this aft motion of the CG occurs, the robot will fall backwards and come to rest upside down, also known as "turtle backing."

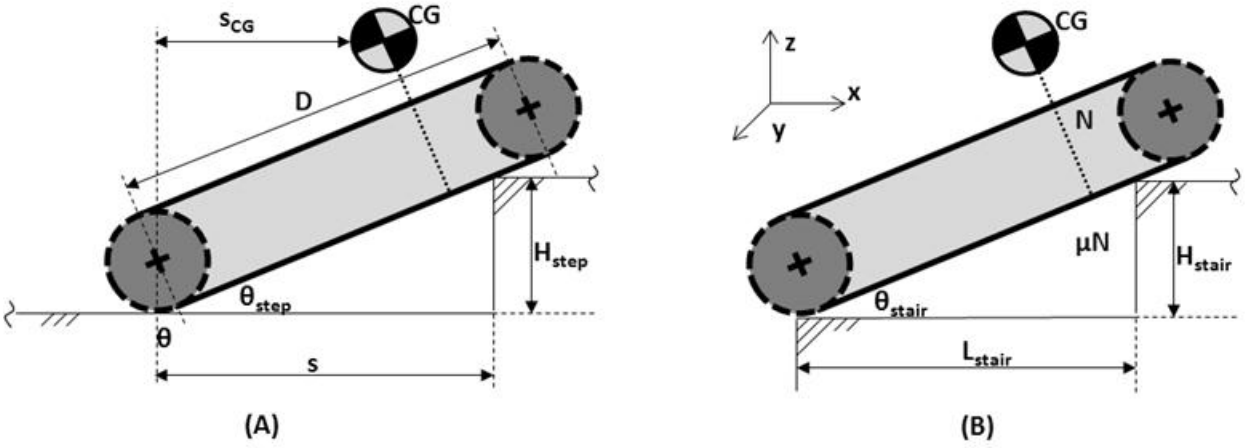

Figure 2-3: Step and Stair Climbing Geometry

The critical condition where the robot is about to pass over the edge of the step occurs when the distance from the aft axle to the CG of the robot, s_{CG}, is equal to the distance from the aft axle to the step face, s. To compute the angle at which this condition occurs, s_{CG}, (Equation [2-17]), and s, (Equation [2-18]), are set equal to one another and rearranged to solve for the height of the step, H_{step}, (Equation [2-19]).

$$s_G = r \cdot \sin(\theta_{step}) + x_{CG} \cdot \cos(\theta_{step}) - z_{CG} \cdot \sin(\theta_{step}) \quad \textbf{[2-17]}$$

$$s = r \cdot \sin(\theta_{step}) + (H_{step} - r + r \cdot \cos(\theta_{step})) \cdot \cot(\theta_{step}) \quad \textbf{[2-18]}$$

To solve this equation for the maximum step height possible, the derivative of Equation [2-19] is taken with respect to θ_{step} to compute the maximum step height as shown in Equation [2-20]. A u substitution is used (Equation [2-21]) where u is set equal to the tangent of half of the angle of the robot, θ_{step}. Using this substitution, Equation [2-22] and [2-23] then describe the substitution for the sin and cos of the step angle. The derivative of Equation [2-19] using the u substitution thus reduces the equation of interest to a quartic equation (Equation [2-24]).

$$H_{step} = r \cdot (1 - \cos\theta_{step}) + x_{CG} \cdot \sin(\theta_{step}) - z_{CG} \cdot \sin(\theta_{step}) \cdot \tan(\theta_{step}) \quad \textbf{[2-19]}$$

$$\frac{dH_{step}}{d\theta_{step}} = x_{CG} \cdot \cos(\theta_{step}) + (r - y_{CG}) \cdot \sin(\theta_{step}) - y_{CG} \cdot \frac{\sin(\theta_{step})}{\cos^2(\theta_{step})} \quad \textbf{[2-20]}$$

$$u = \tan\left(\frac{\theta_{step}}{2}\right) \quad \textbf{[2-21]}$$

$$\sin(\theta_{step}) = \frac{2u}{(1+u^2)} \quad \textbf{[2-22]}$$

$$\cos(\theta_{step}) = \frac{(1-u^2)}{(1+u^2)} \quad \textbf{[2-23]}$$

$$(-x_{CG})u^6 + (-4z_{CG} + 2r)u^5 + (3x_{CG})u^4 + (-4r)u^3 + (-3x_{CG})u^2 - (4z_{CG} - 2r)u - (x_{CG}) = 0 \quad \textbf{[2-24]}$$

A closed-form solution of this equation is not generally possible. However, numerical solutions are readily obtained. If the quartic has four imaginary roots in Equation [2-24], then the robot

cannot climb a step. Otherwise, the smallest real positive root will correspond to an equation for a value u that gives the largest climbable angle, using Equation [2-25].

$$\theta_{\max,step\_CG} = 2\tan^{-1}(u) \quad \textbf{[2-25]}$$

Similar to the step face torque limited condition, Equation [2-26] is the largest angle traversable for a robot with a given mass and available torque.

$$\theta_{\max,step\_torque} = \sin^{-1}\left(\frac{n_{motor}\cdot T_{robot}}{r\cdot m_{robot}\cdot g}\right) \quad \textbf{[2-26]}$$

Finally, Equation [2-27] relates the friction limited maximum angle traversable, $\theta_{max,step\_friction}$. This equation is derived by using a force and moment balance on a static robot in Figure 2-3. This angle represents the point at which the robot's weight is in balance with the frictional forces holding it in place at points "1" and "2" on the right diagram in Figure 2-2.

$$\theta_{\max,step\_friction} = \frac{1}{2}\left(\sin^{-1}\left(\frac{1}{\sqrt{\mu^2+1}}\left(\frac{2s\cdot\left(s_{CG}-\mu\cdot H_{step}\right)}{s_{CG}\cdot\left(s-\mu\cdot H_{step}\right)}-1\right)\right)-\tan^{-1}\left(\frac{1}{\mu}\right)\right) \quad \textbf{[2-27]}$$

The step height, H_{step}, in Equation [2-27] is calculated using the results from the tipping condition angle from the previous limiting climbing condition. The height of the step calculated in Equation [2-28] is a function of both the angle of the robot and the distance from its rear axle to the step face.

$$H_{step} = s\cdot\sin\left(\theta_{\max,step\_CG}\right) \quad \textbf{[2-28]}$$

2.3.3.3 Calculating Step Height

Similar to hill climbing, the robot's CG location, available torque and surface friction are the three limiting factors which determine the largest step the robot can successfully traverse. These three limiting factors occur while the robot is climbing up the step face and over the step.

$$\theta_{\max,step} = \min\begin{pmatrix} \theta_{\max,stepface\_CG}, \theta_{\max,stepface\_torque}, \theta_{\max,stepface\_friction}, \\ \theta_{\max,step\_CG}, \theta_{\max,step\_torque}, \theta_{\max,step\_friction} \end{pmatrix} \quad \textbf{[2-29]}$$

The largest angle the robot can climb is equal to the smallest angle that the robot can climb under all three failure modes in both climbing conditions (Figure 2-2.A-B). The largest step a robot can climb is a function of the angle it can climb, $\theta_{max,step}$. The maximum angle the robot can climb is the smallest of the six possible failure modes, Equation [2-29]. The smallest angle among all these calculated angles corresponds to the first failure mode that will limit the height of the step and therefore the robot's limiting climbing capability. The robot has three possible limiting factors for two conditions, thus giving a total of six possible limiting factors.

$$H_{step} = r + D \cdot \sin(\theta_{\max,step}) \quad \textbf{[2-30]}$$

The height of the step, H_{step} in Figure 2-3, a robot can climb is correlated to the robot's geometry and the largest angle it can climb without tipping over. With the largest angle the robot can climb without failure calculated in Equation [2-29], the corresponding maximum step height the robot can successfully climb is calculated in Equation [2-30]. The largest step a robot can climb is related to the robot's track width of the robot, D, the largest angle traversable, $\theta_{max,step}$, and the robot's drive sprocket radius, r.

2.3.4 Stair Climbing

The traversal of a flight of stairs can be accomplished a number of ways for robots of various sizes and geometries. Attempting to model all possible means of stair climbing is beyond the scope of this work and requires detailed knowledge about the robot design and control. For example, a robot often rocks and tips from one step to another or traverses diagonally across a flight of stairs in a zig-zag pattern to effectively increase the robot's equivalent track length. Such modifications can be very difficult to model.

However, even ignoring these special situations and with limited information about the chassis, it is still possible to provide an accurate estimate of the robot's capabilities. The two climbing conditions which are modeled here include climbing a single step at a time where the

robot can "rest" on each step, and climbing multiple steps at a time where the robot spans across the gaps of at least two steps. For a robot to span multiple steps it must meet the condition in Equation [2-31].

$$D \geq 2\left(H_{stair}{}^{2} + L_{stair}{}^{2}\right)^{\frac{1}{2}} \quad \textbf{[2-31]}$$

With the geometry of both the robot and the step known, one must first determine from geometry whether the robot can span multiple steps. If it cannot span multiple steps, one must calculate the rise and run of the largest step it can climb using the equations in Section 2.3.4.1.

If the robot can span multiple steps at a time, one must calculate the largest steps the robot can climb using Section 2.3.4.2. Each section has criteria which must be satisfied when determining a robot's ability to successfully climb a given flight of stairs.

2.3.4.1 Climbing a Single Step

The rise ($H_{stair,ss}$) of each stair (Figure 2-3) is equal to the maximum traversable height of a single step as shown in Equation [2-32]. In order for the robot to stop on each step, the length of the step, $L_{stair,ss}$, must be greater than the distance from the robot's CG to its front edge (Equation [2-33]). Since a robot can climb stairs backwards, future calculations often consider the minimum distance from the CG to either the front or rear contact point. With the rise and run of the step given, the overall incline of the step is calculated in Equation [2-34].

$$H_{stair,ss} = H_{step} \quad \textbf{[2-32]}$$

$$L_{stair,ss} = \min\left(x_{CG} + r, L_{robot} - \left(x_{CG} + r\right)\right) \quad \textbf{[2-33]}$$

$$\theta_{stair,ss} = \tan^{-1}\left(\frac{H_{stair,ss}}{L_{stair,ss}}\right) \quad \textbf{[2-34]}$$

To successfully climb a flight of stairs which the robot cannot span, two conditions must be satisfied. The first condition is that the actual length of a single stair, L_{stair}, must be greater than the predicted length, $L_{stair,ss}$ (Equation [2-35]). The second condition is that the height of the actual height of a stair, H_{stair}, must be less than the calculated stair height, $H_{stair,ss}$. (Equation [2-36]).

$$L_{stair} \geq L_{stair,ss} \quad \textbf{[2-35]}$$

$$H_{stair} \leq H_{stair,ss} \quad \textbf{[2-36]}$$

2.3.4.2 Spanning Multiple Steps

For spanning multiple steps, the largest height of a stair climbable, $H_{stair,span}$, is equal to the largest step climbable (Equation [2-37]). The new overall incline of the steps, $\theta_{stair,span}$, is now equal to the largest slope traversable from the hill climbing criterion (Equation [2-38]). The new run of the stair, $L_{stair,span}$, is the ratio of the height of the stair and the tangent of the stair incline (Equation [2-39]). Under these conditions, when spanning multiple steps, the length of the new stair, $L_{stair,span}$, should be less than the length of the single stair climbing condition, $L_{stair,ss}$, e.g. the distance required the robot to stop on each step.

$$H_{stair,span} = H_{step} \quad \textbf{[2-37]}$$

$$\theta_{stair,span} = \theta_{\max climb} \quad \textbf{[2-38]}$$

$$L_{stair,span} = \frac{H_{stair,span}}{\tan\left(\theta_{stair,span}\right)} \quad \textbf{[2-39]}$$

To successfully climb a flight of stairs where the robot spans at least two steps at a time, two conditions must be satisfied. The first condition is that the actual length of a single stair must be greater than the predicted length, $L_{stair,span}$ (Equation [2-40]). The second condition is that the height of the actual height of a stair must be less than the calculated stair height, $H_{stair,ss}$. (Equation [2-41]).

$$L_{stair} \geq L_{stair,span} \quad \textbf{[2-40]}$$

$$H_{stair} \leq H_{stair,span} \quad \textbf{[2-41]}$$

2.3.5 Ditch Traversal

A robot's ability to successfully traverse a ditch with an infinite depth is a function of the robot's geometry and CG location as shown in Figure 2-4. The largest ditch a robot can successfully traverse is limited by the shorter of the two distances from the robot's CG to its

front or aft axle (Equation [2-42]). The robot's ability to cross a ditch in this model is independent of both the robot's speed, the friction between the tracks and surface of ditch, and the depth of the ditch.

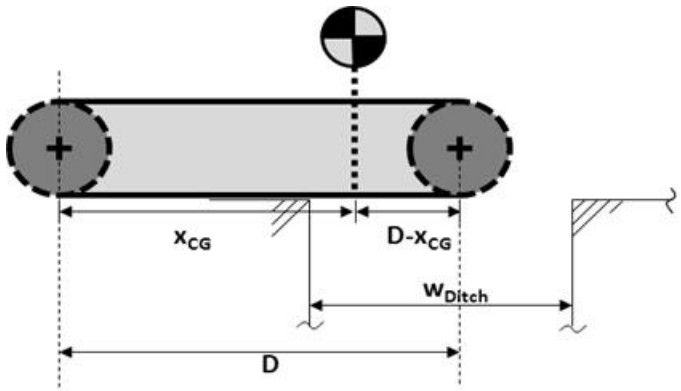

Figure 2-4: Ditch Traversal Capability

$$w_{Ditch} = \min\left(x_{CG}, D - x_{CG}\right) \quad \textbf{[2-42]}$$

2.3.6 Zero Radius Turn

The torque required to turn a skid-steer robot in place, a zero radius turn, is a function of both the width of the robot, W_{robot}, length of the robot, L_{robot}, and it's mass. A static force analysis was used to predict this capability by using measured data from both tracked and wheeled robot platforms (Figure 2-5). During a skid-steer zero radius turn, the assumption is made that the greatest amount of torque required is the amount necessary to initiate the turn and overcome static friction within the robot's drivetrain and between the track and surface.

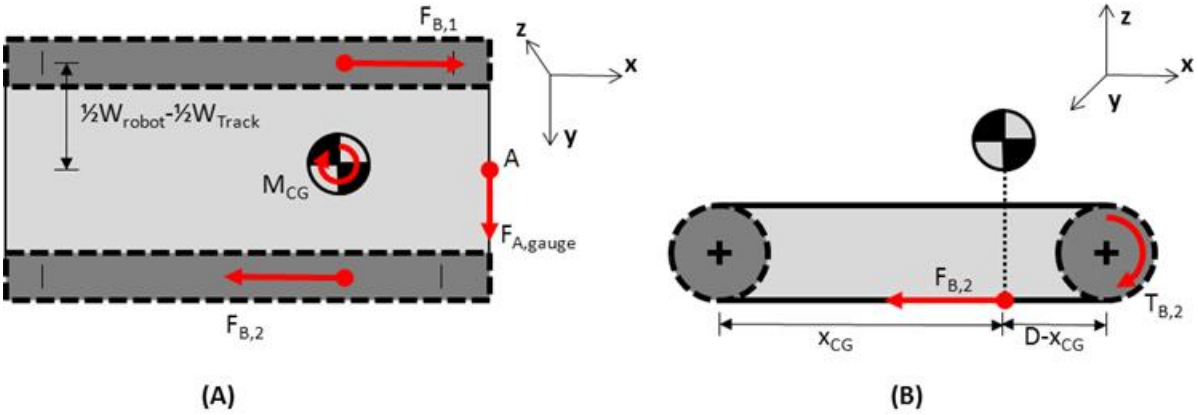

Figure 2-5: Zero Radius Turn Static Analysis

To determine the necessary force for a skid-in place, one must solve a calculus problem representing the changing moment along the contact patch. This is problematic given that the surface friction characteristics of most soils are nonlinear and hard to describe analytically. To overcome these issues and obtain turning force estimates, experiments were conducted to directly measure the moments necessary to spin a robot in place. A load cell was fixed to the front of each platform and aligned with each platform's center of gravity in both the y and z-axis. The maximum force, $F_{A,gauge}$, required to rotate the robot about its center of gravity occurs at the instant prior to each platform beginning to turn. This measurement was repeated several times on three different platforms on both grass and asphalt. The required force, $F_{A,gauge}$, for each platform was found to be well described by a linear fit to the mass of the platform on both surfaces. Through a static force analysis, using the geometry of each robot, the required force was related to the required torque for each motor, $T_{motor,ZRT}$, shown in Equation [2-43]. This model assumes that the forces exerted by the tracks occur at half the robot's track width, W_{track}.

$$T_{motor,asphalt} = \frac{\alpha \cdot r \cdot m_{robot} \cdot g \cdot \left(L_{robot} - x_{CG} - r\right)}{n_{motor} \cdot \left(\tfrac{1}{2} W_{robot} - \tfrac{1}{2} W_{tracks}\right)} \qquad \textbf{[2-43]}$$

In Equation [2-43], α represents the experimental coefficient relating the measured force, $F_{A,gauge}$, to the mass of the robot. The coefficient α was found experimentally to be 0.36 for asphalt and 0.53 for grass. The coefficient α represents the experimental ratio of force required to pull the robot in a circle, $F_{A,gauge}$, to the force of the robot exerted by its mass. This relationship implies that, on asphault, for every Newton of force exerted by the robot's mass, there will be 0.36

Newtons of force required to spin the robot in place, if the force is applied to the front edge of the robot. Again, this coefficient is the average obtained from several experimental data sets taken from a number of ground robots.

2.3.7 Drag Capability

A robot's ability to drag an object is limited either by the drive motor torque or the friction between the robot and the surface it is traversing. The amount of weight a robot can drag behind it, limited by friction, is proportional to the mass of the robot and varies with terrain. Using a load cell, the force required for each robot to overcome static friction was measured on three robot platforms and seven different surfaces. The robot drag coefficients, $\delta_{terrain}$ in Table 2-1, relate the mass of a robot, m_{robot}, to the experimentally measured force required to drag the robot, $m_{drag,force}$.

$$m_{drag,\,friction} = \delta_{terrain} \cdot m_{robot} \qquad \textbf{[2-44]}$$

Table 2-1: Drag Force Coefficients

Surface	$\delta_{Terrain}$
Tile	0.42
Concrete	0.59
Grass	0.72
Asphalt	0.59
Gravel	0.59
Dirt	0.60
Clay	0.61

The amount of weight a robot can drag when limited by torque, $m_{drag,torque}$, is proportional to the number of motors, drive motor torque and track geometry seen in Equation [2-45]. The actual mass a robot can drag, m_{drag}, is the smallest value between the torque and friction limited conditions (Equation [2-46]).

$$m_{drag,torque} = \frac{n_{motor} \cdot T_{motor}}{r \cdot g} \quad \textbf{[2-45]}$$

$$m_{drag} = \min\left(m_{drag,friction}, m_{drag,torque}\right) \quad \textbf{[2-46]}$$

2.3.8 Hallway Maneuverability

The ability of a robot with a given width, W_{robot}, to successfully maneuver through a hallway of a given width, $W_{hallway}$, is a function of both the geometry of the robot and the ability of the operator to limit the robot's lateral deviation from its desired path. Unlike maneuvering in between two point objects, the ability of a robot to maneuver through an infinitely long straight hallway is more complex than confirming that the robot's width is less than that of the hallway. To use a more familiar analogy: a tractor trailer requires a much wider road to drive down that the vehicle's physical width because any small deviation from the path in the front of the vehicle corresponds to a magnified lateral deviation of the trailer from the path. While being operated, tractor trailers, cars and robots alike all deviate from the desired path while maneuvering down a road or hallway. This deviation coupled with the amount of room it takes for the robot to correct its path, combine to give a minimum with of a hallway a robot can maneuver through without hitting the walls. Figure 2-6 demonstrates how a robot commonly oscillates about the path while attempting to be driven in a straight line.

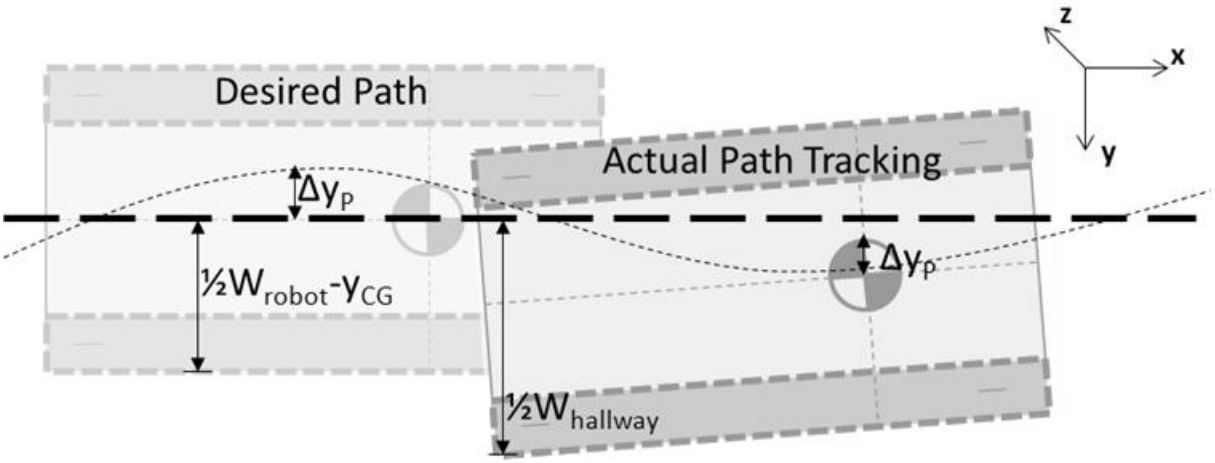

Figure 2-6: Hall Way Maneuverability

The robot's deviation from its desired path is both quantifiable and useful to predict what the minimum hallway width through which it can maneuver without hitting the hallway's walls. The minimum width of the hallway is related to the robot geometry (length, width, CG location) and the amount of lateral deviation of the robot from tracking a straight path. This lateral path deviation or the ability of an operator to track a given path is an experimentally determined value.

To examine later path deviation, four different robots of varying geometries were driven at varying speeds. Their deviation was noted by placing a marker at the CG of each robot (assumed to be the pivot point for a skid-steer robot) and measuring each robot's ability to track a straight path under human control. For each experiment the lateral deviation was recorded in interval equal to approximately the robot's length. It was observed that each robot oscillates to the right and left of the desired path with periods approximately equal to the length of the robot. This means that for example in a distance equal to twice the length of the robot, the robot would usually cross the desired path being track twice. Additionally, it was observed in these fixed-speed experiments that the average lateral deviation is not significantly affected by the size of the robot, the human operator (assuming he/she is trained), nor the operator interface (R/C controller, OCU controllers for disposal robots, etc). The Bombot robot platform (13.2 kg)

traveling at the same speed as a RONS robot platform (307.5 kg) was seen to have approximately equivalent maximum path deviations of 0.057m. Figure 2-7 shows how the width of the hallway, $W_{hallway}$, relates to the measured maximum path deviation, Δy_P, width of the robot, W_{robot}, and length of the robot, L_{robot}.

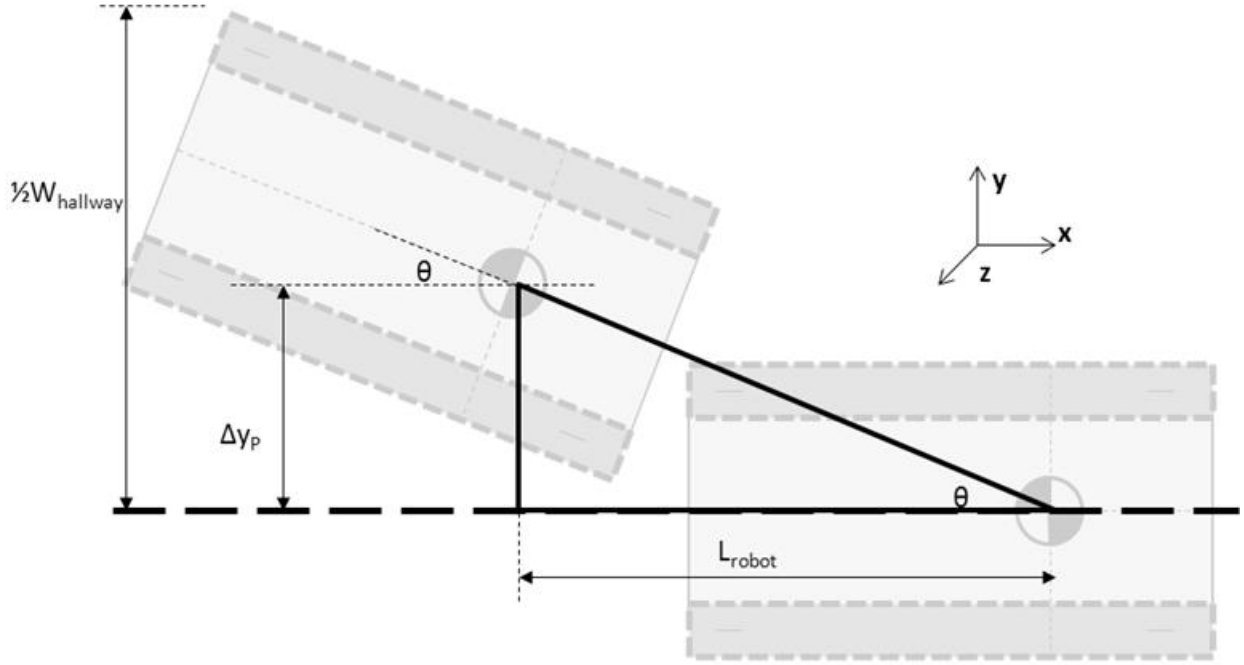

Figure 2-7: Hallway Path Deviation Correction

Experimental results show that a given robot is able to correct its deviated path by the time it travels a distance equal to its length. This allows a relationship between the lateral deviation and hallway width using the angle of rotation, θ. Using this geometric relationship, Equation [2-47] relates the minimum width of a hallway to the geometry of the robot.

$$W_{hallway} = W_{robot} + 2\left(\beta \cdot \sin\left(\tan^{-1}\left(\frac{\Delta y_P}{L_{robot}}\right)\right) + \Delta y_P\right) \qquad \textbf{[2-47]}$$

In Equation [2-47], the maximum path deviation, Δy_P, was measured for each available platform and averaged to be equal to 0.06 m. Assuming the CG is laterally symmetric about the y-axis, the

variable β represents the maximum distance from either the center of gravity in the x-axis to the front or to the rear of the robot as seen in Equation [2-48].

$$\beta = \max\left(x_{CG} + r, L_{robot} - \left(x_{CG} + r\right)\right) \quad \textbf{[2-48]}$$

2.3.9 Doorway Maneuverability

A robot's ability to enter a room or hallway and turn a corner, is modeled as a geometric fit problem in Figure 2-8 assuming that the robot pivots exactly about its CG. Similar to the hallway maneuvering capability, fitting through a doorway or turning a corner in a hallway requires a more complex model than maneuvering between two point objects in an open space. This capability is critical for longer robots which may be capable of climbing a flight of stairs, but incapable of turning on a landing of the stairs.

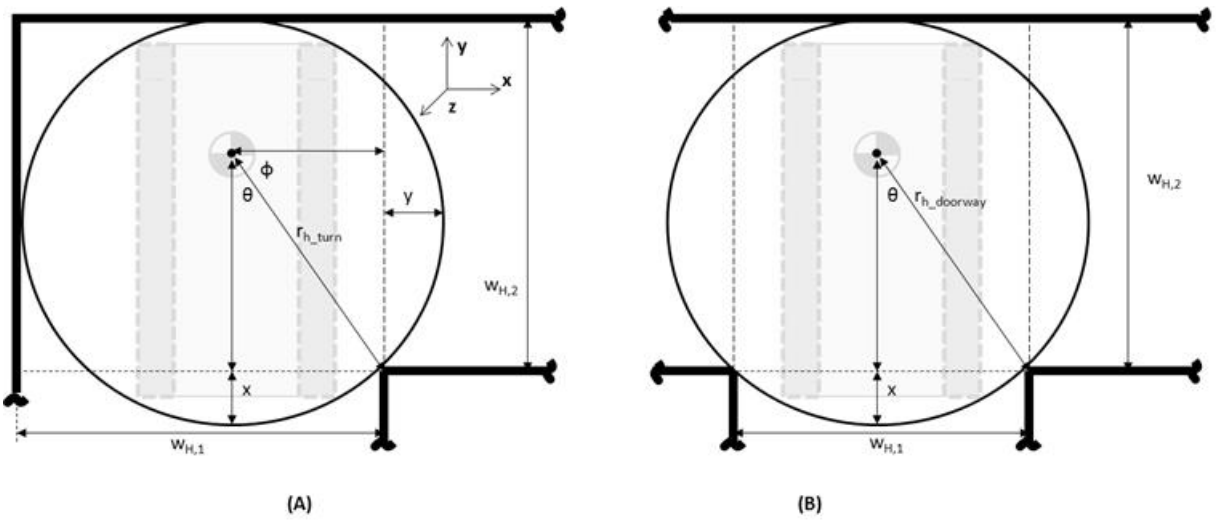

Figure 2-8: Doorway Maneuverability

The width of the doorway/hallway, $W_{H,1}$ and $W_{H,2}$, (Figure 2-8) through which a robot can successfully maneuver, is related to the size of the robot and its CG location. In order to make this geometric correlation, the robot is assumed again to be skid-steer (either track or wheeled) and it can pivot about its CG. The robot is assumed to drive into the open doorway or hallway, pause, pivot to the appropriate turning position, and then drive out of the area.

$$r_{h\_turn} = W_{H,1} + W_{H,2} \pm \sqrt{2 \cdot W_{H,1} \cdot W_{H,2}} \quad \textbf{[2-49]}$$

$$r_{h\_doorway} = \frac{1}{2} W_{H,1} + \frac{1}{8} \frac{{W_{H,1}}^2}{W_{H,2}} \quad \textbf{[2-50]}$$

Equations [2-49] and [2-50] relate the width parameters of the doorway, $W_{H,1}$ and $W_{H,2}$, to the turning radius required for a turn in a hallway, r_{h_turn}, and in a doorway, $r_{h\_doorwary}$. Being that the two widths are independent of one another, there are numerous variations of the two parameters which yield the same radius. The equivalent radius of the robot, r_{robot}, (Equation [2-51]), is the calculated hypotenuse of the two longer sections of the robot's center of gravity in the x and y-axis (Equations [2-52] and [2-53]). To successfully maneuver, the equivalent radius of the turn and doorway must be greater than the equivalent radius of the robot.

$$r_{robot} = \sqrt{\beta^2 + \gamma^2} \quad \textbf{[2-51]}$$

$$\beta = \max\left(x_{CG} + r, L_{robot} - \left(x_{CG} + r\right)\right) \quad \textbf{[2-52]}$$

$$\gamma = \max\left(\tfrac{1}{2} W_{robot} - y_{CG}, \tfrac{1}{2} W_{robot} + y_{CG}\right) \quad \textbf{[2-53]}$$

If the hallways are assumed to be symmetrical, where $W_{H,1}$ is equal to $W_{H,2}$, then Equations [2-49] and [2-50] can be rewritten to relate the width of the turn or doorway to the geometry of the robot. The width of the turn, W_{h_turn}, in Equation [2-54], and the width of the doorway, $W_{h\_doorway}$, in Equation [2-55], then becomes a function of the largest distances of the CG to front and side of the robot, β and δ.

$$W_{h\_turn} = \frac{\sqrt{\beta^2 + \gamma^2}}{2 + \sqrt{2}} \quad \textbf{[2-54]}$$

$$W_{h\_doorway} = 1.6\sqrt{\beta^2 + \gamma^2} \quad \textbf{[2-55]}$$

2.4 Fielded Mobile Robots

As noted previously, the four platforms used to both generate and validate the models in Section 2.4 are the Talon, Tankbot, Bombot, and RONS. A list of the relevant parameters which were used to predict their capabilities are listed in Table 2-2.

Table 2-2: Robot Platform Characteristics

Robot Test Platforms				
	Talon	Tankbot	Bombot	RONS
m_{robot} (kg)	51.50	74.40	13.18	307.50
W_{robot} (m)	0.57	0.55	0.47	0.72
L_{robot} (m)	0.86	0.96	0.64	0.74
x_{CG} (m)	0.30	0.26	0.14	0.21
y_{CG} (m)	-0.01	-0.18	0.00	-0.02
z_{CG} (m)	0.37	0.33	0.21	0.40
D (m)	0.63	0.70	0.31	0.55
r (m)	0.12	0.05	0.10	0.13
n_{motor}	2	2	2	2
T_{motor} (N-m)	45.20	22.98	0.00	0.00
$\mu_{rubber\ to\ concrete}$	0.59	0.59	0.59	0.59
$\mu_{rubber\ to\ grass}$	0.71	0.71	0.71	0.71
W_{track} (m)	0.16	0.08	0.13	0.07

Each of the measurements were taken prior to the robot testing. Not all of the platforms used for testing have the same track configuration assumed in the models, so it was assumed for all tracked models that there is one primary drive sprocket in the front of the platform and another equally large tensioner in the aft of the platform. Sample pictures of each platform tested are displayed in Figure 2-9.

Figure 2-9: Mobile Robot Platforms Used for Testing

The Talon exhibits the track configuration used during modeling while the Tankbot, as seen in Figure 2-9, has a track configuration which closely mirrors that of a tank, e.g. a smaller drive sprocket which is raised above ground level causing a slanted track configuration in the front. The Bombot is a front-wheel-steered, four-wheel-drive platform. The RONS is a tracked vehicle in a similar configuration to the Talon with the addition of articulating tracks also known as “flippers” at the front and the rear. The test data gathered with the RONS was obtained with the flippers raised so that these unmodeled elements are not affecting the platform’s capabilities.

2.5 Results

Using the measured parameters of each available platform (enumerated in Table 2-2), the predicted performance for each platform was calculated. The performance of a platform is separated into three categories. The performance can be a numerical value such as the incline of a slope the robot can climb, a failure mode such as the robot is limited by either friction, torque or by its CG location or as a binary yes/no capabilities such as the robot's ability to turn in place. The comparison of the numerical predicted results the experimental data collected during testing is shown in Figure 2-10-Figure 2-13. Both the failure mode comparison and binary capabilities are location in Table 2-3.

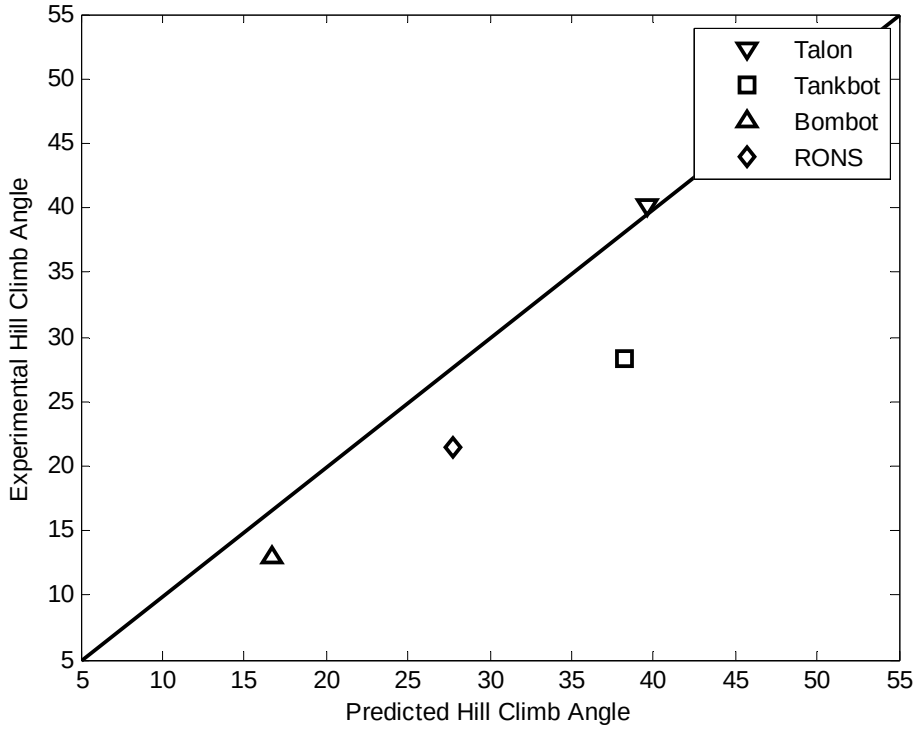

Figure 2-10. Robot Hill Climbing Model Comparison

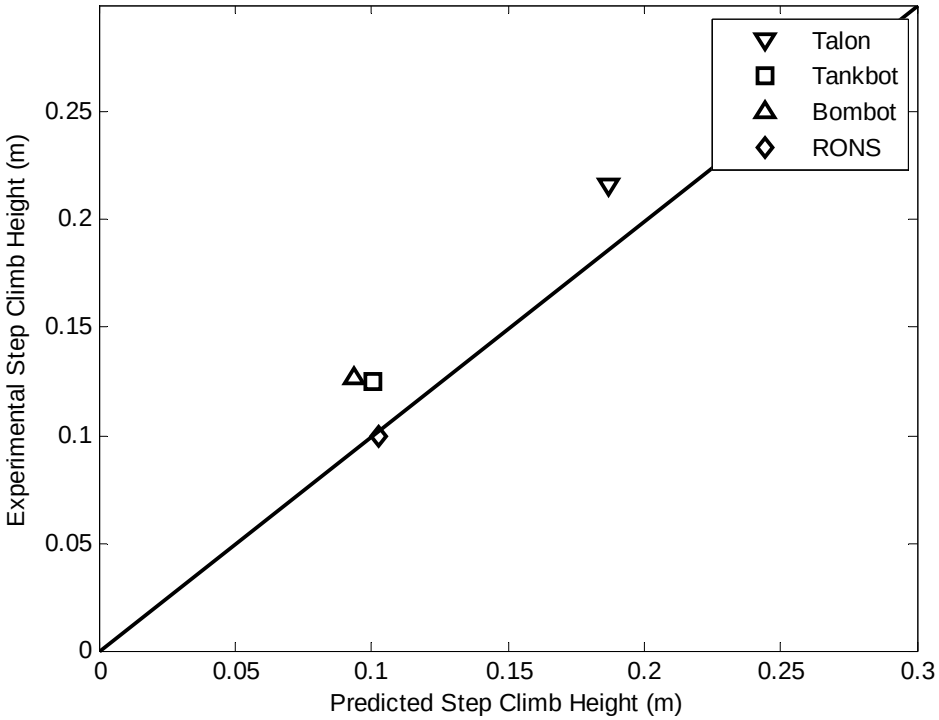

Figure 2-11. Robot Step Climbing Model Comparison

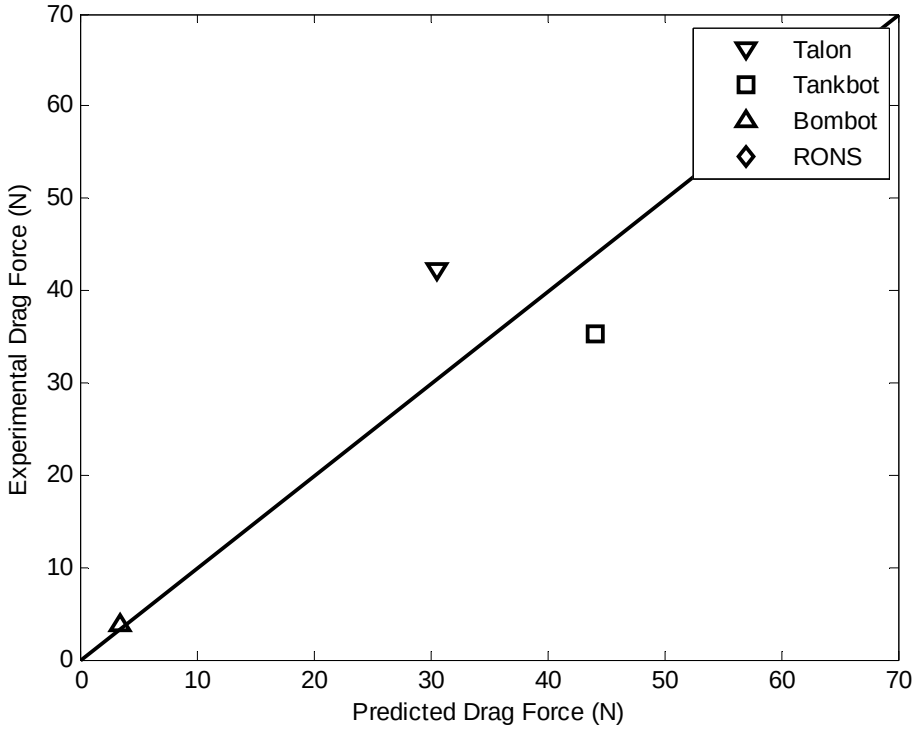

Figure 2-12. Robot Drag Force Model Comparison

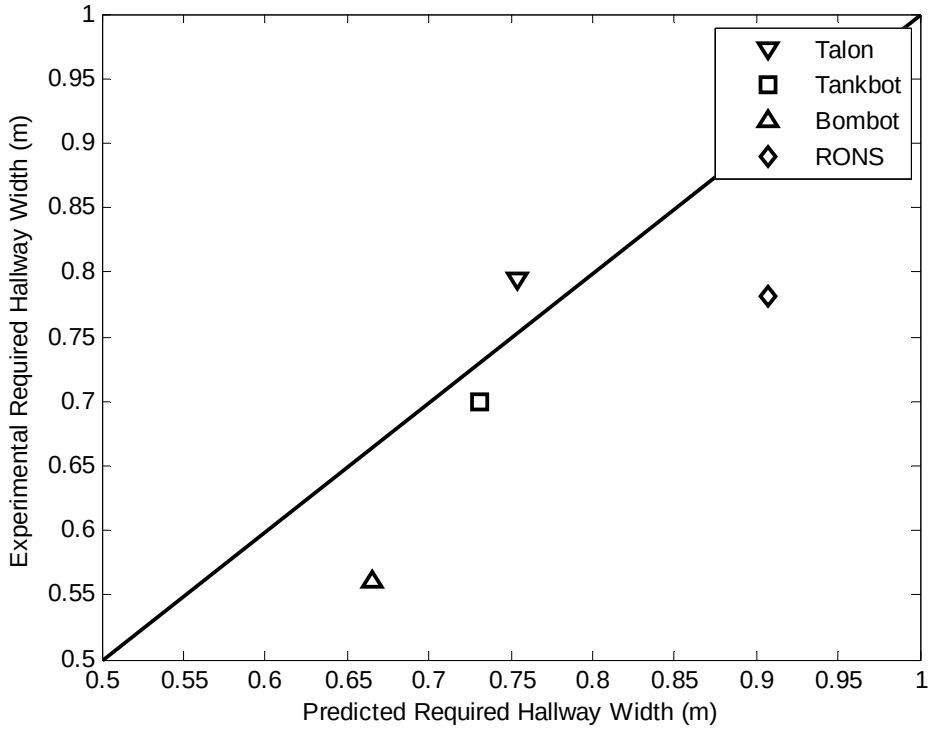

Figure 2-13. Robot Hallway Maneuverability Model Comparison

Overall the predicted performance of the robots is a reasonably accurate approximation to the actual performance. The model approximately predicts the numerical performance limitations of each robot, and also the exact type of performance limitation (torque, CG or friction). There is not a discernable bias towards the predicted performance always being greater than or less than the test data. In one instance the Bombot's actual step climbing ability deviates from the model prediction by approximately 25%. This deviation however only corresponds to a 3cm difference between the tested and actual values and likely is affected more by the deformability of the tire (pneumatic) and the treads of the tire.

Table 2-3: Robot Binary (y/n) Capabilities and Failure Mode Comparison

	Talon		Tankbot		Bombot		RONS	
	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual
Hill Climb Failure Mode	CG	CG	CG	CG	Torque	Torque	CG	CG
Step Climb Failure Mode	Friction	Friction	CG	CG	Torque	Torque	CG	CG
Stair Climb (m) (y/n)	Yes	Yes	No	No	No	No	No	No
Zero Turn Radius (y/n)	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Drag Failure Mode	Friction	Friction	Friction	Friction	Torque	Torque	Friction	-

The primary deviation between the predicted and actual values of the tracked vehicles most likely occurs from the inability to predict the interaction between the tread features and the terrain. Each robot tested has a unique track configuration and tread pattern which is not captured in this modeling effort. As an indicator of this effect: the treads on each robot range in size from a few millimeters, to a track with thin rectangular cleats which sick out over 11 cm from the tread band. If one were to approximate these tread feature interactions during climbing, one could significantly increase the accuracy of these models. Additionally, some of the empirically generated models would benefit from additional data points taken from more robot platforms than the ones used here. This additional work could significantly prove the accuracy of these predictions through the refinement of these predictive capability models.

2.6 Conclusions

Using these models to predict a robot's ground capabilities allows designers to estimate a platform's performance limitations during the design phase rather than during prototype testing. This allows a concept to be evaluated to determine its ability to successfully complete a given mission. These models show that a small change in a design (such as the robot's CG location) can have a significant and quantifiable change in its performance. Moreover a change in one robot characteristic has the potential to propagate changes across numerous capabilities. Many of these capabilities are in opposition with one another. For example: increasing a robot's wheel base helps it climb and traverse ditches, but this change inhibits the robot's ability to maneuver

indoors. These equations can be thus be used to consider performance tradeoffs between desirable capabilities for new robot designs.

2.7 References

[1] Bekker, M. (1969). *Introduction to Terrain-Vehicle Systems*. Ann Arbour: The University of Michigan Press.

[2] Bekker, M. (1960). *Off-the-Road Locomotion*. Ann Arbour: The University of Michigan Press.

[3] Bekker, M. (1969). *Theory of Land Locomotion*. Ann Armbour: The University of Michigan Press.

[4] Wong, J. (2001). *Theory of Ground Vehicles Third Edition.* New York: John Wiley & Sons, Inc.

[5] Ahlvin, R and Haley, P. (1992). NATO Reference Mobility Model Edition II, NRMM User's Guide, Technical Report. GL-92-19.

[6] Hetherington, J. and White, N. (2002). An investigation of pressure under wheeled vehicles. *Journal of Terramechanics,* 39:85-93.

[7] Hetherington, J. (2001). The application of the MMP concept in specifying off-road mobility for wheeled and tracked vehicles. *Journal of Terramechanics*, 38:63-70.

[8] Hetherington, J. (1991). Electric, All-Wheel-Drive, Tracked Vehicles. *Journal of Terramechanics,* 28:79-85.

[9] Richmond, P., Mason, G., Coutermarsh, B., Pusey, J., and Moore, V. (2009) Mobility Performance Algorithms for Small Unmanned Ground Vehicles. Technical Report. ERDC TR-09-6.

[10] McBride, B., Longoria, R., AND Krotkov, E. (2003) Measurement and Prediction of the Off-Road Mobility of Small Robotic Ground Vehicles. *The 3rd Performance Metrics for Intelligent Systems Workshop (PerMIS03).*

Chapter 3

Allometric Design Principles of Ground Robot Powertrains

The purpose of this chapter is to develop general scaling principles for each component used for mobile ground robot locomotion. These components are then combined to create a system model used to predict overall robot locomotion performance, for example the endurance distance of the platform. The robot's powertrain is defined as the components used for locomotion including the robot's power source, motor controller, drive motor, gearbox, and drivetrain (treads or wheels). For each component, scaling factors are developed in this chapter to produce rule sets which describe how each component's physical characteristics (volume and mass) scales as a function of its performance (e.g. power, efficiency, speed, etc.). These rule sets are developed through the use of first principles physics derivation, and the use of experimental data obtained from research into commercially available components. The predicted platform performance using allometric powertrain component models is then validated against a number of currently fielded Explosive Ordnance Disposal (EOD) mobile robots and a custom skid-steer tank-like robot. With these scaling rules derived, a power flow model is then developed to describe how much power is required for a given mission, and in turn how large each component needs to be to achieve a given capability (e.g. climbing, traversal, peak speed, a desired operating distance, etc.).

3.1 Introduction

Ideally, if one can mathematically predict how component-level powertrain changes influence the system-level performance of a robot across a wide range of size scales, then one can implement formalized methods for designing a robot system to achieve a required performance. In order to obtain such mathematical relationships, the linkage of each component in the robot's powertrain must be modeled. This interdependence can be represented as a mechanical and/or electrical chain of once component influencing another, starting from the batteries and ending at the wheels or treads. By specifying the output performance of the system at the wheels/treads side of this causal change, for example a desired speed and/or torque, then each component of the powertrain can be optimally scaled for a robot of a given mass.

One difficulty with the above idealization is that the overall design of such a system is an iterative process. This is due to the cyclical dependencies which exist between each component's mass, power and efficiency. For example, a speed requirement on a robot of a given mass may result in a much larger motor and/or battery being required, which changes the mass of the robot thus necessitating a repeat of the calculation, until convergence is obtained. Typically, it is this iterative dependence of factors on each other that makes the mechanical design of a robot so difficult in practice. However, if one can specify the nonlinear scaling principles relating power to size, speed, and efficiency, then one transform an iterative physical design/build/test process into a software-based computational iteration. This is a fundamental goal of this thesis work.

The most studied component in a robot powertrain is arguably the electric motor. Numerous groups have examined the scaling limitations of Direct Current (DC) motors and other components in a robot's powertrain. Nicoud, for example, examined the point at which micro-scale robotic drivetrains are no longer governed by classical physics and are rather governed primarily by nano-scale physics [1]. Caprari, et al, examined how the power required for motion scales with respect to the size of micro-scale mobile robots [2]. Their work shows a positive quantifiable linear correlation between size and volume for micro-scale electromagnetic motors. Automotive research into electric motors have sought to determine the upper limitations of DC motors to determine their feasibility to power electric vehicles [3]-[4]. By examining the required load characteristics of an automotive drive cycle, large scale DC motors are compared with internal combustion engines and other motors to determine each motor's relative performance.

Others have examined the scaling laws governing entire powertrain systems for vehicle-sized systems. For example, Huei Peng, et al, performed extensive research into subsystem scaling, power management, and design optimization of hybrid electric vehicles with varied powertrain configurations [5]-[6]. In doing so, scaling principles were obtained as a function of power for DC motors, IC engines, fuel cells, batteries, and hydraulic drivetrains. System modeling for each involved the utilization of Simulink, Vehicle System Modeling (VESIM), and/or physics-based modeling [7]. Petersheim et al, developed pi parameters for the scaling of electric vehicle components, principally for batteries, ultracapacitors, and DC motors, to aid hardware in the loop simulations [8].

For robot actuators, including designs without rotational motors, some have considered how these systems compare to biological counterparts. Madden compared various mechanical and electrical power sources, such as DC motors and actuators, to determine which mechanical system most effectively mimicked the power density of human muscles [9]. One conclusion is that DC electric motors thus far do not have the power density to run a human-sized humanoid robot. Marden conducted research to develop universal scaling principles for motors which range in size from a strand of muscle to linear actuators to rockets [10]-[11]. In Equation [3-1], these scaling principles show a correlation between motor mass, M (kg), and maximum force output, F (N), a relationship that appears to hold in size domains ranging from flying insects to electric motors to jets.

$$F = 55M^{0.999} \quad \textbf{[3-1]}$$

In a similar line of study, Waldron and Hubert examined the similarities between a variety of larger and smaller robotic mechanisms to determine scalability [12]. The mechanisms range from determining how deflection scales with size, to the allometry of a linear actuator. Their research found that, for certain types of DC motors, a generalized scaling correlation can be formulated to relate the motor's diameter to its corresponding peak power, stall torque, and no-load speed.

In the 20th century, significant research went into predicting how large-scale tracked and wheeled vehicles interact with their environment in an effort to predict vehicle capabilities and performance, and more recently the environmental damage caused by such vehicles. Bekker studied automobiles, tracked vehicles, large scale construction equipment and military transports among other large scale vehicles in an effort to describe how a vehicle's performance scales as a function of its size [13]-[14]. The majority of Bekker's contributions to the field are obtained by measuring the properties of fielded vehicles. He looked into a generalized method based on geometric features to predict off-road failure modes to predict a vehicle's ability to traverse uneven surfaces. Bekker developed a equation to relate the power required, P_d (watts), to drive over a given surface. His resulting power is related to the force available from the drive motors, F (Newton), the summation of the resistive forces experienced, R (Newton), the vehicle's

velocity, V_t, and tire/surface slip, i, as shown in Equation [3-2]. The resistive forces include rolling resistance, aerodynamic drag, soil compaction and gradient resistance.

$$P_d = (F - \Sigma R) \cdot V_t \cdot (1 - i) \quad \textbf{[3-2]}$$

Wong's recent research furthers Bekker's results with additional focus on large scale vehicles with a primary emphasis on higher order modeling of vehicle behaviors and dynamics [16]. The NATO Reference Mobility Model (NRMM) and the Army Reference Mobility model add to the experimental data available for off-road vehicles by characterizing soil-terrain interaction for fielded military vehicles [17]. The focus of NATO's and Wong's work was to describe and parameterize large scale off-road vehicle performance as a function of characteristics such as: vehicle mass, vehicle speed, terrain, moisture composition, and obstacle being traversed. However, due to the fact that models from both sources are primarily derived through experimental data on large scale vehicles, they are far less accurate in predicting capabilities of small mobile robots.

Dr. J.G. Hetherington, *et al*, at Cranfield University developed similar work in the field of terramechanics which described modeling techniques for the soil-terrain interaction of tanks and other large scale vehicles [18]-[19]. His modeling of these interactions even extends to describing soil-terrain interaction and performance of small scale electrically-driven skid-steer robots [20]. However, these modeling techniques are highly detailed and thus require nearly every parameter of the robot's drivetrain to be defined for the model to be utilized. Such modeling is thus not generally possible until after all major design decisions have already been made, limiting the utility of these methods for design iteration.

Much of the published research on the scaling of robotic components is too vague to be directly applicable for the design of a mobile robot. For example, as stated earlier, the power of a DC motor scales with its mass without consideration of the effects on the motor's efficiency, volume, or operating speed. While these seem to be nuances, one must know speeds to determine gear ratios for a motor to operate near its peak efficiency, or the performance will suffer. Similarly, the volume of the motor imposes strict constraints on the minimum dimensions of a robot chassis.

Although there has been extensive research into robotic components (batteries, motors, etc.) for automotive applications, not all of this work is directly applicable for smaller scale robots, particularly in regard to the expected loads required for terrain traversal. Empirical data used to predict scalability of an automotive component's performance on asphault is not necessarily accurate for a robot one hundredth times the mass of an automobile, driving mostly on grass or dirt surfaces. Similarly, much of Bekker's work is only applicable and accurate for heavy transport vehicles and not robot's two orders of magnitude smaller.

The research presented in this chapter seeks to develop a more comprehensive powertrain model which is applicable to the sizes of components associated with mobile robots while not requiring that every parameter of the robot be specified. In addition, the interdependencies between each of these components will be explained to demonstrate how to scale robot components to develop a predictive model for the entire robot system. This system model is then used to predict the range, maximum power, etc characteristics as a function of robot mass, battery types, etc.

To summarize later sections in this chapter, Section 3.2 decomposes the robot's powertrain into the essential components showing the interdependencies between each. Section 3.3 presents each component and shows how each component scales. It also presents the method used to generate each scaling rule. Experimental validation of these component-level design rules using four currently fielded mobile robots is then discussed in Section 3.4. Section 3.5 shows a comparison of the model-predicted system-level results to experimentally measured robot system behavior. Finally, the conclusions of this powertrain allometry study are then presented in Section 3.6.

3.2 Powertrain Subsystems

A robot's powertrain is defined hereafter as the components needed to achieve mobility of the platform. The powertrain of an electrically-driven robot is shown in Figure 3-1, and is generally comprised of a power source, a motor amplifier, DC motor(s), gearbox(s), and ground-contact drivetrain components (tracks or wheels). Each component has a corresponding mass, volume and efficiency, each which scales differently with respect to the robot's overall mass and

performance requirements. While the power source, motor amplifier and DC motor scale with power, the gearbox mass and efficiency are in general determined by the DC motor and optimal gear ratio to maximize cruising efficiency. The physical geometry of the drivetrain scales with physical geometry of the robot's chassis, and the losses associated with the drivetrain are a function of both the mass of the robot and the robot's velocity. Similarly, the losses due to the interaction of the drivetrain with the terrain are related to the mass of the robot and the robot's ground velocity. In addition to the components required for mobility, a mobile platform also has auxiliary power needs which are modeled here as a black box of a given mass, volume, and power draw. These auxiliary devices are most often communication equipment, sensors, or manipulator on a mobile robot.

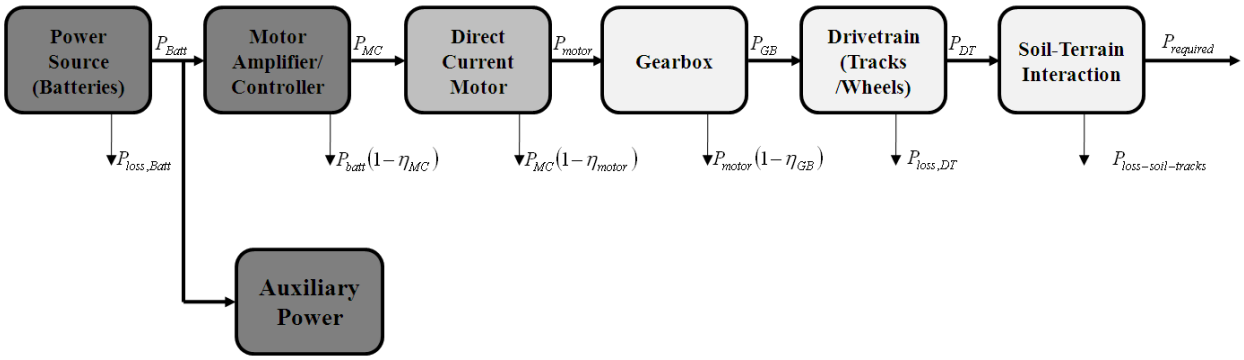

Figure 3-1: Mobile Robot Power Architecture

A robot is generally designed to complete a given mission which includes performance requirements and size requirements. Required performance capabilities can include climbing, speed, traversal, and endurance distance. Size requirements include the ability to fit within particular openings (pipes, doorways, etc), lifting / dragging / payload requirements (which impose mass constraints on the robot), etc. Both performance and sizing requirements can be used to define the torque and speed the platform needs to operate, once the relative size of the platform is known. Using this information, in addition to the amount of energy available on the robot, each component can be scaled accordingly.

While the above methodology appears straightforward, in practice this is quite difficult and iterative because each component's characteristics are dependent upon one another and change with a robot's power and mass. Knowing the given mission scenario of the platform, the

designer can develop an optimal design using the scaling principles explained in the following sections.

3.3 Power Allometry

3.3.1 Power Source (Battery)

The specific energy and the energy density of a battery are known constants based on the chemistry of a given battery, and these generally do not change significantly with battery size (at least, in the range of sizes of typical robots). The practical, specific energy and the energy density for a Lithium-ion battery are 150 Wh/kg and 0.4Wh/m^3 respectively [21]. Even though both the mass and the volume of any batteries chemistry will scale linearly with energy, the efficiency (equivalent energy in the battery) is not a fixed value across size scales. A 200kJ battery that is sized for 100 W of draw will perform much worse on an energy basis than a 200kJ battery that is sized for 200 W of draw, thus producing much less actual energy. The reason for this discrepancy in reported versus actual energy is because the actual energy which can be drawn from a battery is not linearly related to the amount of power being drawn at a given time. An experimental constant known as the Peukert number relates a battery's change in energy capacity as a function of its energy draw rate [22], and becomes critical for situations where a battery's power draw is a significant fraction of the rated power draw.

To define a battery's capacity mathematically, one first defines the change in a battery's energy, $\Delta E_{battery}$ (joules), as the product of the battery's voltage, V (volts), current being drawn, I, and the duration of the current draw, Δt (seconds), as shown in Equation [3-3]. The percent of the energy drawn, %E, is the ratio of the duration of the current drawn, to the rated battery life under a given current draw, T, in Equation [3-4]. The change in battery energy, $\Delta E_{battery,SOC}$, in Equation [3-4] is therefore the product of the percent energy and its nominal battery capacity, E_{nom}. Equation [3-5] then simplifies through substitution to Equation [3-6] which relates the battery's change in energy to the duration of current drawn, rated battery life and nominal battery capacity.

$$\Delta E_{battery} = V \cdot I \cdot \Delta t \quad \textbf{[3-3]}$$

$$\%E = \frac{\Delta t}{T} \quad \textbf{[3-4]}$$

$$\Delta E_{battery,SOC} = (\%E) \cdot (E_{nom}) \quad \textbf{[3-5]}$$

$$\Delta E_{battery,SOC} = \left(\frac{\Delta t}{T}\right) \cdot (E_{nom}) \quad \textbf{[3-6]}$$

At this point, one can introduce Peukert's correction that accounts for rate of power draw. Equation [3-7] relates the theoretical battery capacity, C (amp-hrs), to the actual battery life, T, through the use of the Peukert exponential, N [22]. The theoretical capacity of the battery is usually supplied by battery vendors which state the amp-hour rating of the battery over the given period of the test, R (sec). The Peukert exponential can then be calculated by determining the decrease in the actual battery capacity as the current being drawn increases, but is generally constant for a given battery chemistry and technology of battery construction. The change in the actual energy capacity of a battery from its theoretical capacity is due to the losses associated with increasing the rate of chemical reactions within the battery. The Peukert exponential varies from 1.1 to 1.3 for battery chemistries varying from Lithium-ion to Lead Acid [22].

$$T = \frac{R}{\left(\frac{I}{C/R}\right)^N} \quad \textbf{[3-7]}$$

The purpose of deriving the aforementioned equations is ultimately to relate the change in the battery's state of charge, $\Delta E_{batt,SOC}$, to the amount of energy which is available from the battery. Only at the c-rating of the battery are these two equal. Equation [3-7] is rewritten to

solve for the current draw. This equation is substituted into Equation [3-3] to relate the Peukert number to energy draw rather than current draw. This new equation is substituted with Equation [3-6] to form Equation [3-8] which relates change in battery energy state of charge with the amount of energy available from the battery. At the point where the current drawn is equal to the rated current draw (the one used to determine amp-hour ratings of a battery), then the change in battery's state will be equal to the energy available from the battery. For current draws above this rated value, the net energy out of the battery will be less than what is rated to provide ($\Delta E_{battery} < \Delta E_{battery,SOC}$). And conversely, for current draws below this level, the battery will be able to provide more energy than it is rated for, ($\Delta E_{battery,SOC} < \Delta E_{battery}$).

$$\Delta E_{\text{battery}} = V \cdot \Delta t \cdot \left(\left(\frac{C}{R} \right) \cdot \left(\frac{\Delta E_{battery,SOC} \cdot R}{\Delta t \cdot E_{nom}} \right)^{\frac{1}{N}} \right) \quad \textbf{[3-8]}$$

As a final simplification to more clearly describe the exponential dependence of energy storage on energy change, Equation [3-8] can be rewritten and simplified to Equation [3-9] using the battery constant, k_B, in Equation [3-10]. k_B is a function of the duration of power draw and constant properties of a given battery.

$$E_{\text{supplied\_from\_battery}} = k_B \cdot \left(\Delta E_{Battery\_C10} \right)^{\frac{1}{N}} \quad \textbf{[3-9]}$$

$$k_B = V \cdot \Delta t \cdot \frac{C}{R} \cdot (R)^{\frac{1}{N}} \left(\Delta t \cdot E_{nom} \right)^{\frac{-1}{N}} \quad \textbf{[3-10]}$$

3.3.2 Motor Controller/Amplifier

The purpose of a Direct Current motor controller (often called a motor amplifier) is to allocate power from the batteries to the DC drive motors in a controllable manner. Through a review of available vendor data, correlations between the amplifier's output power and the amplifier's mass and volume were generated. These mass and volume results are presented in Figure 3-2 and Figure 3-3 respectively. Each data point represents a commercially available motor controller, and each line in Figure 3-2 and Figure 3-3 represents the linear correlation between power and the motor controller's mass and volume.

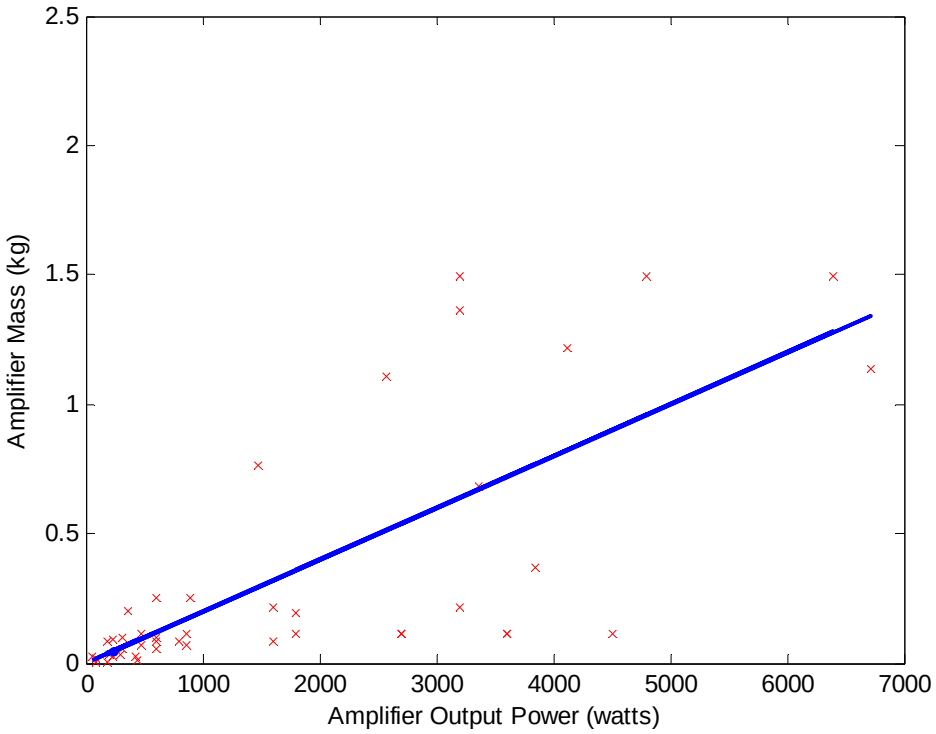

Figure 3-2: Motor Amplifier Mass Scalability with Power

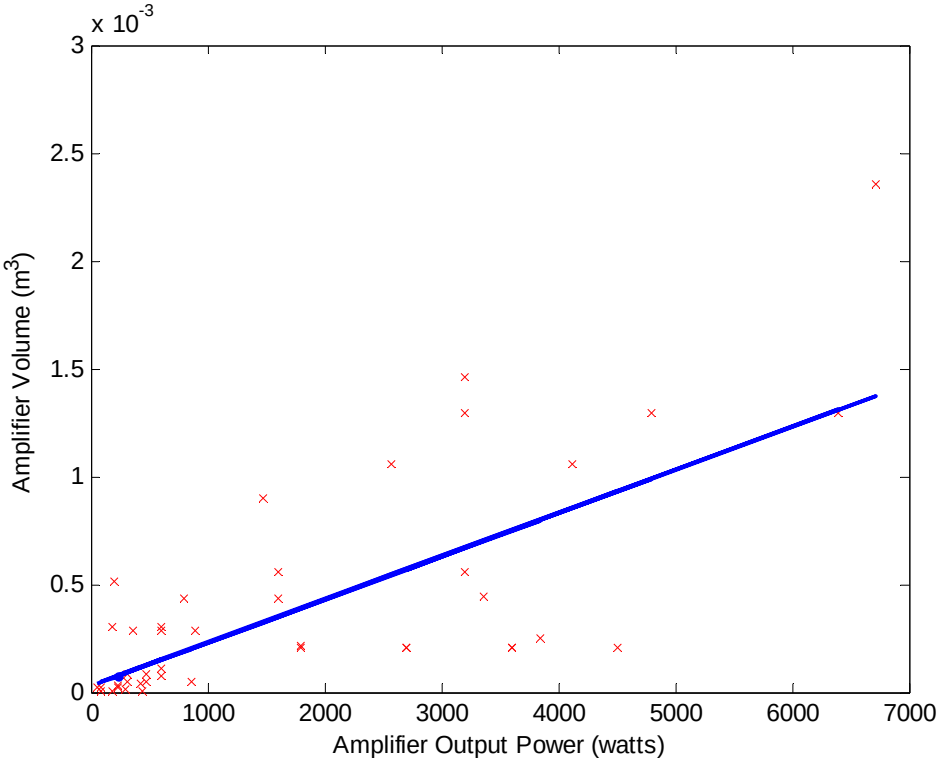

Figure 3-3: Motor Amplifier Volume Scalability with Power

Equations [3-11] and [3-12] are the corresponding mass, m_{MC} (kg), and volume, V_{MC} (m^3), required for a motor controller of a given output power, P_{MC}. The linear fits shown in Figure 3-3 are not closely correlated to the data points because motor controllers of equal power often have different types of packaging for heat dissipation, overload and short protection, cooling fans, etc based on their intended applications. However, the mass of a motor controller is usually quite small relative to a motor/battery combination, so such errors generally fall into the range of error commonly seen on other powertrain components.

$$m_{MC} = 0.0002\,P_{MC} + 0.0042 \quad \textbf{[3-11]}$$

$$V_{MC} = 0.0000002\,P_{MC} + 0.00003 \quad \textbf{[3-12]}$$

While many vendors provide the physical specifications of their products, the change in efficiency as a function of a motor controller's power is rarely if ever provided. For this reason, this efficiency was measured experimentally for a representative high-current (> 100A) amplifier. These measurements were made using a Roboteq AX2550 Motor Controller [23]. The motor controller powers a custom built skid-steered tracked robot platform known as the Tankbot. Further discussion of the characteristics of the platform can be found in Section 3.5.

By measuring the power into and out of the motor controller during a number of driving scenarios, the motor controller efficiency, η_{MC}, was determined. The resulting trend line shows that efficiency changes significantly but in a repeatable manner versus the power drawn. Each point in Figure 3-4 represents a power measurement taken during testing, while the solid line represents a second order fit curve expressed in Equation [3-13]. The peak efficiency of this curve is 93.2% and occurs at approximately 523 watts for the Roboteq AX2550 motor controller. Ideally, a motor controller should be physically sized to be able to supply the required power to each of the drive motors and at the same time be sized to operate at peak efficiency during this condition. For the purposes of scaling, it is assumed that the motor controller is scaled for a given power draw and assumed to operate at this peak efficiency.

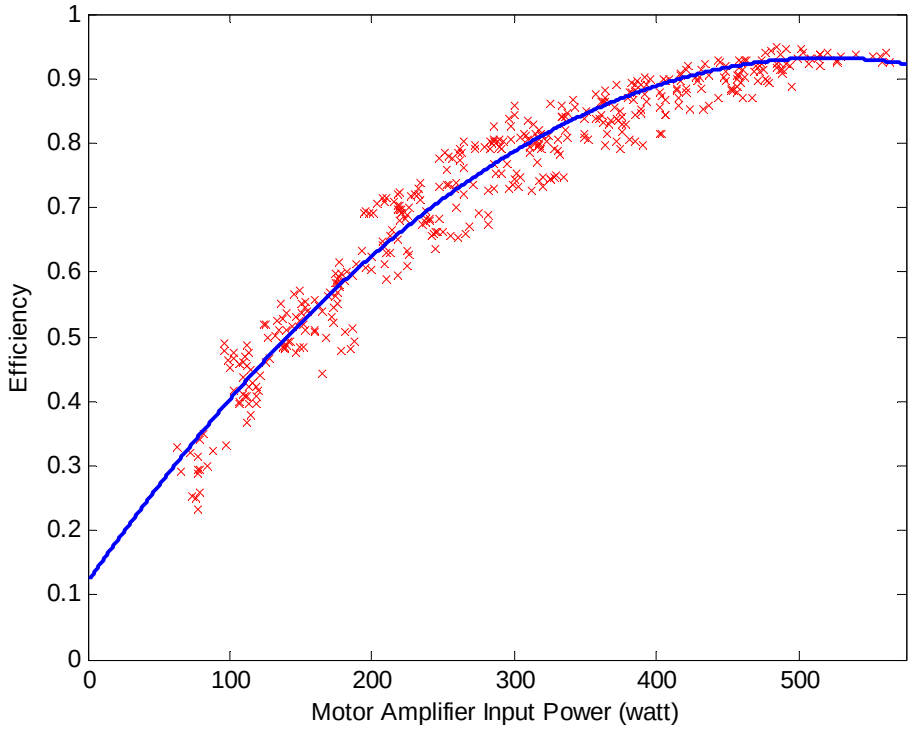

Figure 3-4: Motor Amplifier Efficiency

$$\eta_{MC} = -0.0003 P_{MC}^{\ 2} + 0.3121 P_{MC} + 12.015 \qquad \textbf{[3-13]}$$

3.3.3 Direct Current Motor

In order to optimally size a DC motor, the performance of a given motor must be predicted to understand how motor performance scales with size and usage. Kenjyo and Nagamori derived equations to predict steady-state performance of DC motors using first principles of physics [24]. Figure 3-5 and Figure 3-6 describe how a motor's power, efficiency, speed, and current draw are a function of the changing motor torque. Each point in Figure 3-5 and Figure 3-6 are measured experimentally from a typical motor from a common robot motor vendor (Pittman). The solid lines in these figures represent the predicted motor performance calculated using available motor characteristics such as: rated voltage, no-load current and speed, torque constant, stall current, and peak power output. The points represent actual motor

performance data taken from the motor vendor while the solid lines are the predicted motor performance. The agreement between first-principles derivation and experimental measurement is quite obvious.

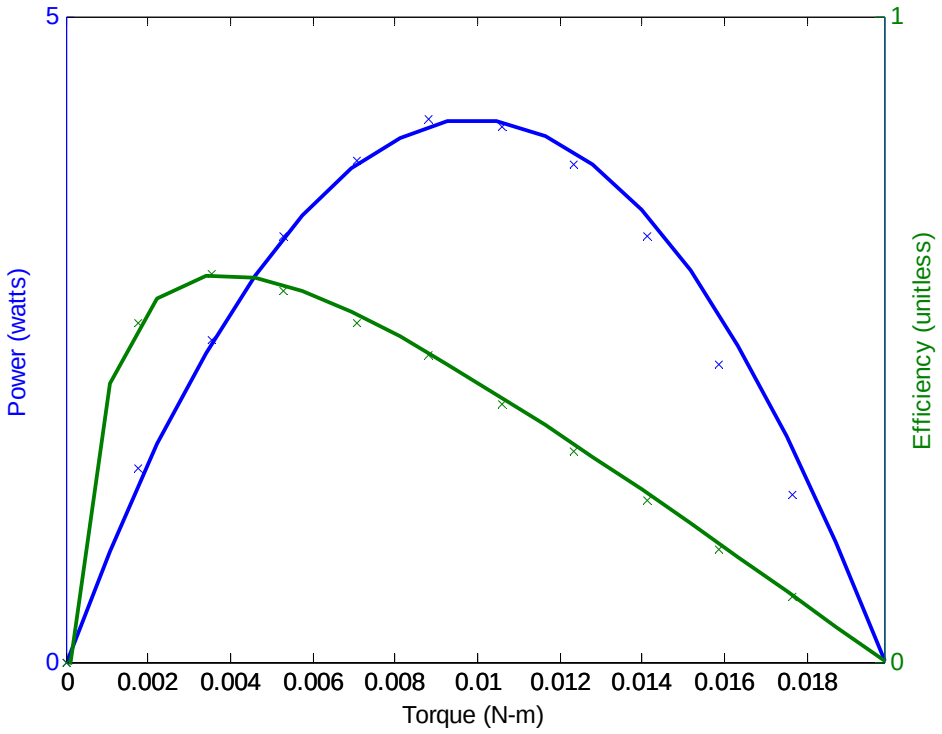

Figure 3-5: Steady State DC Motor Power and Efficiency Performance

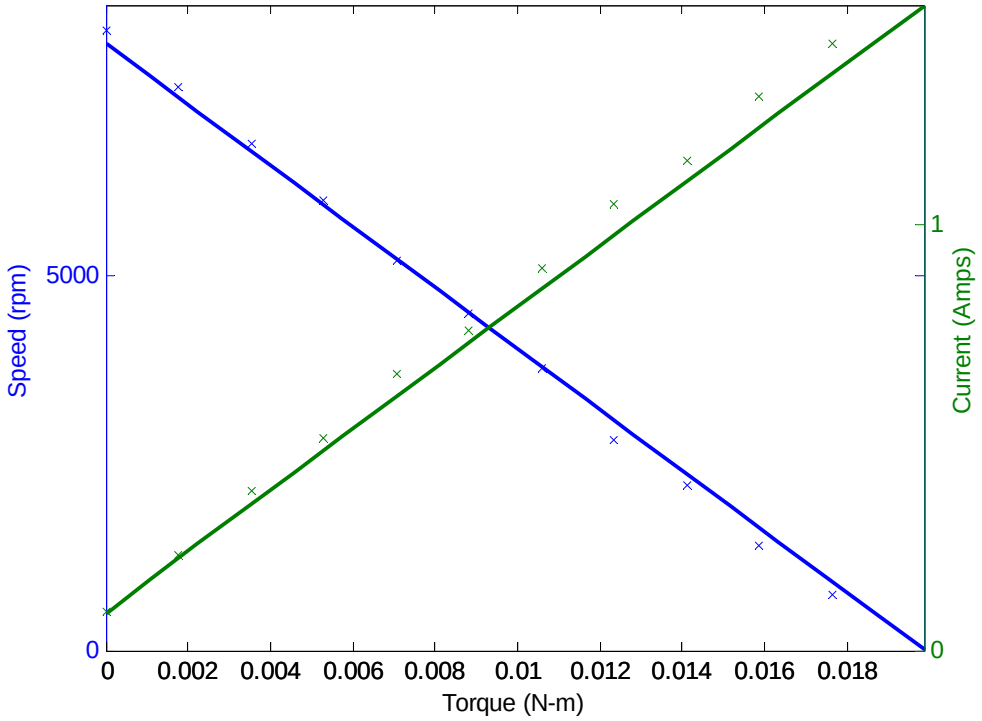

Figure 3-6: Steady State DC Motor Speed and Current Performance

To predict steady-state performance, a DC motor is modeled as an electric circuit, as seen in Figure 3-7, where each resistor is a different loss associated with operating the motor. Both electrical and mechanical losses are modeled including joule heating, windage loss, mechanical loss, and iron loss.

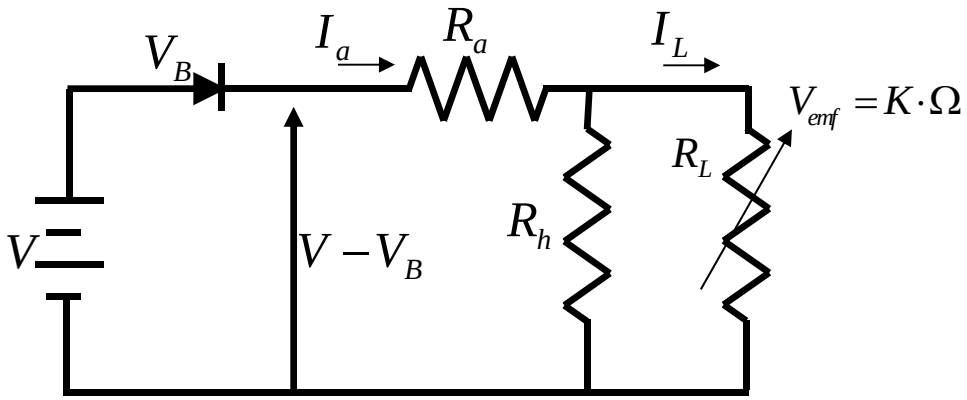

Figure 3-7: DC Motor Circuit Model

Using Ohm's law, Equation [3-14] can be derived that relates the armature resistance to the ratio of the rated motor voltage and the stall current. The voltage drop across the brushes, V_B, is modeled as a diode. Using Kirchhoff's voltage and current laws, V_B is shown in Equation [3-15] is derived to be a function of the motor characteristics: voltage, V, the armature resistance, R_a, no-load current, I_0, motor constant, K, and no-load speed, Ω_0. The term R_h is a grouping of speed invariant losses such as windage, mechanical, and iron, and it is calculated using a combination of Ohm's law and Kirchhoff's laws at no-load in Equation [3-16]. The motor constant, M, describes the relationship between current and speed; and through substitution, Equation [3-17] is reduced to be a function of the resistive forces within the motor.

$$R_a = \frac{V}{I_{stall}} \quad \textbf{[3-14]}$$

$$V_B = V - R_a \cdot I_0 - K \cdot \Omega_0 \quad \textbf{[3-15]}$$

$$R_h = \left(\frac{V - V_B}{I_0} \right) - R_a \quad \textbf{[3-16]}$$

$$M = \sqrt{\frac{R_h + R_a}{R_a}} \quad \textbf{[3-17]}$$

In Equation [3-18], the electromechanical force, V_{emf}, is the parameter which relates the change in voltage with speed. Through the further use of Kirchoff's laws, Ohm's law, energy balance equations, and algebraic manipulation, Equations [3-19]-[3-23] can be obtained that describe how the motor's output performance can be predicted as shaft speed varies from no-load to stall.

$$V_{emf} = K \cdot \Omega \quad \textbf{[3-18]}$$

$$I_{motor} = \frac{V - V_B - V_{emf}}{R_a} \quad \textbf{[3-19]}$$

$$P_{motor,out} = V_{emf} \cdot I - \frac{V_{emf}^{\ 2}}{R_h} \quad \textbf{[3-20]}$$

$$P_{motor,in} = V \cdot I \quad \textbf{[3-21]}$$

$$\eta_{motor} = \frac{P_{motor,out}}{P_{motor,in}} \quad \textbf{[3-22]}$$

$$T_{motor} = \frac{P_{motor,out}}{\Omega_{motor}} \quad \textbf{[3-23]}$$

The equations used to predict steady-state motor performance are also used to calculate motor performance at peak efficiency as well. Equation [3-22] is rewritten using substitution into Equations [3-19]-[3-21], to obtain Equation [3-24] below. By taking the derivative of Equation [3-24] with respect to the electromagnetic voltage, and setting it equal to zero, an equation is derived to calculate the electromagnetic voltage at peak efficiency, $V_{emf,max}$. The electromagnetic voltage at peak efficiency is simplified in Equation [3-25]. Through substitution of V_{emf} for $V_{emf,max}$ the original efficiency equation, Equation [3-22], reduces into Equation [3-26] which describes the peak efficiency of a motor.

$$\eta = \frac{P_{out}}{P_{in}} = \frac{V_{emf} \cdot \left(V - V_{emf}\right) + \frac{V_{emf}}{R_h}}{V \cdot \frac{V - V_{emf}}{R_a}} = \frac{V_{emf}}{V} - \frac{R_a}{R_h} \cdot \frac{{V_{emf}}^2}{V \cdot \left(V - V_{emf}\right)} \quad \textbf{[3-24]}$$

$$V_{emf,\max} = V \cdot \frac{M - 1}{M} \quad \textbf{[3-25]}$$

$$\eta_{motor,\max} = \left(\frac{V - V_B}{V}\right)\left(\frac{K - 1}{K + 1}\right) \quad \textbf{[3-26]}$$

The motor current and velocity at peak efficiency, $I_{motor,maxeff}$ and $\Omega_{motor,maxeff}$ respectively, are both computed by examining the ratio of their governing equations at peak efficiency over their no load conditions shown in Equations [3-27] and [3-28]. With $V_{emf,max}$ expressed above, each equation reduces to Equation [3-29] and [3-30] which describe the motor's current and speed at peak efficiency.

$$\frac{I_{motor,\max eff}}{I_0} = \frac{V - V_{emf\max}}{V - V_{emf}} \quad \textbf{[3-27]}$$

$$\frac{\Omega_{motor,\max eff}}{\Omega_0} = \frac{V_{emf\max}}{V_{emf,0}} \quad \textbf{[3-28]}$$

$$I_{motor,maxeff} = K \cdot I_0 \quad \textbf{[3-29]}$$

$$\Omega_{motor,maxeff} = \left(\frac{K}{K + 1}\right) \times \Omega_0 \quad \textbf{[3-30]}$$

The torque at peak efficiency, $T_{motor,max\ eff}$, and the power at peak efficiency, $P_{motor,max\ eff}$, are generated using the laws of physics. Each equation reduces to a generalized form in Equations [3-31] and [3-32] respectively. Through examination of the experimentally generated performance curves supplied by vendors, the equations to predict motor performance are shown to be highly accurate for a range of different motor sizes.

$$T_{motor,maxeff} = \frac{(M - 1) \cdot \left(V - V_B\right) \cdot I_0}{\Omega_{0(ungeared)}} \quad \textbf{[3-31]}$$

$$P_{motor,maxeff} = T_{motor,maxeff} \cdot \Omega_{motor,maxeff} \quad \textbf{[3-32]}$$

A market survey of DC motors was conducted to record all relevant information necessary to predict each motor's performance in the above equations. Using this data, each motor's performance at peak efficiency was calculated to develop parameters to scale DC motor power. With the required output power at peak efficiency specified, $P_{motor,max\ eff}$, the corresponding mass and volume of the motor is predicted using Equations [3-33]. The data collected for each of these correlations are shown in Figure 3-8 and Figure 3-9. Each point represents a commercially available motor and Equations [3-33] and [3-34] are represented at the curves.

$$V_{motor} = 0.000008 P_{motor,\max eff}^{0.8253} \quad \textbf{[3-33]}$$

$$m_{motor} = 0.0386 P_{motor,\max eff}^{0.8975} \quad \textbf{[3-34]}$$

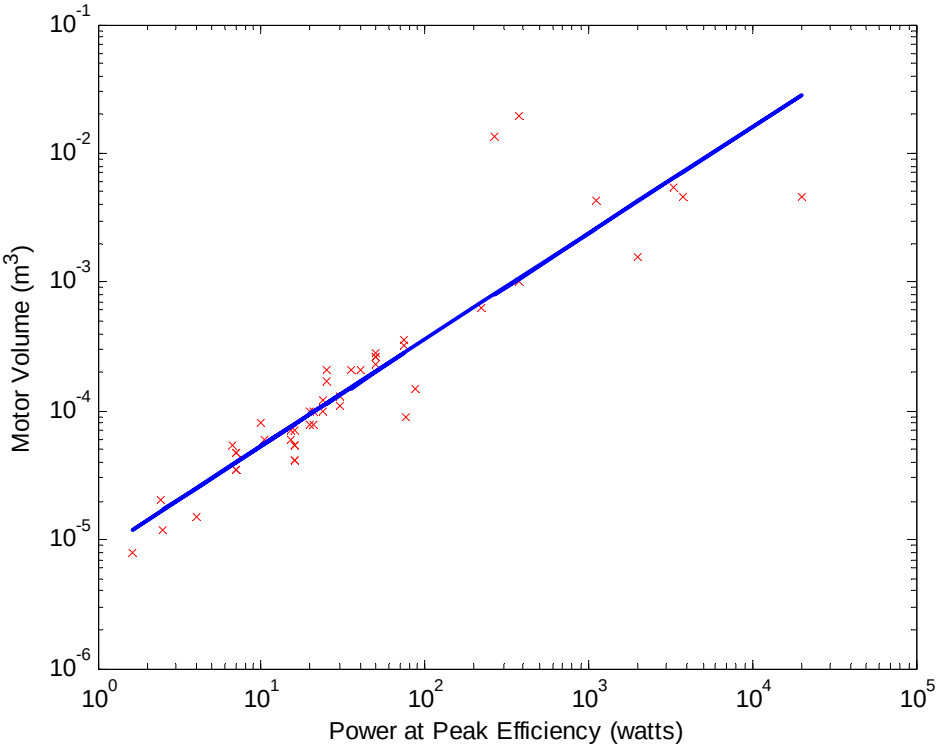

Figure 3-8: Direct Current Motor Volume Scaling with Power

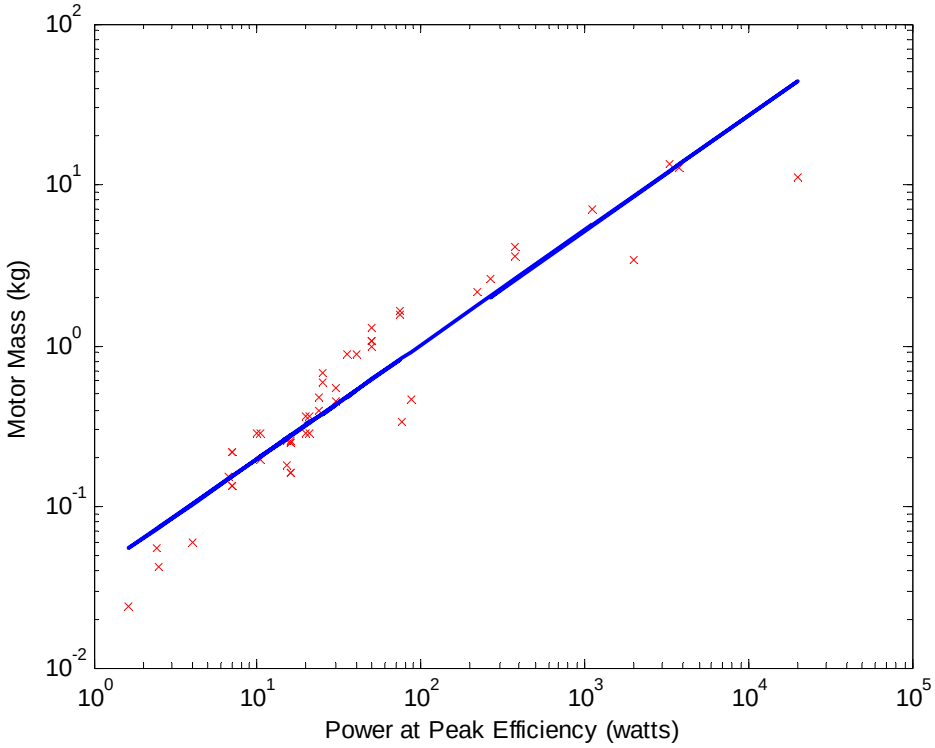

Figure 3-9: Direct Current Motor Mass Scaling with Power

To optimally scale the entire platform of the robot, motors are sized by torque and speed to meet a required power demand. Equations [3-35]-[3-38] show how the motor performance characteristics scale with its peak power. Equation [3-35] shows that the overall efficiency of the DC motor, $\eta_{motor,maxeff}$, increases with total output power. While Equation [3-36] shows that speed at peak efficiency decreases with power, Equations [3-37] and [3-38] show positive relationships. Scaling motors based on peak efficiency indicates that the motor which draws the least amount of power to operate at the desired output is chosen rather than simply the smallest motor necessary for a specified torque.

$$\eta_{motor,\max eff} = 0.5901 P_{motor,\max eff}^{0.056} \quad \textbf{[3-35]}$$

$$\Omega_{motor,\max eff} = 7099 P_{motor,\max eff}^{-0.095} \quad \textbf{[3-36]}$$

$$T_{motor,\max eff} = 0.0016 P_{motor,\max eff}^{1.0379} \quad \textbf{[3-37]}$$

$$I_{motor,\max eff} = 0.1708 P_{motor,\max eff}^{0.7249} \quad \textbf{[3-38]}$$

3.3.4 Gearbox

A DC motor operating at peak efficiency will have a shaft speed that is often an order of magnitude greater than the operating wheel shaft speeds for many mobile robots, requiring the use of a gear reduction. While gear systems greatly increase the efficiency of a motor, they reduce the efficiency of the powertrain due to gear losses. Unfortunately, these gear loses are almost never specified. Fortunately, a number of vendors distribute custom-built direct-drive gearboxes coupled with DC motors. Enough data is provided about the motor itself to calculate its performance using the aforementioned equations in Section 3.3. With the efficiency of the system, η_{system}, experimentally measured by the vendor and the efficiency of the motor, $\eta_{motor,maxeff}$, very accurately calculated, one can closely estimate the efficiency of commercial gearboxes, η_{GB}, using Equation [3-39].

$$\eta_{GB} = \frac{\eta_{system}}{\eta_{motor,\max eff}} \quad \textbf{[3-39]}$$

This estimation process is repeated for a number of different gear ratios and on a number of different size geared motors whose data is available in manufacturer literature. Figure 3-10 shows that there is a correlation between gear ratio and efficiency which is independent of motor characteristics such as size, mass, speed, and torque. There is however a clear negative correlation between gear ratio and gearbox efficiency shown in Figure 3-10 and quantified in Equation [3-40]. These results agree with the general estimates provided by Clark and Owings [25] that explain that, the larger the gear ratio, the more loses one should expect in the geartrain.

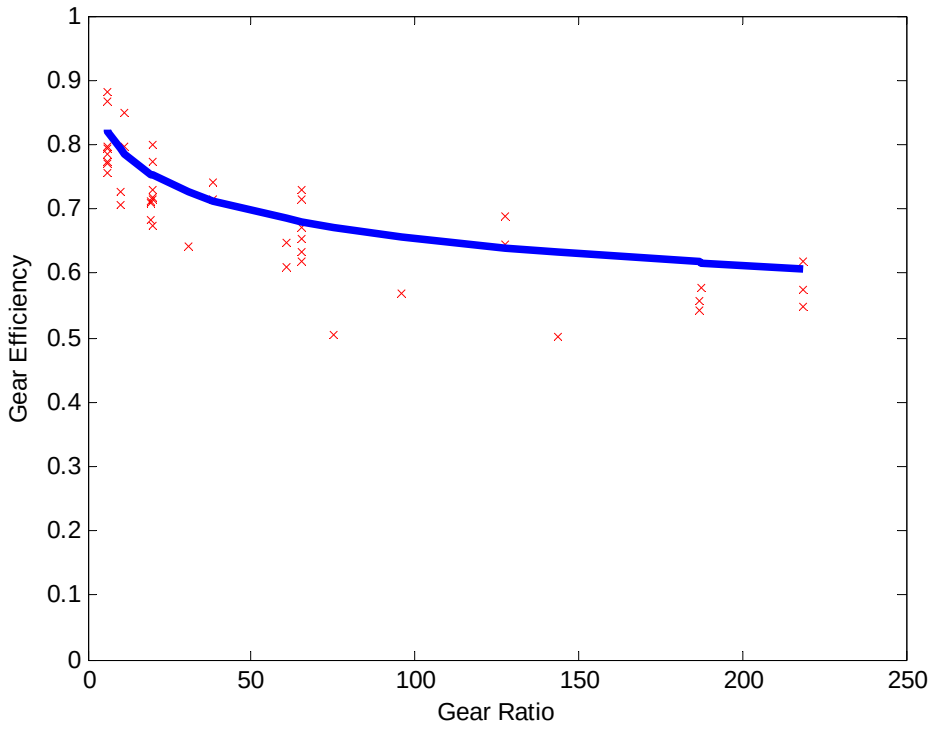

Figure 3-10: Gearbox Efficiency as a Function of Gear Ratio

$$\eta_{GB} = -0.06 \ln\left(N_{GB}\right) + 0.93 \quad \textbf{[3-40]}$$

The mass of the motor's gearbox (kg), m_{GB}, was also computed using vendor data and the first principles from Section 3.3. Vendors often build a number of gearboxes for a given motor. With the mass of the motors calculated and the mass of the system known, the mass of the gearbox can then be calculated. The mass of many different vendor gearboxes are shown in Equation [3-41], and are seen to be both a function of the gear ratio and the mass of the motor.

$$m_{GB} = m_{motor}\left(0.004\, N_{GB} + 0.56\right) \quad \textbf{[3-41]}$$

3.3.5 Drivetrain

The drivetrain of a robot is any mechanical component appearing between the output shaft of the drive motors and the contact point of the robot to the ground. The section discusses how to size wheels and tracks for a ground robot and the corresponding power losses associated with each. The models to predict the sizing of these components are derived from observations made about existing mobile robot platforms, considering specifically robots similar in size to those shown in Figure 3-11. Power draws estimates are made using a custom built skid-steer tracked platform. The sizing of the drivetrain components are derived to be a function of the chassis geometry and mass. However, the actual size and placement of equipment within a robot is, in practice, sometimes more a design issue than an exact science predicted by scaling theory, as manufacturers often package multiple items alongside the motor including limit switches, encoders, etc. The chassis however, must at the very least contain the mechanical and magnetic components of the motor as predicted by scaling theory, and therefore the chassis must at least be greater than or equal in size to the sum of its powertrain components previously listed.

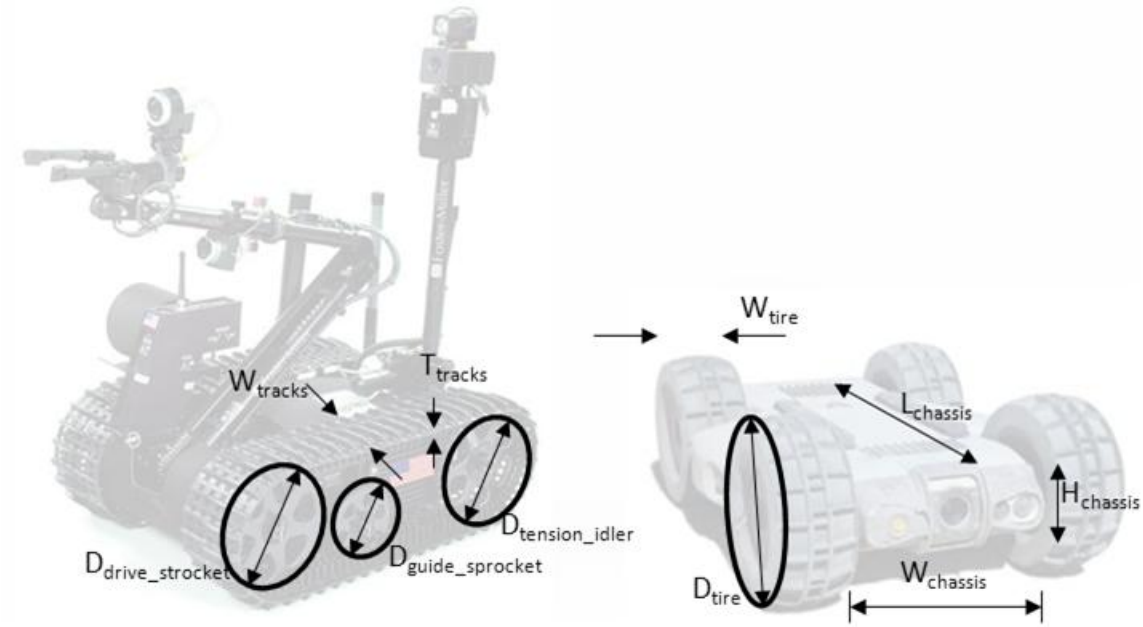

Figure 3-11: Tracked and Wheeled Robot Diagram

3.3.5.1 Tracks
Sizing Tracks

A tracked mobile robot is modeled with three sprockets (drive sprocket, tensioner/idler and guide sprocket) and a track made of a single pieces of rubber. The proportion of sprocket diameter to chassis size is assumed to be a fixed ratio, and this is taken directly from measurements from the Talon platform. Equation [3-42] through Equation [3-48] are derived by making correlations between the geometry of the platform and components sizes. All of these proportions were calculated using SI units relating mass, kg, and length, m. Equations [3-42]-[3-44] show the linear correlation between sprocket diameter and chassis geometry. Further, it shows experimentally-measured approximations between the mass of the tracks are versus the robot chassis' mass.

$$D_{drive\_sprocket} = D_{tension\_idler} = 1.25H_{chassis} \quad \textbf{[3-42]}$$

$$D_{guide\_sprocket} = 0.75H_{chassis} \quad \textbf{[3-43]}$$

$$m_{sprockets} = 0.02055m_{chassis} \quad \textbf{[3-44]}$$

In this thesis, the track length refers to the total length of the tread if it was separated at one point and laid flat. The track length is assumed to circumscribe the chassis and is therefore a function of the sprocket diameters computed above and the length of the robot's chassis (Equation [3-45]). The corresponding width of the tracks is computed to achieve an equivalent ground pressure to what is seen on commercially available EOD robots. Based on the measurements taken on four robot platforms, it was seen nearly all EOD robots maintain approximately equivalent ground pressure, despite changes in the robot's weight changing from 13.2 to 307 kg. Equation [3-46] relates the required width of the tracks, W_{track}, to be a function of the robot's chassis mass, $m_{chassis}$, geometry and number of track, N_{track}. The track thickness in Equation [3-47] is a derived to be a portion or the robot's chassis mass. The mass of the track themselves is the product of the track volume and the tracks density (Equation [3-48]). The model assumes that robots use rubber tracks with a nominal density of 687 kg/m^3.

$$L_{track} = 1.7 L_{chassis} + 1.25\pi \cdot H_{chassis} \quad \textbf{[3-45]}$$

$$W_{track} = \frac{m_{chassis}}{225.935 L_{chassis} \cdot N_{track}} \quad \textbf{[3-46]}$$

$$T_{tracks} = (8.93E-5) m_{chassis} \quad \textbf{[3-47]}$$

$$m_{track} = L_{track} \cdot W_{track} \cdot T_{track} \cdot \rho_{track} \quad \textbf{[3-48]}$$

Internal Resistance of Running Tracks

The internal resistance of running track as defined here is the resistive force or power loss associated with the turning the tracks with a drive sprocket. A custom-built tracked robot was used to measure the typical power lost due to the resistive forces within the tracks as a function of speed. This robot was suspended in midair allowing the tracks to driven freely while power into each of the motors was measured at varying speeds. After calculating the losses from the motor and gearbox using the previous work from Section 3.3 and 3.4, Equation [3-49] describes the power loss, P_{track_loss}, as a function of the robot speed, V_{robot} and is scaled with the mass of the robot. The track losses are modeled here as a power loss which is subtracted from the power out of tracks, rather than as efficiency loss. The resistance in the tracks is modeled this way because during testing it was the output to the free spin test.

$$P_{loss,DT-track} = 0.01268 m_{robot} \cdot V_{robot}^{2.06} \quad \textbf{[3-49]}$$

3.3.5.2 Wheels

The scaling rules to predict the sizing of the wheels for a mobile robot assumes a wheeled configuration similar to that of the Dragon Runner shown in Figure 3-11 [26]. By specifying the ground clearance, and with the chassis height known, the required diameter of the tires can be calculated using Equation [3-50]. A survey of commercially available inflated rubber off-road tires with metal hubs was conducted. Candidate tires ranged from radio-controlled hobby cars to automobile-sized truck tires. The results show that there is a linear correlation between diameter and width of tires, a relationship which is captured in Equation [3-51].

$$D_{tire} = H_{chassis} + 2H_{ground\_clearance} \quad \textbf{[3-50]}$$

$$W_{tire} = 0.425\, D_{tire} \quad \textbf{[3-51]}$$

Equations [3-52] and [3-53] are the basic relationships relating the volume and corresponding mass of the tires. The density of the inflated tires, $\rho_{avg,tire}$, is assumed to be a constant which encompasses the net influences of the metal wheel hub, air and rubber (313.67 kg/m^3). This constant is derived using the same vendor data used to generate the correlation between tire width and tire diameter.

$$V_{tire} = \pi\left(\tfrac{1}{2} D_{tire}\right)^2 W_{tire} \quad \textbf{[3-52]}$$

$$m_{tire} = V_{tire} \cdot \rho_{avg,tire} \quad \textbf{[3-53]}$$

Internal Resistance of Running Wheels

The internal resistance of running wheels is often referred to as rolling resistance, e.g. the resistive forces associated with the deformation of a tire as it rolls along a surface [16]. Unlike the internal resistance of running tracks, a direct drive system between the output shaft of a gearbox and a wheel has virtually no losses. For this reason, the internal resistance of running wheels is approximated to be zero and therefore negligible. The losses associated with rolling resistance should therefore be included when determining the total losses associated with tire and soil deformation , a factor which is considered in other sections.

3.3.6 Soil Terrain Interaction

3.3.6.1 Tracks

All of the losses associated with the soil interacting with the ground can be approximated together into one power loss which changes as a function of the robot's velocity, mass and soil type. The resulting loss equation is given in Equation [3-54]. The losses associated with soil-terrain interaction include the rolling resistance loss, wind resistance, soil compaction, soil plowing/bulldozing, and gradient resistance. According to previous research in this area, losses

due to rolling resistance, soil compaction and the soil plowing effect require numerous parameters to be specified in order to calculate these losses [13]. These parameters include the percentage of moisture in the soil and location of the pressure points within the soil. Instead, the power into the drive motors of a representative robot (the TankBot) was measured experimentally while both speed and terrain were varied independently. Doing so allowed the measurement of power required as a function of platform velocity. After accounting for the losses associated with the inefficiencies in the drive motors, gearboxes, and from running tracks, Equation [3-54] describes the power required to overcome the resistive forces from the tracks interacting with the terrain. This is a linear correlation with the constants representing the slope and intercept vary with the soil type, shown in Table 3-1.

$$P_{loss,soil-tracks} = m_{robot} \cdot \left(m_{terrain} \cdot V_{robot} + b_{terrain} \right) \quad \textbf{[3-54]}$$

Table 3-1. Soil-Terrain Interaction Loss Constants for Tracked Vehicles

Terrain	$m_{terrain}$	$b_{terrain}$
Asphalt	1.18	0.15
Grass	1.21	0.11
Tile	1.00	0.20
Dirt	1.12	0.31
Gravel	0.81	0.01
Brush	1.53	0.66

3.3.6.2 Wheels

The power loss associated with a robot's wheels interacting with the terrain is the product of the summation of the resistive forces and the robot's velocity (Equation [3-55]). The summation of these losses shown in Equation [3-56] includes the rolling resistance (R_{RR}), air drag (R_A) and gradient resistance (R_G).

$$P_{loss,soil-wheels} = R_{tot} \cdot V_{robot} \quad \textbf{[3-55]}$$

$$R_{tot} = R_{RR} + R_A + R_G \quad \textbf{[3-56]}$$

In Equation [3-57] the rolling resistance is the product of the coefficient of rolling resistance, C_{RR}, and the normal force of the robot. According to Wong, the coefficient of rolling resistance varies from 0.0133 to 0.35 depending on terrain, the contents of which are captures in Table 3-2 [16]. The assumption is made here that this constant incorporates the losses associated with the deformation within the tire, soil compaction and the bulldozing effect.

$$R_{RR} = C_{RR} \cdot m_{robot} \cdot g \quad \textbf{[3-57]}$$

Table 3-2. Coefficient of Rolling Resistance for Varied Terrains

Terrain	C_{RR}
Asphalt	0.0133
Unpaved Road	0.05
Gravel	0.02
Field	0.1-0.35

The aerodynamic resistance is described in Equation [3-58], where ρ is the mass density of air, C_D is the coefficient of aerodynamic resistance, A_{robot} is the equivalent surface area of the front of the robot and $V_{robot\text{-}wind}$ is the speed of the vehicle relative to the wind [16]. The gradient resistance, R_G in Equation [3-59], is the losses associated with driving on a surface with angle, θ.

$$R_A = \frac{\rho}{2} C_D A_{robot} V_{robot-wind}^{\ 2} \quad \textbf{[3-58]}$$

$$R_G = m_{robot} \cdot g \cdot \sin(\theta) \quad \textbf{[3-59]}$$

3.4 System-Level Robot Modeling for Endurance Distance Predictions

3.4.1 Robot Mass Prediction

The mass of a robot is defined in Equation [3-60] as the summation of its components used for locomotion (m_{MC}, m_{motor}, m_{GB}, $m_{drivetrain}$), the robot's chassis ($m_{chassis}$) and any auxiliary mass (m_{aux}) which includes controllers, sensors and manipulators for example. Both the chassis mass and mass of the auxiliary equipment varies based on application, and exact choice on chassis layout and the allowable payload is more of a design and control issue than something that can be calculated apriori. Thus, the payload in particular included later in the design process as a user-defined term that can be arbitrarily selected.

$$m_{robot} = m_{MC} + m_{motor} + m_{GB} + m_{drivetrain} + m_{chassis} + m_{aux} \quad \textbf{[3-60]}$$

3.4.2 Robot Power Requirements Prediction

The power required to achieve a given speed for a robot of a given mass, $P_{required}$, is calculated by incorporating all of the losses and inefficiencies within the powertrain as enumerated in Section 3.3. The required power is separated in Equation [3-61] into the power required to overcome the track losses and the soil-terrain interaction, the inefficiencies in several of the powertrain components, η_{tot} in Equation [3-62], and the auxiliary power draw. With power required for locomotion at a given speed already calculated, and the energy available known, the duration of operation (or endurance time) can be calculated using Equation [3-7] in Section 3.1. The battery endurance time coupled with the known robot velocity can be used to calculate the endurance distance using Equation [3-63].

$$P_{required} = \frac{P_{loss,soil-terrain} + P_{loss,DT}}{\eta_{tot}} + P_{aux} \quad \textbf{[3-61]}$$

$$\eta_{tot} = \eta_{MC} \cdot \eta_{motor} \cdot \eta_{GB} \quad \textbf{[3-62]}$$

$$\Delta x_{robot} = T \cdot V_{robot} \quad \textbf{[3-63]}$$

3.5 Fielded Mobile Robots

Four different robots were used to experimentally validate the powertrain scaling rules and endurance range predictions described above. These robots are the Tankbot, Talon, Bombot, and RONS. The Tankbot is a 74.1kg skid steer robot with a tank-like tread configuration as seen in the top right of Figure 2-9. The Tankbot was designed as an instrumentation platform and for purposes of this research used to record data such as power and velocity. This platform was used to generate the motor controller efficiency curves, the drivetrain and terrain power loss correlations. The Talon is one of the currently fielded EOD robots used for this study. In addition to using a track configuration, the Talon has a number of auxiliary sensor equipment and a manipulator shown in the top left image in Figure 2-9. The Bombot is a small scout EOD which utilizes an all-wheel drive configuration and front-wheel steering shown in the bottom left of Figure 2-9. The RONS is the largest EOD robot currently field with a mass of 307.5kg. In addition to a large manipulator the RONS, shown in the bottom left of Figure 2-9, employs two articulating tracks to improve its mobility.

Figure 3-12: Mobile Robot Platforms Used for Testing

The robot characteristics which are used to determine the endurance range of each platform are listed in Table 2-2. The power losses due to both soil terrain interaction and track loss are a function of the robot's mass and velocity. The gearbox efficiency as noted previously is calculated using the known gear ratio. The motor efficiency was approximated using the known motor characteristics for each platform. The motor controller efficiency was assumed to be operating at peak efficiency.

Table 3-3: Robot Characteristics

	Talon	Tankbot	Bombot	RONS
m_{robot} (kg)	51.5	74.4	13.2	307.5
V_{robot} (m/s)	1.00	2.18	1.23	0.80
$V_{battery}$ (V)	24	36	12	24
$C_{battery}$ (Amp-hr)	18.0	20.8	7.2	54.0
R (Amp-hr)	20	20	20	20
N	1.1	1.3	1.1	1.3

3.6 Results

Each of the four robot platforms were teleoperated around an asphalt track at a constant speed. The robots each started with fully charged batteries and the test was not considered completed until each robot was unable to maintain the speed or in some cases powered off. During each test, the distance traveled and duration of test were recorded and used to calculate the speed for each robot as recorded in Table 2-2. Either through review, or user manuals, or through physical examination, each property from Table 2-2 was measured and used to predict each robot's cruising distance. Figure 3-13 shows the predicted endurance distance plotted against the experimental results. There is a very close correlation between the predicted cruising distance and the actual test data collected for each of the four robots ranging in mass between 13.2 to 307.5 kg.

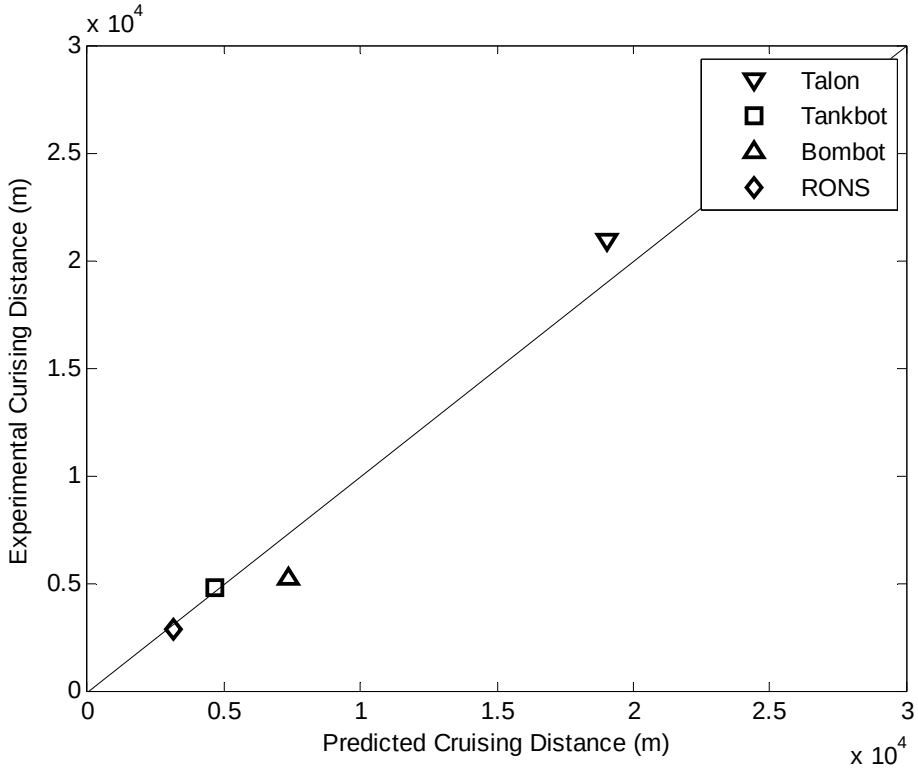

Figure 3-13. Endurance Distance Predicted vs. Actual

3.7 Conclusion

With very few initial characteristics of a mobile robot platform known a priori, a close approximation can be obtained that predicts the robot's performance and size of its components required for mobility. As shown in Figure 3-13 the most accurately predicted endurance range is that of the Tankbot. This is due in part to the use of the TankBot to generate the power loss models for the soil terrain and with the tracks. Even without accounting for variations in drivetrain compositions, materials used, motor types, types of gear reductions used, and auxiliary power drawn, these results demonstrate that it is still possible to closely predict the range a robot can drive by modeling the drivetrain power losses. The models derived in this work provide a close initial approximation which can be used to determine a robot's feasibility early in the design process. Rule based designs can thus cut down on development time and therefore costs by significantly reducing the duration of the prototype and test phases of robot development.

3.8 References

[1] Nicoud, J. (1995). Microengineering: when is small too small? Nanoengineering: when is large too large? *Int. Symposium on Micro Machine and Human Science*, 1-6.

[2] Caprari, G., Estier T., and Siegward R. (2001). Fascination of Down Scaling – Alice the Sugar Cube Robot. *Journal of Micromechatronics*, 1:177-189.

[3] Ehsani, M. and Rahman, K. (1996). Performance Analysis of Electric Motor Drives for Electric and Hybrid Electric Vehicle Applications. *IEEE Power Electronics in Transportation,* 49-56.

[4] Rahman, Z., Ehsani, M., and Butler, K. (2000). An Investigation of Electric Motor Drive Characteristics for EV and HEV Propulsion Systems. *SAE Future Transportation Technology Conference*, Costa Mesa, CA.

[5] Kim, M. and Peng, H. (2006). Power management and design optimization of fuel cell/battery hybrid vehicles. *Journal of Power Sources,* 165:819-832.

[6] Lin, C., Kang, J., Grizzle, J., and Peng, H. (2001). Energy Management Strategy for a Parallel Hybrid Electric Truck. *American Control Conference,* Arlington, VA.

[7] Lin, C., Filipi, Z., Louca, L., Peng, H., Assanis, D., and Stein, J. (2004). Modeling and control of medium-duty hybrid electric truck. *International Journal of Vehicle Design*, 11:349-370.

[8] Petersheim, M. and Brennan, S. (2008). Scaling of Hybrid Energy Vehicle Powertrain Components for Hardware-in-the-Loop Simulation. *IEEE International Conference on Control Applications*, San Antonio, TX.

[9] Madden, J. (2007). Mobile Robots: Motor Challenges and Material Solutions. *Science Magazine*, 318:1094-1097.

[10] Marden, J. (2004). Scaling of maximum net force output by motors used for locomotion. *Journal of Experimental Biology*, 208:1653-1664.

[11] Schilder, R. and Marden, J. (2004) A hierarchical analysis of the scaling of force and power production by dragonfly flight motors. *The Journal of Experimental Biology*, 207:767-776.

[12] Waldron, K. and Hubert, C. (2000). Scaling Robotic Mechanisms. *IEEE International Conference on Robotics & Automation*, San Francisco, CA.

[13] Bekker, M. (1969). *Introduction to Terrain-Vehicle Systems*. Ann Arbour: The University of Michigan Press.

[14] Bekker, M. (1960). *Off-the-Road Locomotion*. Ann Arbour: The University of Michigan Press.

[15] Bekker, M. (1969). *Theory of Land Locomotion*. Ann Armbour: The University of Michigan Press.

[16] Wong, J. (2001). *Theory of Ground Vehicles Third Edition.* New York: John Wiley & Sons, Inc.

[17] Ahlvin, R and Haley, P. (1992). NATO Reference Mobility Model Edition II, NRMM User's Guide, Technical Report. GL-92-19.

[18] Hetherington, J. and White, N. (2002). An investigation of pressure under wheeled vehicles. *Journal of Terramechanics,* 39:85-93.

[19] Hetherington, J. (2001). The application of the MMP concept in specifying off-road mobility for wheeled and tracked vehicles. *Journal of Terramechanics*, 38:63-70.

[20] Hetherington, J. (1991). Electric, All-Wheel-Drive, Tracked Vehicles. *Journal of Terramechanics,* 28:79-85.

[21] Q. Gong, Y. Li, and Z. Peng, "Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles." in *IEEE Transactions on Vehicular Technology*, pp. 3393-3401, 2008.

[22] Peukert Number Derivation (2008) Retrieved January 20 2010 from http://www.smartgauge.co.uk/peukert2 .html.

[23] Roboteq Motor Controller Data Sheet (2008) Retrieved March 05 2008 from http://www.roboteq.com/ brushed-dc-motor-controllers/brushed-dc-motor-controllers-selector.html.

[24] Kenjo, T. and Nagamori, S. (1985). *Permanent-Magnet and Brushless DC Motors*. Oxford: Oxford University Press.

[25] Clark, D. and Owings, M. (2002). Building Robot Drive Trains. New York: McGraw-Hill.

[26] Schempf, H., Crawley, W., Gasior, C., and Moreau, D. (2003). *Ultra-rugged Soldier-Robot for Urban Conflict Missions.* Unmanned Systems Conference, Baltimore, MD.

Chapter 4

System-Level Robot Modeling and ATSV

To design a robot at a system level, component level models for each part of the robot must be linked together in a meaningful way that is representative of the system. The linking of these individual components and evaluation of system's performance is presented in this chapter using a model-based MATLAB® design environment called as Simulink®. Within Simulink®, the interdependencies between each component can be defined so that the power, mass and efficiency of a single component affects both the parts of the robot it is physically connected to, and the design as a whole. While Simulink® can model entire robot designs, it still requires user-defined inputs for many of the subsystems. To automate the entry of data within the model and the visualization of the resulting robot system performance, the Applied Research Laboratory (ARL) Trade Space Visualization (ATSV) tool is used. This software, combined with the Simulink® robot model, allows the designer to "steer" the generation of additional designs, consider design modifications, and view tradeoffs between different criteria and design a robot or group of robots to meet their objective goals.

4.1 Introduction

A comprehensive robot model is too complex to manipulate algebraic equations for individual components from Chapter 2 and Chapter 3 in order to have one equation to optimize the robot design as whole. Each scaling principle from Chapter 3 describes how a robot's powertrain component's mass, volume and, in some cases, efficiency scales as a function of power. Chapter 2 describes how each robot's capability is directly related to its mass, size and available torque and power. Clearly, many of the robot's components and capacities are directly dependent upon one another, so much so that a closed-form equation governing the entire system quite difficult to calculate.

This chapter seeks to bridge the disconnect between component level view and system-level performance. Because size, mass, efficiency, and power are all intertwined in a successful robot design, performance criteria must be designed using models which relate each component's

scaling and performance together. These models will, in many design cases, require iterative performance analysis and parameter variation before the design converges to a solution. MATLAB®, Mathematica and, to a lesser extent, Excel are all numerical simulation packages which can be used to iteratively solve for an optimal robot design, and all have been used to various extent in early software implementations of the rules described in Chapters 2 and 3.

The method discussed here focuses on the implementation of the robot design principles within a visual model design environment MATLAB® known as Simulink® [1]-[2]. The generation of robot designs within Simulink® is driven by a visual steering tool known as ATSV. ATSV (the ARL Trade Space Visualization tool), provides software links to both the inputs and outputs of the Simulink® diagram. ATSV is used to vary the model inputs and view any model outputs. With the link between these two tools established, one or more robot designs can be optimally designed through a trade space study of conceptual simulated designs.

Section 4.2 discusses in more detail the Simulink® simulation environment. Section 4.3 discusses the uses and background of ATSV and the uses of visually steering and trade space exploration. Section 4.4 goes into more detail as to how the components in a robot's powertrain are circularly dependent on one another. This section also describes how these powertrain components can be iteratively sized to optimize the system. With the individual drivetrain components optimized, Section 4.5 discusses how to calculate the geometry of the robot's chassis and the entire platform. Section 4.6 considers the accuracy of the model in predicting system-level physical dimensions and capabilities, by comparing the results to the previously discussed four robot platforms. The relative performance of these predictions can be found in Section 4.7. Conclusions drawn from the Simulink® model evaluation can also be found in Section 4.7.

4.2 Simulation Environment

Simulink® is used to primarily simulate, model and analyze dynamic systems using a graphical representation [2]. Modeling in Simulink® handles both linear and nonlinear systems using both continuous time and discrete time models. Simulink® was chosen for this work

primarily due to its computational flexibility, and because its graphical representation of complex systems facilitates model debugging and understanding.

The robot model is represented in an interconnected block diagram structure shown at a very high level in Figure 4-1. The diagram is broken into a total of fifteen subsystems: twelve of these subsystems represent physical components modeled on the robot, while the other three subsystems represent capability predictions or evaluation calculations.

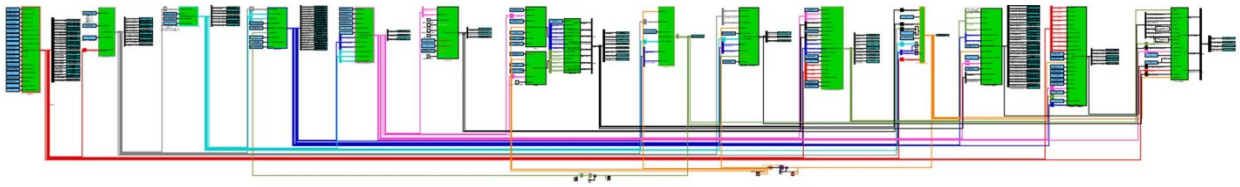

Figure 4-1. Simulink® Robot Model System View

The block diagram in Figure 4-1 is built and evaluated primarily from right to left. The first set of blocks in this figure represents user defined inputs. The architecture of the robot model represented in Figure 4-1 closely follows the power flow architecture in Figure 3-1. The block diagrams moving from the left to right include the battery modeling, motor controller, DC drive motors, gearbox, chassis, structure, and wheels/tracks.

As an example of just one of the blocks, Figure 4-2 provides a closer look at the DC drive motor subsystem. This subsystem uses an embedded function to calculate the characteristics of the motor using curve fits and the set of motor equations from Chapter 2. The outputs from the subsystem are both stored and used in the calculation of gear ratio selection, track sizing and cruising distance calculations. The inputs to the model come from the battery and motor controller subsystems and from circular dependencies of the capabilities subsystems.

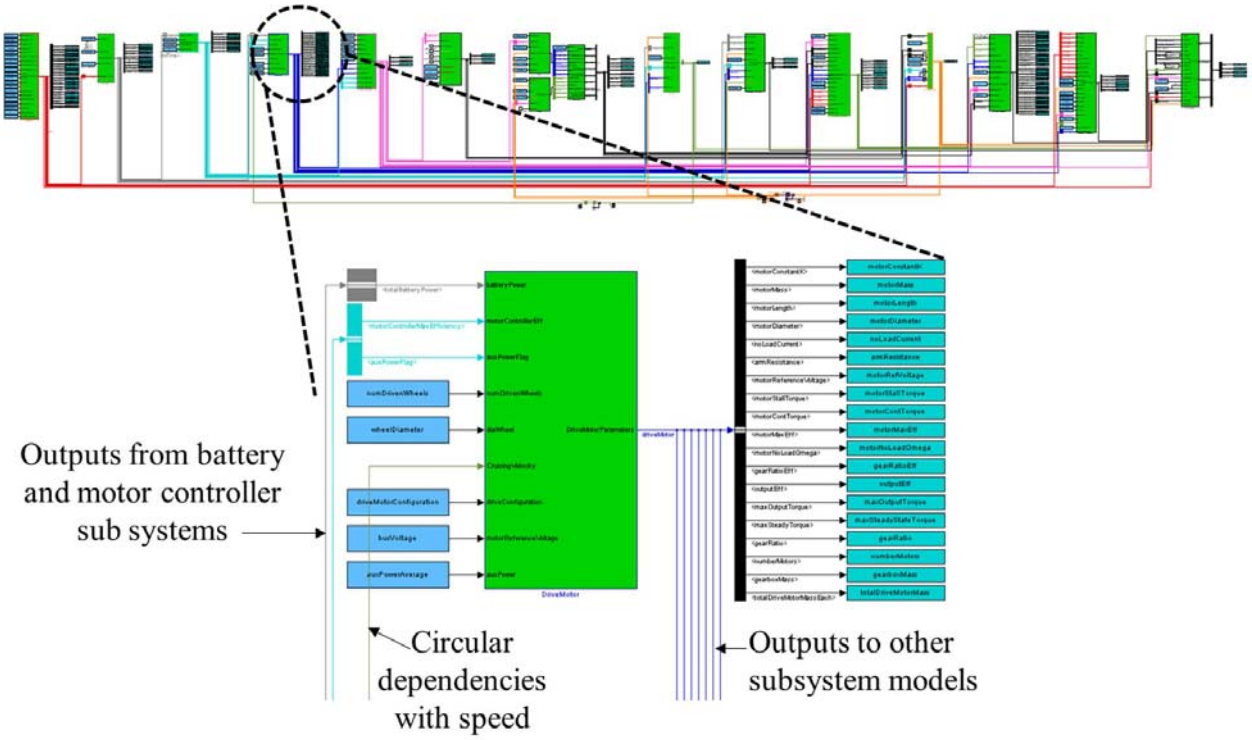

Figure 4-2. Motor Subsystem Simulink® Model

As discussed previously, these subsystems do not link directly left to right in an open loop format. Each subsystem is linked together in feedback loops that iterate on power, efficiency, mass, and volume variables to converge to the design satisfying the user-specified inputs. Circular dependencies exist between each component because each component's power, size and mass are interrelated.

4.3 Visual Steering

ATSV is a Java-based application which is used for design optimization through multi-dimensional visual exploration. Designers can use ATSV to identify optimal conceptual models by plotting identifying relationships between different design variables, applying constraints and applying preferences in real time [3]. ATSV allows the facilitation of the "design by shopping" paradigm where designers can determine the best solution through visualization of tradeoffs in conceptual robot designs in a multidimensional space [4]. Designs can be examined using glyph

plots, histogram plots, parallel coordinate plots, scatter matrices, and linked views [3]. In addition to conventional three-dimensional glyph plots, the graphs' color, orientation, and transparency can be used to represent additional dimensions representing design characteristics of interest [5]. Designers can visually shade their preferences when viewing these designs or utilize tools which allow for the visualization of the Pareto frontier to aid in the decision making process [3], [4].

Visual optimization with ATSV can be enhanced by linking ATSV to a mathematical engine. In this project, ATSV was linked to the mathematical engine used to generate the robot designs, with the result that ATSV can actually drive the generation of additional robot designs. To begin this process, the first step is to explore the trade space and determine the user's preferences and the approximate location of optimal designs in the parameter space. Through the use of attractors, preference samplers and guided Pareto samplers, additional design can be generated around parameter areas representing preferred designs. While there are a number of mathematical engines which can be linked to ATSV, this project compiles the Simulink® model into an executable which can be interfaced directly with ATSV. Thus, without prior knowledge of how each model or component is physically modeled, designers can determine the best system iteratively.

Additionally, optimization through ATSV can be performed on a single design or on a family of products. Optimization of a product family consists of balancing the tradeoffs between performance and commonality across a product family [6]. The insight gained through the examining tradeoffs between performance capabilities across many different missions and/or robot variations allows the designer to iteratively consider the family of robots which balance family commonality and individual mission performance [6].

4.4 System-Level Robot Powertrain Calculation

As noted in Chapter 3, many powertrain components (battery, motor controller, DC drive motors, gearboxes, and drivetrain tracks/wheels) scale as a function of power. Similarly, the losses associated with traversal across a given surface also change as a function of the robot's speed and loading conditions. Figure 4-3 demonstrates the cyclical interdependencies which exist

within each robot that account for these effects. Each line shows dependencies between components, specifically how the output of one component affects another. The arrows in this Figure represent the directions of the interdependency used in the model derivation. For example, the mass of the motor controller, M_{MC} (kg), is scaled as a function of the power of the motor controller, P_{MC} (watts). Thus, the arrow is drawn to point from power to mass in the "Motor Amplifier/Controller" block. Solid lines represent mass, thin double lines represent power, and dotted lines represent speed.

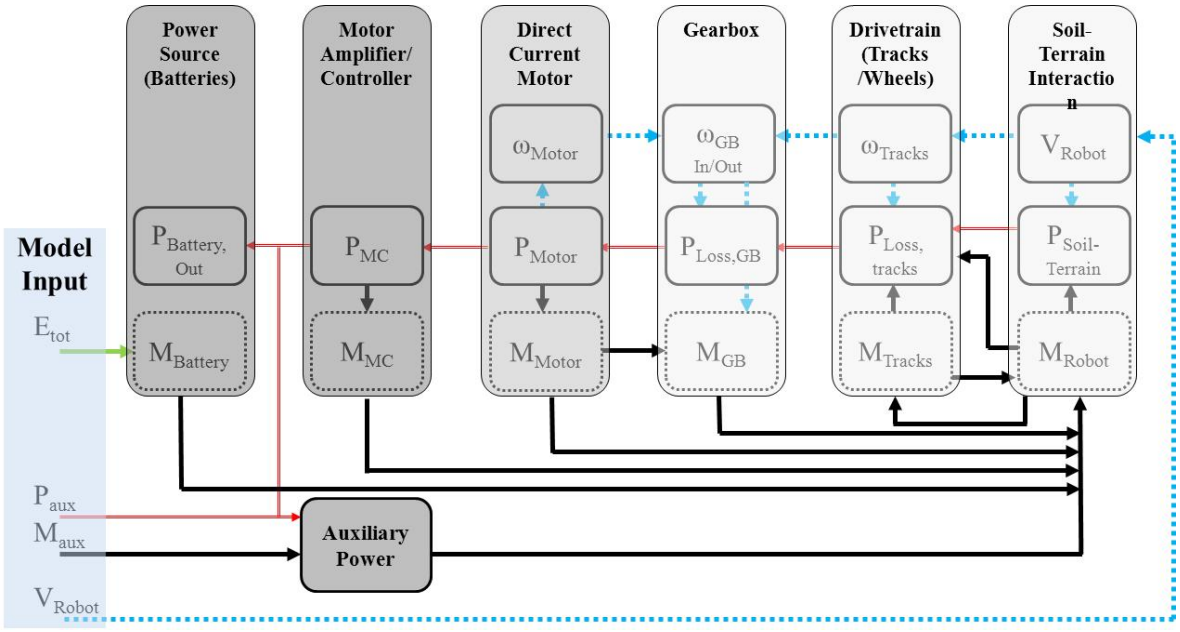

Figure 4-3. Powertrain Subsystem Breakdown

Figure 4-3 shows a number of cyclical dependencies between different subsystems. The first dependency is that of mass and power. Working from the right of the diagram to the left, the power required, $P_{Soil\text{-}Terrain}$, to traverse a given terrain at a given velocity, V_{Robot}, is a function of both the velocity of the robot, the terrain (a user-defined term whose input is not shown passing into this block), and the robot's mass, M_{robot}. The robot's velocity is a model input in this figure, and the total mass of the robot is iteratively solved for to satisfy all the equations of the robot system, starting from a user-specified initial guess and proceeding until convergence criteria are met.

Once the robot's power, mass, and speed are calculated or specified, the characteristics of the drivetrain (wheels/tracks) can be calculated. The drivetrain power losses are a function of both the speed of the robot and its mass as discussed in Chapter 3. The sizes of the tracks/wheels are calculated using user-specified values, and the resulting sizes are used to calculate the robot's linear velocity from the driveshaft rotational velocity. The width of the tracks or wheels are calculated by determining the size that achieves an equivalent ground pressure to a user-specified value. With the diameter and width known, the mass of the drivetrain can be calculated.

The gearbox is designed to match the rotational speed of the tracks/wheels with the rotational speed of the DC drive motors at peak efficiency, assuming a cruising speed of the robot. The efficiency of the gearbox is calculated to be a function of the gearbox ratio per Chapter 3 allometric relationships. Because the motor's speed at peak efficiency decreases with an increase in power, a small interdependent loop is formed wherein the power losses of the gear system will affect the design of the gear system in that a larger motor is required, causing the peak efficiency of the motor to occur at a slightly lower speed. The power and speed of the motor quickly converges, thus giving the final "best design" of the motor gearbox ratio.

With the power of the motor at peak efficiency calculated using the aforementioned feedback loop, all of the necessary characteristics of the motor are calculated using fit curves. The mass, volume, peak efficiency and speed at peak efficiency are all characteristics of the motor that can be calculated using fit curves with power as the independent variable.

The output motor power and efficiency are used to size the motor controller from experimentally-derived fit curves (See Chapter 3). Both the mass and power into the motor controller are thus calculated. The input power to the motor controller can be combined with the user-specified estimates of the auxiliary power draw of the sensor equipment to obtain a total estimate of the power drawn from the battery.

The next step is to calculate the robot's total mass by summing the mass of all previous subsystems as well as the user-specified payload mass. The component masses will change with each changing velocity iteration from the previously mentioned velocity feedback loop. Hence, the total mass of the robot will change and converge as the velocity converges. This convergence

is generally stable as long as the iteration-to-iteration changes in mass are kept small. Of course, each new mass will change the power required to traverse a given terrain, thus starting the feedback loop again. Again, convergence in both mass and velocity are generally observed in a relatively short number of cycles (several hundred) and easily checked either graphically or numerically.

With the major feedback loops explained above, one must note that Figure 4-3 is a simplified loop and does not capture all of the attributes of each of the components nor the system as a whole. There has been little discussion as to how physical size affects this system. To add another level of complexity to this loop, the chassis height is a function of both user inputs and characteristics of the system such as motor diameter and battery volume. The size of the chassis dictates the size of the tracks and wheels. The tracks and wheels dictate the total mass of the robot and vary the required gearbox ratio and motor sizing. Mass, volume, efficiency, power, and speed are all interdependent and need to be solved iteratively for each component in the robot's drivetrain to develop a convergent optimal robot design.

Once all variables in the iterative robot design model have converged, one can use the resulting speed, mass, etc. to evaluate system-level capabilities. The total cruising distance is calculated using the battery equations which predict battery life as a function of power draw from Chapter 2, coupled with the required input power to the motor controller and auxiliary power draw. Gross estimates of the robot's geometry can be obtained using the component scaling models mentioned earlier, and the user-defined payload volume. The drivetrain characteristics that result from the converged design allow one to obtain estimates of the robot's cruising distance.

4.5 System-Level Robot Dimensions

The robot's core chassis is assumed to enclose the robot's batteries, motor controller, drive motors, drive motor gearboxes, and any user-defined internal payloads. The robot's physical size is the summation of the chassis and the tracks (or wheels) which are assumed to protrude outside of the chassis. This section discusses how to compute the chassis and total vehicle properties with the internal components computed using the method above.

4.5.1 Chassis

The maximum width of the chassis of a robot, $W_{Chassis}$, is computed as the maximum of three internal components as expressed in Equation [4-1]. The widest component is either the battery, $W_{Battery}$, the payload, $W_{Payload}$, or the initial width approximation constant, $W_{Chassis,int}$. The initial width approximation is calculated by summing the payload length, $L_{Payload}$ and the battery Length, $L_{Battery}$, and dividing it by the length-width skid-steer vehicle ratio, R_{LW} (Equation [4-2]). The length-width skid-steer ratio constant for skid-steer vehicles is obtained by examining many different successful robot and tank designs, and is found to be approximately 1.65.

$$W_{Chassis} = \max\left(W_{Battery}, W_{Payload}, W_{Chassis,\text{int}}\right) \quad \textbf{[4-1]}$$

$$W_{Chassis,\text{int}} = \frac{L_{Payload} + L_{Battery}}{R_{LW}} \quad \textbf{[4-2]}$$

The length of the chassis, $L_{Chassis}$, is the product of the length to width ratio constant and the computed width of the chassis (Equation [4-3]). Similarly the height of the chassis, $H_{Chassis}$ in Equation [4-4], is the maximum height of either the payload, $H_{Payload}$, or the height of the motor controller, H_{MC}, coupled with the diameter of the motor, D_{Motor}.

$$L_{Chassis} = R_{LW} \cdot W_{Chassis} \quad \textbf{[4-3]}$$

$$H_{Chassis} = \max\left(H_{Paylaod}, H_{MC} + D_{Motor}\right) \quad \textbf{[4-4]}$$

4.5.2 Total Robot Dimensions

The width of the robot with tracks, $W_{Robot,Tracks}$ in Equation [4-5], is calculated as the width of the chassis combined with the width of a single track. Only a single track is used as this model assumes that half of each track covers the chassis, as this is common on EOD robots for space savings. The width of a wheeled robot, $W_{Robot,Wheels}$ in Equation [4-6], is calculated as the summation of the chassis width and both of the widths of the wheels, W_{Tire}. This model assumes that the wheels do not cover any portion of the width of the chassis.

$$W_{Robot,Tracks} = W_{Chassis} + W_{Tracks} \quad \textbf{[4-5]}$$

$$W_{Robot,Tire} = W_{Chassis} + 2 \cdot W_{Tire} \quad \textbf{[4-6]}$$

The length of the robot, L_{Robot}, is the summation of the length of the chassis and the diameter of the wheels or tracks less the diameter of the motor. Half of the track sprockets (or tires) are assumed to protrude past each end of the robot's chassis. The total length of the robot is calculated using Equation [4-7]. The robot's height, H_{Robot}, is either equal to the height of the robot's track/tires or the height of the top of the robot's chassis to the ground, whichever is higher. This is represented by the calculation of $H_{Chassis\text{-}Ground}$ in Equation [4-8].

$$L_{Robot} = L_{Chassis} + D_{Tracks\ /\ Tires} - D_{Motor} \quad \textbf{[4-7]}$$

$$H_{Robot} = \max\left(D_{Tracks\ /\ Wheels}, H_{Chassis\ -\ Ground}\right) \quad \textbf{[4-8]}$$

4.6 System-Level Robot Powertrain Calculation

The list of user-specified inputs required to generate a robot model is listed in Table 4-1. These inputs include the robot's power system characteristics, internal payload specification, drivetrain configuration, and some additional chassis geometries. These specifications are the only information required to perform the iterative optimization outlined in Section 4.4 and the system-level chassis and robot calculations discussed in Section 4.5. The outputs to this model include the physical characteristics of each drivetrain component as discussed in Chapter 3. These outputs, coupled with the specified payload characteristics, are used to build the system performance and overall robot geometry used to evaluate the capabilities of the platform as discussed in Chapter 2.

Table 4-1: Simulink® Model Inputs

Power System	Battery Type (L-ion, NiMH or Lead Acid)
	Bus Voltage (V)
	Battery Capacity (A-hr)
Payload	Auxilary Power Draw (watts)
	Payload Length (m)
	Payload Width (m)
	Payload Height (m)
	Payload Mass (kg)
Drivetrain	Wheels or Tracks
	Drive Motor Configuration
	Number of Driven Wheels/Sprockets
	Number of Wheels
	Wheel/Sprocket Diameter (m)
	Wheel/Sprocket Material
	Ground Pressure (kPa)
	Sprocket Configuration (disk, I-beam, drum)
Chassis Geometry	Additional Front Length (m)
	Ground Clearance (m)

4.7 Simulink® Modeled System Results

The work in this thesis was conducted alongside a graduate student, Aaron Bobuk, whose study focused on the design process of a robot, and so many details of the generation of an executable model from the design rules from Chapter 2 and Chapter 3 can be found in his M.S. thesis [7]. A more detailed discussion of the subsystem models are discussed therein as well, focusing on how each subsystem pertains to the system as a whole. Attention is placed on specific Simulink® coding practices and details which allow the implementation of the rule sets from the previous chapters.

A key goal of this thesis was to verify the model predictions both at a component level and a system level, and verification was assessed by comparing the model-predicted results with measurements obtained from field tests of physical robots. The same model comparisons from Chapter 2 and Chapter 3 are performed here using the predicted results from each of the four robot platforms discussed previously. These robot designs are generated only by varying the Simulink® model inputs discussed in Section 4.6. Some of these Simulink® model inputs are

generated from physical measurements taken from each platform. And others, such as payload characteristics, are varied to achieve a robot Simulink® robot model similar to the actual robot and thus infer the payload specification of a presently fielded robot. Table 4-2 contains both the model-predicted robot results and actual measured data from each platform.

Table 4-2: Simulink® Model Comparison Results

	Talon			Tankbot			Bombot			RONS		
	Predicted	Actual	% Diff.	Predicted	Actual	% Diff.	Predicted	Actual	% Diff.	Predicted	Actual	% Diff.
Battery Mass (kg)	7.305	7.47	2.21%	10.8	10.64	1.50%	1.44	2.668	46.03%	33	37.358	11.67%
Battery Volume (m^3)	0.0037	0.0067	44.78%	0.0043	0.0047	8.51%	0.006	0.0018	233.33%	0.0132	0.0174	24.14%
Drive Motor Mass (kg)	1.66	2.023	17.94%	1.9	2.03	6.40%	0.25	0.25	0.00%	8.88	10.28	13.62%
Drive Motor Volume (cm^3)	629.06	227.12	176.97%	737.49	693.99	6.27%	69.04	68.28	1.11%	4423.06	1814.92	143.71%
Drive Motor Gearbox Ratio	20.01	20	0.05%	9.78	10	2.20%	6.18	6	3.00%	19.05	20	4.75%
Drive Motor Gearbox Mass (kg)	2.72	4.05	32.84%	3.04	4.06	25.12%	0.39	0.5	22.00%	14.53	15.42	5.77%
Chassis Width (m)	0.44	0.44	0.00%	0.37	0.37	0.00%	0.13	0.14	7.14%	0.4	0.4	0.00%
Chassis Height (m)	0.19	0.19	0.00%	0.33	0.17	94.12%	0.13	0.13	0.00%	0.32	0.25	28.00%
Chassis Mass (kg)	9.27	9.26	0.11%	9.89	9.88	0.10%	2.98	2.98	0.00%	18.68	18.68	0.00%
Robot Width (m)	0.6	0.57	5.26%	0.45	0.55	18.18%	0.37	0.47	21.28%	0.47	0.72	34.72%
Robot Length (m)	0.9	0.864	4.17%	0.96	0.96	0.00%	0.64	0.64	0.00%	0.8	0.74	8.11%
Wheelbase Length (m)							0.43	0.31	38.71%			
Wheelbase Width (m)							0.09	0.13	30.77%			
Wheel Mass (kg)							3.47	2.99	16.05%			
Track Width (m)	0.155	0.155	0.00%	0.083	0.083	0.00%				0.07	0.07	0.00%
Track Mass (kg)	1.21	1.2	0.83%	0.64	3.2	80.00%				2.54	1.91	32.98%
Sprocket Mass (kg)	2.12	2.1	0.95%	3.29	2.39	37.66%				16.7	13.43	24.35%
Maximum Velocity (m/s)	1.6	2.25	28.89%	1.47	2.37	37.97%	1.45	0.97	49.48%	4.53	4.52	0.22%
Endurance Time (hrs)	2.58	2.67	3.37%	1.36	1.32	3.03%	1.31	1.19	10.08%	1.05	0.99	6.06%
Endurance Distance (km)	20.27	20.92	3.11%	4.89	4.75	2.95%	5.79	5.24	10.50%	3.03	2.85	6.32%

4.7.1 Battery Model

To predict the size and mass of the battery, the battery composition (lead acid, nickel metal hydride or lithium-ion), voltage (V) and capacity (A-hr) are required. This data is collected through physical measurements and by vendor data. Figure 4-4 and Figure 4-5 show that the Simulink® model accurately predicts both the mass and volume each of the batteries. Each deviation from the predicted values is biased towards the actual batteries being heavier and larger. The models developed to scale batteries are based on the physical limitation of the chemistry of the battery. The actual batteries which are fielded will weigh more and are larger due to packaging, physical interfaces (connectors) and in some cases monitoring or protection equipment that are integrated with the battery.

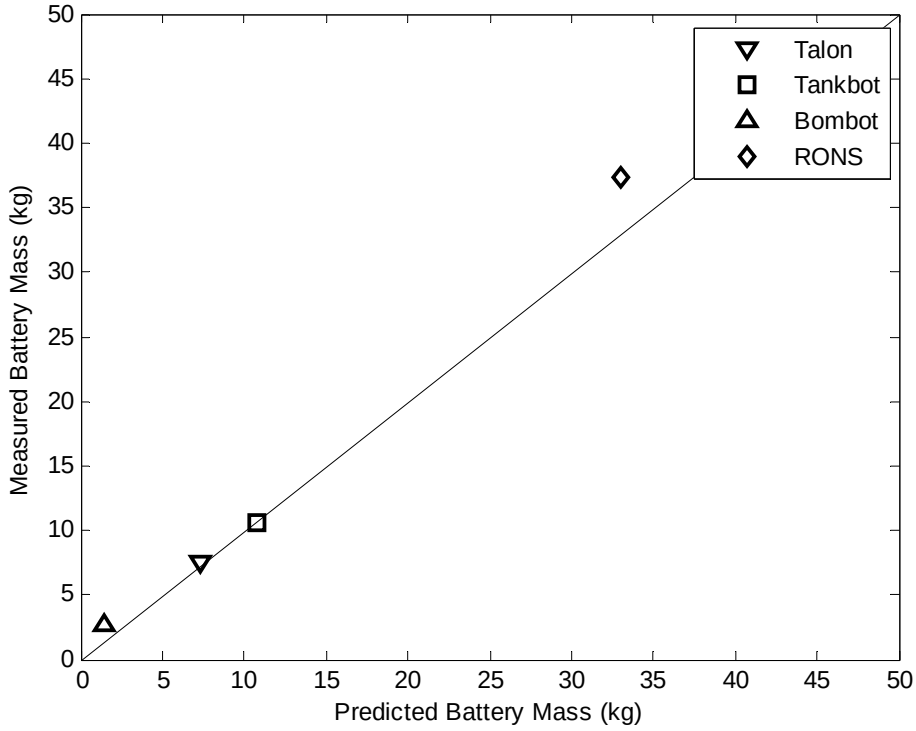

Figure 4-4. Comparison of Simulink® Battery Mass (kg) Prediction to Actual Data

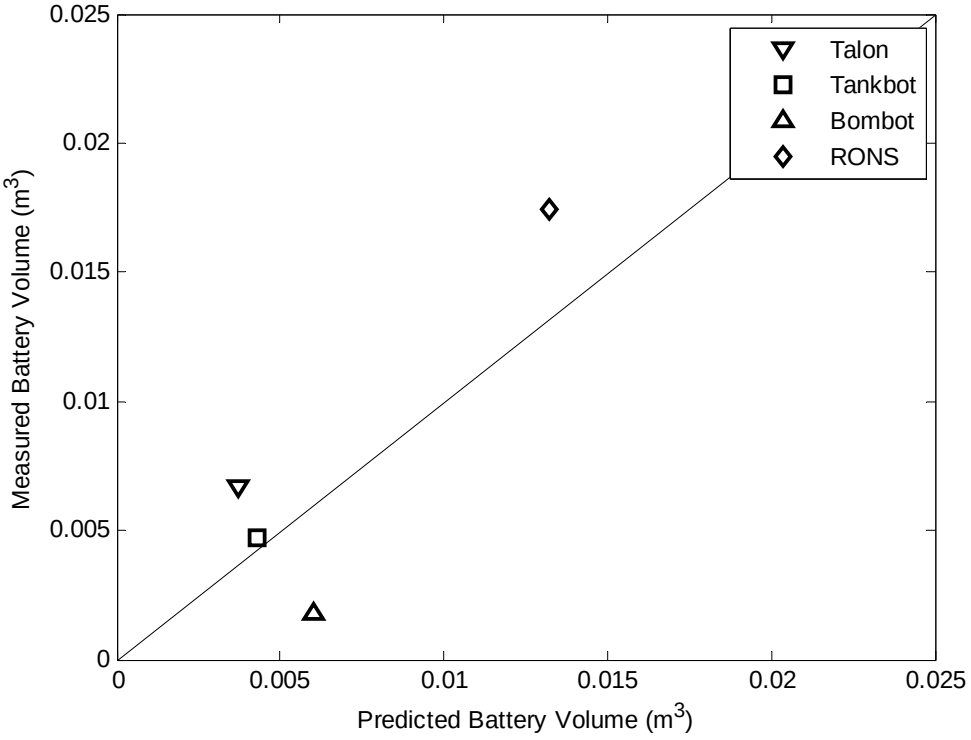

Figure 4-5. Comparison of Simulink® Battery Volume (m^3) Prediction to Actual Data

4.7.2 Drive Motor Model

Each drive motor characteristic is scaled as a function of the power of the motors using curve fits generated from vendor data as noted in Chapter 3. The comparison of the Simulink® model predicted data and physical robot measurements can be found in Figure 4-6 and Figure 4-7. While the models are a good approximation for the physical characteristics of the motor, there is some deviation in the larger robots. The inability to more accurately predict the performance of the drive motors of the robot is due to insufficient vendor data about larger scale motors. While there is an abundance of motor designs and vendors for smaller motors, very few brushed DC motor designs exist in the power level comparable to those needed by the RONS. This is the physical size range which is most often dominated by induction motors and alternative sources for propulsion such as internal combustion engines.

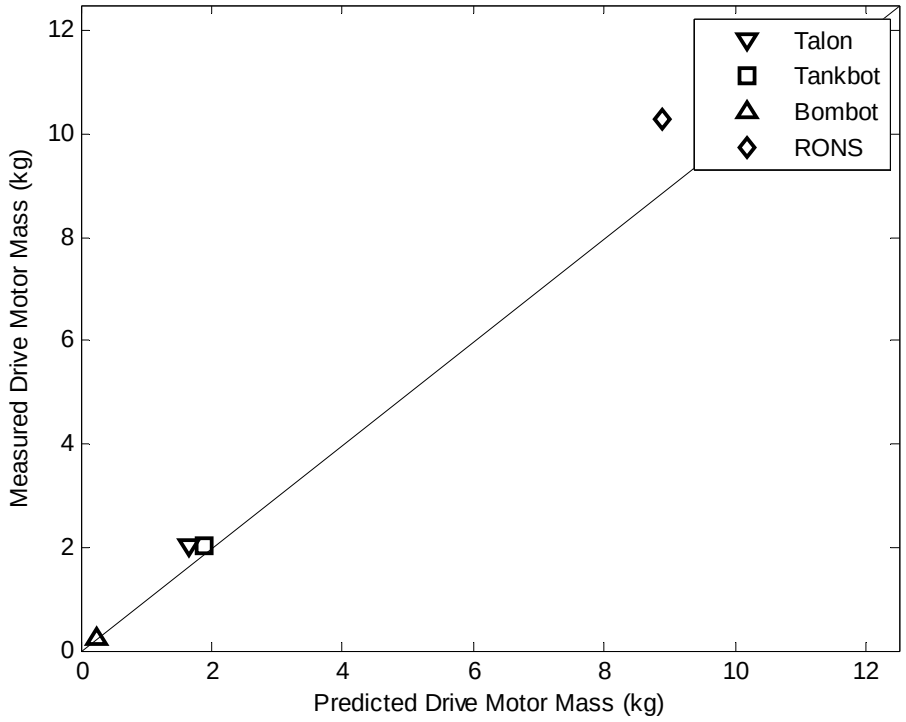

Figure 4-6. Comparison of Simulink® Motor Mass (kg) Prediction to Actual Data

The other source of error in this comparison comes from how motors and gearboxes are sized for each conceptual robot design. Robot's endurance cruising distance was measured experimentally at a given speed. The speed for this test was chosen to be equal to a human's pace and a speed which could be kept constant and monitored as the battery discharged. Each conceptual robot was then optimized to meet that speed. Optimization in this context means that each component of the robot's powertrain was scaled to operate at peak efficiency at this speed. The motors were sized to be big enough to meet these speeds and the gearbox ratios were matched to allow the motor to operate at peak efficiency. The speeds each conceptual robot design was optimized around however may be different than the speed each physical robot was built for. The difference between the speed each physical robot and corresponding conceptual robot design were optimized for corresponds to a variation in the size of the motor, gearbox and other powertrain components.

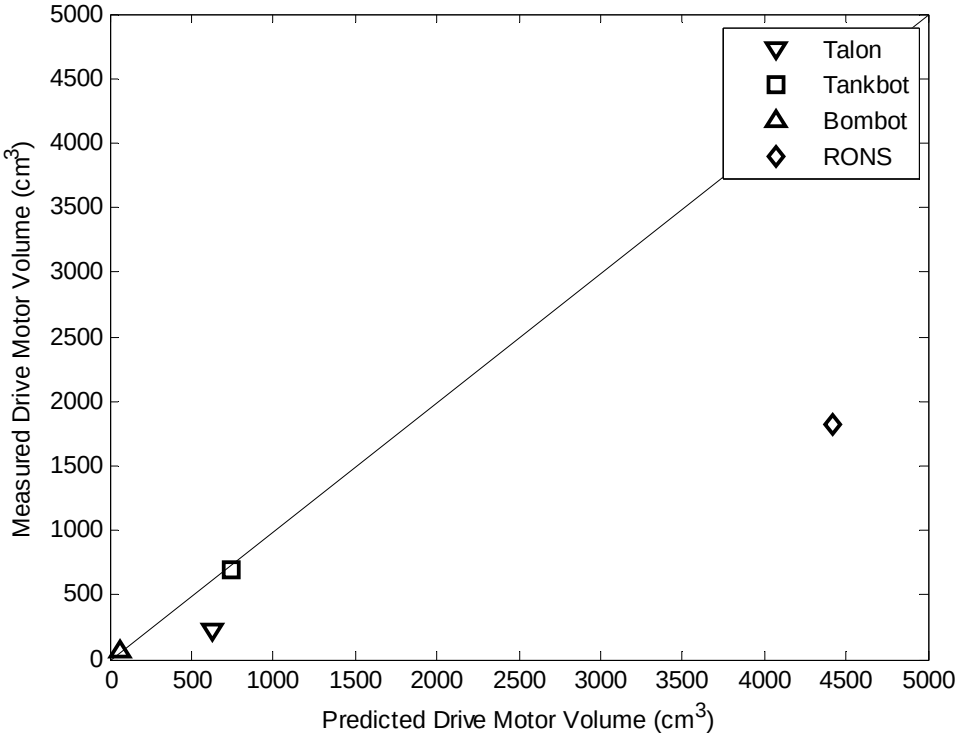

Figure 4-7. Comparison of Simulink® Motor Volume (cm^3) Prediction to Actual Data

The cause for the Simulink® model to over predict the volume in Figure 4-7 may be due in part to the physical construction of each motor. Each of the drive motors that are largest anomalies to the curve fits were custom built for their respective robot. These motors may be more efficiently packaged with copper cores wound more tightly than is typical in commercial generic motors.

4.7.3 Gearbox Model

The gearbox ratio is calculated to match the cruising velocity of the robot to the speed the motor assuming the motor operates at peak efficiency at the cruising velocity of the robot. The gearbox efficiency is solely a function of the gearbox ratio. As stated previously, the operating speed obtained from the Simulink® model after optimization for peak cruising efficiency may be

different than the intended operating speed. This would account for the discrepancy between the model-predicted and actual gearbox ratios shown in Figure 4-8.

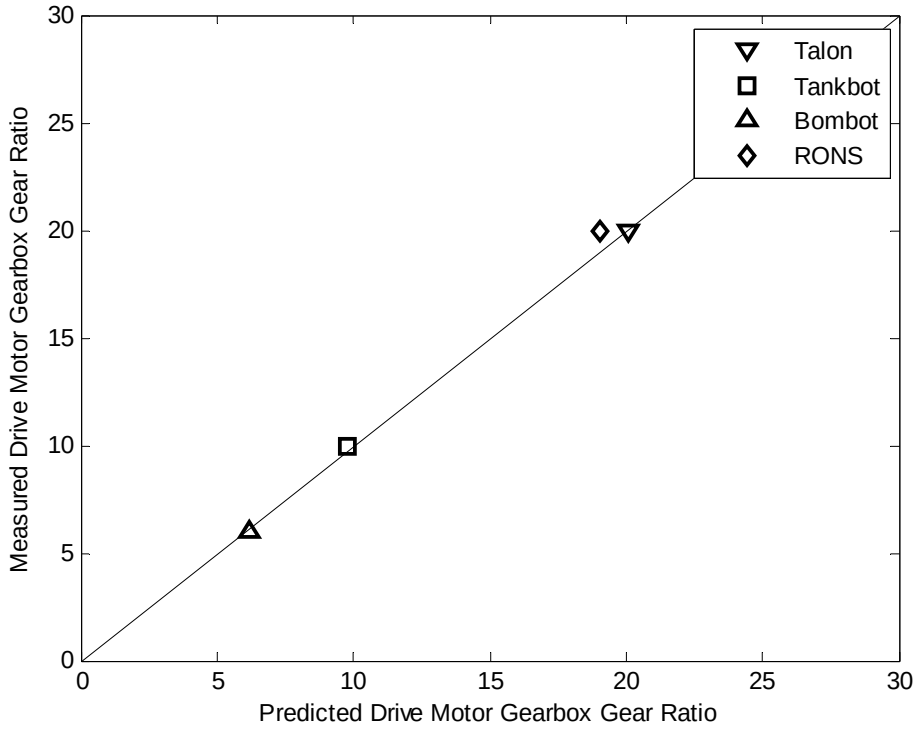

Figure 4-8. Comparison of Simulink® Gearbox Ratio Prediction to Actual Data

The mass of the gearbox is calculated using the gearbox ratio and mass of the drive motors. The model error from the gearbox gear ratio and drive motor mass predictions may account for the variation between the predicted gearbox mass and actual mass shown in Figure 4-9. The mass of the actual drive motors from Figure 4-6 are greater than the predicted values. The actual gearbox masses of these same robots will thus also be greater than the predicted values.

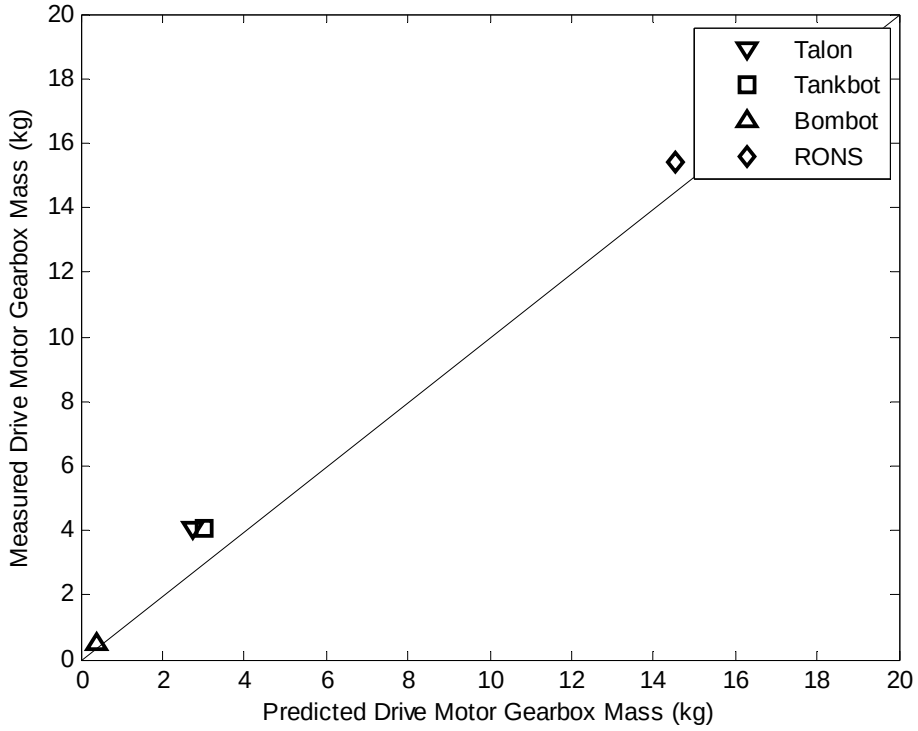

Figure 4-9. Comparison of Simulink® Gearbox Mass (kg) Prediction to Actual Data

4.7.4 Chassis Model

The chassis model predictions are simplistic views of the complexity of the internal components within a robot and the packaging solutions used to build an integrated system. With all of the additional equipment typically found on a robot, it can be difficult to objectively partition components that definitely belong to the chassis losses versus components that are better described as external payloads. Figure 4-10 and Figure 4-11 show that each robot's chassis height and width can be accurately predicted through manipulation of the payload dimensions. The perfect fit does not imply an accurate model, but simply that there is some user-specified payload that would generate a height that is equivalent to these four particular robots.

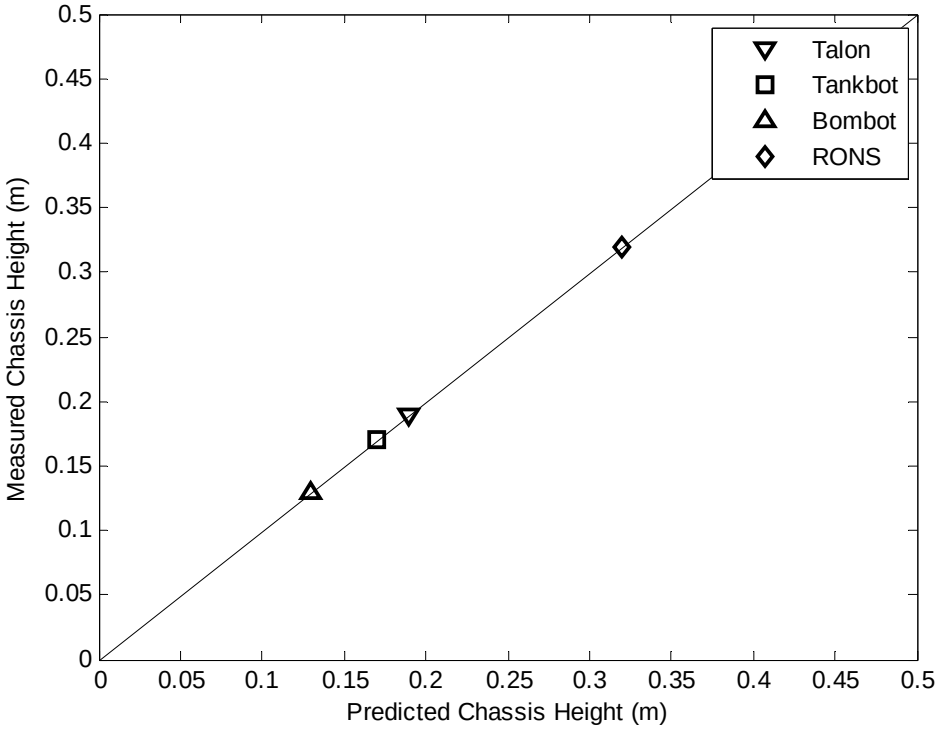

Figure 4-10. Comparison of Simulink® Chassis Height (m) Prediction to Actual Data

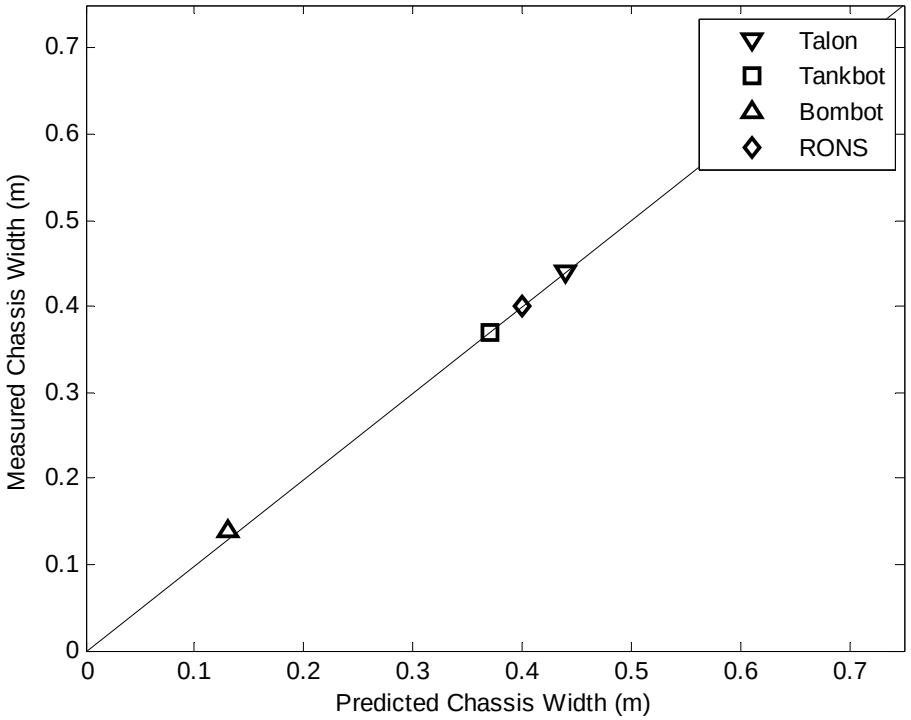

Figure 4-11. Comparison of Simulink® Chassis Width Prediction to Actual Data

The payload mass for each conceptual robot was varied as a user input to achieve equivalent chassis masses as shown in Figure 4-12. Without specifying a payload, each conceptual robot design is optimized to move only the powertrain components required for locomotion. These conceptual designs may be the same dimensions as the physical robot's they are representing but are grossly underweight. Adding payload mass to the robot serves two purposes. The first benefit of matching these chassis masses occurs when testing system performance capabilities. Doing so allows the comparison of system performance to be focused on modeling the capabilities rather than on possible biases due to inaccurate robot system comparisons. The second purpose of matching chassis masses comes from the chassis mass's effect on component scaling. Robot power requirements to traverse a given terrain scale with mass. Increasing the chassis mass of the conceptual robot design, will correspondingly increase the mass of the motors, gearboxes and other powertrain components.

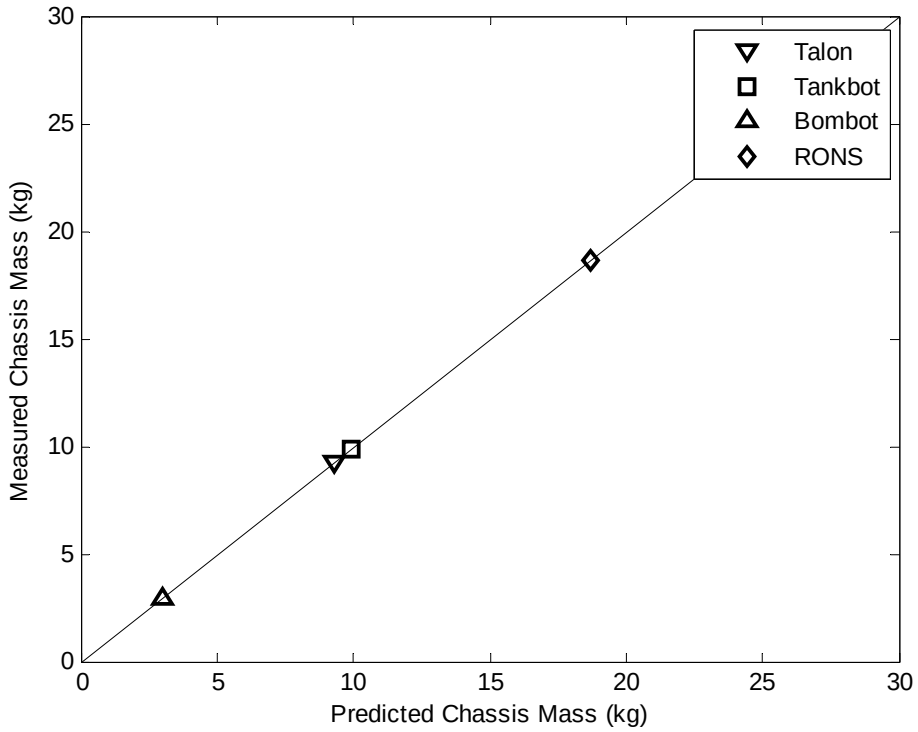

Figure 4-12. Comparison of Simulink® Chassis Mass Prediction to Actual Data

4.7.5 Track/Tire Model

The track width is varied to achieve an equivalent ground pressure with a known vehicle length as noted in Chapter 3.The internal payload mass and geometry was varied as a user input to match vehicle and track lengths. Using the equivalent ground pressure calculation Figure 4-13 shows an agreement between the model and actual values.

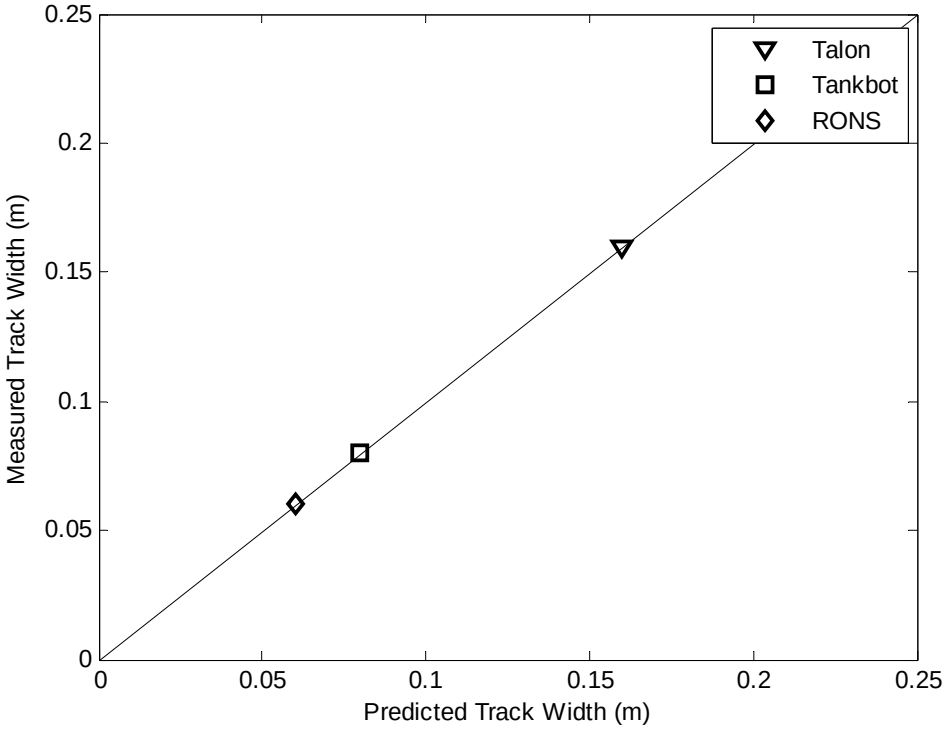

Figure 4-13. Comparison of Simulink® Track Width (m) Prediction to Actual Data

The model accurately predicts the mass of both RONS and talon's track mass in Figure 4-14. The track scaling model is based on a model with the talon's proportions and track configuration. The RONS has a similar track configuration and the actual tracks are made of the same materials as the model predicts. The Tankbot, on the other hand, has a much heavier track mass than the predicted value because the tracks are rubber bonded to plastic rather than pure rubber tracks. This additional material accounts for the increased mass. It also demonstrates a useful feature of the model: to clearly point out portions of a robot that may be grossly overbuilt, as are the tracks on this particular robot.

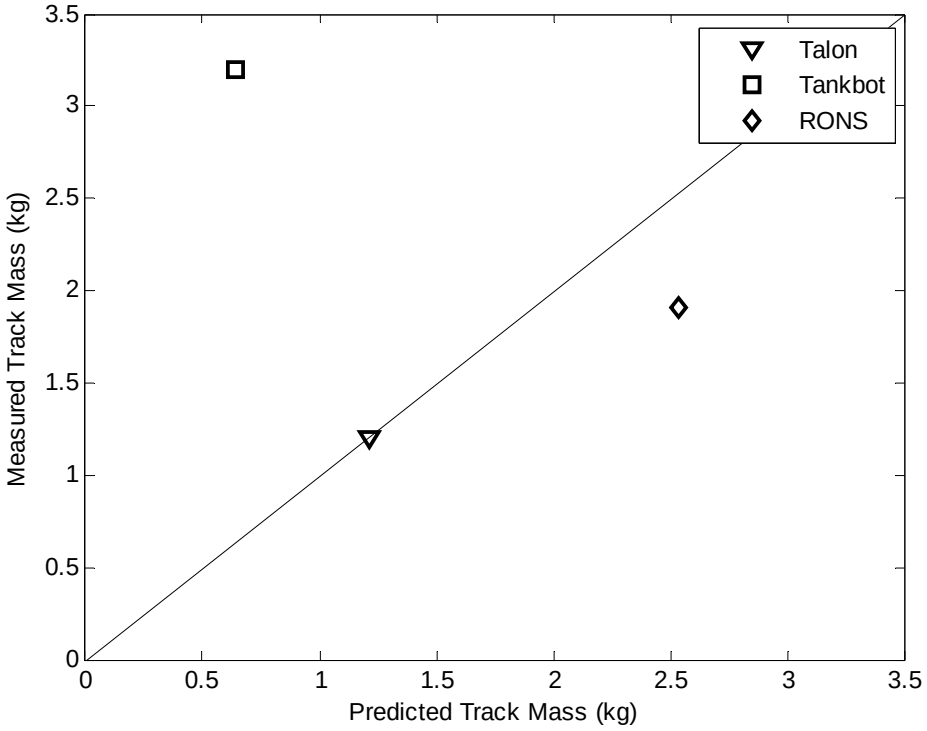

Figure 4-14. Comparison of Simulink® Track Mass (kg) Prediction to Actual Data

Figure 4-15 shows a close agreement between the mass of the sprockets of Simulink® modeled robots and the actual modeled data. The accuracy of this approximation is due to the accuracy of the chassis predictions and track sizing work above. The model over predicts the mass of the RONS's sprockets because it does not take into consideration the hollow drum construction of the sprocket.

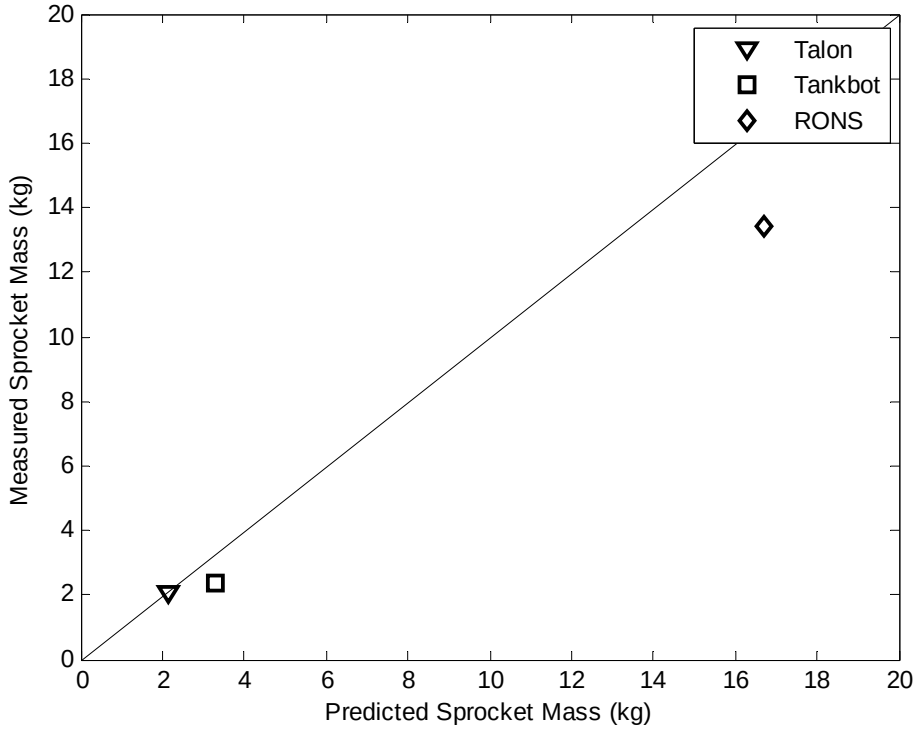

Figure 4-15. Comparison of Simulink® Sprocket Mass (kg) Prediction to Actual Data

4.7.6 Robot Model

Figure 4-16 and Figure 4-17 show that while there is a good agreement from each of the subsequent subsystems, the overall vehicle properties do not match exactly. This is due in part to variations in design choices for each robot. Each robot has a different track configuration and component packaging configurations. While it is possible to vary the internal payloads to match some properties, there is still a certain level of disagreement between actual builds and simulation predictions. This is because of certain inherent assumptions made with regard to the length to width ratios of the robot, component packaging, and overall design of the robot. But even while these robots are built to accomplish different tasks and thus exhibit such variations, they are still approximated remarkably well by a single Simulink® robot model capable of predicting their overall design.

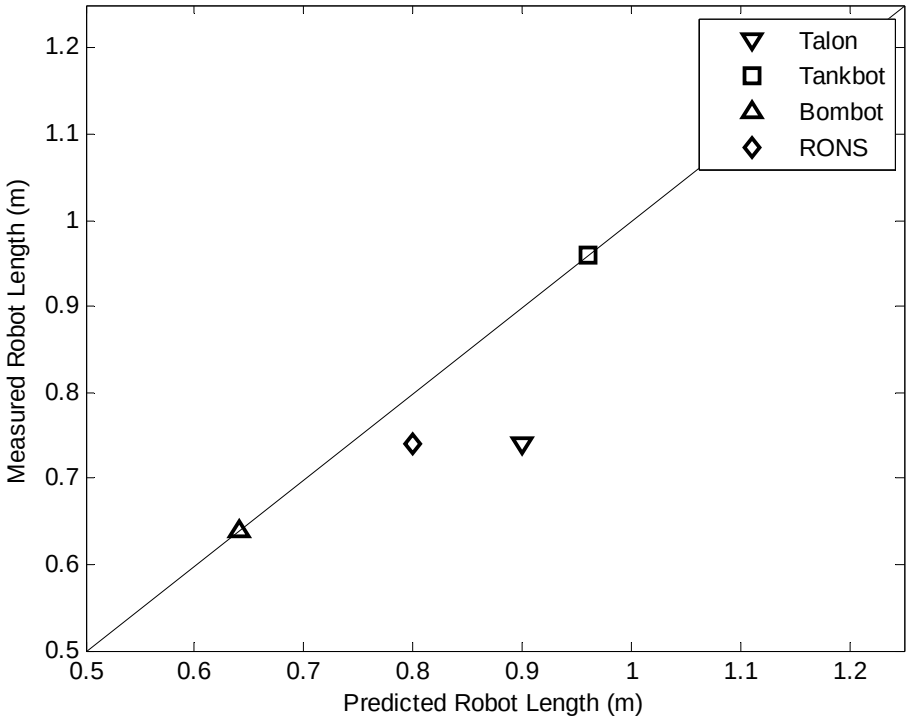

Figure 4-16. Comparison of Simulink® Robot Length (m) Prediction to Actual Data

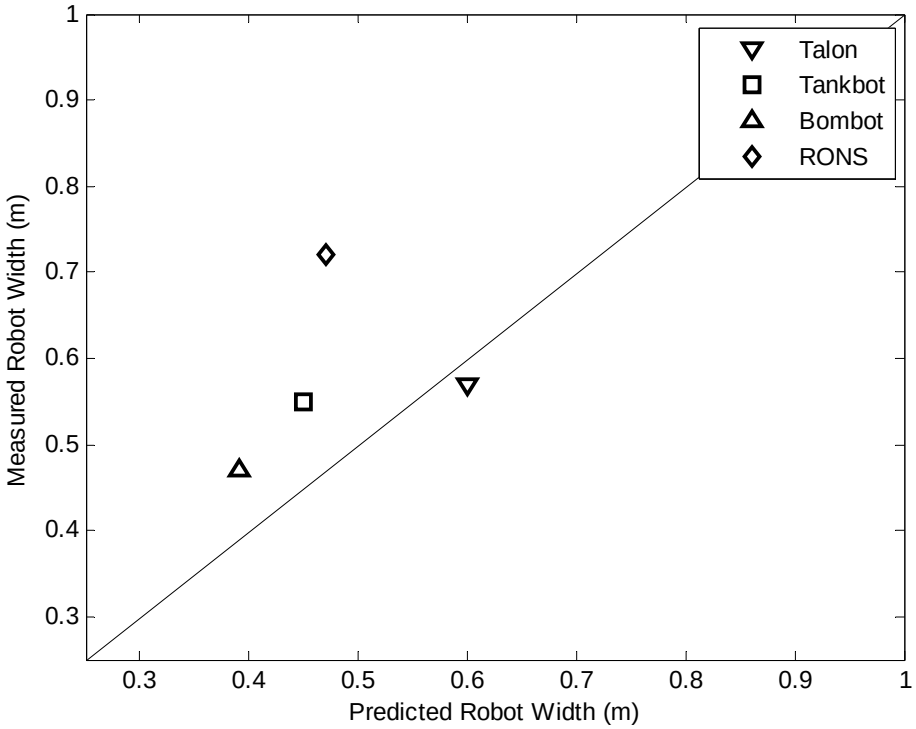

Figure 4-17. Comparison of Simulink® Robot Width (m) Prediction to Actual Data

4.7.7 Robot Maximum Velocity Model

Figure 4-18 shows the accuracy of the maximum velocity prediction equations. The variation of the model-predicted performance to the actual performance of each platform is caused by the method of scaling the drive system to operate at peak efficiency. As stated previously, each conceptual robot model is optimized to operate at peak efficiency at the speed used to test cruising distance. The speed the conceptual robot models were optimized around was chosen because the speed could be kept constant throughout the discharge cycle of the battery, easily measured and equivalent to human walking speeds. The speed each conceptual robot design was optimized around was not the speed that the physical robot was built around. There is a known correlation which is derived using the laws of physics between speed at peak efficiency of a motor and maximum speed of the motor. Figure 3-5 and Figure 3-6 show speed, power and efficiency curves for a brushed direct current motor. Peak efficiency of that motor always occurs at approximately one third the maximum speed of the motor. A variation in the conceptual

robot's speed at peak efficiency from the physical robot's speed at peak efficiency will correspond to a variation in the robot's peak velocity capability.

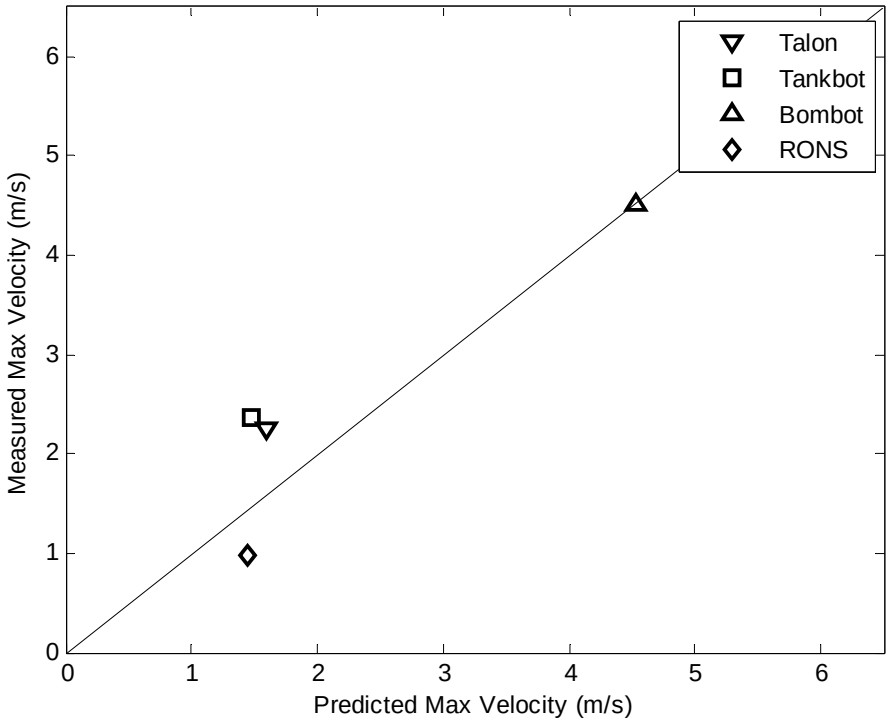

Figure 4-18. Comparison of Simulink® Maximum Velocity (m/s) Prediction to Actual Data

Figure 4-18 supports this argument. The RONS, a relatively slow robot, was tested at a speed near its peak velocity which causes the model to over predict its total possible speed. The Talon and Tankbot were tested at approximately the same cruising speed as the RONS. This cruising speed was much lower than either platforms' maximum speed which caused the model to under predict each platforms maximum velocity.

4.7.8 Robot Endurance Model

The distances each robot can travel without charging its batteries, endurance distance, and the duration of that operation, battery endurance time, are calculated using the subsystem

models in Chapter 3, along with the system-level models and iterative dependencies discussed in this chapter. While not all of the subsystem models or physical system-level models are precise matches, both the endurance distance and endurance time are predicted quite accurately in the Simulink® robot models. Figure 4-19 shows that the Simulink® robot model accurately predicts the endurance time of the robot traveling at a constant velocity. This calculation is made primarily by calculating the power required to traverse over a given terrain and through modeling all of the efficiency losses throughout the powertrain. The battery model in Chapter 3 shows that the equivalent battery capacity changes as power is drawn beyond what the battery's rated power. The corrected power, coupled with the battery model, allow a new battery operation time to be calculated. This operation time is represented in Figure 4-19.

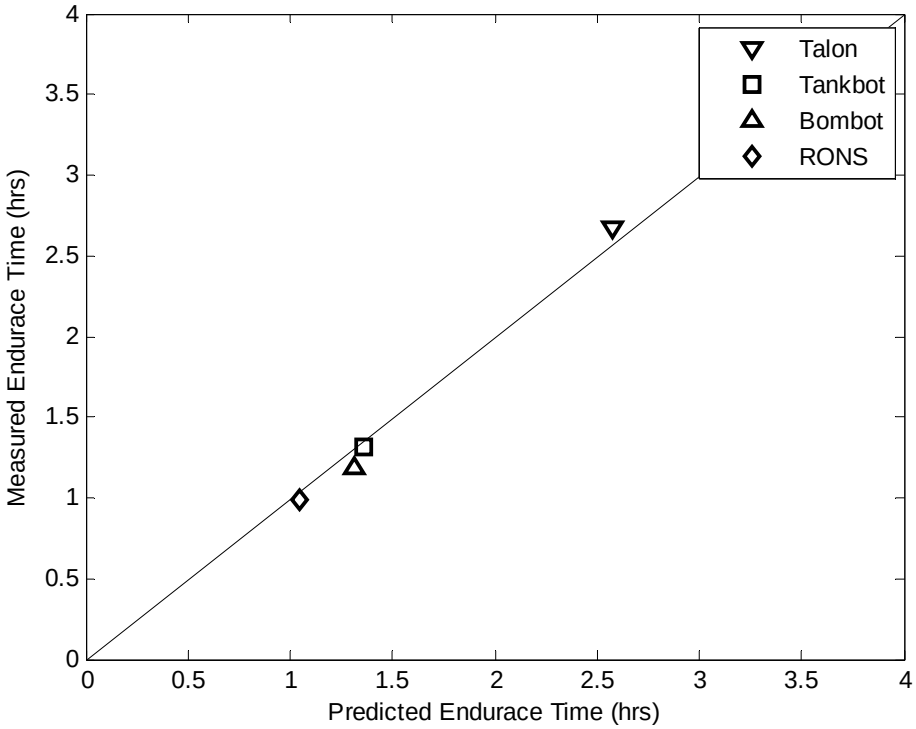

Figure 4-19. Comparison of Simulink® Battery Endurance Time (hrs) Prediction to Actual Data

The product of the endurance time and the robot velocity for which it is being optimized yields the robot endurance distance shown in Figure 4-20. The endurance time and distance are accurately predicted for the four robots being modeled. Unlike other model approximations,

there is not a direct bias towards the model over- or under-predicting this capability. Figure 4-20 shows that the model over predicts the RONS endurance distance but under predicts that of the Bombot.

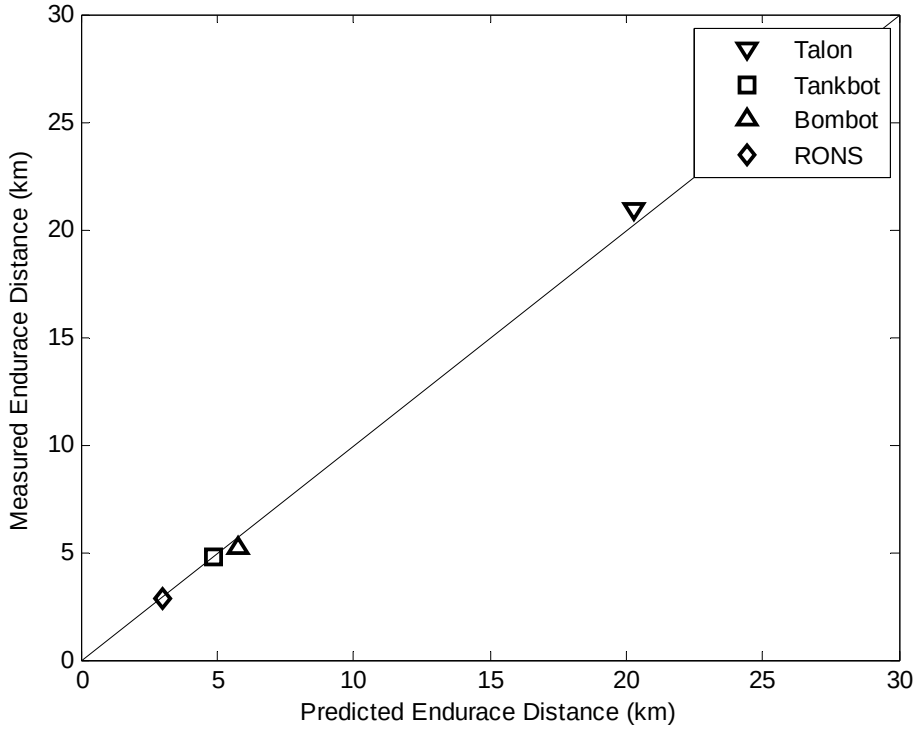

Figure 4-20. Comparison of Simulink® Battery Endurance Distance (km) Prediction to Actual Data

4.8 Relative Performance of Conceptual Robot Models

The previous section demonstrates the validity of the conceptual design models by comparing them with measured characteristics and performance capabilities of existing physical mobile ground robots. This section compares the performance of these existing physical robots with a design space populated with a number of conceptual designs. The purpose of doing so is to show how well randomly sampling inputs into the conceptual robot design space captures the

performance of actual robots. In the cases where a physical robot's performance or characteristics is not captured within the design space, this section discusses weather this is a error in modeling or with the manner in which the robots are sampled.

Results from the population of conceptual robot models viewed using ATSV can be found in Figure 4-4 through Figure 4-20. Each small dot in these figures represents a conceptual robot designs which was generated randomly using the Simulink® robot modeling effort. The larger geometric shapes have been added over each point which represents the performance of physical EOD robots taken from experimental data. These experimental results are the same data used to validate the modeling efforts in this and the last previous two chapters. The following figures are a sample of the total possible tradeoffs and comparisons which could be made. Each comparison (x and y-axis) was chosen to show a possible trade-off in a conceptual robot design and to demonstrate how well the population of conceptual robot's capture real robots.

Some of the conceptual robot designs in Figure 4-21 demonstrate a positive correlation between the battery capacity, batteryUnitCapacity (A-hr), and the maximum robot velocity, maxCruiseVelocity (m/s). Other designs however do not achieve a greater velocity corresponding to more available power. This is due to the method in which the designs were generated rather than showing a tradeoff in performance.

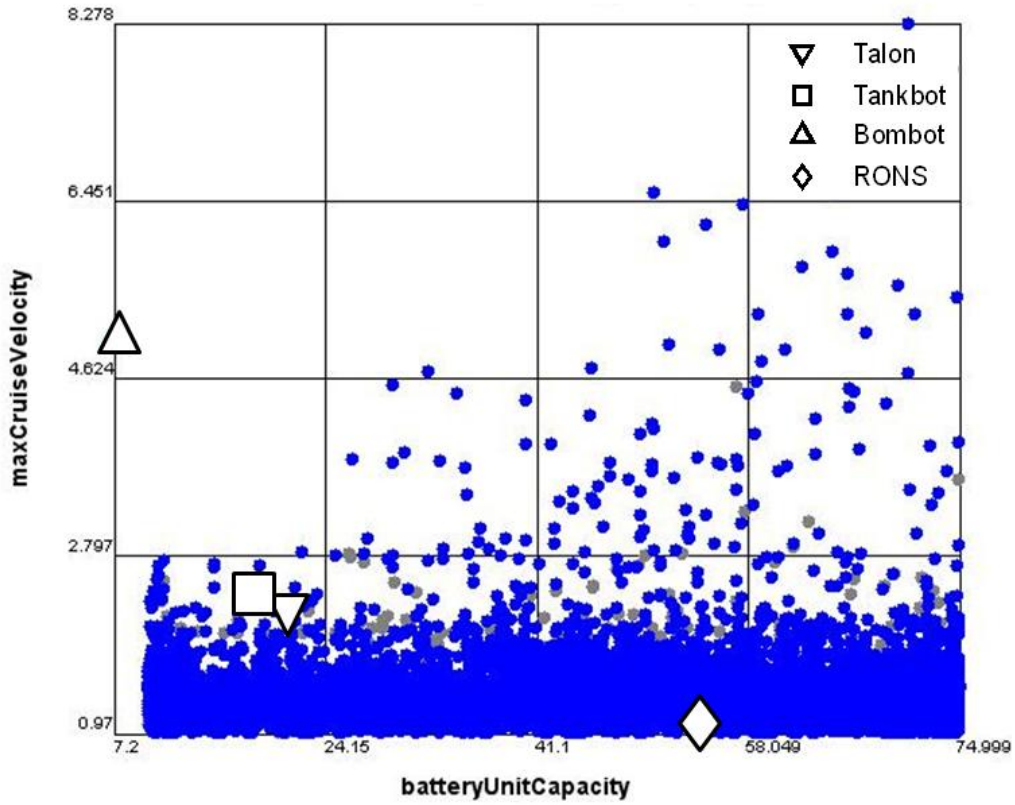

Figure 4-21. EOD Robot Performance Comparison with Simulink® Robot Results

Three of the physical EOD robot models fall within the design space of the conceptual models. The only design which does not is the Bombot. The Bombot cannot be captured within the current robot design space because conceptual robots were not populated with such small batteries. This can be corrected by sampling over smaller discrete values of battery unity capacities and sampling a greater number of designs within the mass ranges of the Bombot. The Bombot does not follow the positive correlation between battery capacity and maximum cruising speed. This can also attributed to the number of conceptual robots populated within the design space. A conceptual robot with very small battery capacity which operates at high speeds is a more difficult robot design and is not captured within the randomly sampled design space.

Figure 4-22 shows that among the conceptual robot designs that, as expected, there is a positive correlation between robot length, vehicleLength, and height of a step it can climb, heightStep. Each of the physical robot's climbing ability is less than numerous conceptual robot designs with the same robot length. These results also show conceptually that there are robots which can climb more effectively than the currently existing robot designs, without a necessary increase in length. While this comparison does not show any other performance tradeoffs, it shows that currently existing robots possess suboptimal climbing capabilities.

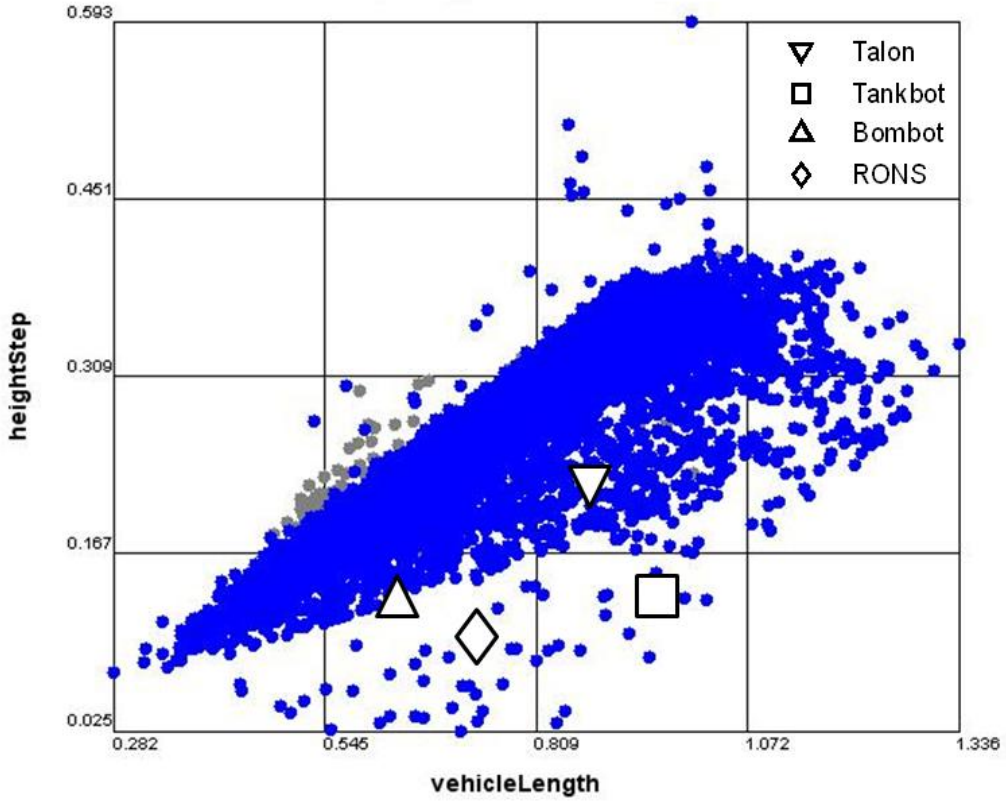

Figure 4-22. EOD Robot Performance Comparison with Simulink® Robot Results

Chapter 2 discusses the correlation between robot geometry and the required size of a hallway the robot can successfully maneuver through. Figure 4-23 demonstrates this positive

correlation between robot length and the width of the hallway, WHallway. All four physical robot designs fall within the design space of the conceptual robot designs. Some of these conceptual models require a smaller hallway while others require a wider hallway. This is because the true correlation between robot size and hallway width cannot be capture used a two dimensional plot of this nature. While influential, robot length is not a large contributing factor into the required hallway maneuverability width.

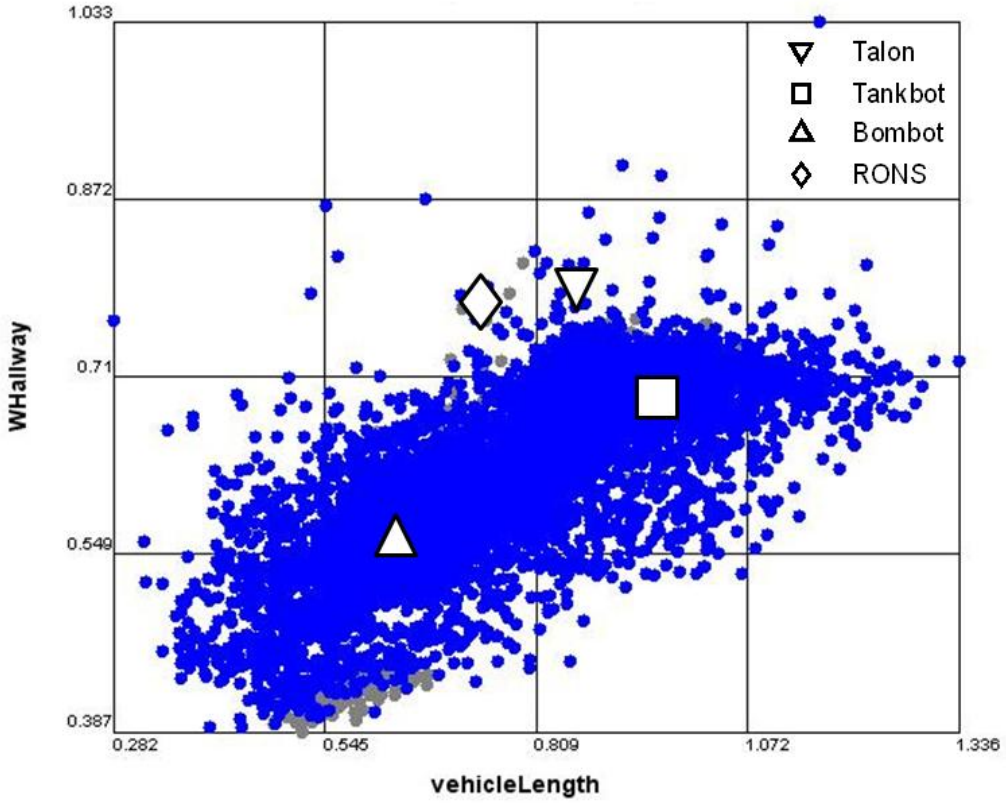

Figure 4-23. EOD Robot Performance Comparison with Simulink® Robot Results

There is an intuitive correlation between the capacity of the robot's battery and its endurance cruising distance, batteryDistance, supported by Figure 4-24. Both the Talon and

Tankbot fall within the middle of the design space where each out performs and underperforms conceptual designs of equivalent battery capacity.

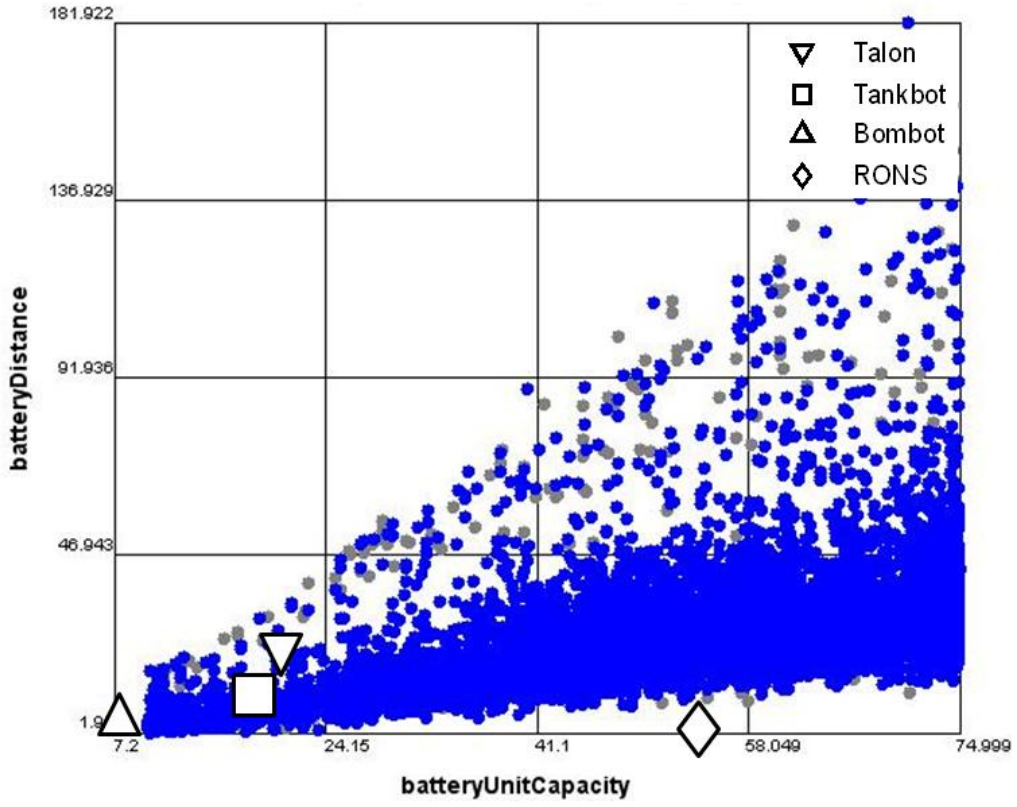

Figure 4-24. EOD Robot Performance Comparison with Simulink® Robot Results

The Bombot falls outside of this space because the model generated inputs to the model were not sampled at a low enough battery capacity. The Bombot does however follow the correlation between battery capacity and endurance distance. The RONS is not captured within the design space populated by conceptual robot models either. The conclusion which can be drawn is that the RONS is a suboptimal design and is not optimized for a prolonged operation time. This explanation is supported by both the age of the platform and the additional equipment on the platform.

4.9 Conclusion

Section 4.8 demonstrates that it is possible to generate representative models of currently fielded EOD robots using the Simulink® models. By varying key model inputs, these conceptual Simulink® models mirror physical robot designs both at the component level and as a system. More importantly these conceptual models' performance, Figure 4-19 and Figure 4-20, is very closely approximated.

Not all of these characteristics within the conceptual EOD robot models are approximated precisely. For example, neither the total robot length nor width is concisely approximated. One possible cause of this is due to propagation of modeling error from the modeling error of the components which make up the total robot system. A small error in the chassis modeling coupled with an error in the track width modeling can have a significant effect on the total robot width.

The other possible cause for these errors is due to the limitation of the model to capture the unique ways each robot is modeled. These model variations consist to both assumption made during the modeling process or by incorrect component packaging models. One such example is that the model assume that a portion of the treads overlap and cover a portion of the chassis. While this is true for the Talon it is not accurate for the Tankbot.

Error within the model also occurs due to the way measurements were taking to perform this comparison coupled with the method in which the model was driven. As stated earlier, much of the robot's powertrain is optimized to achieve the greatest endurance distance possible. The endurance speed is a user input and is used to calculate maximum platform speed. During robot testing the speeds chosen for the endurance testing were not chosen with regard to the maximum robot speed but were selected at a relative consistent pace of a human. The conceptual robot models were optimized to perform at peak efficiency at these speeds while the physical robots were not. This is therefore the primary cause for the error seen in Figure 4-18. Even with these

sources of errors and limitations in the flexibility of the conceptual robot models, the total system overall performance is approximated fairly closely.

Figure 4-21 through Figure 4-24 in Section 4.8 show that physical robots do fall within the conceptual robot design space. The instances when the physical robots do not however can be traced to either how the model is being driven or by the number of sampled robot designs. As state previously error in how the model is being driven occurs in cruising distance and maximum velocity predictions. An example of error due to limited sampling occurs with the comparison between cruising distance and battery capacity.

Increasing the number of conceptual designs would show the ability to capture physical robot's performance more conclusively. Many intuitive correlations between robot design and capacities are supported through exploration of the conceptual robot design trade space. Some of the robot designs populated within this trade space outperform their physical counterparts with the same physical characteristics. A more conclusive exploration of the design space would have to be performed to conclusively say that these conceptual models outperform currently existing models. This study validates the system-level modeling efforts by first showing the ability to recreate the physical robot's characteristics and performance. The system-level modeling effort is also validated by showing that physical robot designs fall within a design space populated by randomly sampled conceptual robots.

4.9 References

[1] "MATLAB." Internet: http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/f0-14059.html, 2010 [June 20, 2010] .

[2] "Using Simulink." Internet: http://www.mathworks.com/access/helpdesk_r13/help/toolbox/simulink/ug/preface2.html, 2010 [June 20, 2010].

[3] G.M. Stump, M.A. Yukish, J.D. Martin, and T.W. Simpson. "The ARL Trade Space Visualizer: An Engineering Decision-Making Tool," in the *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA-2004-4568.

[4] G.M. Stump, M. Yukish, T.W. Simpson, and L. Bennett. "Multidimensional Visualization and Its Application to a Design by Shopping Paradigm," in the *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, AIAA, AIAA-2002-5622.

[5] T.W. Simpson, D.B. Spencer, M.A. Yukish, and G. Stump. "Visual Steering Commands and Test Problems to Support Research in Trade Space Exploration," in the *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA, AIAA-2008-6085.

[6] R. Khire, J. Wang, T. Bailey, Y. Lin, and T.W. Simpson. "Product Family Commonality Selection Through Interactive Visualization," in the *Proceedings of the ASME International Design Engineering Technology Conference & Computer and Information in Engineering Conferences*, IDETC/CI 2008.

[7] A.M. Bobuck, 2010, Mechanical Engineering. University Park, The Pennsylvania State University. Master of Science.

Chapter 5

Optimal Hybrid Power Component Selection for Mobile Ground Robots

The purpose of this chapter is to develop methods optimize the component mix and power distribution of a robot with a hybrid power source when the robot's mission is known. The hybrid power source is assumed to be comprised of the following power sources: BB2590 Lithium-Ion batteries, ultracapacitors, and gasoline-powered generators. The ground robot being discussed weighs approximately 52kg with a power source weighing 5.6kg (the equivalent mass of 6 BB2590 batteries). Dynamic Programming (DP) is a primary numerical optimization technique used to yield an optimal control strategy for a known mission and predetermined hybrid system composition. A faster algorithm than DP, one which uses a current splitting control strategy, is also presented and used to compare a variety of hybrid system's ability to complete the same mission. Each system is constrained to have an equivalent mass while varying the percentage of the total power source mass taken up by each power source. Using the faster power-splitting algorithm implementation, the results of this work show that an optimal hybrid system for a robot to perform a monitoring mission differs from that of a traversal or climbing oriented mission. Thus, the mission characteristics can significantly change the "best" mix of power sources to use on a robot.

5.1 Introduction

Research into the automotive powertrains show an increase in fuel economy and extension of driving range through use of a hybrid powertrain [1]. This is done often through the principle of load sharing between an internal combustion engine and electric motor [2]. Hybrid technologies are being applied to power sources ranging from man-portable devices to powering homes in an effort to increase the effective energy density, reduce costs or extend the life expectancy of the power source [3]-[5]. These benefits can only be reached through proper selection and control of the hybrid devices being used.

There is great interest in applying the same hybridization principles and techniques used in the hybrid automotive industry to power mobile ground robots. Ultracapacitors have as much

as 10 times the power density of batteries, yet have significantly lower energy densities due to the form of energy storage [6]-[8]. The rate at which power is drawn from a battery affects its capacity [9]. A battery's effective capacity will decrease when power is drawn above its c-rating. The battery's effective capacity will be greater than its rating when power is drawn below it rating. The battery's effective capacity changes because of the inefficiencies associated with the electro-chemical reactions which occur when power is drawn from or supplied to a battery.

Ultracapacitors don't suffer the same losses as batteries because energy is stored as an electrical field created by charge separation, rather than by ion transfer as is the case in batteries [7]. Diesel fuel has a power density of 46 MJ/kg, a specific energy value which is nearly 64 times higher than the leading lithium ion battery at 0.72 MJ/kg [10]. But even though there are clearly advantageous uses of more power- and energy-dense sources, mobile ground robots with weights ranging from approximately 9.6 to 308kg almost exclusively use batteries as their power source.

The benefits from these types of energy sources can be leveraged through control strategies and the hybrid systems composition. Hybrid system composition refers to the size of each hybrid component but also the manner in which each power source is connected. Typically a hybrid system consists of a battery, internal combustion (IC) engine and electric motor. Based on the arrangement of these components the hybrid system is said to be a parallel or series configuration [2]. A parallel hybrid splits the drivetrain torque between two drive systems, typically a direct current motor and internal combustion engine. A series hybrid typically uses an internal combustion engine coupled with a alternator to charge a battery which is used to power a direct current motor while powers the drivetrain. The governing benefit of a hybrid automotive system is to reduce fuel consumption of IC engines. The advantages of hybrid powertrains are determined by the efficiency of the components used and the algorithms used to control the system. Hybrid systems managed by using intelligent controllers have the potential to increase fuel efficiency and lower emission [2], [11]. Increased fuel efficiency can be achieved through control of not only the hybrid components such as IC engines and electric motors but also drivetrain components including gear selection and gearbox switching rates for the automobiles transmission [12]. The allocation of each power source varies based on the relative importance of

conservation of each fuel type [13]. A controller which seeks to minimize fuel for the IC engines operates differently than the system which minimizes the use of the electric motor. Intelligent controllers have been shown to optimize fuel efficiency and decrease the emissions for non-hybrid systems with IC engines [12].

Advanced algorithms such as Dynamic Programming have been used optimize a system's performance to the extent that smaller powertrain components can achieve equivalent performance to a larger system in a hybrid system for a given drive cycle [14]. These numerical optimization techniques can be used to show how the optimal control strategy, when each power source is turned on and off during a given drive cycle, varies based on the size of each hybrid component [15]. For the same drive cycle and vehicle, the optimal time to discharge and charge the battery will vary based on the size of the engine or battery in the system. These same numerical optimization techniques can be used to determine if a hybrid configuration (both types and sizes of components) is even feasible for a given vehicle completing a predetermined drive cycle [16]. If a battery and/or IC engine is undersized, the vehicle may not be able to achieve a typical drive cycle.

There are two types of techniques which optimize the control of hybrid systems. Controllers which can be used in real time, during the operation of a hybrid system, are traditionally rule-based [17]. As an example of a rule-based system: a charge-sustaining rule-based controller constantly discharges and charges the vehicle's battery in a manner so that, at the end of any drive cycle, the battery is not significantly drained relative to the start of the drive cycle [13], [18]. Model based predictive control has been used to optimize the control of hybrid electric vehicles through the use of telematics to communicate future driving conditions [19]. Other rule-based controllers for parallel hybrid vehicles seek to increase the average fuel efficiency by first draining the battery by entirely driving with the electric motor, and then using the IC engine of the vehicle if the drive cycle is not completed [18], [20]. Many rule-based controllers use human-designed heuristics [21] and often incorporate fuzzy logic [22]. To improve these controllers, some research has been performed to extend rule based controllers to pattern-learning fuzzy logic [23], [24].

In contrast to real-time controllers, offline optimization techniques are computationally inefficient and cannot be used for real-time control. Instead, these are algorithms used to optimize a control strategy for a given hybrid vehicle with a known drive cycle or trajectory, either before or after a driving cycle has occurred or will occur. These optimization techniques include linear programming [25], optimal control [26] and Dynamic Programming [27], [28].

Dynamic Programming (DP) has a number of uses even though it cannot be implemented for real time control. DP in the context of this work yields a globally optimal solution to the control of each component of a hybrid system. This solution depends on the hybrid system composition, constraints placed on the controller, and a user-specified relative importance of each power source. Dynamic Programming yields the globally optimal solution to a problem and can therefore be used as a method to evaluate the feasibility of a hybrid system for a given drive cycle [13]. Optimal controllers coupled with powertrain modeling can determine the feasibility of novel hybrid systems, thus lowering production cycle time and decreasing design costs [14].

Global optimization using Dynamic Programming offers two benefits for existing hybrid systems. First, the global optimal solution can be used as a benchmark for sub-optimal real-time controllers to determine their relative performance, e.g. the disadvantage of not having future power inputs known in advance when calculating present power mixes [4], [19], [20]. Second, the control methods and trajectories seen from the optimal solution can yield insight into developing new rule-based control strategies which can be implemented in real-time [4], [19].

A number of sub-optimal control strategies have been developed using the results from DP. Stochastic Dynamic Programming (SDP) is a real time implementation of DP where the control predicts the optimal control strategy on an unknown driving cycle using statistics and the results from a number of driving cycles[16], [17]. A database of optimal control strategies is computed using DP. SDP uses the statistical properties of these results and in real time predicts what the next energy demand might be. This prediction uses a probability distribution of next possible energy states using a database generated from the DP results. Model predictive control uses DP and knowledge of the current hybrid system states to formulate a control strategy using finite horizon optimization of the system [11]. Quadratic optimization uses a cost function similar to the cost function used in Dynamic Programming to determine the optimal trajectory

and power allocation from various power sources. Quadratic optimization formulates the optimal trajectory using knowledge of the past and current states of the hybrid vehicle, unlike DP which optimizes with the knowledge about the entire drive cycle [11].

While there are a number of suboptimal controller which can be developed using DP, the main focus of this work is the development and implementation of the DP algorithm. Brahma, et al. have developed a supervisory control system using Dynamic Programming for a series hybrid system [13]. Energy from the generator was partitioned either to or from the battery depending upon the drive cycle. The optimal trajectory varied for the charge sustaining algorithm based upon the relative importance placed on the amount of stored energy within the battery.

Powell, et al. developed a dynamic system model for both traditional combustion engines and hybrid parallel systems [30]. Rizzoni, et al. developed a scalable system view for parallel hybrid systems using scalable system components for the power generation and drivetrain components [31]. These models were then validated through comparison with existing physical hybrid electric vehicle (HEV) models during varied drive cycles.

Using a power flow system rather than a dynamic system, Koot, et al. developed three different controls for parallel hybrid vehicles [11]. The controls developed using Dynamic Programming, Quadratic Programming and Model Predictive Control were all formally developed and evaluated in this work. In addition to the DP formulation was a number of techniques to reduce the computational cost of running the algorithm. The QP based controller transformed the nonlinear dynamics of the hybrid system into quadratic approximations which could be used within a real time controller. Using MPC, which provides an optimal system input for a given time step relevant a limited prediction horizon, the authors showed a quantifiable improvement in fuel efficiency over the controllers currently fielded in hybrid vehicles.

O'Keefe and Markel used Dynamic Programming to examine the relative performance of parallel-hybrid systems with varied component sizes [14]. Their research shows that the optimal controller developed using DP yields equivalent fuel economy for hybrid systems with smaller components. The control strategy changed with the varied sizes of hybrid component, but the

overall performance and fuel economy did not. In essence, they showed that smaller components working more efficiently can reduce production system costs for manufactures.

A DP controller and a real-time controller based on the Principle of Pontryagin were presented by Sinoquet, et al. for a parallel hybrid vehicle [31]. One focus of this work was to approximate the fuel losses associated with turning the engine on and off. The optimal control strategy here is shown to rely heavily on the component size, drive cycle and the relative importance of consuming fuel or electricity.

Kimura, et al. developed a controller using DP for a series-parallel hybrid vehicle [1]. A series-parallel hybrid vehicle operates as a parallel, series or mix of the two through utilization of a clutch located on the output of the vehicle's electric motor. In addition to formalizing the modeling of a hybrid system, Kimura focuses on the impact of battery state of charge on available torque from the electric motor. One conclusion of this work is that the change in battery state of charge needs to be taken into consideration to develop an optimal controller of the IC engine. The controller developed in this work balances torque from varied power sources to meet road load requirements.

Rizoulis, et al. has focused on developing a controller which varies the hybrid vehicle from series to parallel mode [2]. This controller is based on driver inputs and current vehicle states. Optimizing this controller allows the hybrid vehicle to take advantage of the series hybrid configuration for city driving and the parallel configuration for highway driving.

The fuel economy advantage of developing a controller using DP which depletes the charge of a hybrid vehicle's state of charge slowly over the duration of drive cycle was investigated by Sharer, et al. [9]. The results show that while a charge depleting strategy is good when the cycle is known, any deviation from the cycle length leads to a suboptimal control strategy to others. Overall the charge-depleting controller out performs currently fielded hybrid vehicle charge sustaining controllers.

Vahidi's work with Huei Peng, et al. used dynamic programming to manage a hybrid system composed of a fuel cell and ultracapacitors which supply power to an electric motor [16]. Using an ultracapacitor to absorb large current draws, the fuel cell in the vehicle could be

significantly smaller and only need to deliver the bulk of the constant low power needs. Vahidi focused on developing system models for a fuel cell system, ultracapacitor and DC/DC converter load leveling system. The controller formulation was based off of a frequency analysis as it pertained to the control of the ultracapacitor. Min-Joong Kim also working with Huei Peng focused on the implications of component sizing for a fuel cell hybrid vehicle [33]. Kim focused on both the component design but also the controller as system variables which can be varied for system optimization.

Bin Wu along with Huei Peng, et al. developed an optimal control strategy using Dynamic Programming for a hydraulic hybrid delivery truck [34]. The controller optimized on gear selection of the transmission and power selection between the diesel engine and the hydraulic pumps. Simulations using the DP controller show up to a 48% increase in the fuel economy of this truck.

In addition to providing a globally optimal solution, dynamic programming can be used to provide insight into the development of real-time controllers. Gong, et al. have develop a DP based charge-depleting power management algorithm to optimize fuel economy [27]. First a driving profile is collected for a given path and optimized using Dynamic Programming. Then the driver repeats the path and a probabilistic model is generated for each point in time the likelihood of a given power requirement. The DP based control optimizes based on these statistics. Gong's research has shown significant improvement in fuel economy over current rule based controllers within the simulation environment.

Opila, et al. shows a 2-3% performance improvement in a series-parallel hybrid vehicle using shortest path stochastic dynamic programming over current industry controllers [36]. In addition to showing the possible real time performance benefits of using a DP based controller, Opila discusses the tradeoff between performance and drivability. By limiting the number of gear events, the number of transmission shifts in a given discrete amount of time, performance can be lowered while increasing the perceived comfort or drivability of the drive cycle.

Lin, Peng and Grizzle developed a state dependent infinite horizon stochastic dynamic programming using Markov chain modeling [37]. Markov chains are random processes where

past states are not factors in the decision being made; only the current state and future uncertainty are considered. This time invariant, state-dependent controller optimizes engine and battery operations. In this study the results of this stochastic dynamic programming controller yielded better results than sub-optimal rule based controllers tuned using DP.

Tate, Grizzle and Peng discuss the tradeoffs between the implementations of the previously mentioned infinite-horizon stochastic dynamic programming and shortest path stochastic dynamic programming [41]. While both can be implemented in real time and are time-invariant, the benefit of the shortest path formulation is fewer tunable parameters to implement on a given system. Based on the results of this study the shortest-path method yields a more optimal controller with fewer tunable parameters.

The use of Dynamic Programming as a means to optimize hybrid system control within the automotive industry has been performed for over a decade and is becoming something of a mature technology. While there has been research into topics such as path optimization for mobile robots, little research has been performed to date to numerically optimize control of hybrid systems on mobile ground robot platforms [38], [39]. Due to the mass of most mobile robots relative to automobiles, the tradeoffs between power sources and small efficiency gains on vehicles have the potential to become much more significant for robots with power demands that are much more variable. The optimization of a hybrid robot weighing 10kg can consist of controlling power from two sources of power such as an ultracapacitor and a battery. The automotive equivalent of this system could be splitting torque between two different types of motors to propel a 1000kg vehicle.

There is a relatively large discrepancy between the drive cycles which a mobile ground off-road robot would experience and those demanded of hybrid automobiles. The mobile robot is almost exclusively driving over uneven terrain and often climbing over obstacles such as stairs. This is a departure from optimizing around city vs. highway driving patterns. Much of what is currently being studied in the automotive community includes examining drivability/performance tradeoffs. One example of drivability is the number of gear events which affect the comfort of the drive. Conditions such as comfort and gear changes do not affect the control of the vast majority of mobile robots while traversing uneven terrain.

This work seeks to initially determine if a single source of power (batteries, ultracapacitors, or generators) is the optimal power system for a mobile robot weighing 44kg. The results will show that a hybrid system yields better performance than a single power source, and so following research seeks to determine the optimal hybrid power source composition given a fixed power source mass and predetermined mission scenario.

The remainder of this chapter explores these issues in more detail. Section 5.2 gives detail on the robot platform used to gather data. It also explains the hybrid system and the models which will be used to predict each component's performance. Section 5.3 discusses the formulation of the Dynamic Programming controller and methods introduced to increase the computational efficiency of the algorithm. Section 5.4 introduces a rule-based controller that can be implemented in real time. Section 5.5 discusses the power profiles and hybrid systems which are to be used to compare the performance of each controller. Section 5.6 compares the relative performance of each control strategy. Section 5.7 discusses three scenarios which the robot's powertrain will be optimized around. With the numerical controller validated against experimental data, numerical simulations are run in Section 5.8 to determine the optimal hybrid composition for each mission. The conclusions drawn from this work are covered in Section 5.9.

5.2 Hybrid System Decomposition

5.2.1 Robot Platform

The ground robot being used to collect data for each mission in this work is the Talon seen in Figure 5-1. The Talon is a skid-steer tracked ground robot which weighs 44 kg without the battery/power source. All power monitoring done to collect the mission profile occurred between the robot's battery and the power bus using the sensor shown in Figure 5-2. The power monitoring system is a custom-built unit which samples power at 1kHz and records power data onto a micro SD card during each power test. The average power draw for each second was averaged from the data collected over a 1 second interval. All of the modeling presented in this chapter is based on controlling a given number of power (watts) over a 1 second interval. When power is said to be sourced, it is over a 1 second interval. For the purposes of this work, both energy and power allocated in 1 second intervals are used interchangeably. All control strategies

discussed here are energy allocation strategies, and similar nomenclature is used across all controllers to avoid confusion.

Figure 5-1: Talon Ground Robot Platform

Figure 5-2. Talon Power Monitoring System

5.2.2 Power Source System Architecture

The robot's hybrid power system has a total mass of 5.6kg. While the mass of the power source is fixed, the composition or percentage of the mass each component varies anywhere from zero mass to the total mass of the power source. Each power source is connected to the robot's bus through a DC/DC boost buck converter which regulates the current from each source. Each

of these converters are custom built for this project and have corresponding efficiency losses and power draws which is characterized in the next section of this work.

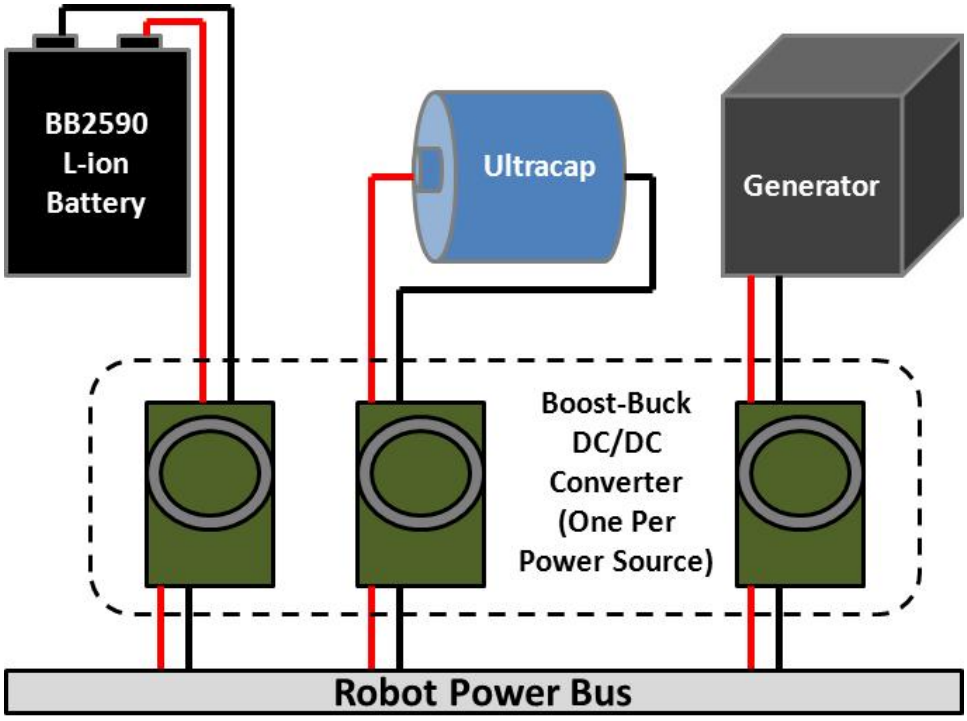

Figure 5-3. Power System Architecture

5.2.4 Power Source System Component Performance Models

Each power source in this section has a corresponding peak and total energy density which scales as a function of the mass of the power source shown in Table 5-1. In addition to the energy densities of these power systems, the corresponding losses associated with overdrawing power from each is also calculated. Each power source (when included in the model) passes through a DC/DC converter which has an efficiency loss which is characterized through experimental data collected on the physical prototype. The peak and total energy available for each power source is shown in Table 5-1 to be a function of the mass of the battery, $m_{battery}$, mass of the ultracapacitor, $m_{Ultracapacitor}$, and mass of the generator, m_{gen}, in kilograms respectively.

Table 5-1: Energy Densities for Each Power Source

	Battery	Ultracapacitor	Generator
Peak Energy Output (joules)	$473.14m_{Battery}$	$10{,}028m_{Ultracap}$	$12.959m_{gen}^{1.38}$
Total Energy (joules)	$533{,}192m_{Battery}$	$19{,}900m_{Ultracap}$	$404{,}930m_{gen}^{1.38}$

5.2.4.1 BB2590 Lithium-Ion Battery

As stated in Chapter 3, the change in the State of Charge of a battery is not equal to the amount of energy from the energy to the system. The cause of this is related to the method of storing energy within a battery. Equation [3-9] and Equation [3-10] are the equations used to relate the change in state of charge of the battery to the amount of energy from the battery.

5.2.4.2 Ultracapacitor

The peak and total energy density of the ultracapacitors was determined through examination of the Maxwell Ultracapacitor specifications [8]. The total and peak energy density from an ultracapacitor is 19,900 joule/kg and 10,028 joule/kg respectively are taken from curve fits using vendor data. Unlike batteries which use chemical reactions to source energy, ultracapacitors have negligible losses when drawing large amounts of power. The model therefore assumes that there is no cost for drawing current between zero and the peak energy available. The requested energy from the ultracapacitor is therefore equal to its change in state of charge with no efficiency losses.

5.2.4.3 Generator

The generator model used is derived from vendor data. A generator of a given mass produces a constant power supply during operation. There are no electric losses associated with the operation of the generator. The peak (constant) energy provided by the generator while running is a function of its mass and is equal to $12.959 \bullet M_{gen}^{1.38}$. For a given total mass of the generator (including fuel) its available total energy $404{,}930 \bullet M_{gen}^{1.38}$.

5.2.4.4 DC/DC Boost-Buck Converter

Each power source provides energy to its individual and identical DC/DC boost-buck converter. The purpose of each is to regulate the voltage level of each source to robot power bus. Through experimentation, the efficiencies of the converters were characterized with the data points in Figure 5-4 by measuring the power into and out of the converter. In Figure 5-4 the fit curve used with the numerical models is shown as the solid line.

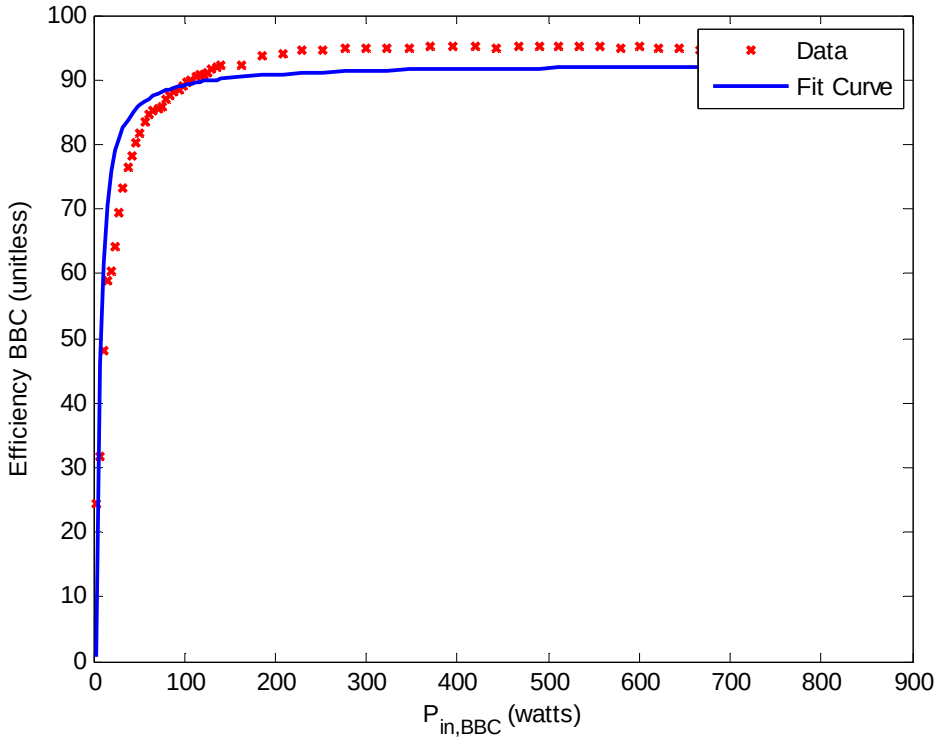

Figure 5-4. DC/DC converter efficiency as a function of the input power

Using the experimental data shown in Figure 5-4, the energy into the DC/DC converter, $E_{in,BBC}$, can be related to the energy out of the DC/DC converter, $E_{out,BBC}$, represented in Equation [5-1] where the coefficients $b_{DC/DC}$ and $m_{DC/DC}$ equal 0.927 and 3.278 respectively.

$$E_{out,BBC} = b_{DC/DC} E_{in,BBC} - m_{DC/DC} \quad \textbf{[5-1]}$$

5.3 Dynamic Programming Algorithm

5.3.1 Background

Dynamic Programming is a numerical optimization technique formalized by Richard Bellman in the 1950s [42], [43] Dynamic programming is a method of breaking a large problem into smaller overlapping subproblems. If a larger problem is broken into 5 smaller subproblems of equal size and complexity, the answer to subproblem 2 can partially be solved without knowledge of subproblem 1. The answer to the entire problem however requires answers to each subproblem and the transition between subproblems [44]. Dynamic programming, unlike many brute force algorithms, solves these subproblems only once and stores the answers in a table, lowering computation time [45].

Traditionally Dynamic Programming has been used in the fields of mathematics, economics and controls, but the method can be applied to any form of numerical optimization where the entire process or profile being optimized is known in advance. Dynamic Programming yields an optimal solution to any convex, well-posed problem because it is formulated from the goal or end state, and works backwards in time to the initial state. While the results of Dynamic Programming can produce the optimal control strategy or solution to a problem, because it is non-causal it cannot be used as a real-time controller. It can, however, be used to compare the relative performance of real-time control strategies and be used to develop suboptimal controllers which produce similar results based on Dynamic Programming.

5.3.2 Method

Dynamic Programming is based around the principle of finding the optimal path or trajectory through a problem. Each step in this path is considered a subproblem. For a given problem, one first defines the final state (goal) and the initial conditions (start) of the overall problem. Next, one defines the number of subproblems or steps into which the problem is going

to be broken. Each step has a chosen number of discrete states which makes up the total number of paths for each step of the problem. All steps have an equal number of states to and from each step. An exception occurs under certain circumstance with the initial and final conditions. Each path has a corresponding numerical value associated with it which is described as its cost. For a given step, there is a cost to transition to and from each path in addition to a cost to be at this given path in some cases. The optimal trajectory therefore is the path chosen from step to step which meets a predetermined criterion. The predetermined numerical criterion used to describe each path is referred to as the cost function. The optimal trajectory typically minimizes or maximizes this cost function based on the problem being solved.

A very simplified example of this method is shown in Figure 5-6. The start and goal are shown on the left and right respectively, and for this example there is only one possible solution to start and end with. While time is defined as left to right in this problem, the problem is solved right to left. There is only one step in this problem and a total of four possible paths. Working backwards from the goal, each path has an associated cost (10, 30, 15, and 19) and a cost to remain at this state (1, 2, 3, and 4). To transition from the step to the start there is also a corresponding cost (18, 2, 15, and 3). The optimal path is to minimize the cost to travel from the goal to the start. The cost in this example is the summation of the costs to transition to and from each state for the single step. The total cost for each path moving from top to bottom is as follows: 29, 34, 33, and 28. The optimal path is therefore the fourth path which is designated as a thicker dotted line in Figure 5-6. The optimal path as defined here is not the intermediate step with the lowest individual costs, but rather the lowest overall cost function.

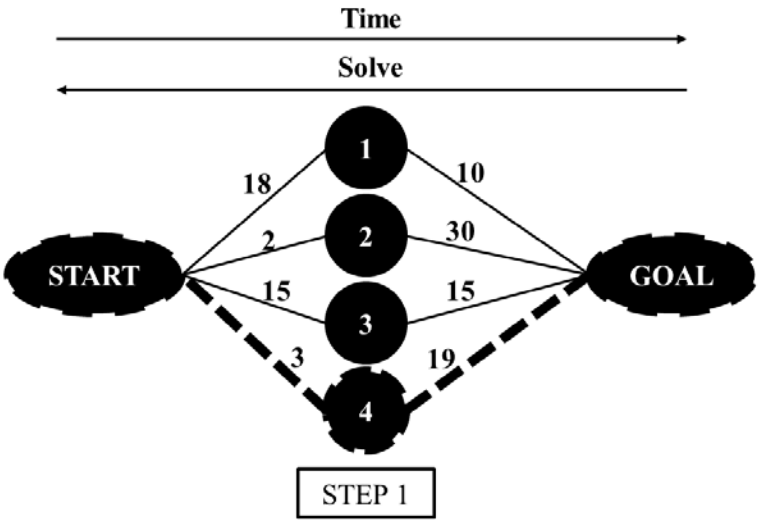

Figure 5-5. Simplified Dynamic Programming Trajectory Example

The simplified example is missing several key components which significantly increase the complexity of the problem. There can be multiple start and goal states each of which can have a corresponding cost function just as a given step would. Problems which require an algorithm as complex as Dynamic Programming has numerous steps which vary in number depending upon the difficulty of the overall problem. There may be orders of magnitude more states per step than the example shown here. Figure 5-7 shows how increasing the number of steps from one to three increases the complexity of the Dynamic Programming example.

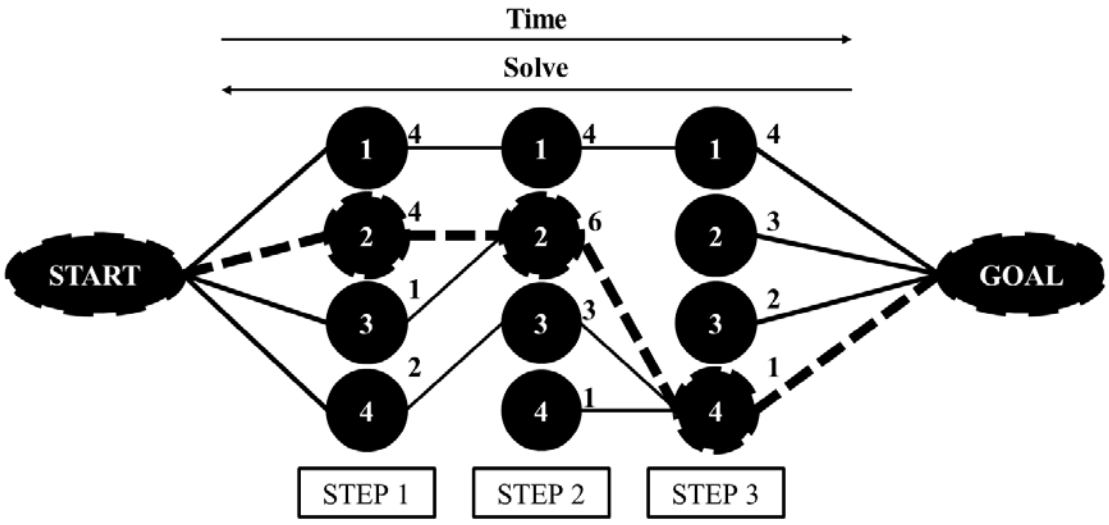

Figure 5-6. Simplified Dynamic Programming Trajectory Example with Multiple Steps

Just as with the first example, the problem is broken into subproblems and solved from the goal backwards in time to the start. The key difference in examples occurs not at the start or goal but rather between Steps 1 through 3. Moving backwards in time from Step 3 to Step 2, a cost function is calculated to transition from each state in Step 3 to each possible state of Step 2. Similar to the last example, there is a cost associated not with occupying a state for each step but rather for a transition. Each state is labeled 1 through 4 to designate its place. The numerical values on the lines leading from each state is the cost associated with moving that state from a previous step.

The path with the lowest cost function from State 1 in Step 2 is to State 1 in Step 3 and designated using a solid line. Only the transition from Step 2 to Step 3 with the lowest cost function is stored. The same procedure is performed between Step 1 and Step 2. Performing this operation produces a set of paths from start to goal equal to the total number of states used to solve a subproblem. In this example the optimal path (designated with a thick dotted line) is to transition from the start to State 2 in Step 1 to State 2 in Step 2 to State 4 in Step 3 which leads to the goal. For each state in Step 1 there is a varied path. Based on the cost function, multiple states in one step can lead to the same state in another step. There is always, however, only one optimal path to lead from one step to another for each state. Because Dynamic Programming

achieves the optimal path by only storing information from one step ahead, it is much more computationally efficient than other algorithms [45].

5.3.3 Dynamic Programming Implementation

5.3.3.1 Battery System Dynamic Programming Algorithm

The first system to test a Dynamic Programming based controller on is a battery only system. Dynamic Programming is used to determine the optimal allocation of energy to meet a predetermined robot power demand profile. The example from the previous section can be extended to describe how Dynamic Programming can be used to control the power from a single BB2590 lithium ion battery which is used to power a ground mobile robot over a given terrain [38]. To begin, we assume that a power profile is measured for a robot which describes the necessary power required to complete a mission. Dynamic Programming is used to determine the optimal change in the state of charge of the battery for each increment of time. Each discrete time increment is equivalent to a step from the previous example while each possible change in state of charge is equivalent to a possible path. The number of discrete time steps depends upon the data collection frequency and the required precision. The number of possible variations in the state of charge of the battery also varies based upon the required accuracy of the algorithm. The change in state of charge of the battery for each step ranges from empty or no charge to full charge.

Figure 5-8 is an example of one time step, between time n and n+1, where the power profile is requires 100 joules, $E_{Demand,n\rightarrow n+1}$, in 1 second. Focusing on one possible state of charge of the battery, $E_{SOC\_2,n}$, of 1000 joules, there are n_B states that the battery can transition to. In this example, n_B represents the number of possible discrete states of charge of the battery ranging from 0 to 1200 joules. For each possible path a cost function, J, is calculated to determine the relative cost to transition from state $E_{SOC\_2,n}$ to each possible state in time n+1. A battery which has a maximum charge of 1200 joules where each state in a given step varies by 1 joule has 1200 possible states. There are therefore 1200 cost functions which are calculated to transition from 1 state in time step n to each possible state in step n+1. There are a total of 1,440,000 cost functions which are calculated for this single step in time.

Though within the context of this problem it is not likely the battery will change its state of charge from empty to full within a single increment of time, each possible path is examined. The cost function will show that there are paths which can meet the demanded energy more effectively while limiting the battery's change in state of charge.

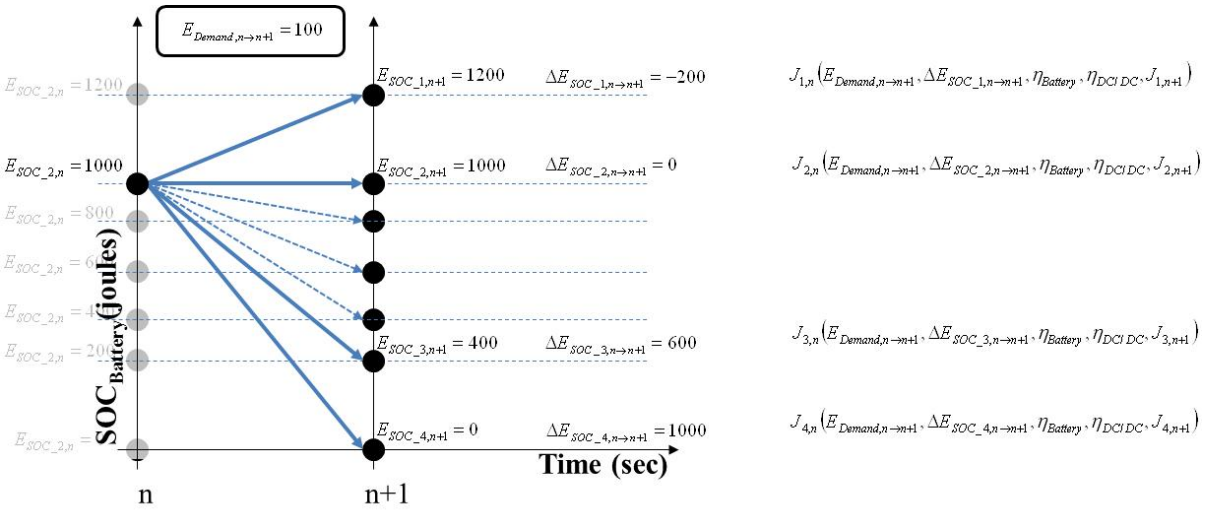

Figure 5-7. Single Power Source Path Matrix

The cost function in this example is primarily a function of the change in the state of charge of the battery, $\Delta E_{battery,SOC}$, and whether or not the power profile is being met. To minimize the cost function, the battery's change in state of charge should only be great enough to meet the demand of the power profile for a given step. Too much energy out of the battery wastes energy while too little energy does not meet the required energy demand.

The change in the battery's state of charge does not equal the amount of energy which reaches the bus from the battery. Chapter 3 covers the derivation relating the change in the battery's state of charge with the energy out of the battery. The relationship between energy into and out of the DC/DC converters is explained in Section 5.2.4. The change in the state of charge of the battery between time n and n-1 is described in Equation [5-2]. The losses experienced by overdrawing energy from the battery are expressed in Equation [5-3] and Equation [5-4]. The

relationship between the energy into and out of the DC/DC converter derived previously can be found in Equation [5-5].

$$\Delta E_{battery,SOC} = E_{battery,SOCn} - E_{battery,SOCn-1} \quad \textbf{[5-2]}$$

$$\Delta E_{out,battery} = k_B \cdot \Delta E_{battery,SOC}^{\frac{1}{N}} \quad \textbf{[5-3]}$$

$$k_B = V \cdot \Delta t \cdot \frac{C}{R} \cdot (R)^{\frac{1}{N}} \cdot (\Delta t \cdot E_{nom})^{\frac{-1}{N}} \quad \textbf{[5-4]}$$

$$\Delta E_{out,BBC\_battery} = b_{DC/DC} \Delta E_{out,battery} - m_{DC/DC} \quad \textbf{[5-5]}$$

The cumulative cost function for time n, J_n, is broken into three parts in Equation [5-6]. The first part describes how well the power demanded is being met by the battery. The second part describes the change in the state of charge of the battery. The final portion of the cumulative cost function is the cost to arrive at this step. The optimal path minimizes all three parts of this cost function. The DP optimization process therefore seeks to meet the demands of the profile while changing its state of charge just enough to accomplish this.

$$J_n = \left(E_{Demand} - \Delta E_{out,BBC\_battery}\right)^2 + \left(\Delta E_{battery,SOC}\right)^2 + J_{n+1} \quad \textbf{[5-6]}$$

The cumulative cost function is calculated in this manner for each step in time except for the final time step. For the final time step, or the first cost function solved, J_{n+1} will be replaced with a cost to finish the profile at this given state of charge.

5.3.3.2 Battery and Ultracapacitor System Dynamic Programming Algorithm

The second system to be examined is a hybrid power source composing of both a battery and ultracapacitor. Using two power sources to meet the demands of a power profile significantly increases the number of computations for a given segment in time. The most significant change to the algorithm is that each possible state of the battery and ultracapacitor must be examined to determine which the more efficient split of the power is.

Figure 5-8 shows how a similar path matrix to the previous system is altered to incorporate a second power source. Similar to the previous power system, each change in state of charge of the battery is examined for time step from n to n+1. There is however a second source of power which can also now change its state of charge and must therefore be accounted for. The

search matrix shown in Figure 5-8 is very similar to Figure 5-7 because there is no change in state of charge of the ultracapacitor. Between n and n+1 the charge of the ultracapacitor remains 400 joules. This search matrix needs to be repeated for each possible change in the state of charge of the ultracapacitor.

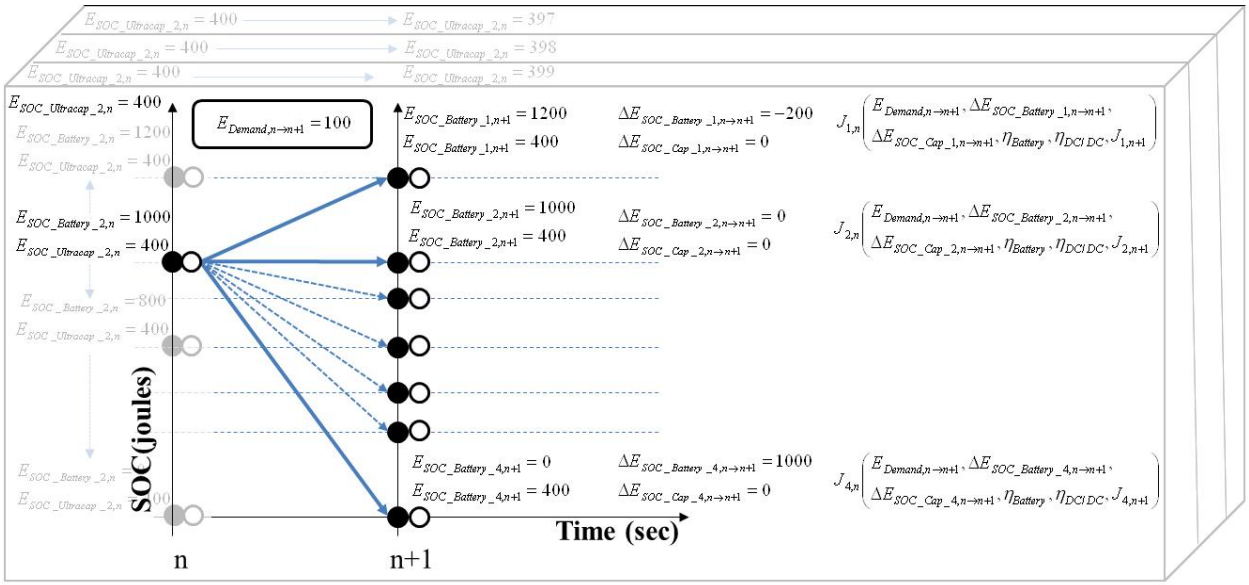

Figure 5-8. Hybrid Power Source Path Matrix

In Figure 5-8, if the ultracapacitor has a total possible charge of 400 joules and the battery has a total possible charge of 1200 joules, if the state of charge increases in 1 joule increments, there are a total of 230,400,000,000 cost functions calculated for a single increment of time. The total possible combination of the battery is 1,440,000 combinations which are multiplied by the total possible combinations of the ultracapacitor which is 160,000.

Similarly to the previous system, Equation [5-7] describes the change in state of charge of the ultracapacitor, $\Delta E_{cap,SOC}$, as simply the change in the state of charge, E_{capSOC}, from time n to n-1. While the ultracapacitor doesn't experience the same losses as the battery, the change in state of charge still passes through its own DC/DC converter. The losses of this converter are the same as that of the battery system and are expressed in Equation [5-8].

$$\Delta E_{cap,SOC} = E_{cap,SOCn} - E_{cap,SOCn-1} \quad \textbf{[5-7]}$$

$$\Delta E_{out,BBC\_cap} = b_{DC/DC}\Delta E_{cap,SOC} - m_{DC/DC} \quad \textbf{[5-8]}$$

For each change in the state of charge of the battery and the ultracapacitor, a cost function calculates the relative performance of each possible path. Equation [5-9] is broken down in to three groups which correspond to how well the energy demand is being met, the charge in each power source's state of charge, and the cumulative cost from the previous calculation at that step.

$$J_n = \left(E_{Demand} - \Delta E_{out,BBC\_battery} - \Delta E_{out,BBC\_cap}\right)^2 + \left(\Delta E_{battery,SOC}\right)^2 + \left(\Delta E_{cap,SOC}\right)^2 + J_{n+1} \quad \textbf{[5-9]}$$

Adding a second power source increases the number of computations required, but it does not drastically increase the level of complexity of the fundamental problem being solved. To add another power source which is treated similarly to the battery and the ultracapacitor in the system would require two changes. The first of which would be to change the cost function to include additional terms. The number of computations, $N_{Computations}$, performed per time step is a function of the number of states, λ_{States}, being searched through and the number of power, N_{PS}, sources being used shown in Equation [5-10].

$$N_{Computations} = \left[\lambda_{States}{}^{N_{PS}}\right]^2 \quad \textbf{[5-10]}$$

For this equation to be valid, the number of states has to be equal for each power source. For example a two power source system containing a 100 joules and 1200 joule system must have the same number of states being searched through. One solution is use 100 states per time step. The smaller power source has 1 joule increments while the larger power source has 12 joule increments between each state. The size of the discrete points between states varies depending upon the level of accuracy required in the solution.

For a single power source system with 1200 states per time step requires 1,440,000 computations per time step. A two power source system using 1200 states per step requires 2,073,600,000,000 computations per step.

5.3.4 Dynamic Programming Derivative Method Implementation

5.3.4.1 Derivative Method Battery Only

To increase the computational efficiency of the Dynamic Programming algorithm, the cost function can be manipulated to eliminate suboptimal paths for each step in time. As mentioned previously, the cost function is minimized when the change in the state of charge of the battery is such that the amount of energy exits through the DC/DC converter is equal to the demanded energy at each given point in time. The correlation between the change in state of charge of the battery and the demanded energy can be found in Equation [5-11].

$$\Delta E_{battSOC} = \left(\frac{E_{Demand} + m_{DC/DC}}{b_{DC/DC} \cdot k_B} \right)^N \quad \textbf{[5-11]}$$

Equation [5-11] accounts for the losses due to the inefficiencies from overdrawing current from the battery in addition to the losses from the DC/DC converter. This equation is derived by taking the derivative of the cost function from the first battery-only system, Equation [5-9], with respect to energy and setting it equal to zero. The optimal change in state of charge for each step in time changes as a function of the demanded energy and has a corresponding slope of zero at the optimum energy trajectory. Taking the derivative and setting it equal to zero reduces the number of computations from 746,496, in the case of the battery only system (battery capacity of 746,496 joules), to 1 calculated cost function for each step in time. The new cost function in Equation [5-12] is the algebraic relationship above and the cumulative cost function from the previous set.

$$J_n = \left(\Delta E_{batt\ SOC}\right)^2 + J_{n+1} \quad \textbf{[5-12]}$$

5.3.4.2 Derivative Method Battery and Ultracapacitor

The same principles used to reduce the number of computations in the battery-only system can be applied to significantly reduce the number of calculations in the two-source-power system. The first step is still to search though the possible changes in state of charge of one of the power sources, in this case the ultracapacitor because it has a smaller search matrix. The change in the state of charge is still the difference between time n and n+1 (Equation [5-13]).

$$\Delta E_{cap,SOC} = E_{cap,SOCn} - E_{cap,SOCn-1} \quad \textbf{[5-13]}$$

If the ultracapacitor's state of charge is increasing between n and n+1, the ultracapacitor is charging when the condition in Equation [5-14] is satisfied.

$$\Delta E_{cap,SOC} \geq 0 \quad \textbf{[5-14]}$$

Due to the losses passing energy though the DC/DC converter, to increase the state of charge in the ultracapacitor the change in state must be greater than the energy losses through the DC/DC converter. A very small increase in the energy going to charge the ultracapacitor will still result in a decrease of the state of charge of the ultracapacitor due to the nominal draw of the DC/DC converter. For the following equations, the notation for the ultracapacitor is that a positive change in state of charge means the ultracapacitor is charging. A negative value denotes a decrease in energy in the ultracapacitor.

Equation [5-15] describes the amount of energy which must pass from the robot's power bus and into the ultracapacitor's DC/DC converter to increase the ultracapacitor's state of charge. For example, to charge the ultracapacitor by 10 joules, approximately 14.35 joules must enter into the DC/DC converter to account for the converter's losses.

$$\Delta E_{out,BBC\_cap} = \frac{\Delta E_{cap,SOC} + m_{DC/DC}}{b_{DC/DC}} \quad \textbf{[5-15]}$$

When the condition in Equation [5-16] is satisfied, the ultracapacitor is being discharged. The amount of energy on the robot's power bus from the ultracapacitor will be less than the ultracapacitor's change in state of charge. Equation [5-17] describes the loss of energy due to the inefficiencies of DC/DC converter as the ultracapacitor is discharged.

$$\Delta E_{cap,SOC} < 0 \quad \textbf{[5-16]}$$

$$\Delta E_{out,BBC\_cap} = -\left(\Delta E_{cap,SOC} \cdot b_{DC/DC} - m_{DC/DC}\right) \quad \textbf{[5-17]}$$

Similar to the single source system, the optimal trajectory for a two-source system must meet the demand of the power profile. The energy out of the battery on the robot's power bus, $\Delta E_{out,BBC\_Batt}$, must therefore account for both the demand from the power profile and the change in the state of charge of the ultracapacitor during both charging and discharging. In Equation

[5-18] the energy demanded from the battery is the difference of the energy demanded from the power profile and the amount of energy either entering or leaving the ultracapacitor's DC/DC converter:

$$\Delta E_{out,BBC\_Batt} = E_{Demand} - \Delta E_{out,BBC\_cap} \quad \textbf{[5-18]}$$

Based upon the change in the state of charge of the ultracapacitor and the demand of the power profile, the battery in this two-power source system can either be charged or discharged. Similar to the ultracapacitor conditions, when Equation [5-19] is satisfied the battery potentially charges.

$$\Delta E_{out,BBC\_Batt} \geq 0 \quad \textbf{[5-19]}$$

The battery potentially charges due to the inherent energy draw of the DC/DC converter. A small amount of energy is passing from the robot power bus to the DC/DC converter and must be greater than the losses associated with the converter to charge the battery. Equation [5-20] describes the losses experienced while trying to charge the battery. If the energy between the battery and the DC/DC converter, $\Delta E_{BBC\text{-}Batt}$, is positive, the battery is charging. If the energy between the battery and DC/DC converter is negative, then the battery is discharging to supply the necessary energy.

$$\Delta E_{BBC-Batt} = \Delta E_{out,BBC\_Batt} \cdot b_{DC/DC} - m_{DC/DC} \quad \textbf{[5-20]}$$

Providing the condition in Equation [5-21] is satisfied, Equation [5-22] describes the additional losses due to the C-rating of the battery while charging the battery.

$$\Delta E_{BBC-Batt} \geq 0 \quad \textbf{[5-21]}$$

$$\Delta E_{batt,SOC} = k_B \cdot \Delta E_{BBC-Batt}^{\frac{1}{N}} \quad \textbf{[5-22]}$$

If the condition in Equation [5-23] is satisfied rather than Equation [5-21], then the battery must discharge to compensate nominal energy draw from the DC/DC converter.

$$\Delta E_{BBC-Batt} < 0 \quad \textbf{[5-23]}$$

Equation [5-24] describes the necessary decrease in the state of charge of the battery to meet the nominal draw of the DC/DC converter less the energy from the robot power bus from the ultracapacitor.

$$\Delta E_{batt,SOC} = -\left(\frac{\Delta E_{BBC-Batt}}{k_B}\right)^N \quad \textbf{[5-24]}$$

If the criterion in Equation [5-25] is met, the battery will discharge. Equation [5-26] and Equation [5-27] describe the relationship between the amount of energy required from the robot's power bus to meet the profile's demands and the corresponding decrease in state of charge of the battery.

$$\Delta E_{out,BBC\_Batt} < 0 \quad \textbf{[5-25]}$$

$$\Delta E_{BBC-Batt} = \frac{-\Delta E_{out,BBC\_Batt} + m_{DC/DC}}{b_{DC/DC}} \quad \textbf{[5-26]}$$

$$\Delta E_{batt,SOC} = -\left(\frac{\Delta E_{BBC-Batt}}{k_B}\right)^N \quad \textbf{[5-27]}$$

To determine the necessary change in the state of charge to offset the DC/DC converter's needs, the additional energy required due to losses experienced by the DC/DC converters must be computed (Equation [5-26]). For this calculation, the energy required between the battery and DC/DC converter, $\Delta E_{BBC\text{-}Batt}$, is assumed to be greater than the energy required on the bus. The next step is to account for the losses due to the C-rating of the battery (Equation [5-27]) as the battery is discharged. The decrease in the state of charge of the battery should be greater than the energy out of the battery before it enters the DC/DC converter.

Instead of searching through each possible combination of changes in states of charge for the battery and ultracapacitor the simplified version only searches through one power source and then meets the power demand profile with the second. The cost function used in this system therefore has fewer variables than the previous two power source system. The previous system evaluated possible combinations where the power demand profile was not met. Not meeting the power profile will not yield the optimal path, and therefore those options are eliminated to increase the computational efficiency of the algorithm. In the above analysis, the cost function is only evaluating how much energy was used to meet the demands of the profile (assuming it was met) and based on the cumulative cost to reach that state. The optimal path will discharge or charge each component just enough the meet the demands of the system. Charging or

discharging one power source by the other will be penalized using this new cost function in Equation [5-28].

$$J_n = \left(\Delta E_{batt\ SOC} + \Delta E_{cap\ SOC}\right)^2 + J_{n+1} \quad \textbf{[5-28]}$$

5.4 Rule Based Controller

As noted previously, Dynamic Programming cannot be used in its original form as a real-time controller. To overcome this severe constraint, a rule-based controller must be used to implement a control strategy in real time. A rule-based strategy is formulated in this thesis to use three power sources in a hierarchal order, using a decision tree shown in Figure 5-9. To explain this rule-based method, starting at the top left of Figure 5-9, the first step is to select a hybrid system and energy demand profile. Each hybrid system is composed of a generator, a Lithium-Ion battery, and an ultracapacitor. Each of these components can vary its mass from 0 kg to any mass. Based on the mass of each source, Table 5-1 shows the predicted peak and total energy for each source. Both characteristics are important when determining the portion of power which is drawn from each source at each given interval of time.

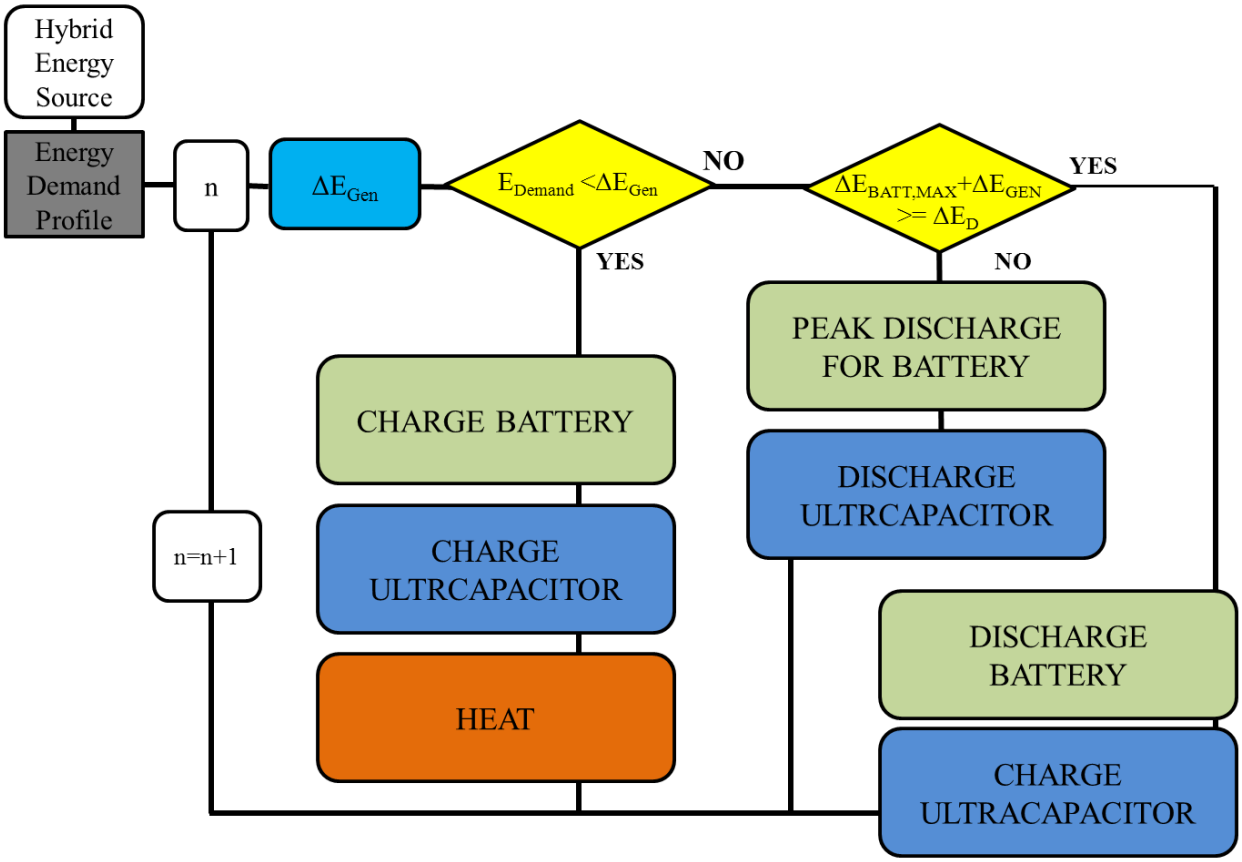

Figure 5-9. Rule Based Hybrid Controller Decision Tree

The rule-based controller allocates power from each source to the robot bus as time progresses, rather than starting with the final state and working backwards as with Dynamic Programming. The generator in this hybrid system is assumed to be always running and sources power to either meet the demands of the power profile, charge the other devices or producing heat.

The first step in the decision tree is to determine if the energy being sourced from the generator meets the demands of the power profile. If this power is sufficient, then the generator will use the remaining energy first to charge the battery and then the ultracapacitor, if needed. If any energy is left after both power sources are charged, then the remaining energy produces heat.

If the available energy from the generator is not sufficient, then there are two possible options for the hybrid power source. If the maximum possible power sourced from the generator

and battery is sufficient to meet the power demand profile, then the controller will discharge the battery appropriately to meet the demand of the profile. In addition to meeting the power demand profile, the battery will also discharge if the ultracapacitor is not fully charged.

If the generator and battery are together unable to meet the power demand profile, the next strategy is to discharge all three power sources to their maximum power draw, if needed to meet the demands of the profile. The basic principle of this rule-based controller is to discharge each power source as much as possible to meet the demanded power and when possible to immediately charge each power source when there is additional available energy from one higher power source to a lower power source. The hierarchy of power draw is to first tax the generator where possible, then the battery, and finally the ultracapacitor.

The generator is drawn from first because the energy source is constantly on. Regardless of the required energy, the generator, based on the controller, is always on and supplying energy. This energy cannot be stored and can only be used to meet the profile's demands or charge other energy sources. The battery is drawn from next because of it possesses a higher energy density than the ultracapacitor. It is a more efficiency process to drawn energy from the battery initially, than to constantly draw energy from the battery into the ultracapacitor and immediately out of the ultracapacitor to meet the demands of the profile.

There are several conditions which may occur during a given power profile which may cause the controller to be unable to meet the power demands of the profile. The first condition occurs when any of three power sources would need to supply more energy than it currently possesses to meet the demand of the profile. The manner in which the hierarchy is structure, this will only occur when there is insufficient energy between the three power sources to continue meeting the demands of the mission. The second condition is when the demanded power from the profile is greater than all three sources can instantaneously draw. Each source has peak limits which are a function of their respective masses. A hybrid system fails when the demand profile is greater than these limits.

All of the same efficiency losses and charging/discharging principles from the previous section are used within this hybrid control strategy. Each power source passes its energy through a DC/DC converter during both charging and discharging cycles.

5.5 Battery Only Controller Comparison

The relative performance of the Dynamic Programming and rule-based power sharing methods are compared with one another using a battery-only power system using a single power profile. First the system composition is discussed. Second the power profile is shown. The relative performance of the Dynamic Programming and rule-based power sharing methods are shown and compared to determine the performance of each.

5.5.1 Battery Only System

The first system examined is a battery-only composition. In this system, two BB2590 Lithium-ion batteries are connected in parallel to double the capacity of the battery. The only possible operation in this configuration is to discharge the battery as shown in Figure 5-10. While this is a trivial profile, it is useful to confirm that the algorithms indeed meet the demands imposed by the user-specified power profiles.

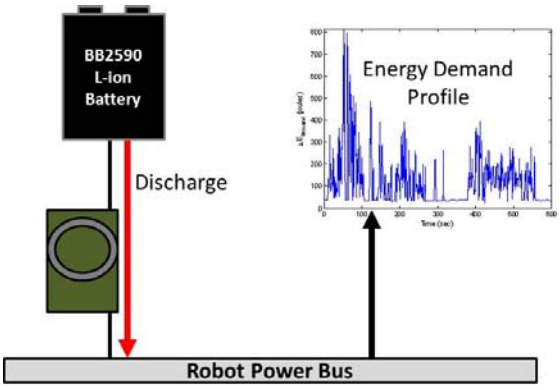

Figure 5-10. Battery Only System

5.5.2 Power Profile

The duration of this particular test profile is 10 minutes and is cycled 15 times, making the total run 150 minutes shown in Figure 5-11. The profile is repeatedly cycled to make discrepancies between each controller more evident. Each controller will be evaluated to determine which uses the least amount of energy while still meeting the demands of the power profile.

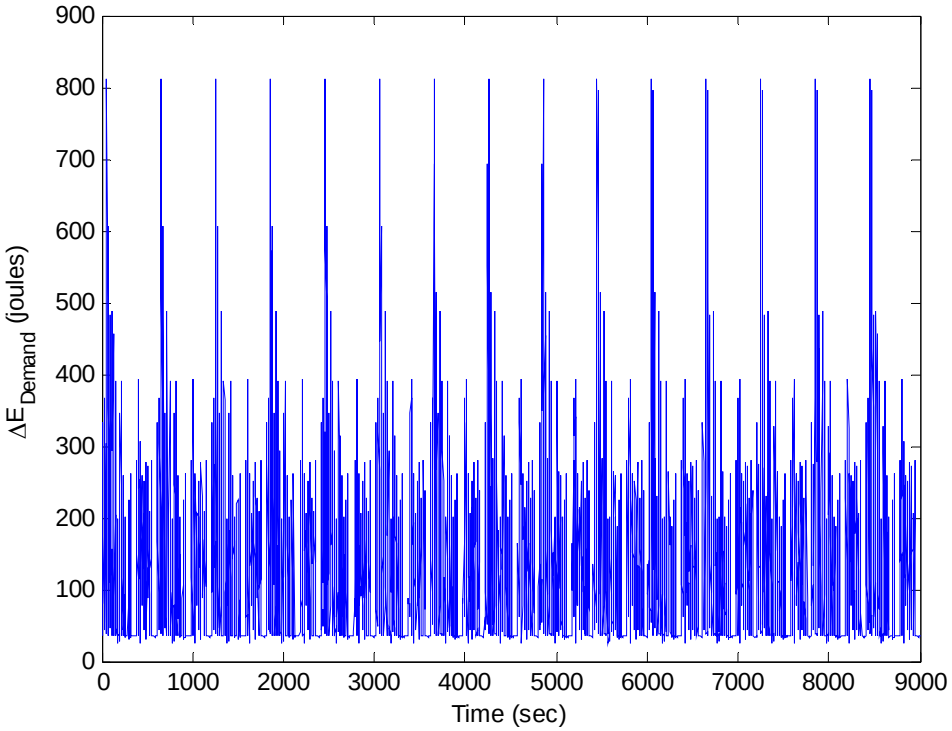

Figure 5-11. Battery Only System Power Profile

5.5.3 Battery-Only Dynamic Programming Results

The results of the Dynamic Programming controller are displayed in Figure 5-12 through Figure 5-15. The top graph in Figure 5-12 shows the energy demand profile while the bottom graph shows the energy which makes it to the robot's power bus from the battery to meet the robot's energy demand profile. These two graphs are identical because there is only one power

source meeting the robot profile power demands. The energy from the battery must be equal to the demanded energy.

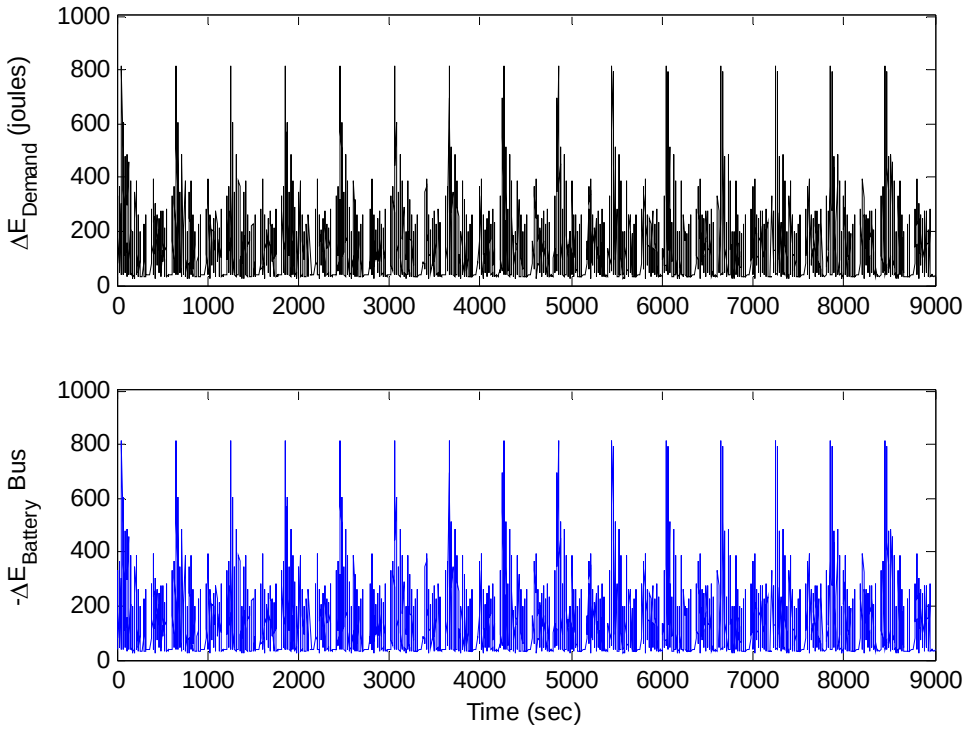

Figure 5-12. Dynamic Programming Results Power System Bus Energy: Battery System

The first graph in Figure 5-13 again shows the energy demand profile. The second graph in Figure 5-13 shows the change in the state of charge of the battery to meet this demanded profile. The change in the state of charge of the battery in Figure 5-13 is greater than the energy demand profile at each instance in time. The change in the battery's state of charge is also greater than the energy on the bus from the battery in Figure 5-12. Both of these relationships occur due to the losses experienced by the battery while overdrawing current and the losses associated with using the DC/DC converter. Figure 5-14 displays the cumulative energy required to meet the power demand profile and the corresponding state of charge of the battery to meet this demand.

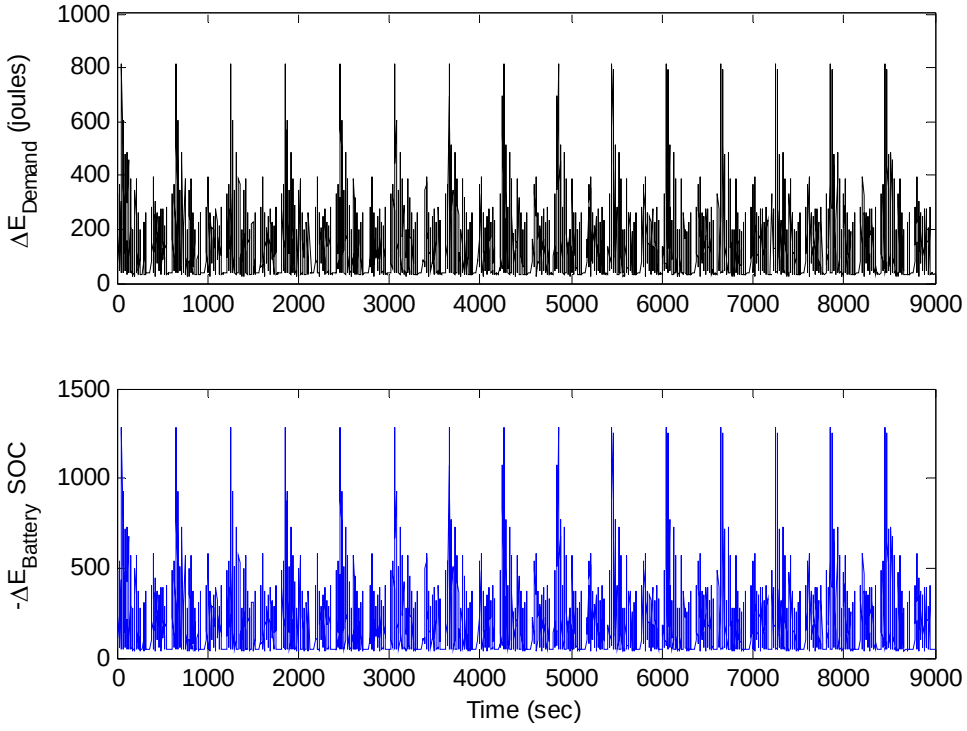

Figure 5-13. Dynamic Programming Results Power System Change in SOC: Battery System

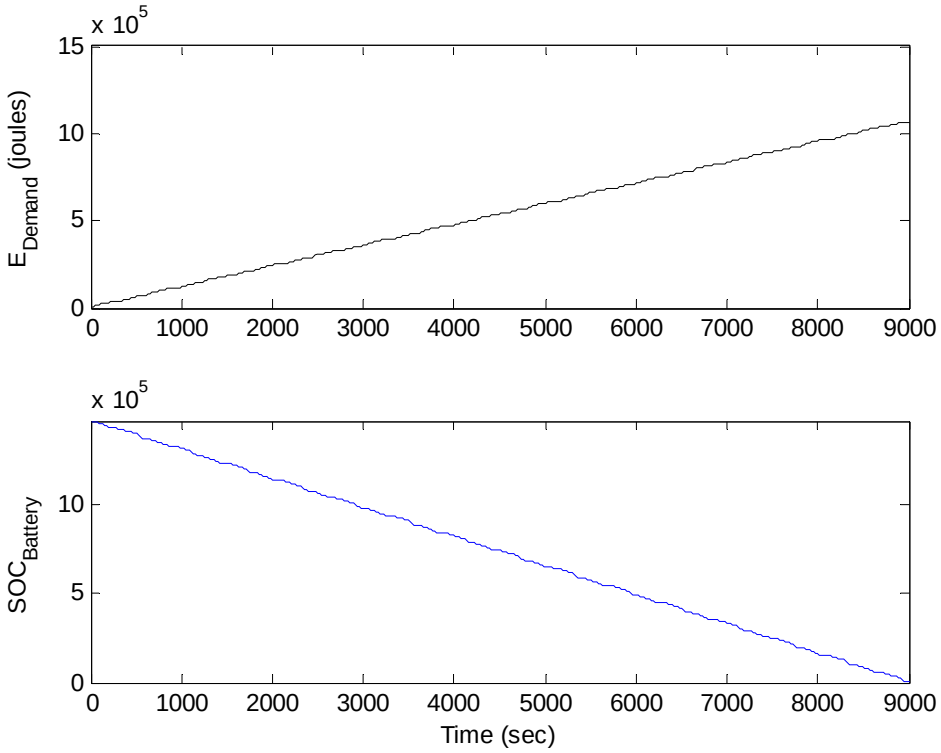

Figure 5-14. Dynamic Programming Results Power System SOC: Battery System

The cumulative cost function is calculated using Equation [5-12]. The cost for each instance in time (non-cumulative) does not include the cost for the previous step, J_{n+1}. The non-cumulative cost for each step in time is shown in Figure 5-15. The total cumulative cost function to meet this profile's needs is 1,477,534. The cost function is greater than the 1,068,187 joules required to complete this power demand profile because it includes losses in the DC to DC converter as well as loses associated with the battery's C-rating.

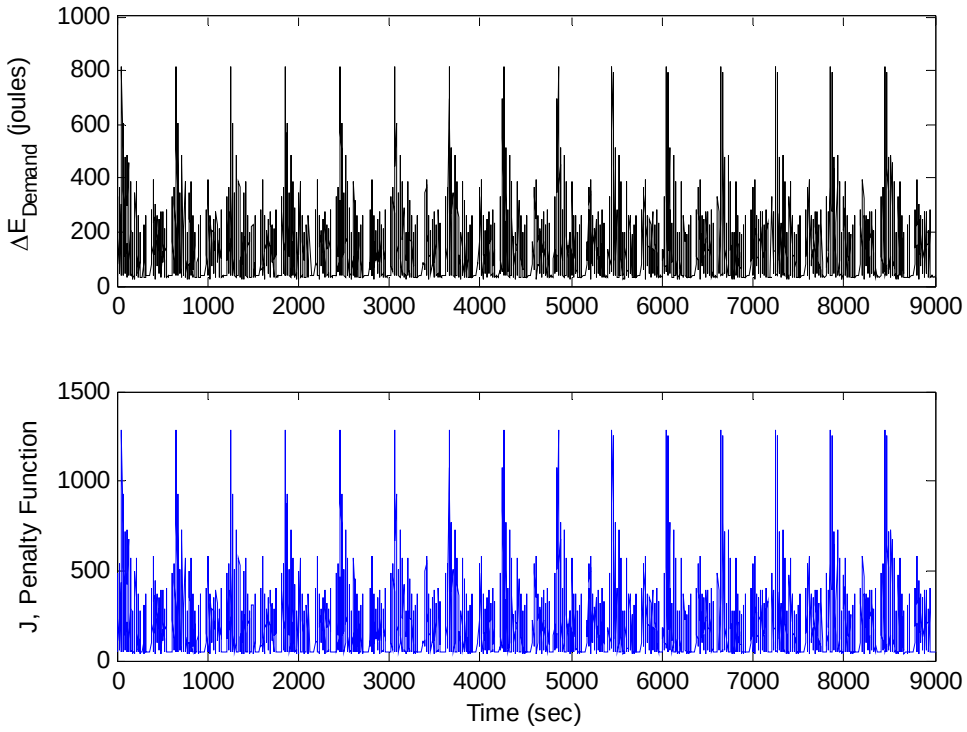

Figure 5-15. Dynamic Programming Results Cost Function: Battery System

The results of the of the Dynamic Programming algorithm for the battery-only power system can be found in Table 5-2. The battery power system consisting of two BB2590 batteries had a theoretical energy capacity of 1,492,992 joules. A total of 15,412 joules remained in the battery at the end of the profile. After complete the profile, the battery had 1.033% of its total charge remaining. The power demand profile has consumed 98.97% of the total energy available

in the battery. The Dynamic Programming MATLAB® script written for a battery-only system can be found in Appendix A.5.1.

Table 5-2: Battery-Only Dynamic Programming 150min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Battery Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	150	1492992	15,412	1.03%

5.5.4 Battery-Only Rule-Based Results

The results of the rule-based Controller are displayed in Figure 5-16 through Figure 5-19. The top graph in Figure 5-16 shows the energy demand profile. The bottom graph shows the energy transferred to the robot's power bus from the battery to meet the robot's energy demand profile. These two graphs are once again identical because there is only one power source meeting the robot profile power demands.

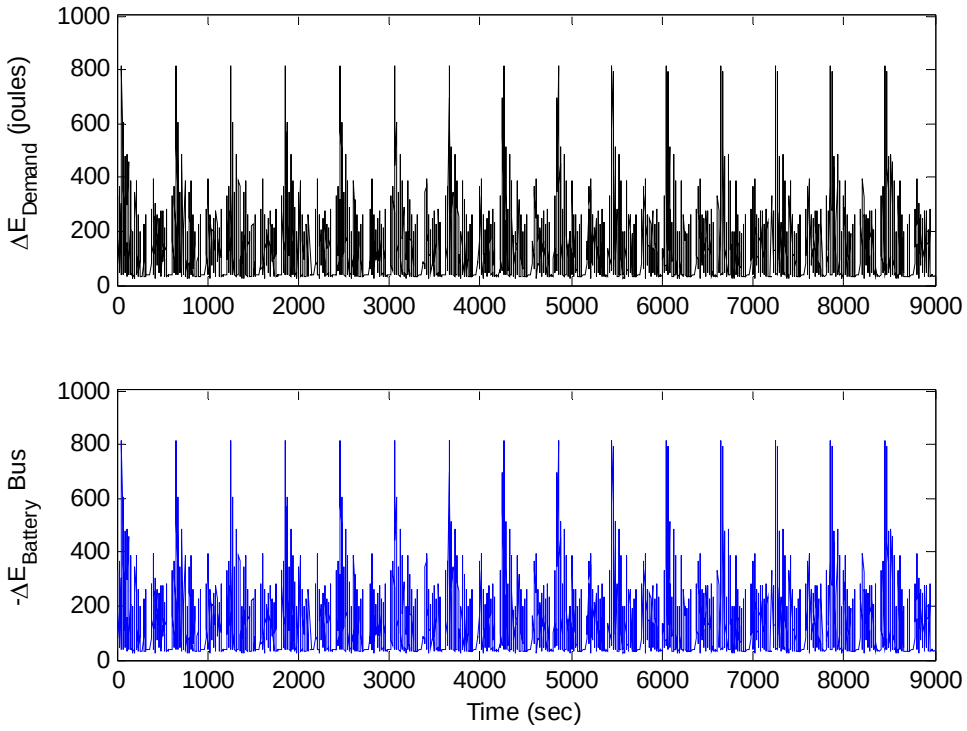

Figure 5-16. Rule Based Controller Results Power System Bus Energy: Battery System

The second graph in Figure 5-17 shows the change in the state of charge of the battery to meet this demanded profile. The change in the state of charge of the battery in Figure 5-17 is greater than the energy demand profile at each instance in time. The change in the battery's state of charge is also greater than the energy on the bus from the battery in Figure 5-16. Figure 5-18 displays the cumulative energy required to meet the power demand profile and the corresponding state of charge of the battery to meet this demand.

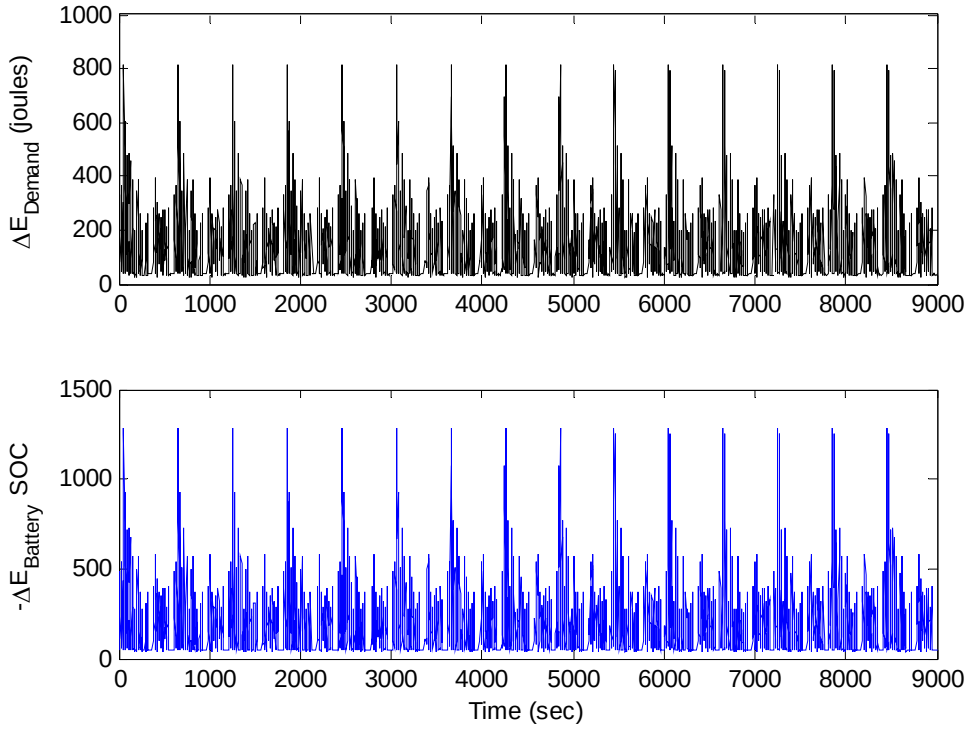

Figure 5-17. Rule Based Controller Results Power System Change in SOC: Battery System

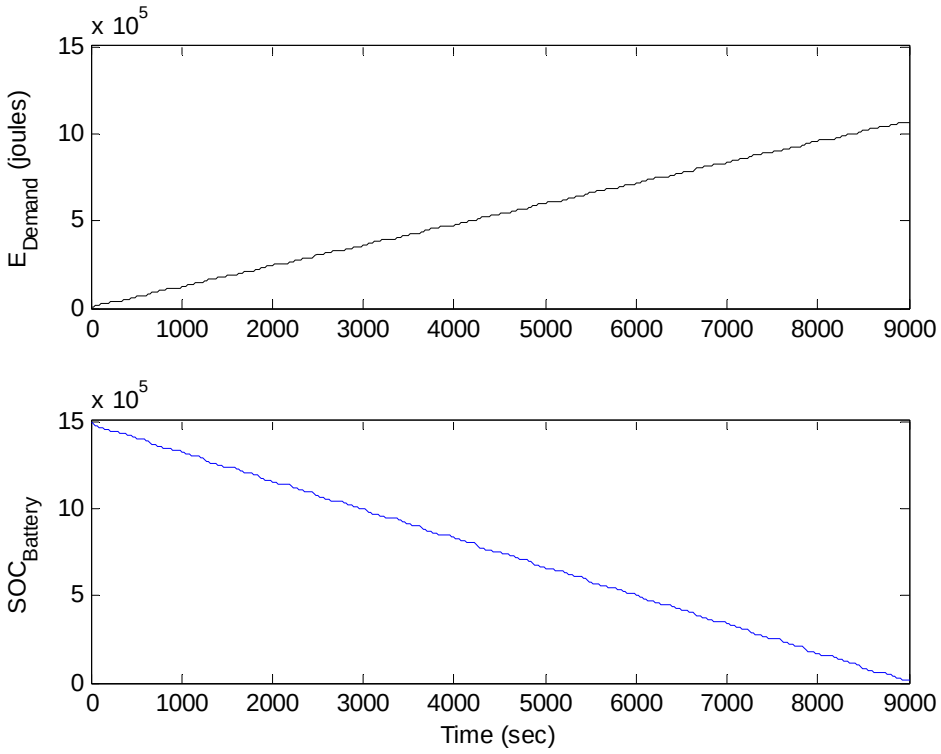

Figure 5-18. Rule Based Controller Results Power System SOC: Battery System

The non-cumulative cost for each step in time is shown in Figure 5-19. The total cumulative cost function to meet this profile is 1,477,585. The cost function is greater than the 1,068,187 joules required to complete this power demand profile.

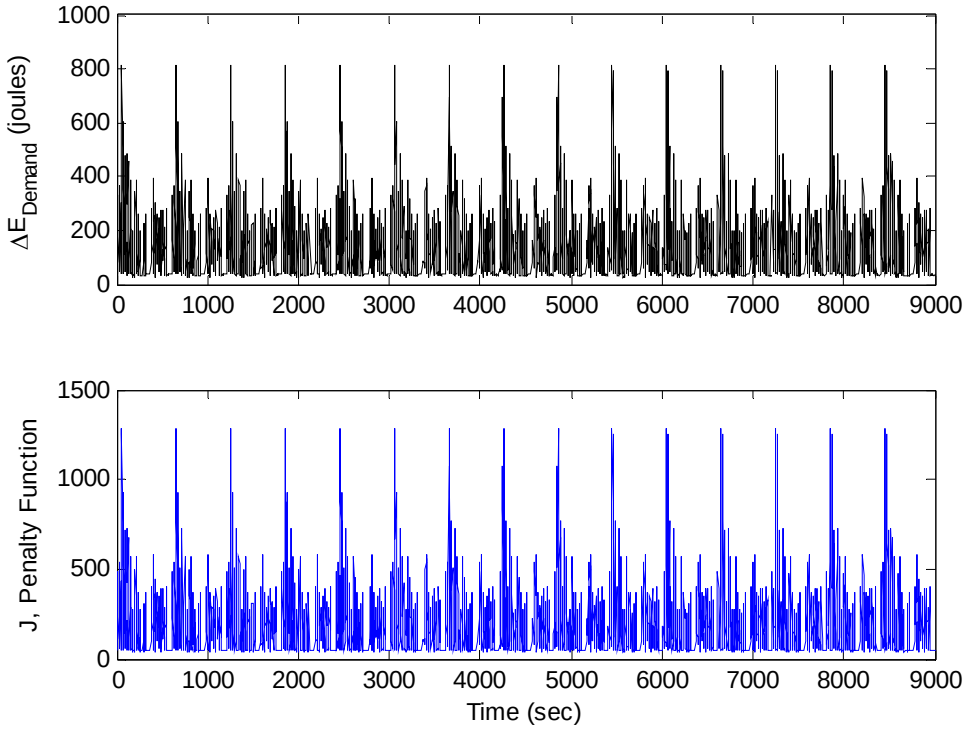

Figure 5-19. Rule Based Controller Results Cost Function: Battery System

The results of the of the Dynamic Programming algorithm for the battery-only power system can be found in Table 5-3. The battery power system consisting of two BB2590 batteries had a theoretical energy capacity of 1,492,992 joules. A total of 15,352.84 joules remained in the battery at the end of the profile. After complete the profile, the battery had 1.033% of its total charge remaining. The power demand profile has consumed 98.97% of the total energy available in the battery, identical (as expected) to the DP result.

Table 5-3: Battery-Only Rule-Based 150min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Battery Energy Remaining (joules)	Percent Energy Remaining
Rule-Based	150	1492992	15,412	1.03%

5.5.5 Relative Performance of Battery-Only System Controllers

The results from the Dynamic Programming and Rule Based Controllers are identical for a single power source system as shown in Table 5-4. Both systems start with identical profiles and power sources. The equations used to compute the change in state of charge as a function of energy on the robot's bus are calculated in the same manner for each algorithm. Each controller uses the same amount of energy and meets all of the demands of the profile. Both controllers use a single power source and are structured to always meet the energy demand profile. Dynamic Programming achieves these results by minimizing a cost function while the rule-based method is structured around subtracting the needed energy from the battery. With only one power source, the Dynamic Programming method can only change/discharge that single power source and will therefore yield the same answer as the rule-based method.

Table 5-4: Battery-Only Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Battery Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	150	1492992	15412	1.03%
Rule-Based	150	1492992	15412	1.03%

5.6 Battery Only Controller Comparison

The relative performance of the Dynamic Programming and rule-based power sharing methods are compared with one another using a battery and ultracapacitor hybrid system. First the system composition and power allocation methods are discussed. Second the power profile is shown. The relative performance of the Dynamic Programming and rule-based power sharing methods are shown and compared to determine the performance of each.

5.6.1 Battery and Ultracapacitor Composition

The second system consists of a hybrid power source with a single BB2590 battery and ultracapacitor with a mass of 0.0603kg. Each of the two power sources are fully charged at the start of the power profile. There are six possible changes in power source states which are illustrated in Figure 5-20. Both the battery and ultracapacitor can charge, discharge, or remain constant based upon how each control strategy is formulated.

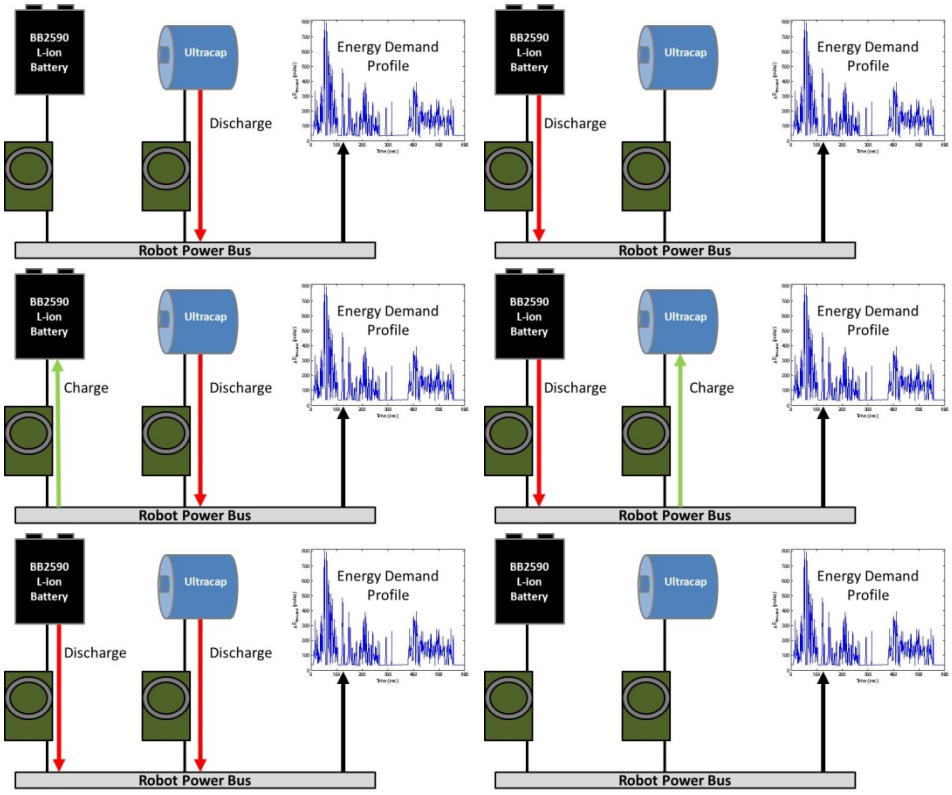

Figure 5-20. Battery and Ultracapacitor Hybrid System

5.6.2 Power Profile

The same 10 minute power profile mentioned earlier is used for the battery and ultracapacitor power profile. This profile is repeated only 6.5 times because, based on the controller results, 6.5 cycles is the limit of profiles which can be run before the battery and ultracapacitor are completely depleted.

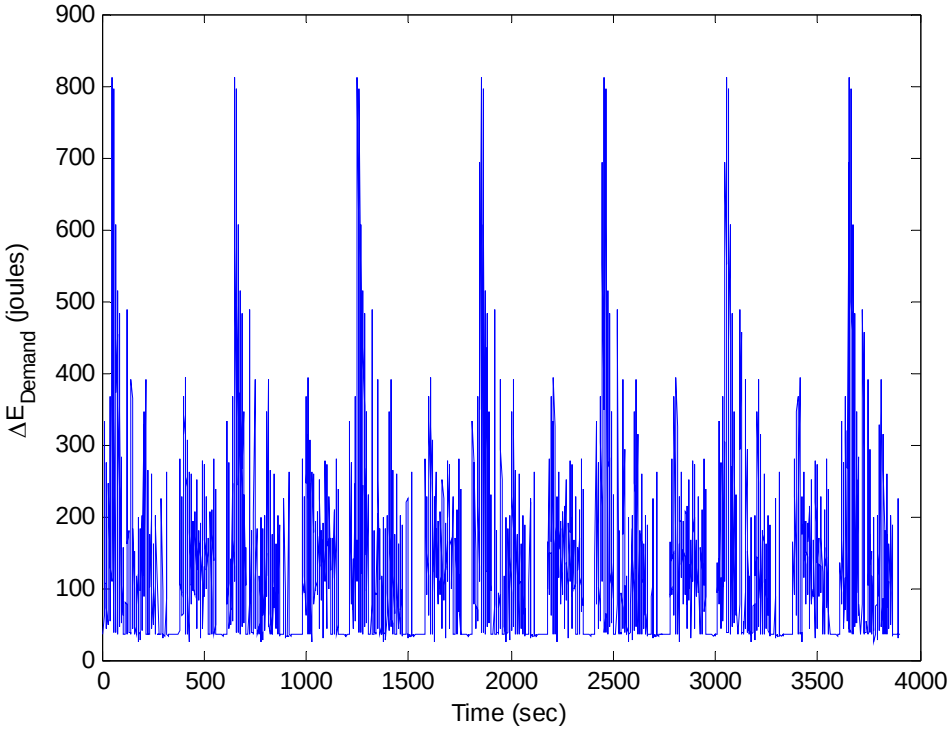

Figure 5-21. Battery and Ultracapacitor System Power Profile

The control strategy which has the most remaining energy at the end of the profile is said to be the most efficient. The total remaining energy at the end of the profile is the summation of the energy available from either source. There is no weight applied to the energy from either source, as any energy from each is considered equally usable at the end of the mission.

5.6.3 Battery and Ultracapacitor Dynamic Programming Results

In this section two power profiles are examined for each control strategy. The first profile is a single 10 minute cycle of the previous power demand profile. The second profile is this same cycle looped 6.5 times. The purpose of the longer cycle is to show how the Dynamic Programming control strategy offers a consistent benefit over the rule-based controller regardless of the duration of the power profile.

10 Minute Profile

The results of a single cycle using Dynamic Programming can be found in Figure 5-22 through Figure 5-25. Figure 5-22 through Figure 5-24 contains three plots each which include the following: the power demand profile, the battery's power demand, and the ultracapacitor's response to the demand profile. The first plot in Figure 5-22 shows the demand profile for the 10 minute power demand profile. The two charts below the demand profile show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure shows that, for the Dynamic Programming result, the bulk of the energy comes from the battery and is supplemented by the ultracapacitor only during times of large energy demand. The energy from the battery onto the robot's power bus is sometimes greater than the demanded profile because the battery is both meeting the profile and adding energy into the ultracapacitor to charge it. During these times, the ultracapacitor also has a negative value in the y-axis which denotes charging. The battery never charges during this profile even though the ultracapacitor does so regularly.

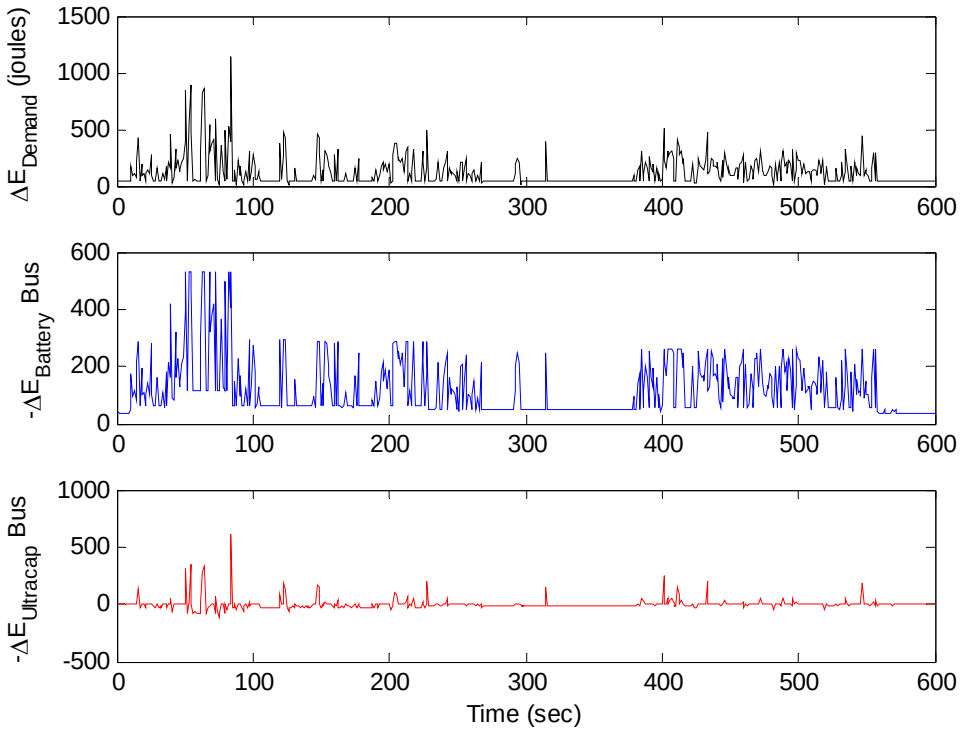

Figure 5-22. Dynamic Programming Controller Results Power System Bus Energy: Battery and Ultracapacitor System

The bottom two graphs in Figure 5-23 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. While the power demand profile graphs in Figure 5-22 and Figure 5-23 are identical, there are a few key differences between the battery and ultracapcitor's change in state of charge and the amount of energy on the bus from each source. The battery's change in state of charge in the figure below is always greater than the energy from the battery on the bus. As noted previously the battery experiences losses from overdrawing current and form the DC/DC converter. The amount of energy leaving the ultracapacitor, while discharging, is also greater than the amount of energy on the robot's power bus from the discharge. This is due to the losses experienced by the DC/DC converter operating during the discharge of the ultracapacitor. While charging however, the amount of energy traveling from the robot's power bus to the ultracapacitor is greater than the change in the ultracapacitor's state of charge. This is again because the DC/DC converter draws power during

charging of the ultracapacitor, e.g. the converter experiences efficiency losses during both charging and discharging.

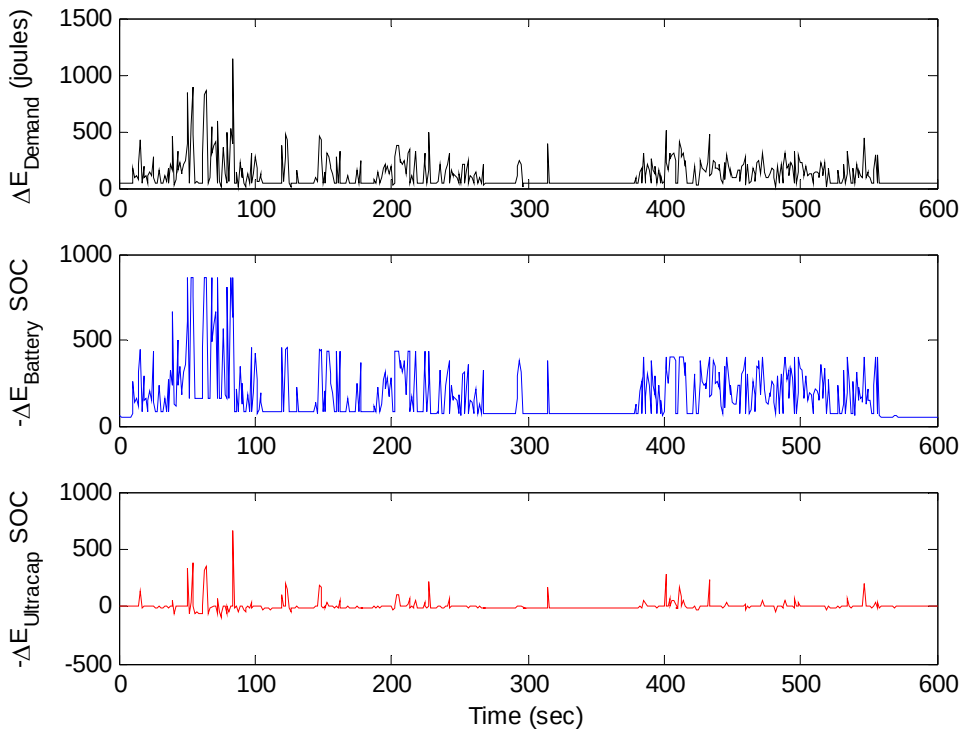

Figure 5-23. Dynamic Programming Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Each power source's state of charge and the cumulative energy demand are shown in Figure 5-24. The energy demand profile is consistently increasing throughout the 10 minute profile. Accordingly, the battery's charge is consistently decreasing. While this particular battery contains 746,496 joules, only the energy required to complete the mission is shown below.

The ultracapacitor is constantly charging and discharging throughout the duration of the energy demand profile. Again, the Dynamic Programming controller discharges the ultracapacitor during instances of large energy demand. During instances of lower energy demand the ultracapacitor is slowly discharged. The rate of each is computed using the cost function described previously. The most efficient use of the ultracapacitor using the Dynamic Programming controller is to have the ultracapacitor completely discharged at the end of the

profile. The relative cost to draw energy from this source is always lower than drawing from the battery because the ultracapacitor experiences fewer losses. Providing the profile is long enough in duration, the ultracapacitor will always be completely discharged by the end of a profile.

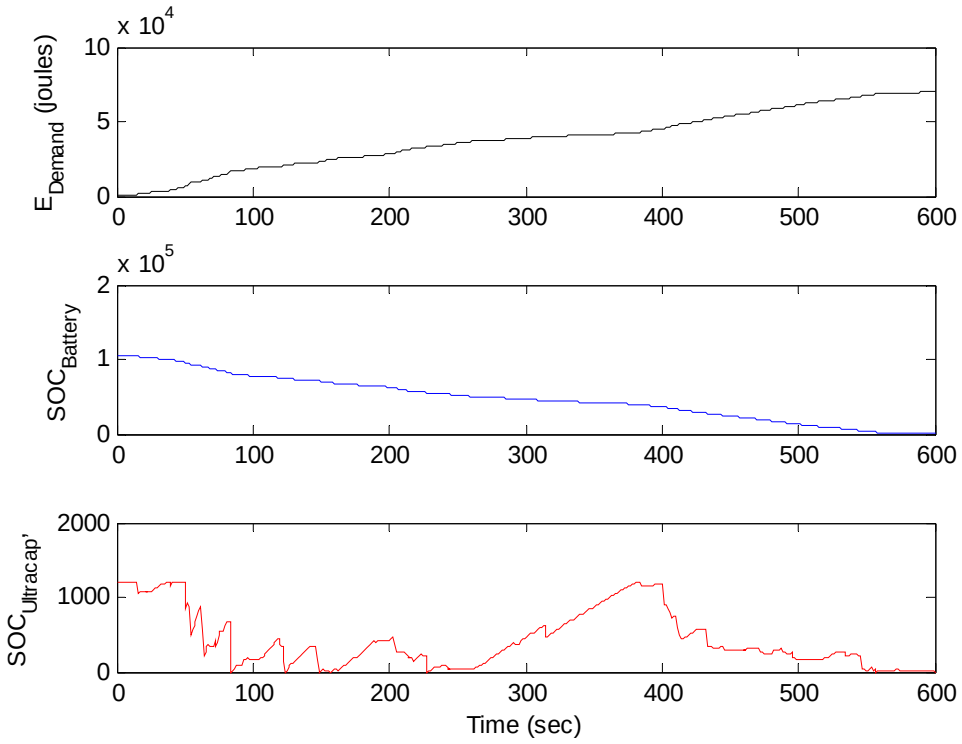

Figure 5-24. Dynamic Programming Controller Results Power System SOC: Battery and Ultracapacitor System

The energy demand profile and the corresponding non-cumulative cost function for each step in time is shown in Figure 5-25. The non-cumulative cost function represents the total change in both the battery and ultracapacitor's state of charge to meet the demanded energy profile. Each power sources experiences losses as mentioned earlier, so the non-cumulative cost function will be greater than the demanded energy for each instance in time.

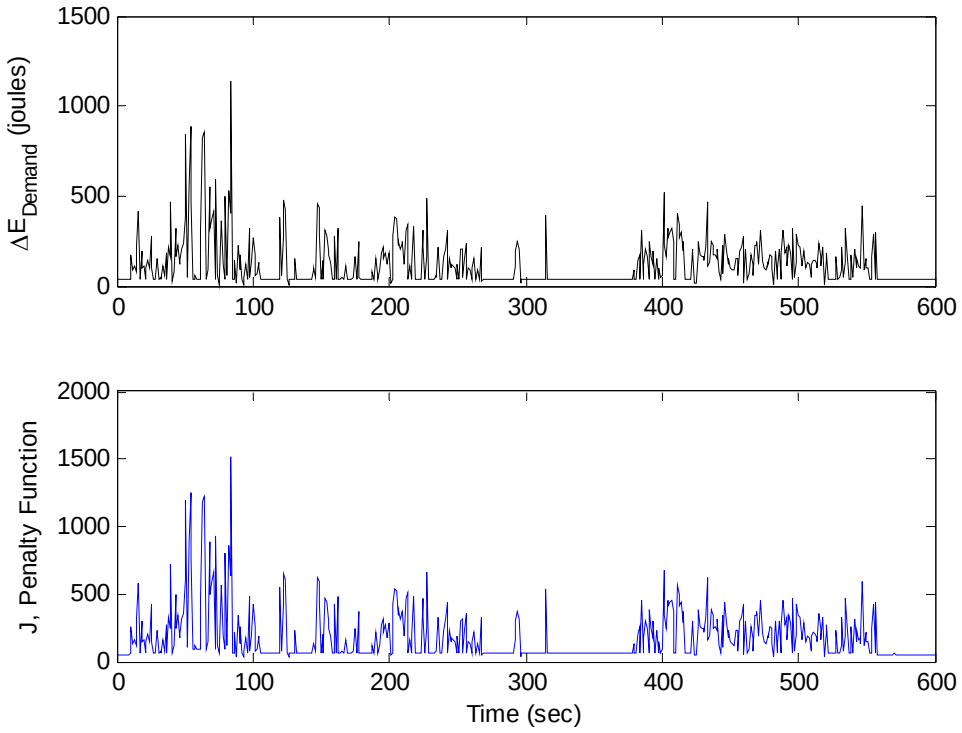

Figure 5-25. Dynamic Programming Controller Results Cost Function: Battery and Ultracapacitor System

The results of the Dynamic Programming algorithm applied to the battery and ultracapacitor hybrid system can be found in Table 5-5. The battery power system consisting of one BB2590 batteries had a theoretical energy capacity of 746,496 joules, which can be combined with an ultracapacitor which has an initial charge of 1200 joules. At the end of the profile, a total of 639893.06 joules remained in the battery and 0 joules remain in the ultracapacitor. The demanded profile requires a total of 70,017 joules while 107,802.94 joules left the two power sources to meet the power demand profile. After the profile, the battery had 85.88% of its total charge remaining. The power demand profile consumed 14.12% of the total energy available from the hybrid system.

Table 5-5: Battery and Ultracapacitor Dynamic Programming 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	10	746496	1200	639,893	0	85.88%

Without the addition of an ultracapacitor, a battery-only power source with a single BB2590 battery would be unable to complete this power profile. The BB2590 has a limit built into the battery which limits it output to 662.4 joules. The peak demand of the above power profile is almost twice the maximum energy available from a single battery. Without the addition of an ultracapacitor, a single BB2590 battery is unable to complete this mission.

65 Minute Profile

The results using Dynamic Programming on a long-duration profile can be found in Figure 5-26 through Figure 5-29, where results are shown for the 65 minute cycle. The first plot in Figure 5-26 is the demand profile for the 65 minute power demand profile. The two charts below the demand profile show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure again shows that the best strategy is to use the bulk of the energy from the battery, and to only supplement this with energy from the ultracapacitor during times of large energy demand. These results look very similar to those of the 10 minute profile. The bottom two graphs in Figure 5-27 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. All of the same comparisons made between Figure 5-22 and Figure 5-23 above for the 10 minute profile can be made for Figure 5-26 and Figure 5-27.

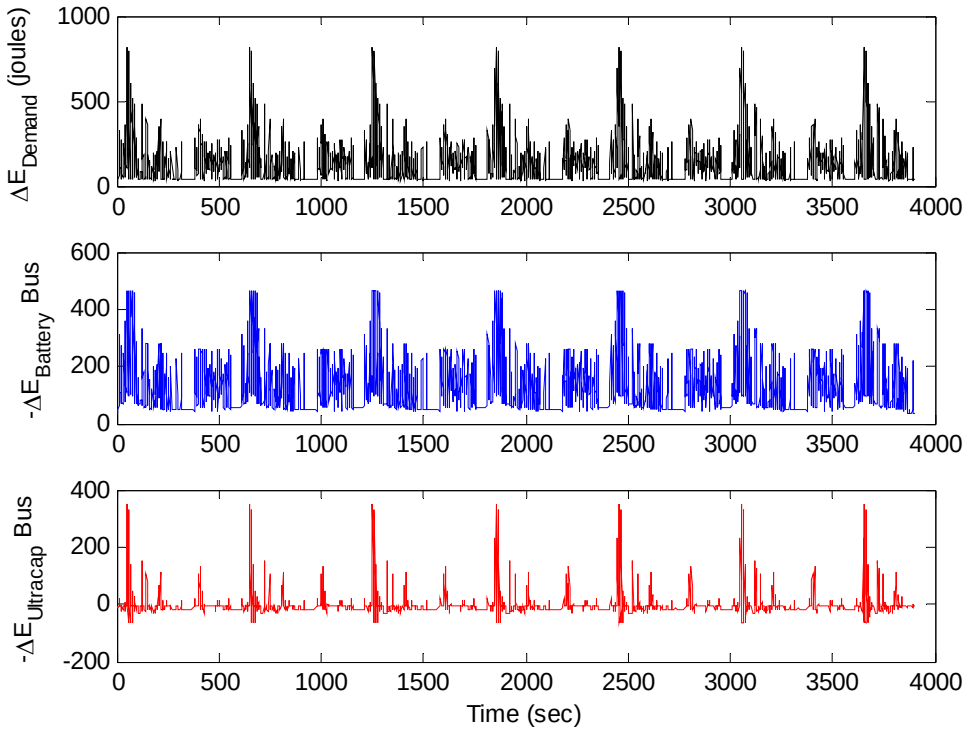

Figure 5-26. Dynamic Programming Controller Results Power System Bus Energy: Battery and Ultracapacitor System

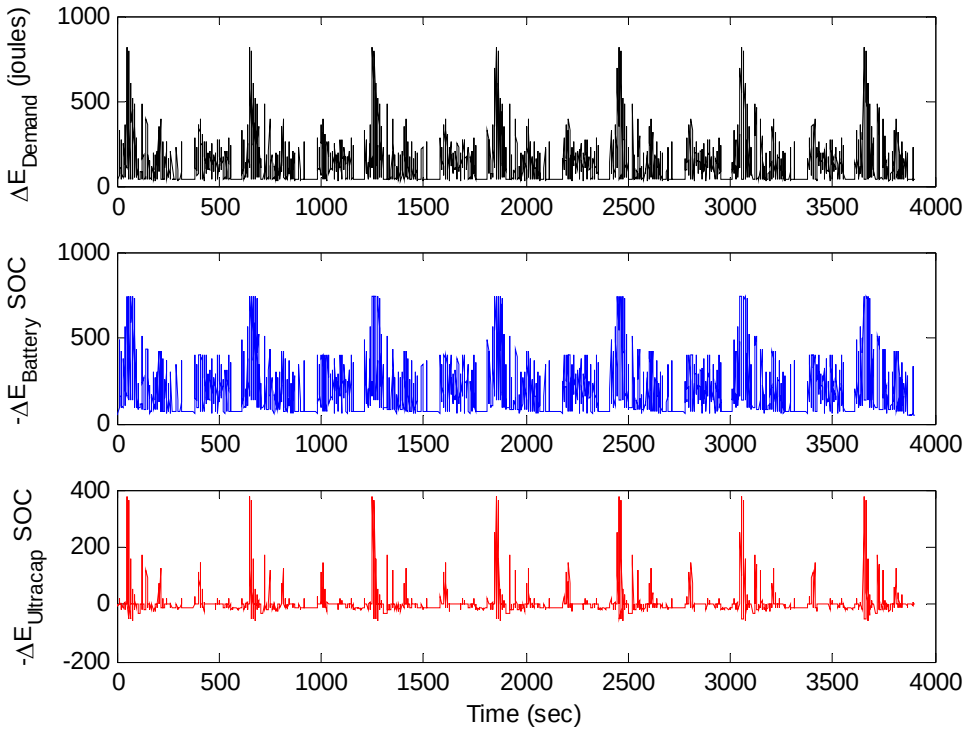

Figure 5-27. Dynamic Programming Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Figure 5-28 shows the discharge cycle for the battery and the charge/discharge cycle for the ultracapacitor for the longer profile. The control strategy for each power source is repeated 6.5 times matching the 6.5 cycles of the same energy demand profile. Figure 5-29 shows a similar trend with regard to the non-cumulative cost function. The optimal control strategy determined using Dynamic Programming provides the same controller for each cycle. Providing enough energy is available, the optimal control strategy does not appear to change despite the requirement of more or fewer energy demand cycles.

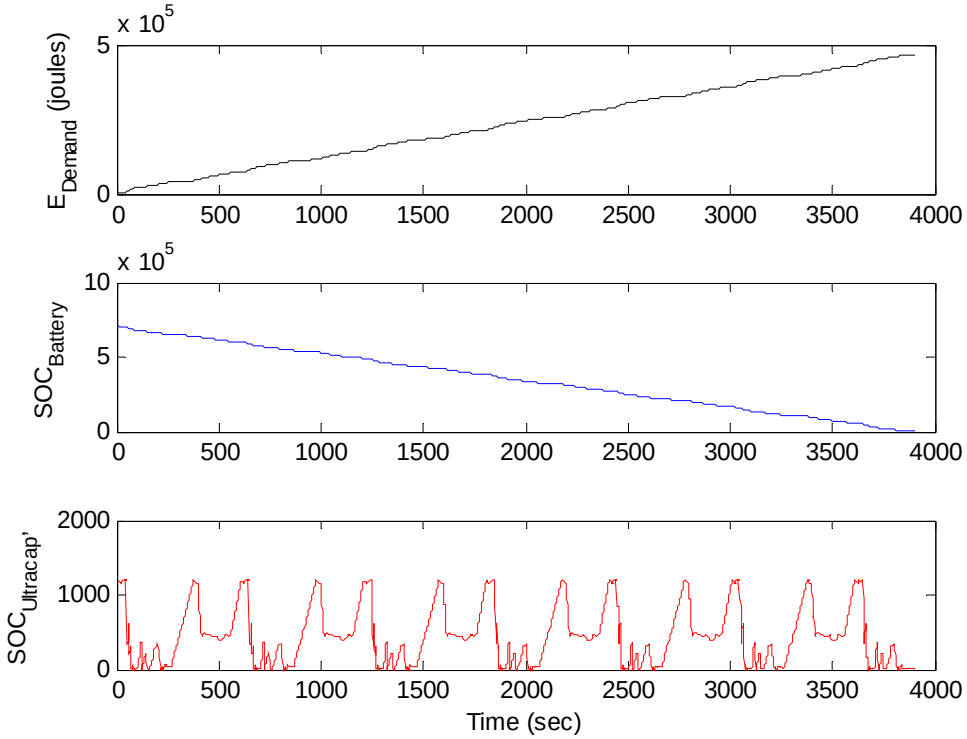

Figure 5-28. Dynamic Programming Controller Results Power System SOC: Battery and Ultracapacitor System

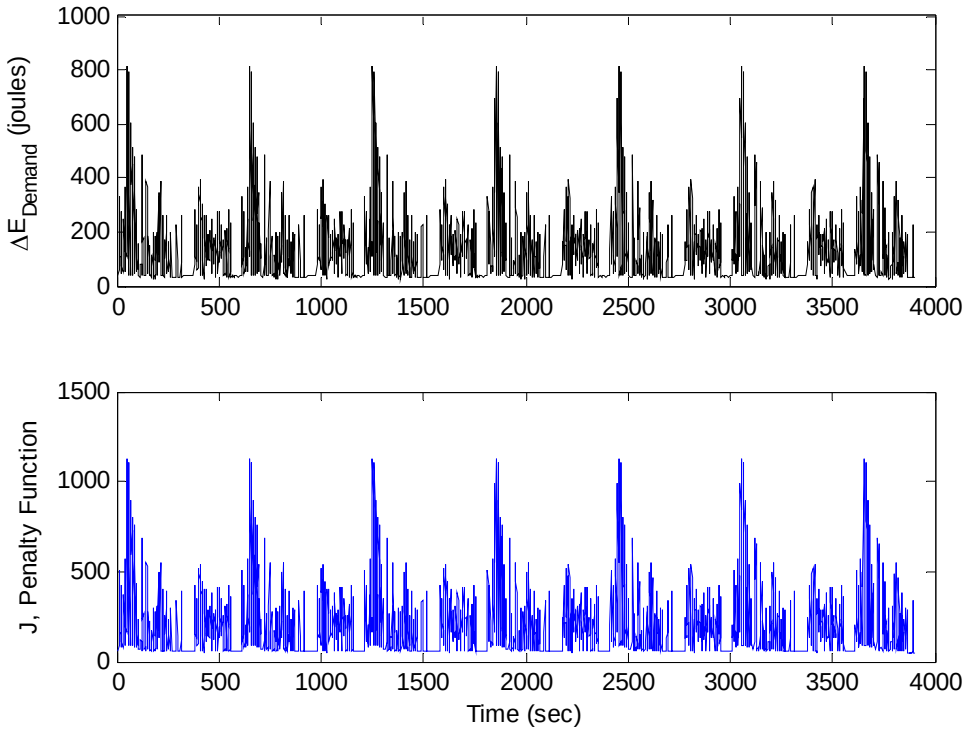

Figure 5-29. Dynamic Programming Controller Results Cost Function: Battery and Ultracapacitor System

The total energy demanded by this power profile is 467,888.27 joules. The total energy required to meet this energy demand profile is 712,629.06 joules total combined from the two power sources (Table 5-6). At the end of the 65 minute profile, only 4.7% of the 747,696 joules available from the hybrid system remain. The percent of the total energy available used to meet the energy demand profile is therefore 95.3%. The ultracapacitor is completely discharged leaving all of the remaining energy in the battery. The Dynamic Programming algorithm and data processing MATLAB® scripts written for a battery and ultracapacitor system which was used for this analysis can be found in Appendix A.5.2 and Appendix A.5.3 respectively.

Table 5-6: Battery and Ultracapacitor Dynamic Programming 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	65	746496	1200	33,867	0	4.70%

5.6.4 Battery and Ultracapacitor Rule-Based Results

10 Minute Profile

The results of a single 10 minute cycle using the Rule Based Controller can be found in Figure 5-30 through Figure 5-33. Just as with the Dynamic Programming results, Figure 5-30 through Figure 5-32 contains three plots each which include the following: the power demand profile, battery's response and ultracapacitors response to the demand profile. The two charts below the energy demand profile in Figure 5-30 show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure shows that the battery is used exclusively to meet the demands of the profile until the energy limit is reached. Once this occurs, the ultracapacitor is used to meet the additional demands. Once additional energy is available from the battery, after meeting the demand of the profile, the ultracapacitor is charged immediately back to its maximum capacity. The energy from the battery onto the robot's power bus is greater than the energy demand profile when this ultracapacitor charging occurs. During these times the ultracapacitor also has a negative value in the y-axis which denotes charging. The battery never charges during this profile.

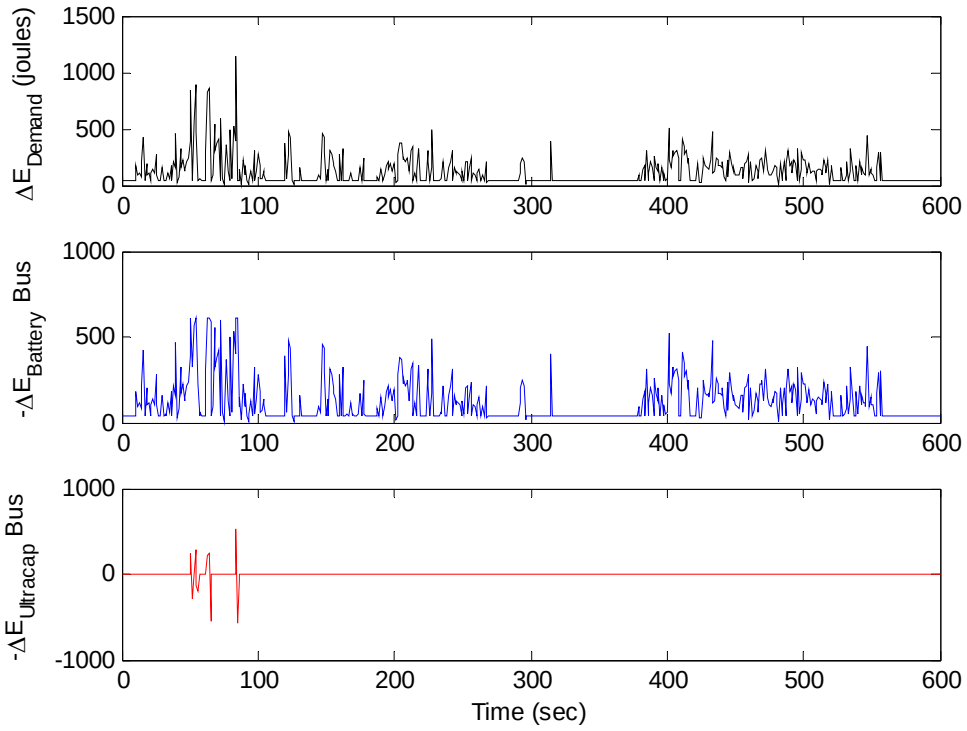

Figure 5-30. Rule Based Controller Results Power System Bus Energy: Battery and Ultracapacitor System

The bottom two graphs in Figure 5-31 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. The battery's change in state of charge in the figure below is always greater than the energy from the battery on the bus. The amount of energy leaving the ultracapacitor, while discharging, is also greater than the amount of energy on the robot's power bus from the discharge. This is due to the losses experienced by the DC/DC converter. While charging, the amount of energy traveling from the robot's power bus to the ultracapacitor is greater than the change in the ultracapacitor's state of charge. This is again because the DC/DC converter draws power and experiences efficiency losses during both charging and discharging. While the times at which each source charges and discharges varies in comparison to the DP results, the rule-based controller seems to utilize similar principles for using the ultracapacitor. This, the losses appear to be consistent between controllers. The energy demand profile is always met using the hybrid system. The relationship between change in state of charge and energy available from the bus is again consistent with the scaling and efficiency equations presented at the beginning of this Chapter.

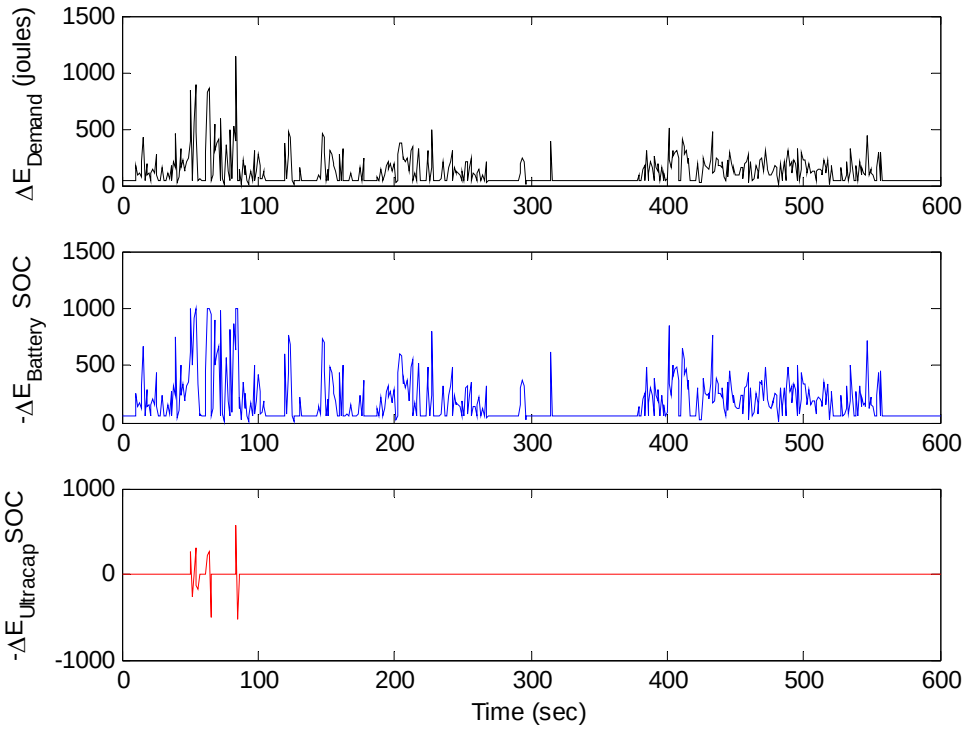

Figure 5-31. Rule Based Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Each power source's state of charge and the cumulative energy demanded are shown in Figure 5-32. The energy demand profile is consistently increasing throughout the 10 minute profile. Just as intuitively, the battery's charge is seen to be consistency decreasing. The ultracapacitor, in comparison, is constantly charging and discharging throughout the duration of the energy demand profile.

The biggest difference between these results and those seen previously, is that, unlike Dynamic Programming, the rule-based controller immediately charges the ultracapacitor to full capacity whenever possible. As a result, at the end of this 10 minute profile the battery is depleted while the ultracapacitor remains fully charged.

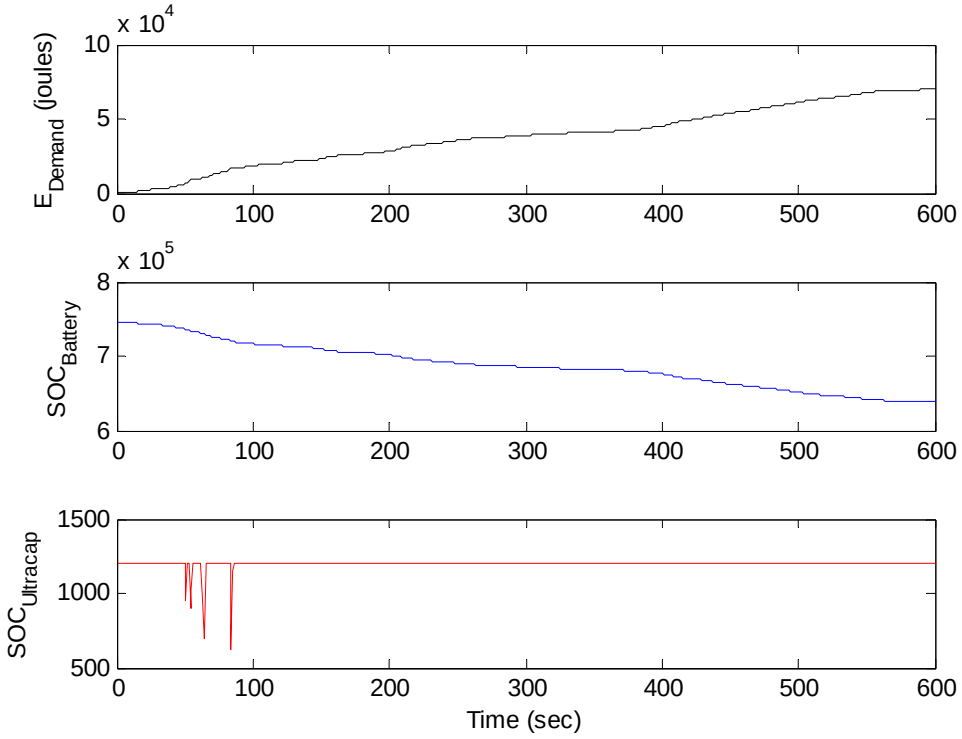

Figure 5-32. Rule Based Controller Results Power System SOC: Battery and Ultracapacitor System

The rule-based controller does not use a cost function to calculate the optimal use of each power source. The cost function can be used to compare the relative performance of each controller. The Dynamic Programming controller minimizes energy use by minimizing a cost function which describes energy use. Both the energy used and the cumulative cost function can therefore be used to evaluate each controller performance. The energy demand profile and the corresponding non-cumulative cost function for each step in time is shown in Figure 5-33. Using the rule-based controller, the non-cumulative cost function will be greater than the demanded energy from the profile.

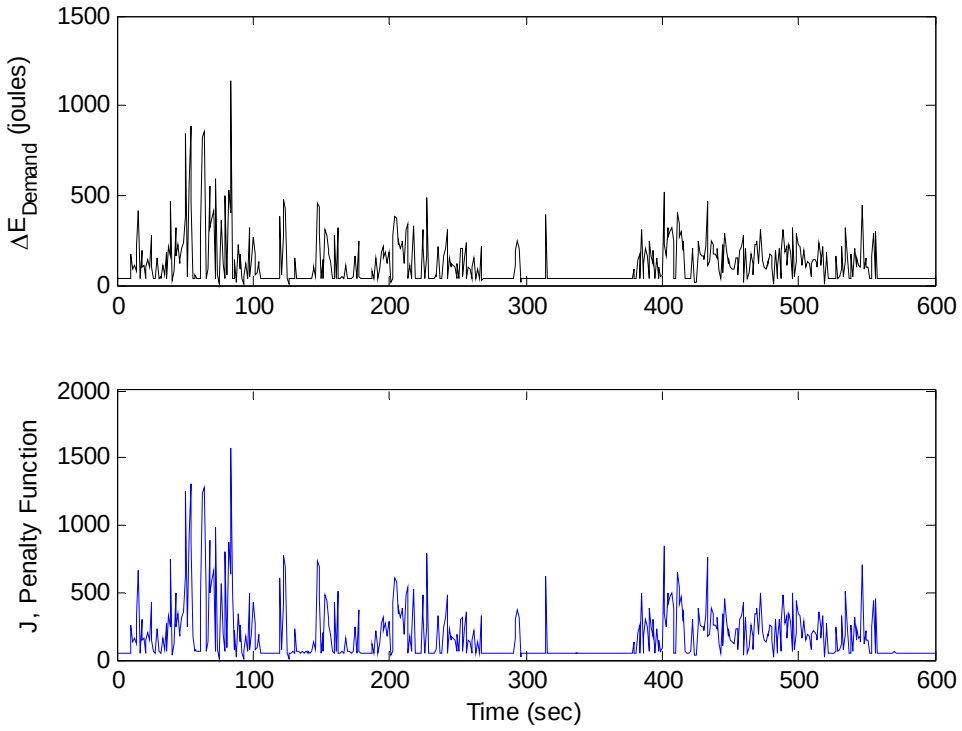

Figure 5-33. Rule Based Controller Results Cost Function: Battery and Ultracapacitor System

Table 5-7 shows that at the end of the 10 minute profile, a total of 639,893 joules remained in the battery and 1200 joules remained in the ultracapacitor. The demanded profile requires a total of 70,017 joules while 107,967 joules left the two power sources to meet the power demand profile. After completing the profile, the battery had 85.74% of its total charge remaining. The power demand profile consumed 14.26% of the total energy available from the hybrid system.

Table 5-7: Battery and Ultracapacitor Rule-Based 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Rule-Based	10	746496	1200	639,893	1,200	85.74%

65 Minute Profile

The results from the 65 minute profile using the rule-based method can be found in Figure 5-34 through Figure 5-37. Figure 5-34 plots the 65 minute energy demand profile and the hybrid system responses using the rule-based controller. Figure 5-34 describes the amount of energy on the power bus from each source while Figure 5-35 shows the change the hybrid system component's state of charge to meet that same demand. The hybrid system's responses are identical to the results for the 10 minute profile, except repeated 6.5 times.

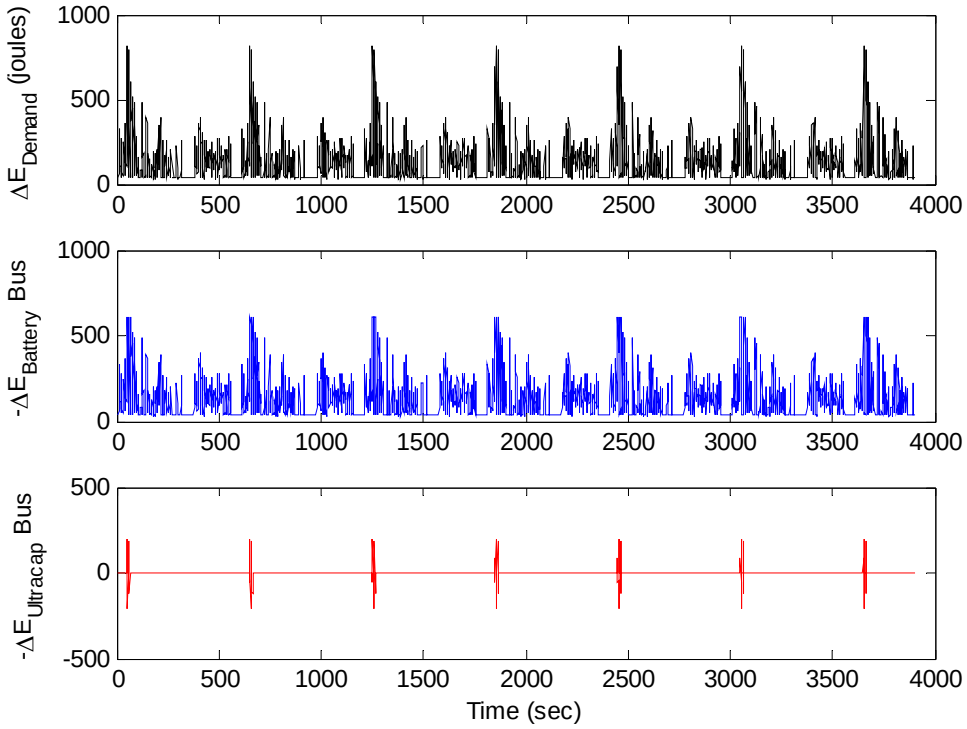

Figure 5-34. Rule Based Controller Results Power System Bus Energy: Battery and Ultracapacitor System

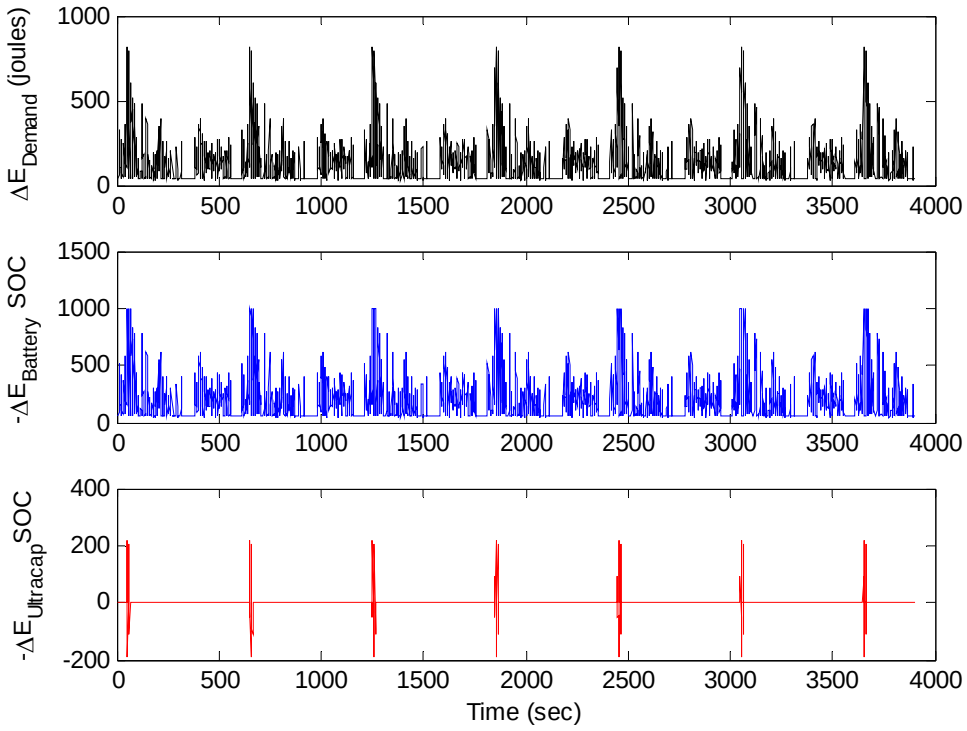

Figure 5-35. Rule Based Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Figure 5-36 shows the discharge cycle for the battery and the charge/discharge cycle for the ultracapacitor for the 65 minute profile. The control strategy for each power source is repeated 6.5 times matching the 6.5 cycles of the same energy demand profile. The battery consistently discharges while the ultracapacitor remains consistently at its peak capacity. Figure 5-37 shows a similar trend with regard to the non-cumulative cost function. The controller results remain the same regardless of the number of power demand cycles repeated.

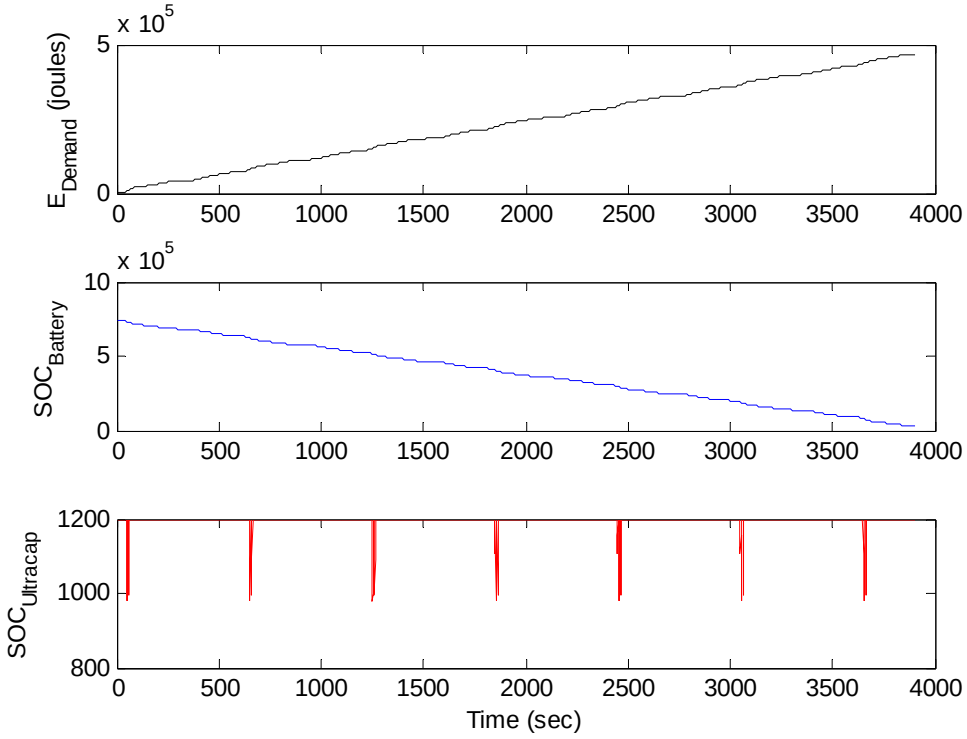

Figure 5-36. Rule Based Controller Results Power System SOC: Battery and Ultracapacitor System

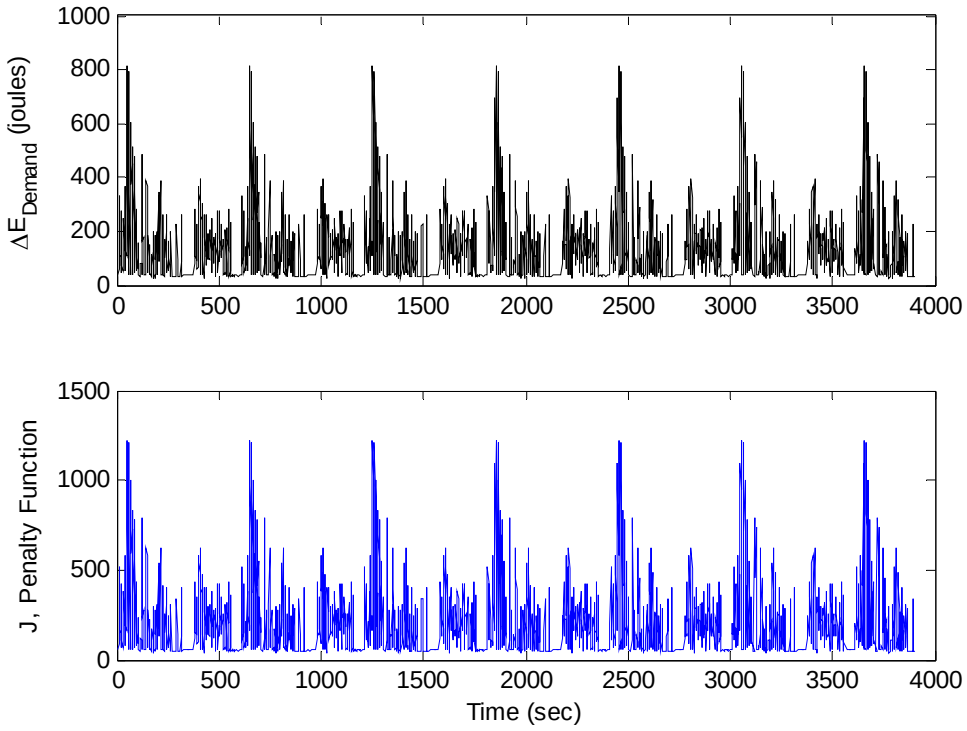

Figure 5-37. Rule Based Controller Results Cost Function: Battery and Ultracapacitor System

The total energy required to meet the 467,888 joules demanded in the profile is 715,808 joules between the two power sources shown in Table 5-8. Only 467,888 joules were used to meet the energy demand profile. The decrease in the state of charge of the hybrid system, however, decreased by 715,808 joules. Only 4.26% of the hybrid system's 747,696 joules remained at the end of the 65 minute profile. The hybrid system depleted 95.74% of its energy to meet the demanded profile using the Rule Based Controller method. The rule-based controller MATLAB® scripts written for a battery and ultracapacitor system which was used for this analysis can be found in Appendix A.5.4.

Table 5-8: Battery and Ultracapacitor Rule-Based 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Rule-Based	65	746496	1200	30,688	1,200	4.26%

5.6.5 Relative Performance of Battery and Ultracapacitor System Controllers

The Dynamic Programming controller outperformed the rule-based controller for both the 10 and 65 minute power demand profiles. The results of the 10 minute profile are listed in Table 5-9. During the 10 minute profile, the percentage of the battery's charge used to meet the profile was 14.12% using the Dynamic Programming controller. The rule based method used 14.26% of the total energy available. The Dynamic Programming controller thus retains an additional 0.1% of its total energy at the end of the short profile. The Dynamic Programming controller uses 1,030 joules less than the Rule based method.

Table 5-9: Battery and Ultracapacitor 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	10	746496	1200	639893.06	0	85.88%
Rule-Based	10	746496	1200	639893	1,200	85.74%

The results over a longer profile show similar results in Table 5-10. The Dynamic Programming controller requires 95.3% of the total energy to meet the 65 minute profile. The rule-based Controller requires 95.74%. This translates into an energy savings of 0.46% or 3,235 joules. While Dynamic Programming yields a more optimal solution than the Rule Based method, the relative difference between the two is nearly negligible. The conclusion which can therefore be drawn from this comparison is that the rule-based controller yields nearly the same results as Dynamic Programming for the profiles that have been considered. The rule-based method is therefore used exclusively from here on to optimize and evaluate a wide range of hybrid systems. The other added benefit to the rule-based method, as noted before, is that it can be implemented in real time and is more computationally efficient than Dynamic Programming.

Table 5-10: Battery and Ultracapacitor 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	65	746496	1200	33866.94	0	4.70%
Rule-Based	65	746496	1200	30688	1,200	4.26%

Additional profiles were run to reinforce the above results showing that Dynamic Programming algorithm always offers a more optimal solution than the rule-based controller. Two additional profiles for a 23.3 and 83.3 minute profile are located in sections A.5.4 and A.5.5 respectively in the Appendix. These power demand profiles are also collected on the Talon platform while it performed climbing traversal and manipulation capabilities. The results show that not only does Dynamic Programming offer a more optimal solution, but the benefit of the solution (energy saved) increases with the duration of the profile.

5.7 Mission Scenarios

The purpose of the hybrid system analysis is to determine the most effective hybrid system composition for three different missions. The total hybrid system mass is a fixed value of 8.4kg which is the equivalent of 6 BB2690 Lithium-ion military batteries (1.4kg each). Each mission is based around repeating a single task: monitoring, traversing a given distance and climbing stairs. The power required to complete each of these tasks was collected on the Talon robot. The power profile for each individual task was then generated and repeated a number of times to create a mission.

A segment of the robot monitoring profile is shown in Figure 5-38. The profile shown in this figure is 12 minutes long. During this mission the robot's drivetrain does not move. The stationary robot does move its mast camera twice accounting for the two spikes in energy draw. The nominal robot draw is approximately between 36 and 37 joules during the monitoring mission. The hybrid system will be optimized to complete this 12 minute mission 60 times. The total length of the monitoring mission will therefore be 12 hours.

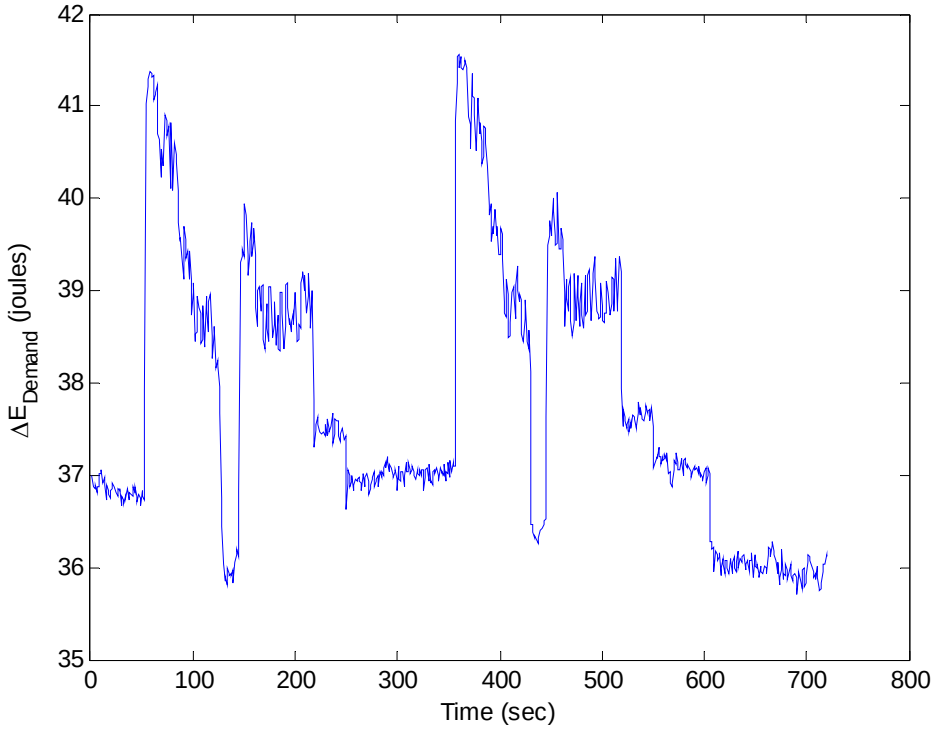

Figure 5-38. Robot Monitoring Energy Demand Profile Segment

The robot traversal mission segment is plotted in Figure 5-39. The robot traversal mission segment is also 12 minutes in duration. The robot travels 122 meters at a rate of 0.55m/s on average. Approximate two thirds of the 122 meters were traversed on grass and the remaining their on pavement. The peak energy of 390 joules occurred while turning. Each energy spike in Figure 5-39 occurred while the robot was going around a turn. The full robot energy demand mission incorporates 30 of these loops over 6 hours.

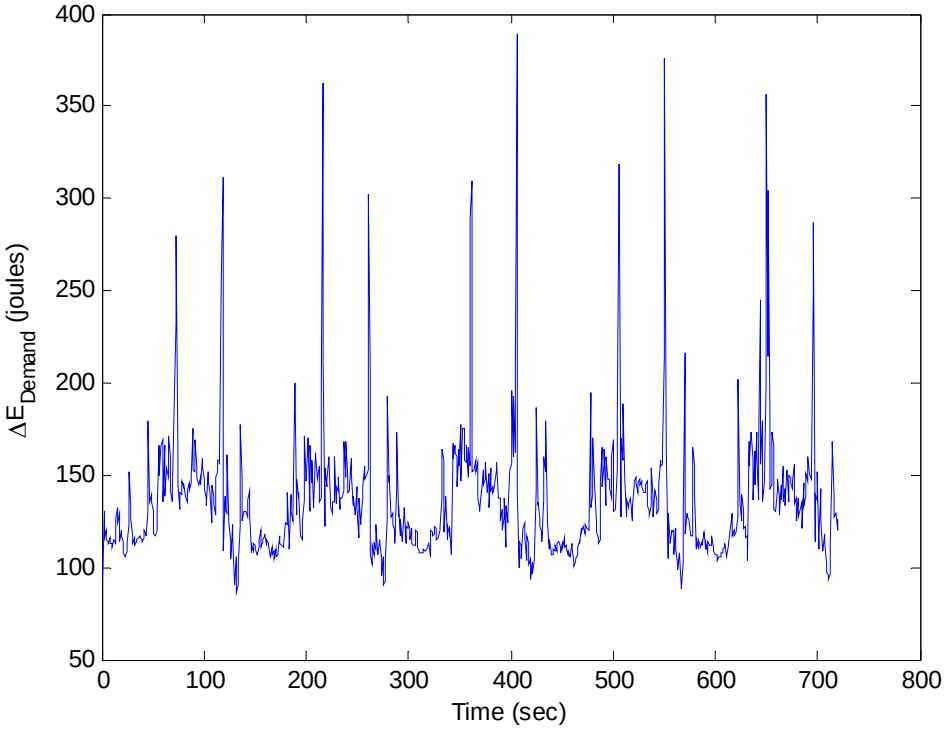

Figure 5-39. Robot Traversal Energy Demand Profile Segment

Figure 5-39 shows a segment of the robot climbing missions. The robot climbs up and down a single flight of stairs twice in five minutes. The robot climbing mission being used to optimize a hybrid system around repeats of this climbing scenario 25 times. The robot, to complete the climbing mission, must climb up and down 50 flights of stairs in 2.5 hours. Each set of energy spikes in Figure 5-40 occur while the robot is climbing up half of the flight of stairs. The dip in the energy demand occurs while the robot is making the turn onto the second half of the flight of stairs. The low energy demands which occur between times 50-175 seconds and 225-300 are the portions of the profile where the robot climbs back down the flight of stairs. During the climbing portion of this mission, the peak demanded energy peaks above 500 joules. This peak demand is over 100 joules greater than the demand from the traversal mission scenario.

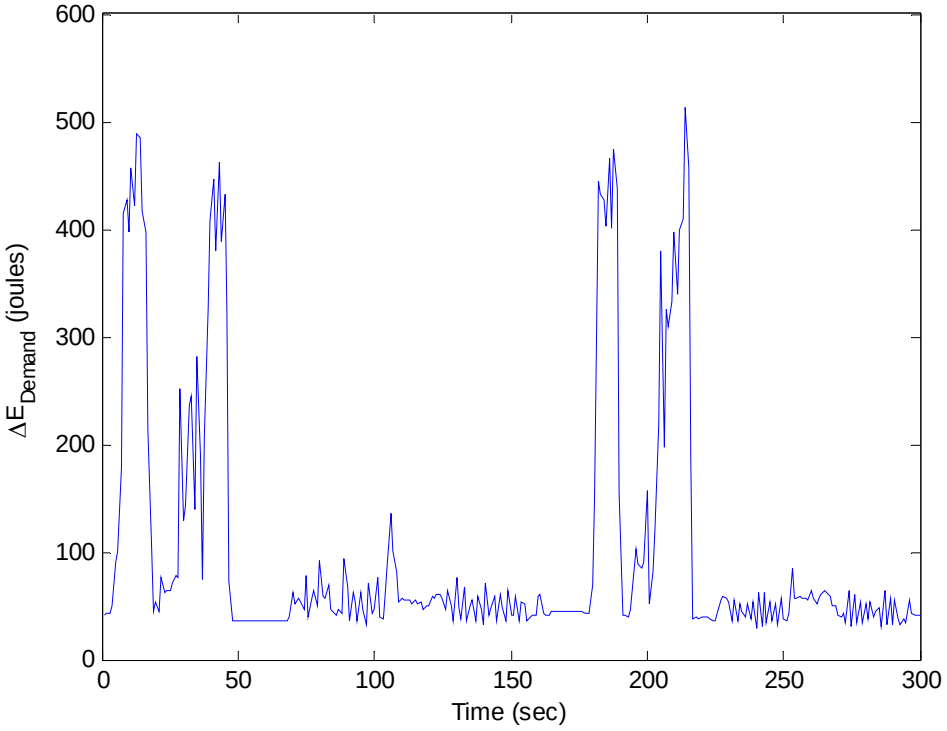

Figure 5-40. Robot Stair Climbing Energy Demand Profile Segment

5.8 Results

The total mass of the hybrid power source is forced to be equivalent to the total mass of six BB2590 batteries. While the mass of all power supplies is held constant, the percentage of each power source is varied, and with each variation the relative performance is compared. For example, the percentage of the total power source's mass which is taken up by the generator is varied from 0 to 100%. So the axis which represents the portion of the power source mass allocated to the generator is denoted "1-%$M_{Tot,Generator}$." A value of 0 denotes the mass of the generator is 8.4 kg while a value of 1 represents a generator mass of 0kg. This value varies in 1% increments between 0 and 100%. The remainder of the robot power sources' mass not allocated to the generator is then divided between the ultracapacitor and the battery, again using percentage allocations. The label in the following figures which denotes the allocation of the rest of the power source's mass is "Battery/Ultracapacitor DOH." A value of 0 indicates the

remainder of the mass not used by the generator is allocated completely to the battery. When "Battery/Ultracapacitor DOH" is equal to 1, the remainder of the power sources' mass is allocated completely to the ultracapacitor. "Battery/Ultracapacitor DOH" also varies in 1% or 0.084kg increments.

The rule-based control strategy for a three power source system (Section 5.4) is used with each different hybrid system topology to test each system's ability to meet the demands of the three missions (Section 5.8). If at any point during the missions the robot's power source is unable to meet the demands for the reasons explained in Section 5.4, then the mission is terminated at that point and is considered a failure. A failed mission is allotted a remaining energy level of zero joules. If the mission is successful, at the end of each completed mission, the additional available energy is recorded along with the hybrid composition. The different hybrid systems are then evaluated for each mission to determine which topology has the optimal hybrid composition. The optimal powertrain in this context will be the hybrid system with the most energy available after completing the given mission.

5.8.1 Robot Monitoring Mission

The full robot monitoring profile is shown in Figure 5-41. The remaining energy available after completing the mission for each hybrid system is shown in Figure 5-42 and Figure 5-43. The x and y-axis show the hybrid system composition while the vertical z-axis show the amount of energy left after the mission is completed.

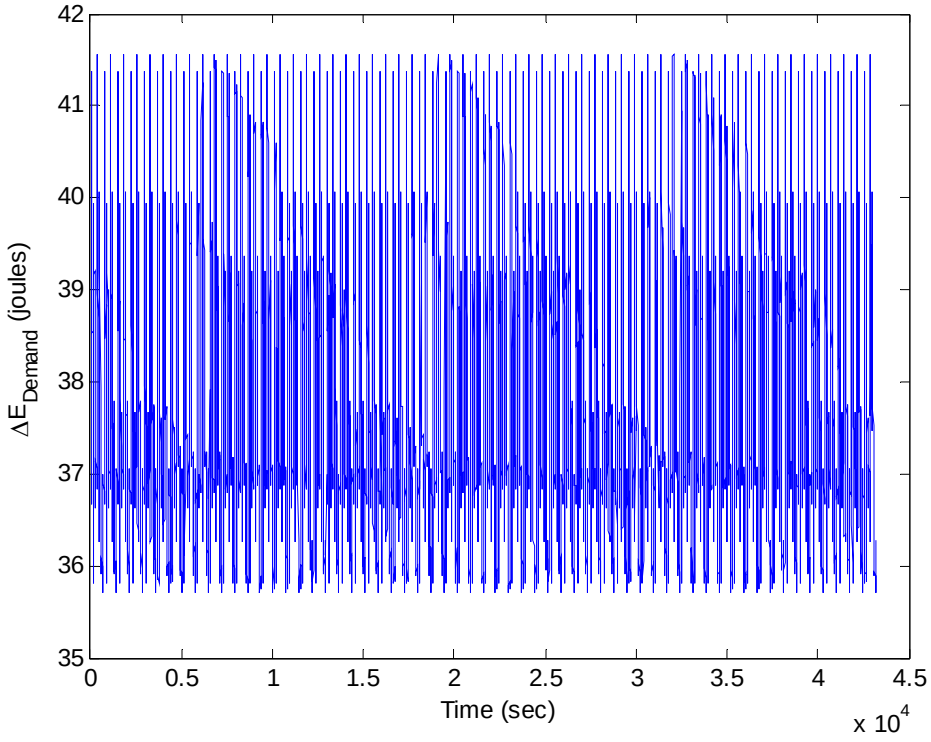

Figure 5-41. Robot Monitoring Energy Demand Full Profile

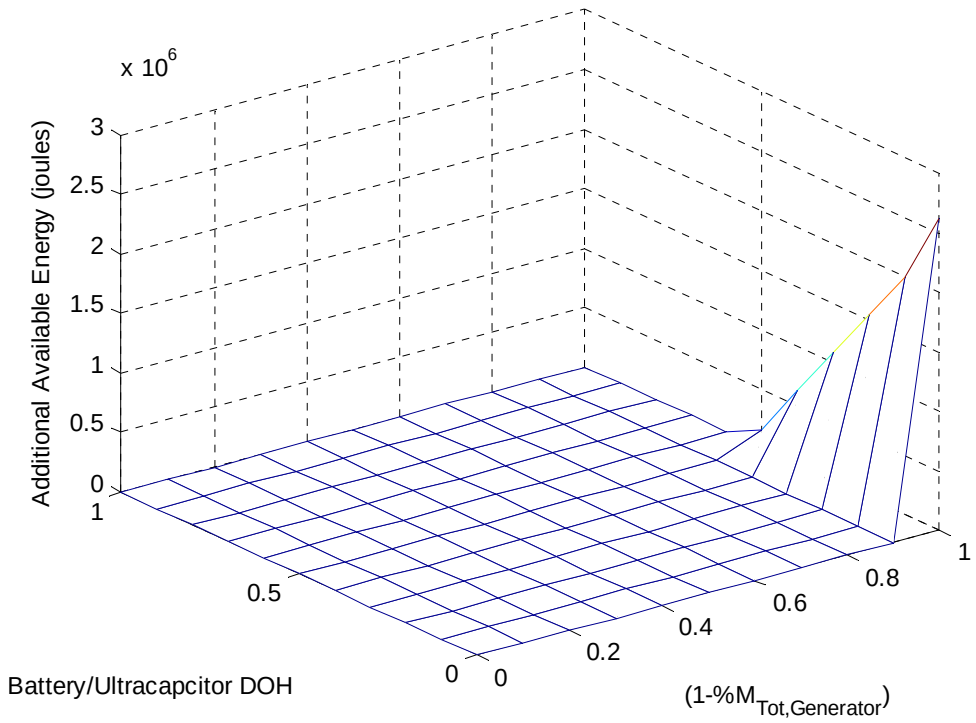

Figure 5-42. Hybrid System Effectiveness Results for Robot Monitoring Mission View 1

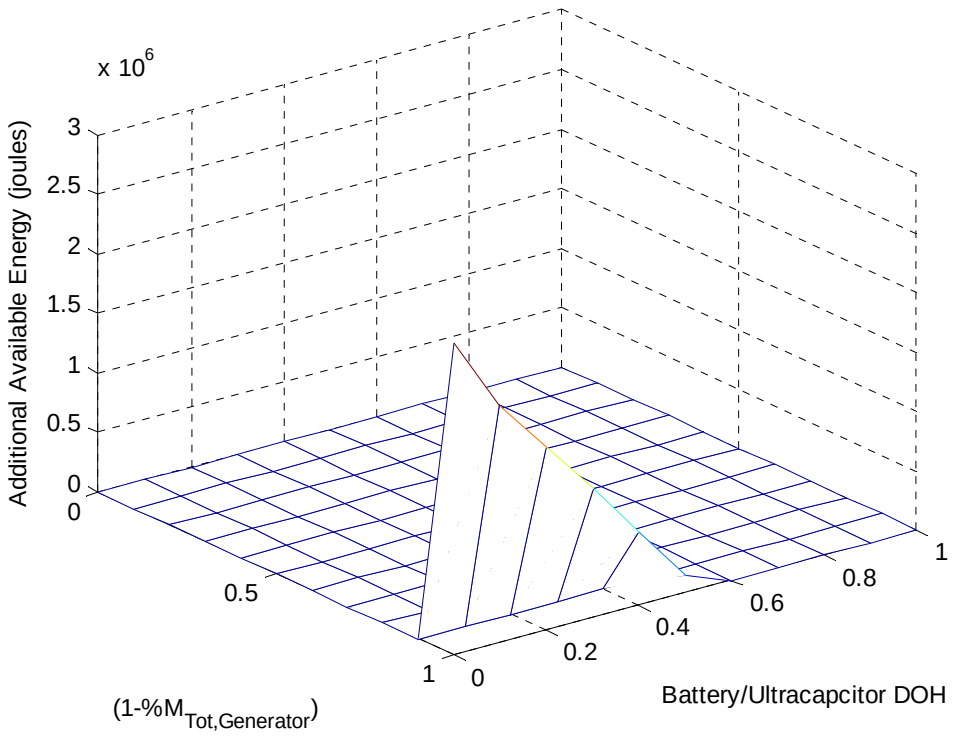

Figure 5-43. Hybrid System Effectiveness Results for Robot Monitoring Mission View 2

Figure 5-42 and Figure 5-43 show the same results with the axis shifted 90^{O}. The results indicate that there are no hybrid systems with generators which are able to complete the monitoring mission. However, hybrid power systems with the majority of the mass allocated to the battery are able to meet the demands of the mission. As the percentage of the hybrid power system's mass allocated to the battery increases, so does the amount of energy available after completion of the mission. The optimal power source for this mission contains only a battery. The remaining energy available at the end of the mission is 2,627,000 joules.

The results of the robot monitoring mission show that a hybrid energy system is not optimal. Instead, the battery only system is the best power source to meet the energy demands of this particular mission. This result does not intuitively make sense, as it would seem that a robot which is going to require low amounts of energy for long periods of time would benefit from having a generator to supply this constant power. However, the generator does not outperform a battery within the context of this mission for a number of reasons. Within the rule-based control

model, if the system has a generator, it is always providing power to either meet the demands of the mission or to charge the other power sources. All additional power is unused and goes to heat. A generator's constant power is also its peak power. Since the generator cannot turn off in this model, a generator-only system would lose more energy to heat than it would use to meet the power demand profile. Both batteries and ultracapacitors have the ability to turn on/off and to vary the energy drawn from each. For a very small generator, small enough to meet the low level energy draw of the profile, a battery is actually a more power-dense source. While a battery's power density scales linearly with mass, a generator's power density does not. The intersection point of where a generator is more efficient than a battery occurs at power draw higher than the average draw of this particular monitoring profile.

As shown in Table 5-1, a generator's peak power and total energy are both scaled as a function of the generator's mass in this model. Both batteries and ultracapacitors have fixed peak power and total energy densities which are functions of their chemical properties. Generators in contrast have the potential to vary both by varying the size of the fuel tank or the size of the physical generator.

A battery-only system outperforms a battery and ultracapacitor hybrid system for two reasons. At very low energy demands, a battery can actually supply a total energy higher than its rated capacity, due to the C-rating effects. Ultracapacitors are always assumed to have uniform 100% efficiency. The presence of a second power source (Ultracapacitor) would require a second DC/DC converter which has both efficiency losses and nominal power draw to operate the device. The addition of the DC/DC converter coupled with the benefit of the battery's C-rating for low energy demand allow a single power source to outperform a hybrid source for the monitoring mission.

5.8.2 Robot Traversal Mission

The complete traversal mission is shown in Figure 5-44. The corresponding hybrid system results are shown in Figure 5-45 and Figure 5-46. Figure 5-45 and Figure 5-46 show the results of same hybrid system analysis from two different views.

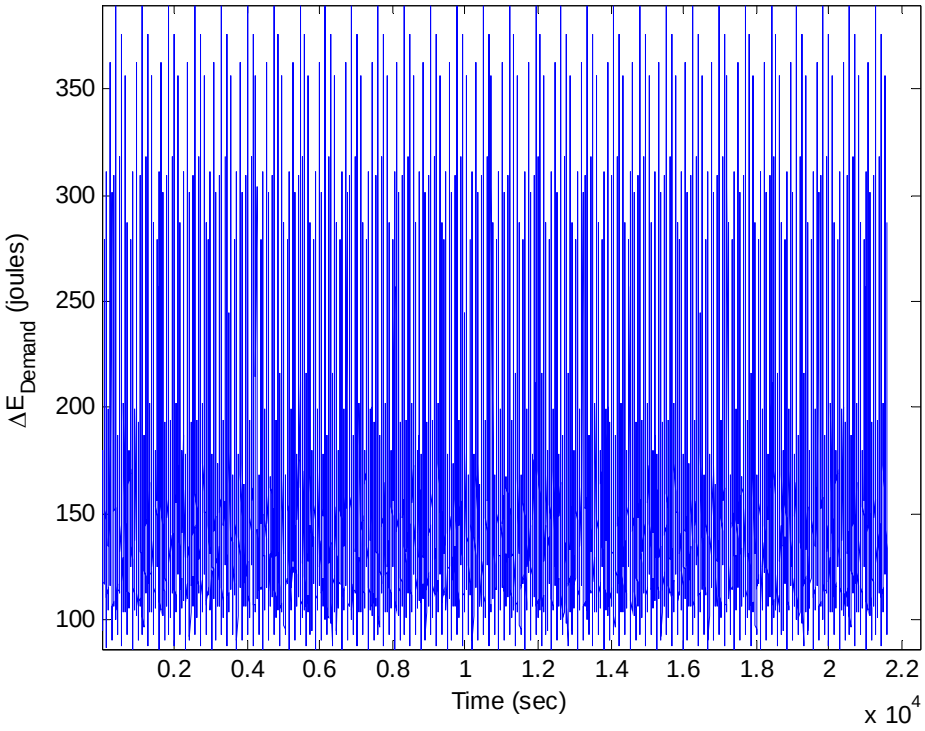

Figure 5-44. Robot Traversal Energy Demand Full Profile

The results indicate that a battery-only power source can complete the traversal mission, but a generator- or ultracapacitor-only power source cannot. The results also show that, while there are a number of hybrid systems which can complete the traversal mission, the optimal hybrid system consists of only a generator and an ultracapacitor. The hybrid system's mass is comprised of 80% generator (6.72 kg) and 20% ultracapacitor (1.68 kg). After completing this mission, the hybrid system still has 2,707,000 joules available.

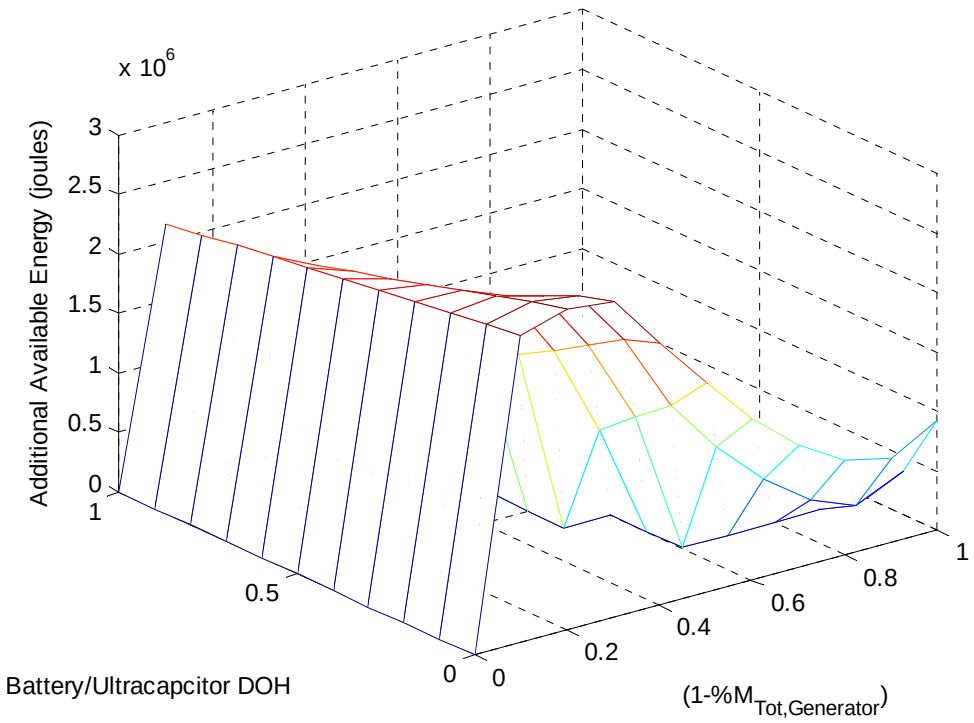

Figure 5-45. Hybrid System Effectiveness Results for Robot Traversal Mission View 1

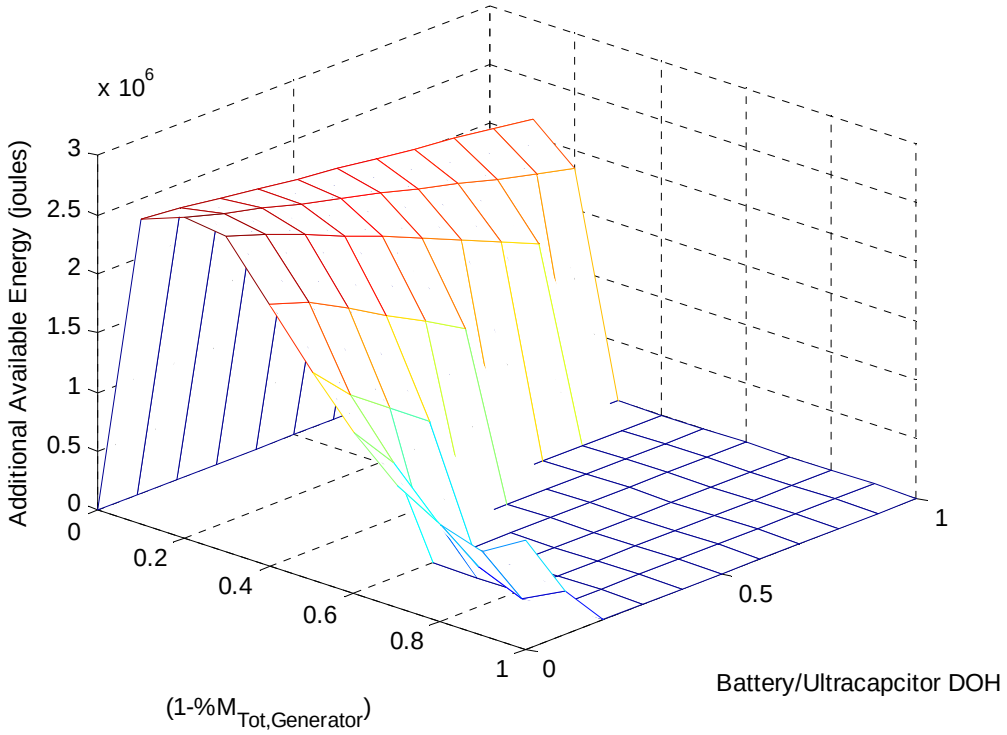

Figure 5-46. Hybrid System Effectiveness Results for Robot Traversal Mission View 2

The optimal power source for the robot traversal mission is a hybrid system comprised of a generator and ultracapacitor. Generators have much higher energy densities than batteries or ultracapacitors in the mass range of 6.72 kg. These same generators have much lower peak energy outputs. For the mass ranges being examined in this study, a generator-only system is unable to meet peak energy demands for the traversal mission. The results of this analysis show that the generator can be supplemented by an ultracapacitor to meet the peak demands of the cycle. There are a number of hybrid system compositions which can meet this mission. A battery-only power source can meet both the peak and total energy demands of the profile as well. At these mass ranges the ultracapacitor and generator can both meet the mission's demands and have an overall greater total energy available than the battery. The hybrid system outperforms the battery because generators have a greater energy density and batteries in this size range take significant efficiency losses when current is consistently drawn at a high rate.

5.8.3 Robot Climbing Mission

The complete robot climbing mission energy demand profile for 50 flights of stairs is shown in Figure 5-47. While the energy draw is in excess of 500 joules, the profile is not dominated by high energy demand. This is because the mission includes a requirement to climb back down the stairs, which requires much less power, and demands much more time. The hybrid system results are captured in Figure 5-48 and Figure 5-49.

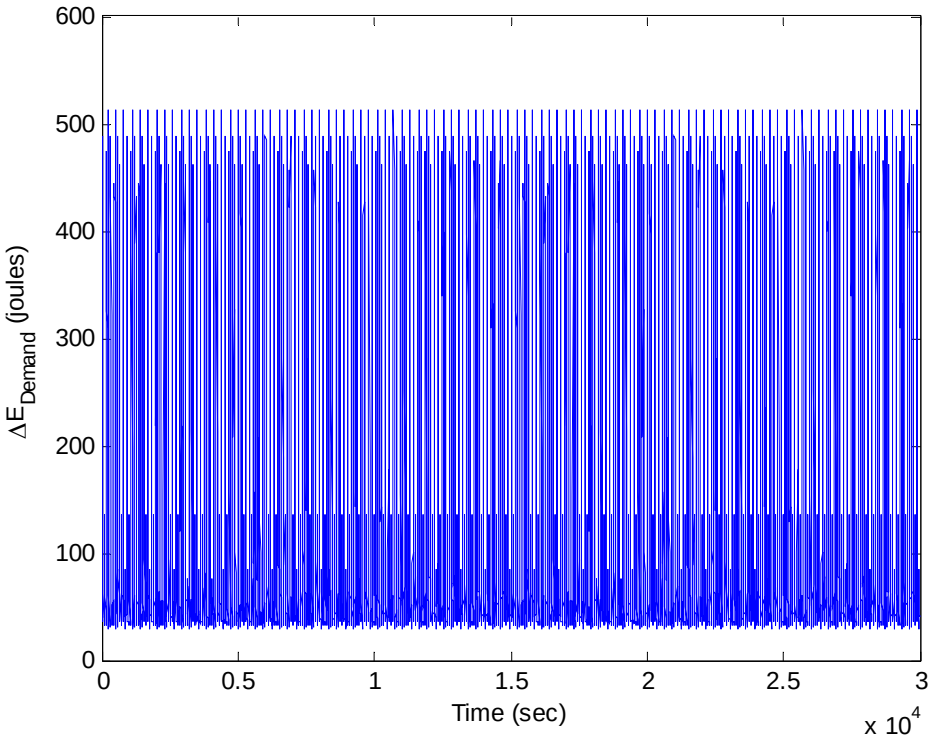

Figure 5-47. Robot Stair Climbing Energy Demand Full Profile

Figure 5-48 and Figure 5-49 show that very few robots were able to complete the mission without a generator. Interestingly, a power source consisting solely of a generator or of an ultracapacitor is also unable to complete the mission. A battery-only power source will be able to complete the mission, but does not provide an optimal solution. The optimal hybrid system for the robot climbing mission is composed of 70% generator (5.88kg) and 30% ultracapacitor

(2.52kg). The maximum remaining energy available after completing the climbing mission is 1,597,000 joules.

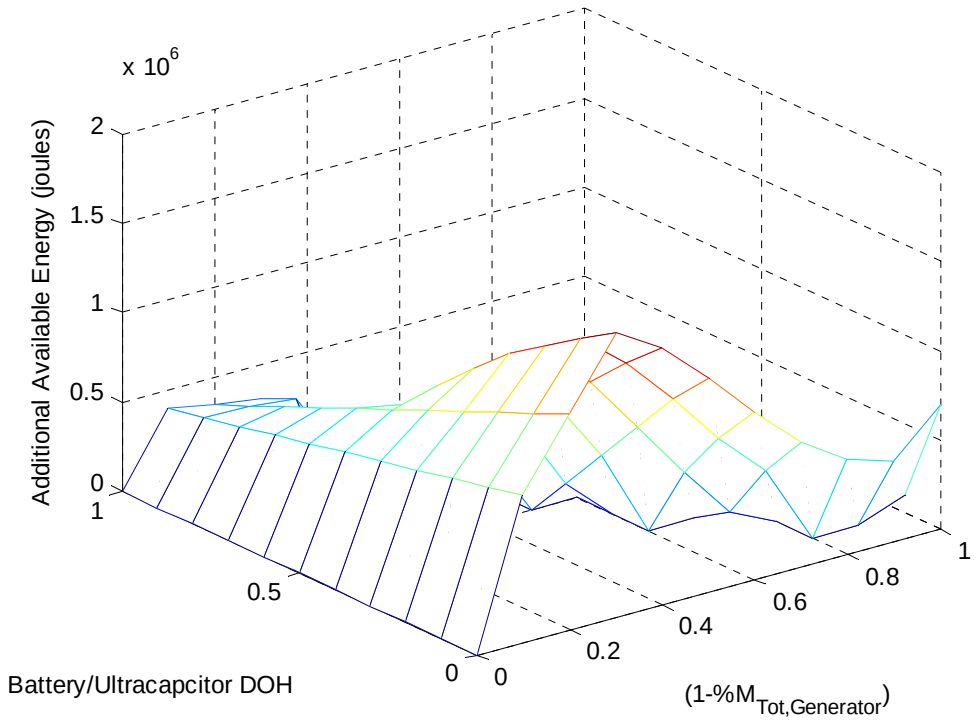

Figure 5-48. Hybrid System Effectiveness Results for Robot Climbing Mission

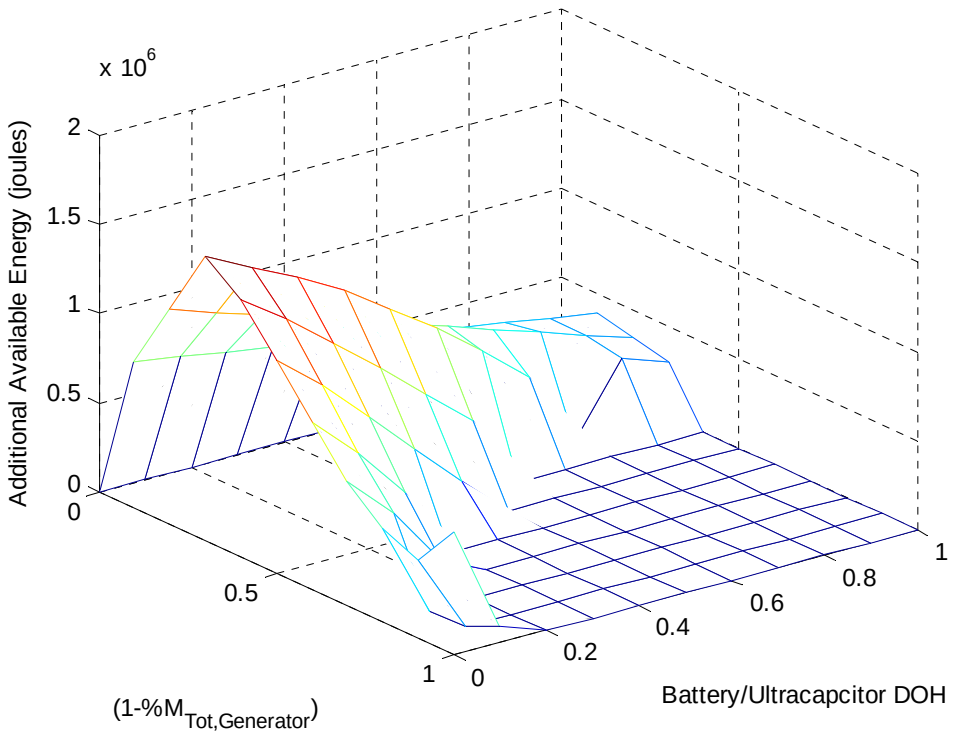

Figure 5-49. Hybrid System Effectiveness Results for Robot Climbing Mission

The hybrid system which best meets the demand of the climbing mission is a hybrid system composed of a generator and ultracapacitor. The climbing mission power profile includes higher energy spikes for longer durations than the traversal mission. The lower energy demand portions of climbing mission also are greater than the low energy demands of the traversal mission. A larger ultracapacitor is therefore needed to meet the higher more prolonged energy demands of the profile. Based on how each power source is scaled, this makes intuitive sense. The optimal hybrid system has a 10% larger ultracapacitor for the climbing mission than the traversal mission. A battery-only system does meet all of the required demands for both missions, but is outperformed on a total additional available energy basis. The rule-based controller MATLAB® script written for a generator, battery and ultracapacitor system used for this analysis can be found in Appendix A.5.7.

5.9 Conclusion

Dynamic Programming serves as a tool to test hybrid systems feasibility and to measure the success of different sub optimal control strategies. Performing this comparison between DP and a rule-based control strategy shows that, within the context of this hybrid system and using a power allocation strategy, the rule-based control strategy performs well enough to be used to examine a wide variety of hybrid topologies. This rule-based controller is validated against Dynamic Programming for a battery-only power source as well as a battery and ultracapacitor hybrid system.

A third power source is then added to rule-based controller. The mass of this three power source hybrid system is fixed while the percentage of the power sources' mass allocated to each power source is varied. Varied hybrid compositions are then evaluated over a three power profiles: monitoring, traversal and climbing. Each mission has a varied and distinct power profile. For a robot which has interchangeable power sources of equivalent mass, the hybrid system composition will change for each mission. While a single hybrid system composition would be able to complete all three missions, no one system yields an optimal solution for more than one mission. Which hybrid system is optimal is directly correlated to the actions being performed by the robot and the corresponding power requirements.

5.10 References

[1] A. Kimura, T. Abe, and S. Sasaki, "Drive force control of a parallel-series hybrid system." in Society of Automotive Engineers of Japan, vol. 20, pp. 337-341, 1999.

[2] D. Rizoulis, J. Burl, and J. Beard, "Control Strategies for a Series-Parallel Hybrid Electric Vehicle." in *Society of Automotive Engineers World Congress*, 2001-01-1354.

[3] L.P. Jarvis, P.J. Cygan, and M.P. Roberts. "Hybrid Power Source for Manportable Applications." in *IEEE AESS Systems Magazine*, vol. 18, pp. 13-16, Jan. 2003.

[4] T.B. Atwater, P.J. Cygan, and F.C. Leung, "Man portable power needs of the 21st century I. Applications for the dismounted soldier. II. Enhanced capabilities through the use of hybrid power sources," in *Journal of Power Sources*, vol. 91, pp. 27–36, 2000.

[5] S.R. Vosen and J.O. Keller, "Hybrid energy storage systems for stand-alone electric power systems: optimization of system performance and cost through control strategies," in *International Journal of Hydrogen Energy*, vol. 24, pp. 1139-1156, 1999.

[6] V. Pop, H.J. Bergveld, D. Danilov, P. Regtien, and P. Notten. *Battery Measurement Systems*. Eindhoven, The Netherlands: Springer, 2008, pp. 14-17.

[7] P. Rodatz, O. Guzzella, F. Buchi, M. Bartchi, A. Tsukada, P, Dietrich, R. Kotz, G. Scherer, and A. Wokaun, "Performance and operational characteristics of a hybrid vehicle powered by fuel cells and supercapacitors," SAE International, Warrendale, PA, Rep. Ro. 2002-01-0418, 2003.

[8] Maxwell Technologies Boostcap Ultracapacitor (2009) Retrieved August 4 2009 from http://maxwell.interconnectnet.com/pdf/uc/Maxwell_UC_comparison.pdf.

[9] Peukert Number Derivation (2009) Retrieved January 20 2010 from http://www.smartgauge.co.uk/peukert2 .html.

[10] M. Fischer, M. Werber, and P. Schwartz. "Batteries: Higher energy density than gasoline?," in *Energy Policy*, vol. 37. pp. 2639-2641, Feb. 2009.

[11] M. Koot, J. Kessels, B. de Jagar, W. Heemels, and P. Bosch, "Energy Management Strategies for Vehicular Electric Power Systems." in *IEEE Transactions on Vehicular Technology*, vol. 54, pp. 771-782, 2005.

[12] I. Kolmanovsky, I. Siverguina, and B. Lygoe, "Optimization of Powertrain Operating Policy for Feasibility Assessment and Calibration: Stochastic Dynamic Programming Approach." in the *Proceedings of the American Controls Conference*, 2002, pp. 1425-1430.

[13] A. Brahma, Y. Guezennec, and G. Rizzoni, "Optimal Energy Management in Series Hybrid Electric Vehicles." in the *Proceedings of the American Controls Conference*, 2000, pp. 60-64.

[14] M.P. O'Keefe and T. Markel, "Dynamic Programming Applied to Investigate Energy Management Strategies for a Plug-in HEV." in *The International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium and Exhibition*, 2006, pp. 1-12.

[15] D. Karbowski, C. Haliburton, and A. Rousseau, "Impact of Component Size on Plug-In Hybrid Vehicle Energy Consumption Using Global Optimization." in *International Electric Vehicle Symposium*, 2007.

[16] A. Vahidi, A. Stefanopoulou, and H. Peng, "Current Management in a Hybrid Fuel Cell Power System: A Model-Predictive Control Approach." in *IEEE Transactions on Control Systems Technology*, pp. 1047-1057, 2006.

[17] N. Jalil, N.A. Kheir, and M. Salman, "A rule–based energy management strategy for a series hybrid vehicle," in the *Proceedings of the American Control Conference*, 1997, pp. 689-693.

[18] P.B. Sharer, A. Rousseau, D. Karbowski, and S. Pagerit, "Plug-in Hybrid Electric Vehicle Control Strategy: Comparison between EV and Charge-Depleting Options." in *Society of Automotive Engineers World Congress*, 2008-01-0460.

[19] M. Back, M. Simons, F. Kirschaum, and V. Krebs, "Predictive control of drivetrains," in the *Proceedings of the IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[20] Q. Gong, Y. Li, and Z. Peng, "Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles." in *IEEE Transactions on Vehicular Technology*, pp. 3393-3401, 2008.

[21] B.M. Baumann, G. Washington, B.C. Glenn, and G. Rizzoni, "Mechatronic design and control of hybrid electric vehicles," in *IEEE Transactions on Mechatronics*, vol. 5. No. 1, pp.58-72, Mar. 2000.

[22] N.J. Schouten, M. A. Salman, and N.A. Kheif, "Fuzzy logic control for parallel hybrid vehicles," in *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 460-468, May 2002.

[23] R. Langari and J.-S. Wong, "Intelligent energy management agent for a parallel hybrid vehicle- Part I: System architecture and design of the driving situation identification process," in *IEEE Transactions on Vehicle Technology*, vol. 54, no. 3, pp. 925-934, May 2005.

[24] J.-S. Won and R. Langari, "Intelligent energy management agent for a parallel hybrid vehicle- Part II: Torque distribution, charge sustenance strategies, and performance results," in *IEEE Transactions on Vehicle Technology*, vol 54, no. 3. pp. 935-953, May 2005.

[25] E.D. Tate and S.P. Boyd, "Finding ultimate limits of performance for hybrid electric vehicles," in the *Proceedings of the SAE Future Transportation Technology Conference*, Aug. 2000, SAE Paper 2000-01-3099.

[26] S. Delprat, J. Lauber, T.M. Guerra, and J. Rimaux, "Control of a parallel hybrid powertrain: optimal control," in *IEEE Transactions on Vehicle Technology*, vol. 53, no. 3, pp. 872-881, May 2004.

[27] M. Back, M. Simons, F. Kirschaum, and V. Krebs, "Predictive control of drivetrains," in the *Proceedings of the IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[28] C.-C. Lin, H. Peng, J.W. Grizzle, and J.-M. Kang, "Power management strategy for a parallel hybrid electric truck," in *IEEE Transactions on Control System Technology*, vol. 11, no. 6, pp. 839-849, Nov. 2003.

[29] T. Hofman and R. van Druten, "Energy analysis of hybrid vehicle powertrains," in the *Proceedings of the IEEE International Symposium on Vehicle Power Propulsion*, Paris, France, Oct. 2004.

[30] B. K. Powell, K.E. Bailey, and S.R. Cikanek, "Dynamic Modeling and Control of Hybrid Electric Vehicle Powertrain Systems." in *IEEE Transaction on Control Systems*, vol. 18, pp. 17-33, 1998.

[31] G. Rizzoni, L. Guzzella, and B. Baumann, "Unified Modeling of Hybrid Electric Vehicle Drivetrains." in *IEEE Transactions on Mechatronics*, vol. 4, pp. 246-257, Sept. 1999.

[32] D. Sinoquet, G. Rousseau, and Y. Milhau, "Design optimization and optimal control for hybrid vehicles." in *Optimization and Engineering*, 2009.

[33] M.-J. Kim and H. Peng. "Power management and design optimization of fuel cell/battery hybrid vehicles." in *Journal of Power Sources*, vol. 165, pp. 819-832, 2007.

[34] B. Wu, C.-C. Lin, Z. Filipi, H. Peng, and D. Assanis, "Optimal Power Management for a Hyraulic Hybrid Delivery Truck." in the *Proceedings of the Advanced Vehicle Control Conference*, 2002.

[35] S.J. Moura, D.S. Callaway, H.K. Fathy, and J.L. Stein, "Impact of Battery Sizing on Stochastic Optimal Power Management in Plug-in Hybrid Electric Vehicles." in the *Proceedings of the IEEE Conference on Vehicular Electronics and Safety*, 2008, pp. 96-102.

[36] D.F. Opila, X. Wang, R. McGee, J.A. Cook, and J.W. Grizzle, "Performance Comparison of Hybrid Vehicle Energy Management Controllers on Real- World Driving Cycle Data." in the *Proceedings of the American Control Conference,* 2009, pp. 4618-4625.

[37] C. Lin, H. Peng and J.Grizzle, "A Stochastic Control Strategy for Hybrid Electric Vehicles," in the *Proceedings of the American Control Conference*, Boston, 2004, MA, 06/30-07/02.

[38] Z. Sun and J. Reif. "On Energy-minimizing Path on Terrains for a Mobile Robot," in *IEEE International Conference on Robotics & Automation*, 2003, pp. 3782-3788.

[39] Y. Mei, Y.H. Lu, Y.C. Hu, and C.S. G. Lee, "Energy-Efficient Motion Planning for Mobile Robots," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 4344-4349.

[40] BB-2590/U Rechargeable Lithium-Ion Battery (2007). Retrieved August 4 2009 from http://www.tacticaleng.com/mc_images/category/5/bren-tronicsbb-2590u.pdf.

[41] E. D. Tate Jr., J.W. Grizzle and H. Peng, "Shortest Path Stochastic Control for Hybrid Electric Vehicles," in *International Journal of Robust and Nonlinear Control*, Vol.18, Issue 14, Sep 2008, pp.1409-1429.

[42] S.R. Eddy, "What is dynamic programming?" in Natural Biotechnology, vol. 22, July 2004 pp. 909-910.

[43] R.E. Bellman, *Eye of the Hurricane: An Autobiography*. World Scientific, Singapore, 1984.

[44] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, *Algorithms*. McGraw-Hill, 2006.

[45] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms (2nd edition)*. MIT Press & McGaw-Hill 2001.

Chapter 6

Conclusions

This section summarizes key conclusions from each chapter and discusses the relative shortcomings of each chapter's modeling efforts. Comparisons between models and experimental data are found at the end of each chapter. However, each section describing robot system performance does not include every variation of a capability the robot can perform to achieve its mission. The focus of this thesis is on the conceptual design of the platform using generalized assumptions of physical composition of the robot system. In nearly all examples in this thesis, the performance is optimized only considering a single criterion. The summary of results and assumptions will suggest methods to improve modeling which conclude this chapter and thesis.

6.1 Geometric Considerations for Ground Mobile Robots

Using quasi-static analysis and geometric considerations, a wide variety of ground robot capabilities were successfully predicted. It is possible to predict the robot's ability to climb, maneuver and traverse within a 20% accuracy with only the bulk properties of the robot specified. Not only are the step, stairs and hill climbing capabilities modeled accurately, but their failure modes are also correctly predicted. While a number of these capabilities predictions are modeled using experimental data from one platform, the rules generated accurately extend to robots of all sizes.

While the predominant failure modes of step, stair and hill climbing were modeled with reasonable accurately, a number of conditions are not captured. All of these climbing scenarios and failure modes assume that the robot approaches the object being climbed or traverse uniformly straight. However, it is possible for a robot to be able to climb some hills or stairs only if it approaches these features at an angle. Additional geometric considerations could be included into the modeling effort to encompass such angled approaches to climbing and traversal.

There are two tread features within skid steer tracked robots which are not modeled. The climbing models used in this thesis assume a completely flat and uniform contact patch on all of the treads and wheels of the robots. This, however, is not the case for any of the robots used during testing. The RONS and the Talon (the latter shown in Figure 6-1) utilize rubber treads with protruding rubber features to allow the robot to climb much higher and pull more weight than the mathematical models predict. A flat contact patch on a tread has a fundamental limitation based upon a coefficient of friction, while treads with additional features gain an advantage through the use of soft material or in situations where the track catches a step or other surface feature for added forward traction.

Figure 6-1. Talon Tread Configuration

The other feature which is not currently in the capabilities modeling is the use of articulating treads, often referred to as “flippers,” shown deployed in Figure 6-2. These are used to increase the capabilities of a robot by extending its effective wheelbase. The RONS robot is able to double both its wheelbase and its climbing capabilities through use of these articulating treads. To improve the robot model, a mass penalty associated with adding flippers and increased wheelbase could be used to capture additional robot designs.

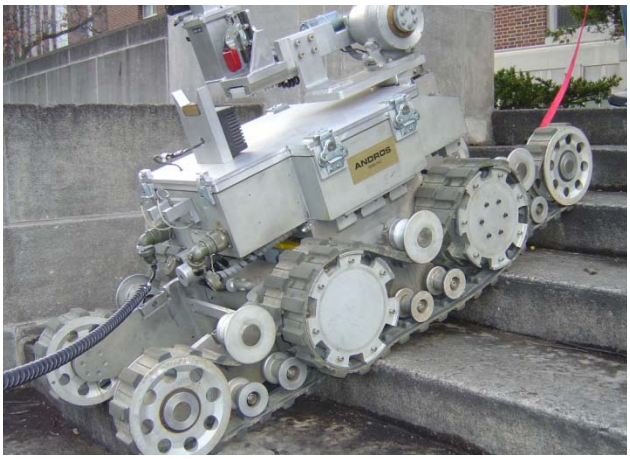

Figure 6-2. RONS Deployed Articulating Track

6.2 Allometric Design Principles for Ground Robot Powertrains

The results from Chapter 3 show that while there is some error exists in the subsystem scaling predictions, the models accurately predict the bulk system properties which correspond to the endurance cruising distance of each robot platform. Both physics-based rules and experimental data are shown to scale accurately within a wide variety of mass ranges.

All of the component-level models are either derived from physics, based on vendor data, or based on experimental data collection. To increase the accuracy of the component-level modeling, the option to be powered by brushless DC motors should be included. Due to the lack of available information from vendors, brushless motors and their performance tradeoffs were not included in the model.

The limitations of the computational robot designs as viewed in ATSV are a result of how each design is derived. The robot's size is computed as the summation of key dimensions of internal components. Some of these components, in particular payload mass and volume, are varied though random or Pareto sampling. Some of the designed robots are therefore larger than each fundamentally has to be. This makes it difficult to compare the relative performance tradeoffs between each robot, independent of payload. Some of the tradeoffs between size, mass

and power may not be as evident with the addition of payloads which have no added performance benefit. Before the effectiveness of each robot can be more accurately assessed, additional measurements need to be included to show a relative increase in capability related to the larger payload.

6.3 System Level Robot Modeling

Through manipulation of the model inputs, it is possible to develop a conceptual robot model which possesses the same physical dimensions and capabilities as the physical robot evaluated. These same physical robots generally fall within a population of randomly sampled conceptual robot designs. This shows that the system level robot performance predictions are reasonable. This also shows the ability of the modeling effort to yield designs which outperform currently existing robot platforms.

In this work all robot models are built using the same packaging configurations, which constrain the drive motors, batteries and other internal components to the same position within each robot. Some packaging of components around others – for example nesting battery compartments inside the tread cavity of a robot – may offer significant space savings. Each existing robot used for testing has batteries, controllers and other varied components oriented differently and in some cases in different locations within the robot. Adding flexibility could be added to the model to consider different types of component packaging, with the potential to yield more unique and optimal designs.

6.4 Optimal Hybrid Power Component Selection for Mobile Ground Robots

Dynamic Programming has been shown to yield an optimal solution to both control and scaling of hybrid components. However, it is still not a real-time controller and therefore cannot be used to actively allocate power from each of the hybrid systems within a robot. The research

presented in Chapter 5 shows that a rule-based controller that is designed in an ad-hoc manner works nearly as well as an optimized Dynamic Programming controller. There are real-time DP implementations, and a natural next step might be to implement these more sophisticated real-time control strategies, known as Stochastic Dynamic Programming. This will lead to greater precision for a wider range of power profiles because each will use statistics on a drive cycle to allocate power rather than a rule-based controller using the properties of the power source.

The results of the rule-based Controller can be improved to more accurately show the benefits of hybrid systems. A key improvement to the control strategy would be to include the ability to turn the generator on and off between a given time interval. The DC/DC converters used for this modeling effort are also applied universally for hybrid components of all sizes. A more accurate representation of a hybrid power source connected to a DC/DC converter would be to scale each converter to fit the size of the corresponding power source. Using converters that are too large for their respective components or loads will tend to bias hybrid system design results away from hybrid topologies when there are small power loads. One can easily fix this bias by including estimates of scaling effects for DC/DC converters, similar to what has been done on motor amplifiers in earlier sections of the thesis.

The hybrid system study should also be extended to consider a wider range of power sources for a variety of robots, at a wider range of size scales. Many of the possible benefits of using hybrid technology may be seen at much smaller power systems. A power source for a 15kg robot is in the range of approximately 2.8kg. At these mass regions, the benefit of generators is much less substantial and current limitations of batteries become more critical to completing a mission. There may be fuel cells or other technologies that do not have such limitations.

This goal of this work has been to develop capabilities and demonstrations of how a robot can be designed through the use of model-based exploration of the design space. Dynamic Programming has shown that mobile ground robots with hybrid power systems are able to surpass their counterparts with a single power source. The optimal hybrid power topology also changes with the mission the robot is undertaking. In this context, hybrid power systems allow the robot to take advantage of each of the energy types' advantages, such as the ability to meet high power demands or supply high energy density for low power demands. The modeling

efforts for the overall designs and hybrid power sources for robots can continue to be refined over time. Doing so will allow designers to explore new designs for robots while increasing the fidelity existing models. It is hopefully clear that this thesis has made significant progress in the effort to generate computational ground robot designs that enable principled design of physical robots.

Appendix

A.5.1 Dynamic Programming Battery Only MATLAB® Script

```
%% Drew Logan
%% HyPER Project
%% DYNAMIC PROGRAMMING: Battery Only System
%% Updated: 2010_09_01
    close all;
    clear all;
    clc;


%% 0.0 CONSTANTS
% 0.1 Power System
        num_BB2590_series=1;
        num_BB2590_parallel=2;
        num_Ultracap_series=1;
        num_Ultracap_parallel=1;
% 0.1 Tunable Variables
        W1=1;
        W2=1;
        m=3.27768;
        b=0.92694;
% 0.2 Constants (BB2590 Battery)
        N= 1.1;                                     % Peukert Number
        R= 20;                                      % hrs
        C= 7.2*num_BB2590_series;                  % Amp-hr
        V= 28.8*num_BB2590_parallel;                % V
        del_t_sec=1;                                % sec
        del_t_hrs=del_t_sec/3600;                   % hrs
        Enom= C*V*3600;    % joules
% 0.4 Constants Derived for Simplicity
    k_B= V*(C/R)*(del_t_sec)*((R)^(1/N))*((del_t_hrs)^(-1/N))*((Enom)^(-
1/N));


%% 1.0 DEFINE POWER PROFILE
% 1.1 Load E_Demand File
    cd('!DATA');
    load('E_Demand1');
    total_E_Demand=(E_Demand)';
    clear E_Demand
% 1.2 Crop Data
    number_of_seconds_time_steps=max(size(total_E_Demand));
    for ZZ=1:number_of_seconds_time_steps
       E_Demand(:,ZZ)=total_E_Demand(:,ZZ);
       ZZ=ZZ+1;
    end
    E_Demand_Matrix_Size=max(find(E_Demand));
% 1.3 E_Demand as Cumulative
    for index_E_Demand=1:number_of_seconds_time_steps;
```

```
        if index_E_Demand==1;
            E_Demand_Cumulative(1,index_E_Demand)=E_Demand(1,index_E_Demand);
        else

E_Demand_Cumulative(1,index_E_Demand)=E_Demand_Cumulative(1,index_E_Demand-
1)+E_Demand(1,index_E_Demand);

        end
    end


%% 2.0 DEFINE SIZE OF MATRIX AND NUMBER OF STEPS WITHIN MIN MAX VALUES
% 2.1 Max Values for both CAPS AND BATTERIES
    ENERGY_BB2590max= Enom;                     %6Batt*14.4A-hr*12V*3600
    ENERGY_ULTRACAP= 15*80*num_Ultracap_series; %15V* 80amp max surge 1 sec
% 2.3 b,c are vectors of the increments
        nb= ENERGY_BB2590max;
        mc= nb;
        bb=[0:ENERGY_BB2590max/nb:ENERGY_BB2590max]';
        cc=[0:ENERGY_ULTRACAP/mc:ENERGY_ULTRACAP]';
        max_b=max(find(bb));
        max_c=max(find(cc));


%% 3.0 DEFINE FINAL MATRIX J_FINAL VALUES
    J_Fc=cc.*10;
    J_Fb=bb.*0;


%% 4.0 SET UP LOOP K
% 4.1 Compute Various Values for the Cap and Battery
    EC=J_Fc(:,1);
    EB=J_Fb(:,1);
% 4.2 Set up matrix of possible values
    EHS(:,1)=EC;
    EHS(:,2)=EB;


%% 5.0 RUN LOOP TO DETERMINE STEP FROM K TO K-1
% 5.1 CUMULATIVE PENALTY FUNCTION MEETING DEMAND
        J_F=0;
    index=E_Demand_Matrix_Size;
    for EE=1:E_Demand_Matrix_Size;
        disp(index)
            del_E_battery(1,index)=((E_Demand(index)+m)/k_B_BBC)^N;
            E_Bus_Battery(1,index)=-m+k_B_BBC*(del_E_battery(1,index)^(1/N));
            J_min_noncum(1,index)= (E_Demand(index)-
del_E_battery(1,index))^2;
            J(index)= del_E_battery(1,index);
        index=index-1;
    end

% Display Data Forward in Time
```

```
    E_Demand=E_Demand(end:-1:1);
    del_E_battery=del_E_battery(end:-1:1);


%% 6.0 CALCULATE BATTERY ENERGY REQUIRED
    ztotal_E_required=sum(del_E_battery)
    ztotal_E_talon=sum(E_Demand)
for EE=1:E_Demand_Matrix_Size;
    if EE==1
        battery_energy_state(EE)=ztotal_E_required-del_E_battery(EE);
    else
        battery_energy_state(EE)=battery_energy_state(EE-1)-
del_E_battery(EE);
    end
end


%% 7.0 BATTERY CAP Plots for Thesis
figure(100)
            subplot(2,1,1)
                plot(E_Demand(end:-1:1),'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(E_Bus_Battery,'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y Bus');
                xlabel('Time (sec)');

figure(200)
            subplot(2,1,1)
                plot(E_Demand(end:-1:1),'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(del_E_battery(end:-1:1),'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y SOC');
                xlabel('Time (sec)');
figure(300)
            subplot(2,1,1)
                plot(E_Demand_Cumulative,'k'); hold on;
                ylabel('E_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(battery_energy_state,'b'); hold on;
                ylabel('SOC_B_a_t_t_e_r_y');
                axis([0 inf 0 inf])
                xlabel('Time (sec)');
figure(999)
            subplot(2,1,1)
                plot(E_Demand(end:-1:1),'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(J,'b'); hold on;
                ylabel('J, Penalty Function');
                xlabel('Time (sec)');
```

```
%% 8.0 DISP RESULTS
Energy_Remaining= Enom - ztotal_E_required
Heat_Losses=ztotal_E_required-ztotal_E_talon
Percent_Used=1-(Enom - ztotal_E_required)/Enom
```

A.5.2 Dynamic Programming Battery and Ultracapacitor System Algorithm Code MATLAB® Script

```
%% Drew Logan
%% HyPER Project
%% DYNAMIC PROGRAMMING: Battery and Ultracapacitor System
%% Updated: 2010_09_01
    close all;
    clear all;
    clc;
    cd;


%% 0.0 DEFINE POWER PROFILE
% 0.1 Load Talon Profile
        load E_Demand1
% 0.2 Resample Data (currently meant to be a 1 second intervals)
% 0.3 Crop Data
% Column Vector into Row Vector
    total_E_Demand=-E_Demand';
    number_of_seconds_time_steps= 600;
% Set up Matrix
    E_Demand=total_E_Demand(:,1:number_of_seconds_time_steps);
% Find size of profile
    E_Demand_Matrix_Size=max(find(E_Demand));
% 0.5 Represent E_Demand as a cumulative process
    for index_E_Demand=1:number_of_seconds_time_steps;
        if index_E_Demand==1;
            E_Demand_Cumulative(1,index_E_Demand)=E_Demand(1,index_E_Demand);
        else

E_Demand_Cumulative(1,index_E_Demand)=E_Demand_Cumulative(1,index_E_Demand-
1)+E_Demand(1,index_E_Demand);
        end
    end


%% 1.0 CONSTANTS
% 1.1 Power System
        num_BB2590_series=1;
        num_BB2590_parallel=1;
        num_Ultracap_series=1;
        num_Ultracap_parallel=1;
% 1.1 Tunable Variables
```

```
        W1=1;
        W2=1;
        m=3.27768;
        b=0.92694;
% 1.2 Constants (BB2590 Battery)
        N= 1.1;                                     % Peukert Number
        R= 20;                                      % hrs
        C= 7.2*num_BB2590_parallel;                 % Amp-hr
        V= 28.8*num_BB2590_series;                 % V
        del_t_sec=1;                                % sec
        del_t_hrs=del_t_sec/3600;                   % hrs
        Enom= C*V*3600;                             % joules
% 1.4 Constants Derived for Simplicity
    k_B= V*(C/R)*(del_t_sec)*((R)^(1/N))*((del_t_hrs)^(-1/N))*((Enom)^(-
1/N));


%% 2.0 DEFINE SIZE OF MATRIX AND NUMBER OF STEPS WITHIN MIN MAX VALUES
% 2.1 Max Values for both CAPS AND BATTERIES
    ENERGY_BB2590max= Enom;                      %6Batt*14.4A-hr*12V*3600
    ENERGY_ULTRACAP= 15*80*num_Ultracap_series; %15V* 80amp max surge 1 sec

% 2.3 b,c are vectors of the increments
        nb= ENERGY_ULTRACAP;
        mc= nb;
        bb=[0:ENERGY_BB2590max/nb:ENERGY_BB2590max]';
            bb=bb(end:-1:1);
        cc=[0:ENERGY_ULTRACAP/mc:ENERGY_ULTRACAP]';
            cc=cc(end:-1:1);
        max_b=max(find(bb));
        max_c=max(find(cc));


%% 3.0 DEFINE FINAL MATRIX J_FINAL VALUES
    % Use this if you want to make the Cap end the mission full. Relative
    % penalty for having anyting in the cap or battery. Intuitively we will
    % always charge deplete b/c it will give us the best solution b/c the
    % cap is more efficient so we will always want to use as much as
    % possible.
    J_N_plus_CAP=cc.*0;
    J_N_plus_BATT=bb.*0;


%% 4.0 SET UP LOOP K
% 4.1 Compute Various Values for the Cap and Battery
    EC=cc;
    EB=bb;


%% 5.0 RUN LOOP TO DETERMINE STEP FROM K TO K-1
% 5.1 CUMULATIVE PENALTY FUNCTION MEETING DEMAND
    index=E_Demand_Matrix_Size;
% Run loop through demand cycle for each second in time and for each
```

```
% variation of the ultracap change in charge.
for EE=1:E_Demand_Matrix_Size;
    disp(index)
    for CC=1:(max_c+1);
        for DD=1:(max_c+1);

% Ultracap State: Vary the change in teh state of the Ultracap.
                del_E_CapSOC(DD,:)= EC(DD)-EC(CC);          %CHANGE IN CAP
SOC
            if del_E_CapSOC(DD,:)>=0 % CHARGE CAP
                del_E_Cap_DCDC= (del_E_CapSOC(DD,:)+m)/b;
                del_E_Cap_Bus(DD,:)= del_E_Cap_DCDC;
            else % DISCHRAGE CAP
                del_E_Cap_DCDC= abs(del_E_CapSOC(DD,:));
                del_E_Cap_DCDC= del_E_Cap_DCDC*b-m;
                del_E_Cap_Bus(DD,:)= -del_E_Cap_DCDC;
            end
% Calculate the amount of energy from battery to bus to meet the Ultrac's
% change in state of charge and the demanded energy.
            E_Demand_C(DD,:)= E_Demand(index) - del_E_Cap_Bus(DD,:);
% Battery State: Calcualte the change in state of charge.
            if E_Demand_C(DD,:) >= 0 % CHARGE BATTERY (MAYBE)
                del_E_Batt_Bus= E_Demand_C(DD,:);
                del_E_Batt_DCDC= del_E_Batt_Bus*b-m;
                if del_E_Batt_DCDC >= 0 %CHARGE BATTERY
                        del_E_BattSOC(DD,:)= k_B*(del_E_Batt_DCDC^(1/N));
                else %DISCHARGE BATTERY
                    del_E_BattSOC(DD,:)= -((-del_E_Batt_DCDC/k_B)^N);
                end
            else %DISCHARGE BATTERY
                del_E_Batt_Bus= -E_Demand_C(DD,:);
                del_E_Batt_DCDC= (del_E_Batt_Bus+m)/b;
                if del_E_Batt_DCDC<=0
                    disp(del_E_Batt_DCDC)
                end
                del_E_BattSOC(DD,:)=-((del_E_Batt_DCDC/k_B)^N);
            end
% PENALTY FUNCTION, J
            J(DD,:)=(-del_E_BattSOC(DD,:) -
del_E_CapSOC(DD,:))^1+J_N_plus_BATT(DD,1);
        end
% Determine Min J Function: MIN DD (time=n+1) FOR LOCATION CC (timee=n)
            J_min=min(J);
            J_locate= find(J==J_min);
                    % Are there multiple optimal solutions
                    if length(J_locate)>1
                        J_trace= min(J_locate);
                    else
                        J_trace= find(J==J_min);
                    end
            J_min_DD(CC,index)=J_min;
% Store Answers
            J_min_DD_trace(CC,index)= J_trace;
            del_E_Cap_trace(CC,index)= del_E_CapSOC(J_trace,:);
    end
```

```
        J_N_plus_BATT(:,1)= J_min_DD(:,index);
        index=index-1;
end


%% 6.0 SAVE WORK
    save RESULTS_NIST_E_Demand_2010_09_00_2RUNS
```

A.5.3 Dynamic Programming Battery and Ultracapacitor System Process and Display MATLAB® Script

```
%% Drew Logan
%% HyPER Project
%% DYNAMIC PROGRAMMING: Battery and Ultracapacitor System
    % Evaluate best solution
%% Updated: 2010_09_01
k=0;
if k==1
    close all
    clear all
    clc
    cd;
    load RESULTS_NIST_E_Demand_2010_08_17_600
end
    keep EC J_min_DD J_min_DD_trace E_Demand E_Demand_Matrix_Size Enom m b
k_B N
    E_Demand_ANS=-E_Demand';

%% 7.0 COMPUTE CUMULATIVE DEMAND
    for HH=1:E_Demand_Matrix_Size
        if HH==1;
            E_Demand_ANS_TOT(HH,1)=E_Demand_ANS(HH,1);
        else
            E_Demand_ANS_TOT(HH,1)=E_Demand_ANS_TOT(HH-
1,1)+E_Demand_ANS(HH,1);
        end
    end

%% 8.0 TRACE MIN J FUNCTION FOR ULTRACAP AND DETERMINE CHANGE IN SOC
% Determine Initial State of Cap
    choose_int_SOC_CAP=1200;
        choose_int_SOC_CAP_LOCATE=find(EC==choose_int_SOC_CAP);
    J_Optimal=J_min_DD(choose_int_SOC_CAP_LOCATE,1);
% Determine path based on Initial State of Cap
    for HH=1:E_Demand_Matrix_Size
        if HH==1
        SOC_CapTrace(HH,1)=max(find(J_min_DD(:,1)==J_Optimal));
        else
        SOC_CapTrace(HH,1)=J_min_DD_trace((SOC_CapTrace(HH-1,1)),(HH));
        end
```

```
    end
% SOC CAP
    for HH=1:E_Demand_Matrix_Size
        SOC_Cap(HH,:)=EC(SOC_CapTrace(HH,:),:);
    end
% DEL SOC CAP
    for HH=1:E_Demand_Matrix_Size
        if HH==1
            del_SOC_Cap(HH,:)=choose_int_SOC_CAP-SOC_Cap(HH,:);
        else
            del_SOC_Cap(HH,:)= SOC_Cap(HH,:)-SOC_Cap(HH-1,:);
        end
    end



%% 8.0 KNOWN SOC OF CAP DETERMINE BATTERY CHANGE IN SOC AND SOC
E_Demand=E_Demand';
for HH=1:E_Demand_Matrix_Size
                del_E_CapSOC(HH,:)= del_SOC_Cap(HH,:); %CHANGE IN CAP SOC
            if del_E_CapSOC(HH,:)>=0 % CHARGE CAP
                del_E_Cap_DCDC= (del_E_CapSOC(HH,:)+m)/b;
                del_E_Cap_Bus(HH,:)= del_E_Cap_DCDC;
            else % DISCHRAGE CAP
                del_E_Cap_DCDCin= abs(del_E_CapSOC(HH,:));
                del_E_Cap_DCDC= del_E_Cap_DCDCin*b-m;
                del_E_Cap_Bus(HH,:)= -del_E_Cap_DCDC;
            end
            E_Demand_C(HH,:)= E_Demand(HH,:) - del_E_Cap_Bus(HH,:);
            if E_Demand_C(HH,:) >= 0 % CHARGE BATTERY (MAYBE)
                del_E_Batt_Bus= E_Demand_C(HH,:);
                del_E_Batt_DCDC= del_E_Batt_Bus*b-m;
                if del_E_Batt_DCDC >= 0 %CHARGE BATTERY
                        del_E_BattSOC(HH,:)= k_B*(del_E_Batt_DCDC^(1/N));
                else %DISCHARGE BATTERY
                    del_E_BattSOC(HH,:)= -((-del_E_Batt_DCDC/k_B)^N);
                end
            else %DISCHARGE BATTERY
                del_E_Batt_Bus= -E_Demand_C(HH,:);
                del_E_Batt_DCDC= (del_E_Batt_Bus+m)/b;
                if del_E_Batt_DCDC<=0
                    disp(del_E_Batt_DCDC);
                end
                del_E_BattSOC(HH,:)=-((del_E_Batt_DCDC/k_B)^N);
            end
            J(HH,:)=(-del_E_BattSOC(HH,:) -del_E_CapSOC(HH,:));

end

% SOC BATT
for HH=1:E_Demand_Matrix_Size
   if HH==1
       SOC_Batt(HH,:)= -sum(del_E_BattSOC);
   else
```

```
        SOC_Batt(HH,:)= SOC_Batt(HH-1,:) + del_E_BattSOC(HH-1,:);
    end
end


%% 9.0 RESULTS
Energy_Remaining= Enom -( SOC_Batt(1,1)+ choose_int_SOC_CAP)
Percent_Used= (SOC_Batt(1,1)+ choose_int_SOC_CAP)/Enom


%% 10.0 PLOT RESULTS
figure(100)
    subplot(311)
        plot(-E_Demand,'k'); hold on;
        ylabel('\DeltaE_D_e_m_a_n_d (joules)');
    subplot(312)
        plot(-E_Demand_C,'b')
        ylabel('-\DeltaE_B_a_t_t_e_r_y Bus')
    subplot(313)
        plot(-del_E_Cap_Bus,'r')
        ylabel('-\DeltaE_U_l_t_r_a_c_a_p Bus')
        xlabel('Time (sec)')
figure(200)
    subplot(311)
        plot(-E_Demand,'k'); hold on;
        ylabel('\DeltaE_D_e_m_a_n_d (joules)');
    subplot(312)
        plot(-del_E_BattSOC,'b')
        ylabel('-\DeltaE_B_a_t_t_e_r_y SOC')
    subplot(313)
        plot(-del_SOC_Cap,'r')
        ylabel('-\DeltaE_U_l_t_r_a_c_a_p SOC')
        xlabel('Time (sec)')
figure(300)
    subplot(311)
        plot(E_Demand_ANS_TOT,'k'); hold on;
        ylabel('E_D_e_m_a_n_d (joules)');
    subplot(312)
        plot(SOC_Batt,'b')
        ylabel('SOC_B_a_t_t_e_r_y')
    subplot(313)
        plot(SOC_Cap,'r')
        ylabel('SOC_U_l_t_r_a_c_a_p,')
        xlabel('Time (sec)')

figure(999)
    subplot(211)
        plot(-E_Demand,'k'); hold on;
        ylabel('\DeltaE_D_e_m_a_n_d (joules)');
    subplot(212)
        plot(J,'b')
        ylabel('J, Penalty Function')
        xlabel('Time (sec)')
disp('DP Results')
```

```
disp('Updated 9/1/2010');
```

A.5.4 Rule Based Controller MATLAB® Script (Single Total Mass with Varied Hybrid Topology)

```
%% 2010_03_26
%% HyPER
%% DREW LOGAN


%% GENERATE PLOT COMPARING MULTIPLE HYBRID SYSTEMS (2 to start)
    locate_file= 0;
    if locate_file==1
        close all;
        clear all;
        clc;


%% 1.0 Load Talon Profile
% LOAD PROFILE
        cd('!DATA NIST EXT')
        load('E_Demand15');
        E_Demand=E_Demand(1:size(E_Demand),:);
    end
    E_Demand=E_Demand';
    for index_E_Demand=1:length(E_Demand);
        if index_E_Demand==1;
            E_Demand_Cumulative(1,index_E_Demand)=E_Demand(1,index_E_Demand);
        else

E_Demand_Cumulative(1,index_E_Demand)=E_Demand_Cumulative(1,index_E_Demand-
1)+E_Demand(1,index_E_Demand);
        end
    end
% TIME INTERVAL
    deltaT=1;


%% 2.0 BATTERY AND ULTRACAP Characteristics
% 2.1 BATTERY (GENERALIZED BB2590)
        Mbatt_BB2590= 1.4;                                  %kg
        Ebatt_BB2590= 746469;                               %joules
        Emaxbatt_BB2590= 23*28.8*1;                         %25amp*28.8V*1sec
        Vbatt_BB2590=28.8;                                  %V
        N= 1.1;                                             % Peukert Number
        Rbatt=20;                                           % Amp-hr
        DensityBattery= Ebatt_BB2590/Mbatt_BB2590;          %joule/kg
        DensityBatteryMax= Emaxbatt_BB2590/Mbatt_BB2590;%joule/kg
% 2.2 ULTRACAP (GENERALIZED MAXWELL ULTRACAP)
```

```
        DensityCap= 0.0199*1000000;                     %joule/kg
        DensityCapMax= 10028;                           %joule/kg
% 2.3 GENERATOR
        % DensityGen= 404930*(x^1.3852);
        % DensityGenMax= 12.959*(x^1.3769);


%% 3.0 DC/DC Converter Parameters
% DCDC=1 if the system is using converters
% DCDC=0 if we do not include losses due to the converters
        DCDC=1;
        if DCDC==1; % YES USE DC/DC CONVERTERS
            b= 0.92694;
            m= 3.27768;
        else
            b=1;
            m=0;
        end


%% 4.0 Power Source Properties
% Initialize Values
% Generator Characteristics
Mgen= 0;
    GenMAXint= 12.959*(Mgen^1.3769);
    if Mgen<=2*m/GenMAXint
        m_gen=0;
        b_gen=1;
    else
        m_gen=m;
        b_gen=b;
    end
        DeltaEgenMAX= 12.959*(Mgen^1.3769);
        DeltaEgenMAX=DeltaEgenMAX*b_gen-m_gen;
        EgenInt= 404930*(Mgen^1.3852);
% BB2590 Battery Characteristics
Mbatt= 1*1.4;
% Mbatt=2*1.4;
    BattMAXint=Mbatt*DensityBatteryMax;
    if Mbatt<=2*m/BattMAXint;
        m_batt=0;
        b_batt=1;
    else
        m_batt=m;
        b_batt=b;
    end
        DeltaEbattMAX= Mbatt*DensityBatteryMax;
            DeltaEbattMAX=DeltaEbattMAX*b_batt-m_batt;
            EbattInt= Mbatt*DensityBattery;
            Cbatt=EbattInt/(Vbatt_BB2590*3600);
            kbatt=
(Vbatt_BB2590)*(Cbatt/Rbatt)*(Rbatt^(1/N))*((deltaT/3600)^(-
1/N))*(EbattInt^(-1/N))*(deltaT);
            if isnan(kbatt)
```

```
                kbatt=1;
                disp('BB2590 ERROR')
            end
% Ultracap Characteristics
% Mcap= 0;
Mcap= 0.0603;
    CapMAXint=Mcap*DensityCapMax;
    if Mcap<=2*m/CapMAXint
        m_cap=0;
        b_cap=1;
    else
        m_cap=m;
        b_cap=b;
    end
            DeltaEcapMAX= Mcap*DensityCapMax;
                DeltaEcapMAX= DeltaEcapMAX*b_cap-m_cap;
            EcapInt= Mcap*DensityCap;
Mcheck= Mgen+Mbatt+Mcap;


%% 5.0 RUN PROFILE
    deltaHEAT=0;
    flag=1;
for EE=1:length(E_Demand);
    disp(EE)
    T(:,EE)=EE;
% STEP 1: GENERATOR MEETS DEMAND- CHARGE BATTERY AND CAP
                        deltaEgen(:,EE)=DeltaEgenMAX;
                if deltaEgen(:,EE)>E_Demand(:,EE);
                    % BATTERY
                        deltaEbatt_y(1,:)= DeltaEgenMAX-E_Demand(:,EE);
                        if EE==1
                        battery_y= EbattInt-EbattInt;
                        else
                        battery_y= EbattInt-Ebatt(:,EE-1);
                        end
                            battery_y= (battery_y/kbatt)^N;
                            battery_y= (battery_y+m_batt)/b_batt;
                        deltaEbatt_y(2,:)=battery_y;
                        deltaEbatt_y(3,:)= (DeltaEbattMAX+m_batt)/b_batt;
                            deltaEbatt_y= min(deltaEbatt_y);
                            deltaEbatt(:,EE)=deltaEbatt_y;
                        deltaEbatt_DCDC= deltaEbatt(:,EE)*b_batt-m_batt;
                        if deltaEbatt_DCDC>=0 % CHARGE BATTERY
                            delta_E_battSOC(:,EE)=
kbatt*(deltaEbatt_DCDC^(1/N));
                        else % DISCHARGE BATTERY
                            deltaEbatt_DCDC= -deltaEbatt_DCDC;
                            deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(:,EE)= -deltaEbatt_DCDC;
                        end
                    % ULTRACAP
                        deltaEcap_y(1,:)= DeltaEgenMAX-E_Demand(:,EE)-
deltaEbatt(:,EE);
```

```
                        if EE==1;
                        deltaEcap_y(2,:)= (EcapInt-EcapInt+m_cap)/b_cap;
                        else
                        deltaEcap_y(2,:)= (EcapInt-Ecap(:,EE-1)+m_cap)/b_cap;
                        end
                        deltaEcap_y(3,:)= (DeltaEcapMAX+m_cap)/b_cap;
                            deltaEcap_y= min(deltaEcap_y);
                            deltaEcap(:,EE)=deltaEcap_y;
                        deltaEcap_DCDC= deltaEcap(:,EE)*b_cap-m_cap;
                        if deltaEcap_DCDC<0
                            deltaEcap_DCDC= (deltaEcap(:,EE)+m_batt)/b_batt;
                            deltaEcap_DCDC= -deltaEcap_DCDC;
                            disp('ERROR')
                        end
                            delta_E_capSOC(:,EE)= deltaEcap_DCDC;
                    % HEAT
                        deltaHEAT_STEP= deltaEgen(:,EE)-E_Demand(:,EE)-
deltaEbatt(:,EE)-deltaEcap(:,EE);
                            deltaHEAT(:,EE)= deltaHEAT_STEP;
% STEP 2: GENERATOR AND BATTERY MEETS DEMAND-CHARGE ULTRACAPACITOR-
                elseif deltaEgen(:,EE)+DeltaEbattMAX>=E_Demand(:,EE)
                    % ULTRACAP
                        deltaEcap_ny(1,:)= deltaEgen(:,EE) + DeltaEbattMAX -
E_Demand(:,EE);
                        if EE==1
                        deltaEcap_ny(2,:)= (EcapInt-EcapInt+m_cap)/b_cap;
                        else
                        deltaEcap_ny(2,:)= (EcapInt-Ecap(:,EE-
1)+m_cap)/b_cap;
                        end
                        deltaEcap_ny(3,:)= (DeltaEcapMAX+m_cap)/b_cap;
                            deltaEcap_ny= min(deltaEcap_ny);
                            deltaEcap(:,EE)=deltaEcap_ny;
                        deltaEcap_DCDC= deltaEcap(:,EE)*b_cap-m_cap;
                            delta_E_capSOC(:,EE)= deltaEcap_DCDC;
                    % BATTERY
                    deltaEbatt_ny= deltaEgen(:,EE)-E_Demand(:,EE)-
deltaEcap(:,EE);
                            deltaEbatt(:,EE)= deltaEbatt_ny;
                        deltaEbatt_DCDC= -deltaEbatt(:,EE);
                        deltaEbatt_DCDC= (deltaEbatt_DCDC+m_batt)/b_batt;
                        deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(:,EE)= -deltaEbatt_DCDC;
                        deltaHEAT(:,EE)=0;
% STEP 3: DISCHARGE ALL TO MEET DEMAND-DISCHARGE ULTRACAPACITOR-
                else
                    % BATTERY
                        deltaEbatt_nn= DeltaEbattMAX;
                            deltaEbatt(:,EE)= -deltaEbatt_nn;
                        deltaEbatt_DCDC= -deltaEbatt(:,EE);
                        deltaEbatt_DCDC= (deltaEbatt_DCDC+m_batt)/b_batt;
                        deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(:,EE)= -deltaEbatt_DCDC;
                    % ULTRACAP
```

```
                        deltaEcap_nn= -E_Demand(:,EE)+deltaEgen(:,EE)-
deltaEbatt(:,EE);
                            deltaEcap(:,EE)= deltaEcap_nn;
                        deltaEcap_DCDC= -deltaEcap(:,EE);
                        deltaEcap_DCDC= (deltaEcap_DCDC+m_cap)/b_batt;
                            delta_E_capSOC(:,EE)= -deltaEcap_DCDC;
                       deltaHEAT(:,EE)=0;
                end
% STEP 4: SHOW CHANGE IN SOC
                    if EE==1
                    Egen(:,EE)= EgenInt - deltaEgen(:,EE);
                    Ebatt(:,EE)= EbattInt + delta_E_battSOC(:,EE);
                    Ecap(:,EE)= EcapInt + delta_E_capSOC(:,EE);
                    else
                    Egen(:,EE)= Egen(:,EE-1) - deltaEgen(:,EE);
                    Ebatt(:,EE)= Ebatt(:,EE-1) + delta_E_battSOC(:,EE);
                    Ecap(:,EE)= Ecap(:,EE-1) + delta_E_capSOC(:,EE);
                    end
    J(:,EE)=(-delta_E_capSOC(:,EE)-delta_E_battSOC(:,EE))^1;
% STEP 5: DISP ERRORS
                % PEAK POWER CONDITION
                if E_Demand(:,EE)>DeltaEgenMAX+DeltaEbattMAX+DeltaEcapMAX
                    disp('PEAK DEMAND FAILURE');
                    flag=0;
                    break
                end
                % TOTAL POWER CONDITION
                if E_Demand(:,EE)>DeltaEgenMAX+Ebatt(:,EE)+Ecap(:,EE)
                    disp('Total Energy Violation');
                    flag=0;
                    break
                end
                % TOTAL GENERATOR ENERGY
                if Egen(:,EE)<0
                    disp('Generator Limited');
                    flag=0;
                    break
                end
                % TOTAL BATTERY ENERGY
                if Ebatt(:,EE)<0
                    disp('Battery Limited')
                    flag=0;
                    break
                end
                % TOTAL ULTRACAP ENERGY
                if Ecap(:,EE)<0
                    disp('Ultracap Limited');
                    flag=0;
                    break
                end
end


%% 6.0 RESULTS
```

```
    zEnergyRemaining= (Egen(:,max(EE))+Ebatt(:,max(EE))+Ecap(:,max(EE)))*flag
    Percent_Used= ((EbattInt+EcapInt)-zEnergyRemaining)/(EbattInt+EcapInt)
    TotalPenalty= sum(J)


%% 7.0 DISPLAY RESULTS
    if Mgen>0 && Mbatt>0 && Mcap>0
        PowerSource=3;
    elseif Mgen==0 && Mbatt>0 && Mcap>0
        PowerSource=2;
    elseif Mgen==0 && Mbatt>0 && Mcap==0
        PowerSource=1;
    else
        PowerSource=0;
        disp('Power Source Error: Wrong Hybrid Composition to be Displayed');
    end

if PowerSource==1
figure(100)
            subplot(2,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(-deltaEbatt,'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y Bus');
                xlabel('Time (sec)');
figure(200)
            subplot(2,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(-delta_E_battSOC,'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y SOC');
                xlabel('Time (sec)');
figure(300)
            subplot(2,1,1)
                plot(E_Demand_Cumulative,'k'); hold on;
                ylabel('E_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(Ebatt,'b'); hold on;
                ylabel('SOC_B_a_t_t_e_r_y');
                xlabel('Time (sec)');
figure(999)
            subplot(2,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(J,'b'); hold on;
                ylabel('J, Penalty Function');
                xlabel('Time (sec)');
elseif PowerSource==2
figure(100)
            subplot(3,1,1)
                plot(E_Demand,'k'); hold on;
```

```
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(3,1,2)
                plot(-deltaEbatt,'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y Bus');
            subplot(3,1,3)
                plot(-deltaEcap,'r'); hold on;
                xlabel('Time (sec)');
                ylabel('-\DeltaE_U_l_t_r_a_c_a_p Bus');
figure(200)
            subplot(3,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(3,1,2)
                plot(-delta_E_battSOC,'b'); hold on;
                ylabel('-\DeltaE_B_a_t_t_e_r_y SOC');
            subplot(3,1,3)
                plot(-delta_E_capSOC,'r'); hold on;
                xlabel('Time (sec)');
                ylabel('-\DeltaE_U_l_t_r_a_c_a_pSOC');
figure(300)
            subplot(3,1,1)
                plot(E_Demand_Cumulative,'k'); hold on;
                ylabel('E_D_e_m_a_n_d (joules)');
            subplot(3,1,2)
                plot(Ebatt,'b'); hold on;
                ylabel('SOC_B_a_t_t_e_r_y');
            subplot(3,1,3)
                plot(Ecap,'r'); hold on;
                ylabel('SOC_U_l_t_r_a_c_a_p');
                xlabel('Time (sec)');
figure(999)
            subplot(2,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(J,'b'); hold on;
                ylabel('J, Penalty Function');
                xlabel('Time (sec)');

elseif PowerSource==3 || PowerSource==0
    figure(100)
        subplot(411)
            plot(E_Demand)
            ylabel('Power Profile (watts)');
        subplot(412)
            plot(deltaEgen-deltaHEAT);
            ylabel('P_G_e_n_e_r_a_t_o_r Used');
        subplot(413)
            plot(-deltaEbatt)
            ylabel('P_B_a_t_t_e_r_y Bus');
        subplot(414)
            plot(-deltaEcap)
            xlabel('Time (sec)');
            ylabel('P_U_l_t_r_a_c_a_p Bus')
    figure(200)
```

```
        subplot(411)
            plot(E_Demand)
            ylabel('Power Profile (watts)');
        subplot(412)
            plot(deltaEgen); hold on;
            plot(deltaHEAT,'r');
            ylabel('P_G_e_n_e_r_a_t_o_r');
            legend('Generator','Heat')
        subplot(413)
            plot(-delta_E_battSOC)
            ylabel('P_B_a_t_t_e_r_y SOC');
        subplot(414)
            plot(-delta_E_capSOC)
            xlabel('Time (sec)');
            ylabel('P_U_l_t_r_a_c_a_p SOC')
    figure(300)
        subplot(411)
            plot(E_Demand_Cumulative)
            ylabel('Energy Profile');
        subplot(412)
            plot(Egen)
            ylabel('E_G_e_n_e_r_a_t_o_r');
        subplot(413)
            plot(Ebatt)
            ylabel('E_B_a_t_t_e_r_y');
        subplot(414)
            plot(Ecap)
            xlabel('Time (sec)')
            ylabel('E_U_l_t_r_a_c_a_p');
    figure(999)
            subplot(2,1,1)
                plot(E_Demand,'k'); hold on;
                ylabel('\DeltaE_D_e_m_a_n_d (joules)');
            subplot(2,1,2)
                plot(J,'b'); hold on;
                ylabel('J, Penalty Function');
                xlabel('Time (sec)');
end

disp('RBM Results');
disp('Updated 2010_09_05')
```

A.5.5 Dynamic Programming and Rule Based Controller Comparison for Additional Robot Mission 1

23.3 Minute Profile

The 23.3 minute robot power profile was collected on the Talon platform. High energy spikes shown at of top of Figure A.5-1 during the beginning of the profile occur during climbing

operations. The other energy spikes throughout the rest of the profile occur during driving over both asphalt and grass. The key difference between this power profile and those discussed in Chapter 5 is number and frequency of large energy demands.

A.5.5.1 Dynamic Programming Results

The results using Dynamic Programming on a new profile can be found in Figure A.5-1 through Figure A.5-4, where results are shown for the 23.3 minute profile. The first plot in Figure A.5-1 is the demand profile for the 23.3 minute power demand profile. The two charts below the demand profile show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure again shows that the best strategy is to use the bulk of the energy from the battery, and to only supplement this with energy from the ultracapacitor during times of large energy demand. The bottom two graphs in Figure A.5-2 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. All of the same comparisons made between the 10 and 65 minute Dynamic Programming results in Chapter 5 can be made for Figure A.5-1 and Figure A.5-2.

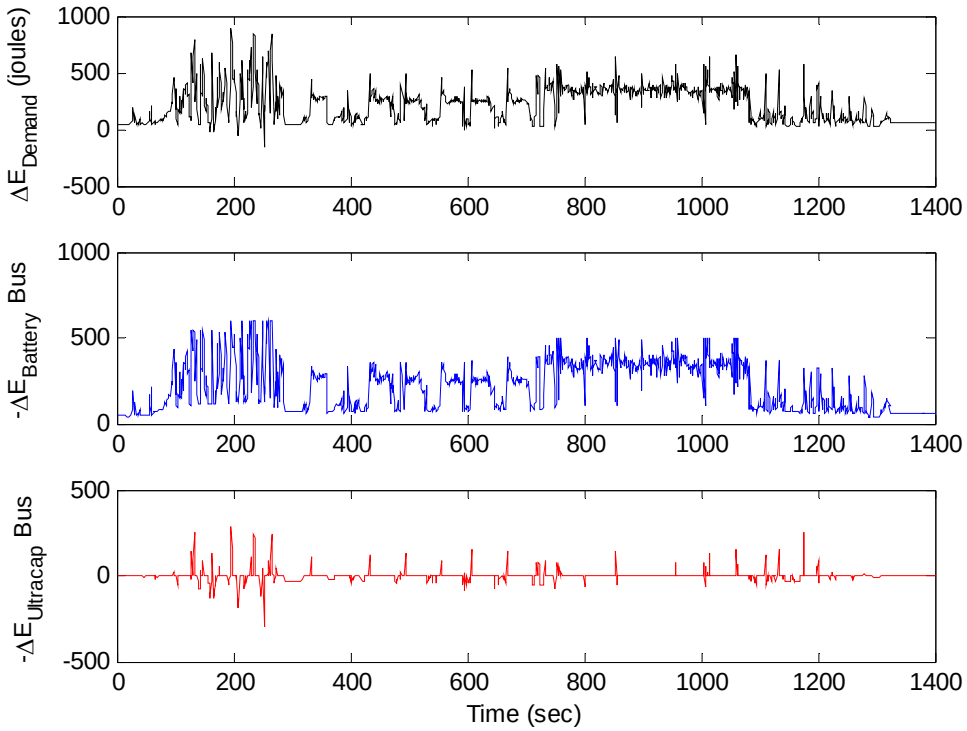

Figure A.5-1. Dynamic Programming Controller Results Power System Bus Energy: Battery and Ultracapacitor System

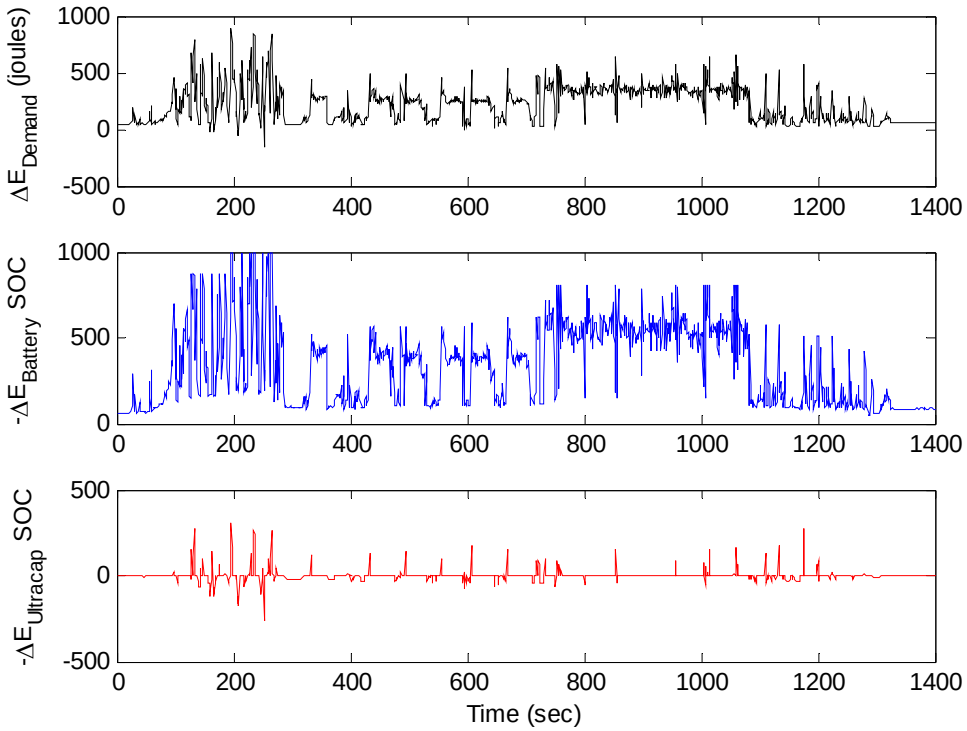

Figure A.5-2. Dynamic Programming Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Figure A.5-3 shows the discharge cycle for the battery and the charge/discharge cycle for the ultracapacitor for the longer profile. Figure A.5-4 shows the non-cumulative cost function.

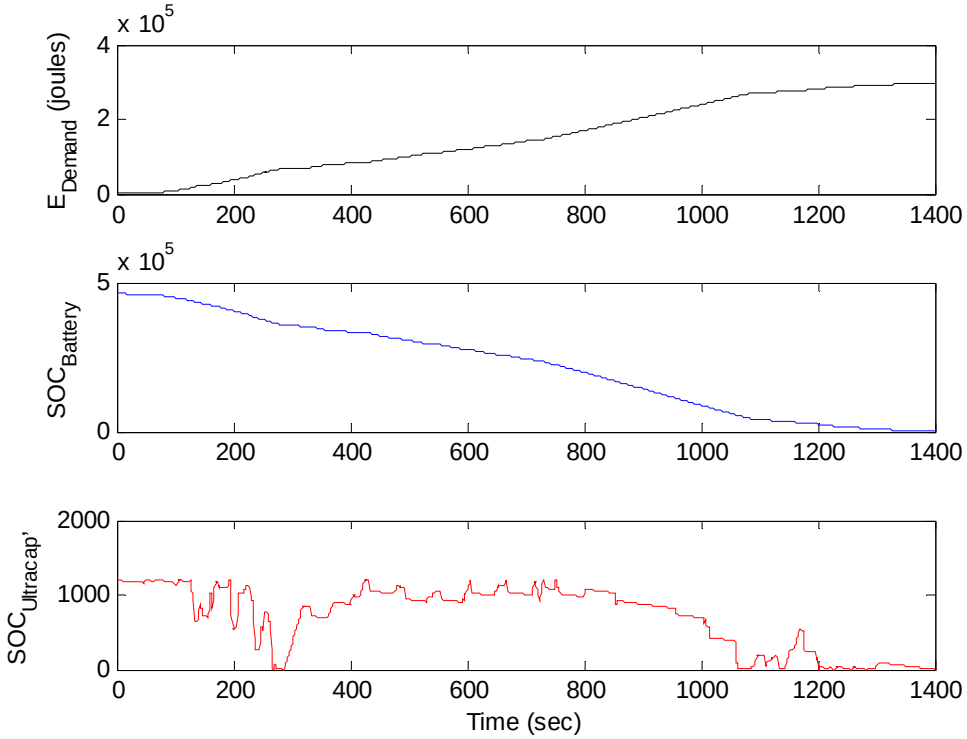

Figure A.5-3. Dynamic Programming Controller Results Power System SOC: Battery and Ultracapacitor System

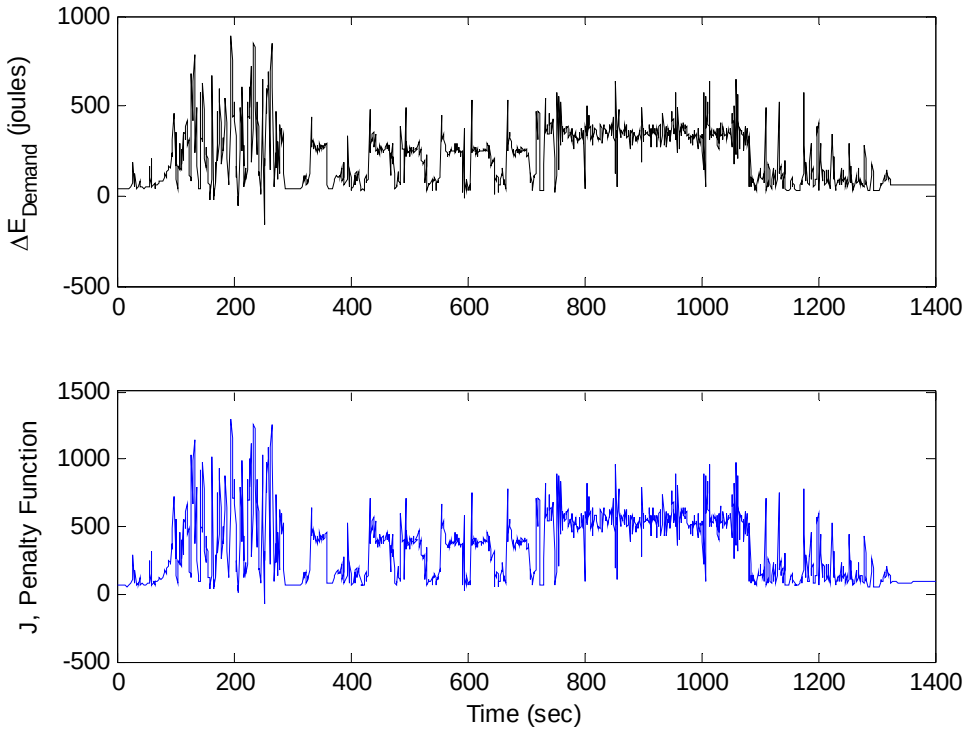

Figure A.5-4. Dynamic Programming Controller Results Cost Function: Battery and Ultracapacitor System

The total energy demanded by this power profile is 297,902.67 joules. The total energy required to meet this energy demand profile is 465,354 joules total combined from the two power sources (Table A.5-1). At the end of the 23.3 minute profile, 37.98 of the 747,696 joules available from the hybrid system remain. The percent of the total energy available used to meet the energy demand profile is therefore 62.02%. The ultracapacitor is completely discharged leaving all of the remaining energy in the battery.

Table A.5-1: Battery and Ultracapacitor Dynamic Programming 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	23.3	746496	1200	282,342	0	37.98%

A.5.5.2 Rule-Based Results

The results from the 23.3 minute profile using the rule-based method can be found in Figure A.5-5 through Figure A.5-8. Figure A.5-5 plots the 23.3 minute energy demand profile and the hybrid system responses using the rule-based controller. Figure A.5-5 describes the amount of energy on the power bus from each source while Figure A.5-6 shows the change the hybrid system component's state of charge to meet that same demand.

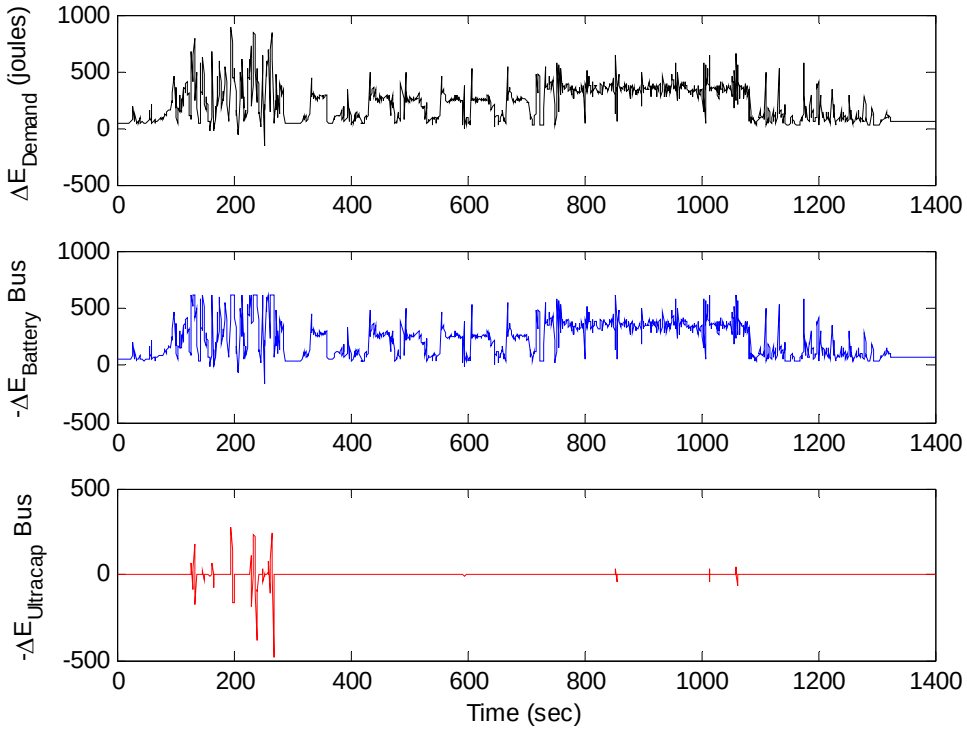

Figure A.5-5. Rule Based Controller Results Power System Bus Energy: Battery and Ultracapacitor System

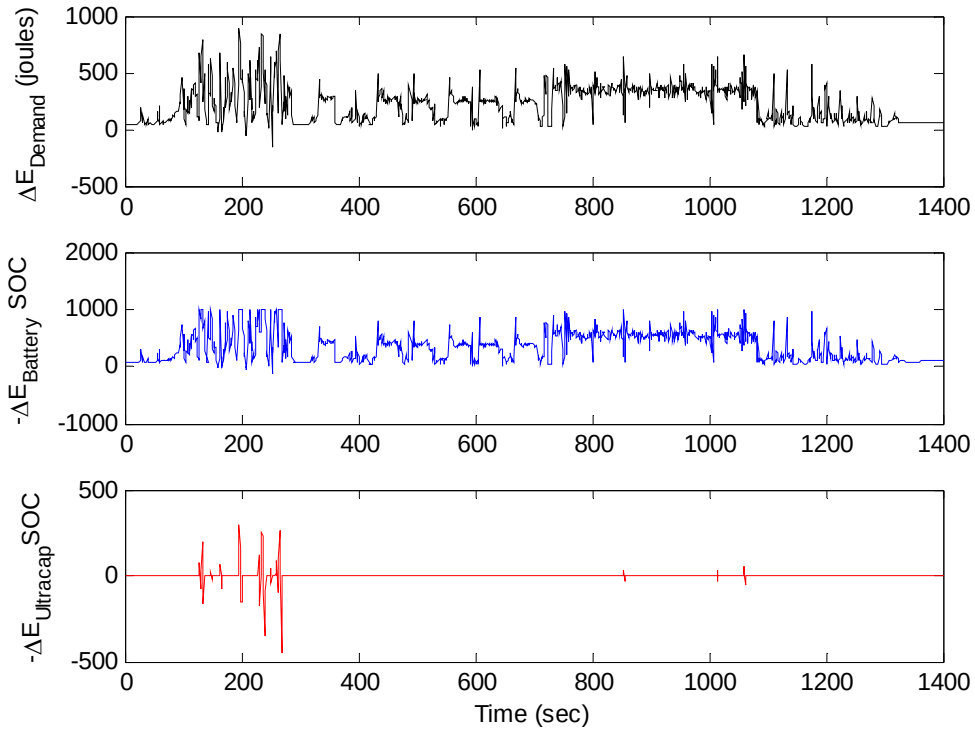

Figure A.5-6. Rule Based Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Figure A.5-7 shows the discharge cycle for the battery and the charge/discharge cycle for the ultracapacitor for the 23.3 minute profile. The battery consistently discharges while the ultracapacitor remains consistently at its peak capacity. Figure A.5-8 shows a similar trend with regard to the non-cumulative cost function.

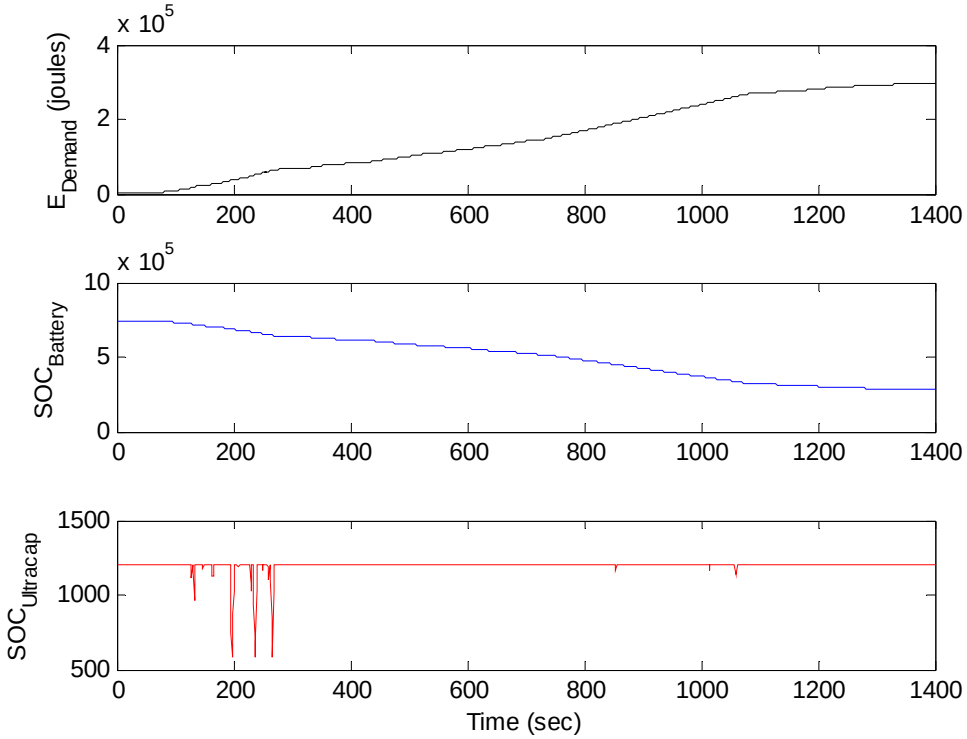

Figure A.5-7. Rule Based Controller Results Power System SOC: Battery and Ultracapacitor System

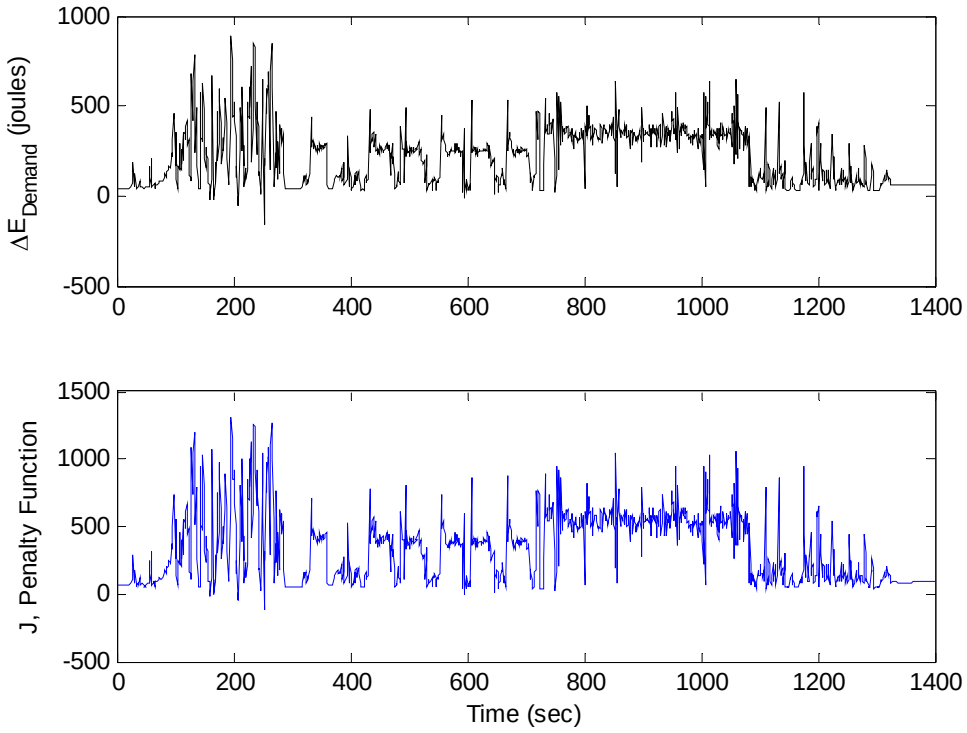

Figure A.5-8. Rule Based Controller Results Cost Function: Battery and Ultracapacitor System

The total energy required to meet the 297,902.67 joules demanded in the profile is 466,609 joules between the two power sources shown in Table A.5-2. The decrease in the state of charge of the hybrid system, however, decreased by 466,609 joules. Less than half of the energy, 37.59%, of the hybrid system's 747,696 joules remained at the end of the 23.3 minute profile. The hybrid system depleted 62.41% of its energy to meet the demanded profile using the Rule Based Controller method.

Table A.5-2: Battery and Ultracapacitor Rule-Based 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Rule-Based	23.3	746496	1200	279,887	1,200	37.59%

A.5.5.3 Relative Performance of Battery and Ultracapacitor System Controllers

The Dynamic Programming controller outperformed the rule-based controller in the 23.3 minute profile just as in the 10 and 65 minute profiles from Chapter 5. The results of the 23.3 minute profile are listed in Table A.5-3. During the 23.3 minute profile, the percentage of the battery's charge used to meet the profile was 62.02% using the Dynamic Programming controller. The rule based method used 62.41% of the total energy available. The Dynamic Programming controller thus retains an additional 0.39% of its total energy at the end of the short profile. The Dynamic Programming controller uses 2,910.62 joules less than the Rule based method.

Table A.5-3: Battery and Ultracapacitor 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	23.3	746496	1200	282341.82	0	37.98%
Rule-Based	23.3	746496	1200	279887	1,200	37.59%

A.5.6 Dynamic Programming and Rule Based Controller Comparison for Additional Robot Mission 2

83.3 Minute Profile

The 83.3 minute robot power profile was collected on the Talon platform. The robot performs climbing, monitoring, manipulation, and traversal capabilities throughout the mission as shown in the top of Figure A.5-9. The key difference between this power profile and those seen in Chapter 5 is that no portion of this profile is cycled. All of the data collected is unique and continuously collected over a wide range of capabilities.

A.5.6.1 Dynamic Programming Results

The results of a single cycle using Dynamic Programming can be found in Figure A.5-9 through Figure A.5-12. Figure A.5-9 through Figure A.5-10 contains three plots each which include the following: the power demand profile, the battery's power demand, and the

ultracapacitor's response to the demand profile. The first plot in Figure A.5-9 shows the demand profile for the 83.3 minute power demand profile. The two charts below the demand profile show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure shows that, for the Dynamic Programming result, the bulk of the energy comes from the battery and is supplemented by the ultracapacitor only during times of large energy demand. The energy from the battery onto the robot's power bus is sometimes greater than the demanded profile because the battery is both meeting the profile and adding energy into the ultracapacitor to charge it. During these times, the ultracapacitor also has a negative value in the y-axis which denotes charging. The battery never charges during this profile even though the ultracapacitor does so regularly.

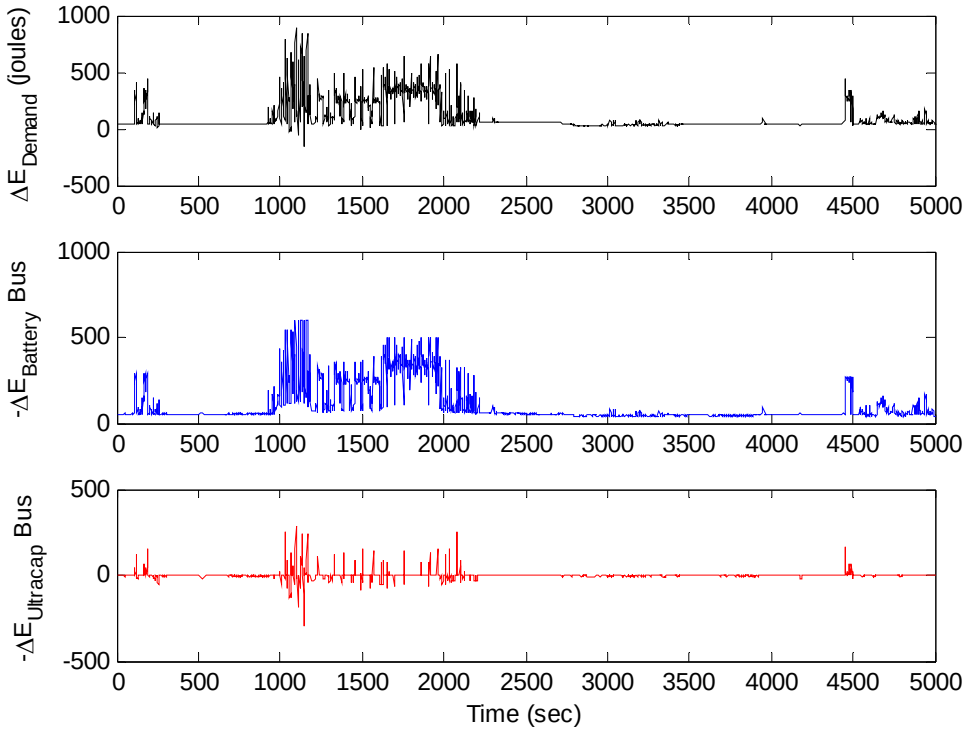

Figure A.5-9. Dynamic Programming Controller Results Power System Bus Energy: Battery and Ultracapacitor System

The bottom two graphs in Figure A.5-9 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. While the power demand profile graphs in Figure A.5-9 and Figure A.5-10 are identical, there are a few key differences between the battery and ultracapcitor's change in state of charge and the amount of energy on the bus from each source. The battery's change in state of charge in the figure below is always greater than the energy from the battery on the bus. As noted previously the battery experiences losses from overdrawing current and form the DC/DC converter. The amount of energy leaving the ultracapacitor, while discharging, is also greater than the amount of energy on the robot's power bus from the discharge. This is due to the losses experienced by the DC/DC converter operating during the discharge of the ultracapacitor. While charging however, the amount of energy traveling from the robot's power bus to the ultracapacitor is greater than the change in the ultracapacitor's state of charge.

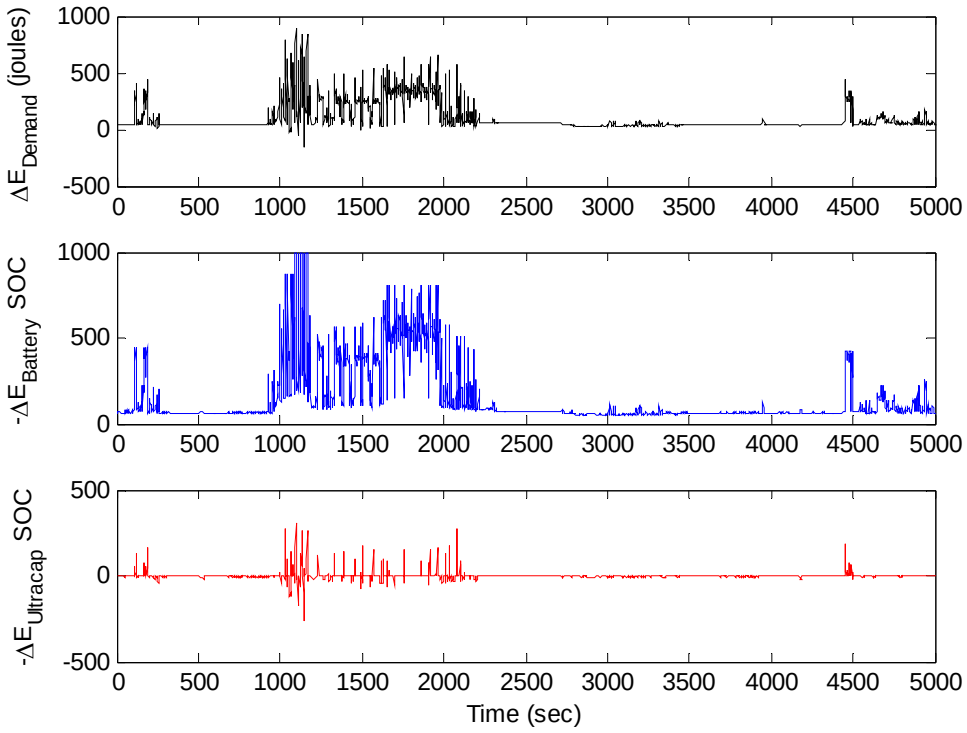

Figure A.5-10. Dynamic Programming Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Each power source's state of charge and the cumulative energy demand are shown in Figure A.5-11. The energy demand profile is consistently increasing throughout the 83.3 minute profile. Accordingly, the battery's charge is consistently decreasing. While this particular battery contains 746,496 joules, only the energy required to complete the mission is shown below.

The ultracapacitor is constantly charging and discharging throughout the duration of the energy demand profile. Again, the Dynamic Programming controller discharges the ultracapacitor during instances of large energy demand. During instances of lower energy demand the ultracapacitor is slowly discharged. The rate of each is computed using the cost function described previously. The most efficient use of the ultracapacitor using the Dynamic Programming controller is to have the ultracapacitor completely discharged at the end of the profile. The relative cost to draw energy from this source is always lower than drawing from the battery because the ultracapacitor experiences fewer losses. Providing the profile is long enough in duration, the ultracapacitor will always be completely discharged by the end of a profile.

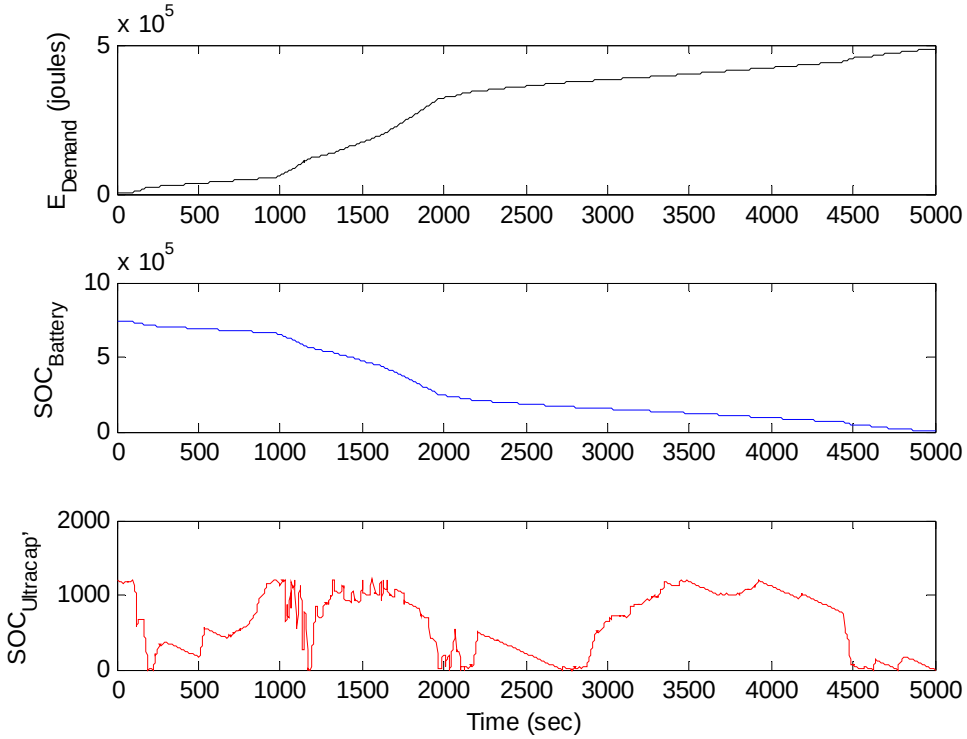

Figure A.5-11. Dynamic Programming Controller Results Power System SOC: Battery and Ultracapacitor System

The energy demand profile and the corresponding non-cumulative cost function for each step in time is shown in Figure A.5-12. The non-cumulative cost function represents the total change in both the battery and ultracapacitor's state of charge to meet the demanded energy profile. Each power sources experiences losses as mentioned earlier, so the non-cumulative cost function will be greater than the demanded energy for each instance in time.

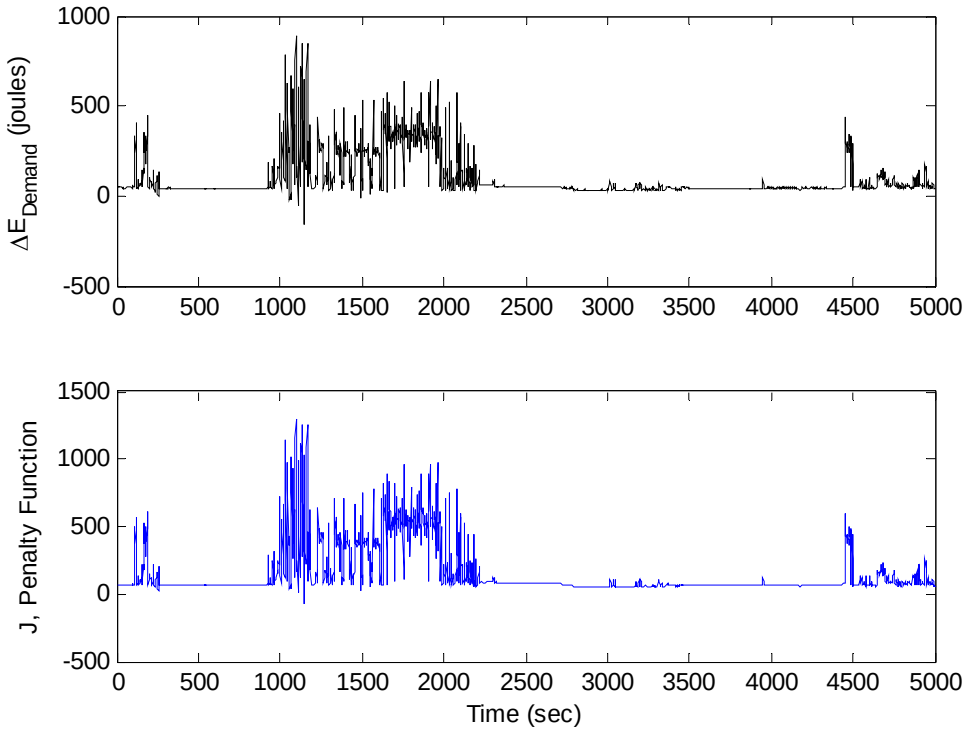

Figure A.5-12. Dynamic Programming Controller Results Cost Function: Battery and Ultracapacitor System

The results of the Dynamic Programming algorithm applied to the battery and ultracapacitor hybrid system can be found in Table A.5-4. The battery power system consisting of one BB2590 batteries had a theoretical energy capacity of 746,496 joules, which can be combined with an ultracapacitor which has an initial charge of 1200 joules. At the end of the profile, a total of 2,119 joules remained in the battery and 0 joules remain in the ultracapacitor. The demanded profile requires a total of 487,263.53 joules while 745,577 joules left the two power sources to meet the power demand profile. After the profile, the battery had 85.88% of its total charge remaining. The power demand profile consumed 99.56% of the total energy available from the hybrid system.

Table A.5-4: Battery and Ultracapacitor Dynamic Programming 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	65	746496	1200	2,119	0	0.44%

A.5.6.2 Rule-Based Results

The results of the 83.3 minute profile using the Rule Based Controller can be found in Figure A.5-13 through Figure A.5-16 show that the rule-based controller is unable to complete the mission. The battery system depletes its energy completely after 4987 seconds. Just as with the Dynamic Programming results, Figure A.5-9 through Figure A.5-10 contains three plots each which include the following: the power demand profile, battery's response and ultracapacitors response to the demand profile. The two charts below the energy demand profile in Figure A.5-13 show the portion of energy from the battery and the ultracapacitor supplied to the robot's power bus to meet the demanded profile. This figure shows that the battery is used exclusively to meet the demands of the profile until the energy limit is reached. Once this occurs, the ultracapacitor is used to meet the additional demands. Once additional energy is available from the battery, after meeting the demand of the profile, the ultracapacitor is charged immediately back to its maximum capacity. The energy from the battery onto the robot's power bus is greater than the energy demand profile when this ultracapacitor charging occurs. During these times the ultracapacitor also has a negative value in the y-axis which denotes charging. The battery never charges during this profile.

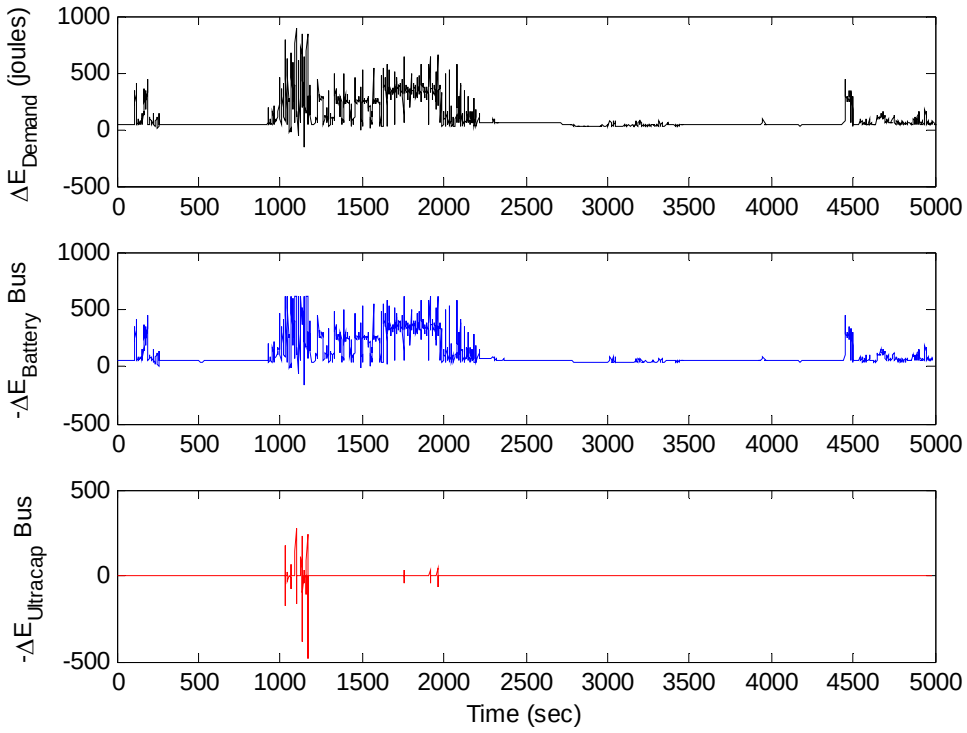

Figure A.5-13. Rule Based Controller Results Power System Bus Energy: Battery and Ultracapacitor System

The bottom two graphs in Figure A.5-14 show the battery and ultracapacitor's change in state of charge to meet the power demand profile in the top graph. The battery's change in state of charge in the figure below is always greater than the energy from the battery on the bus. The amount of energy leaving the ultracapacitor, while discharging, is also greater than the amount of energy on the robot's power bus from the discharge. This is due to the losses experienced by the DC/DC converter. While charging, the amount of energy traveling from the robot's power bus to the ultracapacitor is greater than the change in the ultracapacitor's state of charge. This is again because the DC/DC converter draws power and experiences efficiency losses during both charging and discharging. While the times at which each source charges and discharges varies in comparison to the DP results, the rule-based controller seems to utilize similar principles for using the ultracapacitor. This, the losses appear to be consistent between controllers. The energy demand profile is always met using the hybrid system. The relationship between change in state of charge and energy available from the bus is again consistent with the scaling and efficiency equations presented at the beginning of Chapter 5.

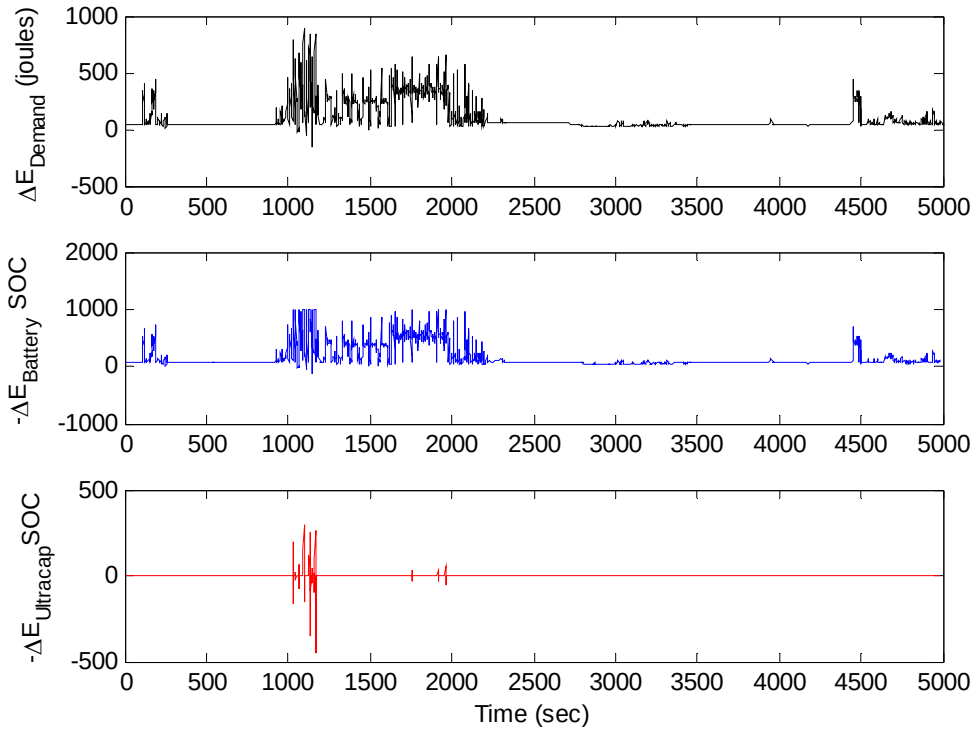

Figure A.5-14. Rule Based Controller Results Power System Change in SOC: Battery and Ultracapacitor System

Each power source's state of charge and the cumulative energy demanded are shown in Figure A.5-15. The energy demand profile is consistently increasing throughout the 83.3 minute profile. Just as intuitively, the battery's charge is seen to be consistency decreasing. The ultracapacitor, in comparison, is constantly charging and discharging throughout the duration of the energy demand profile.

The biggest difference between these results and those seen previously, is that, unlike Dynamic Programming, the rule-based controller immediately charges the ultracapacitor to full capacity whenever possible.

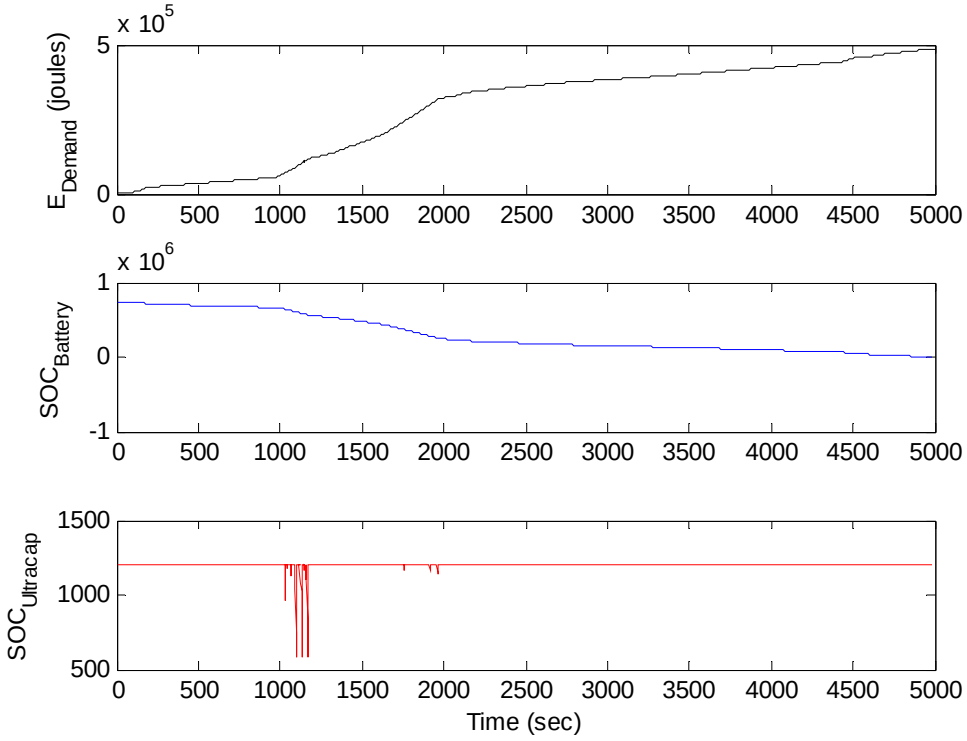

Figure A.5-15. Rule Based Controller Results Power System SOC: Battery and Ultracapacitor System

The rule-based controller does not use a cost function to calculate the optimal use of each power source. The cost function can be used to compare the relative performance of each controller. The Dynamic Programming controller minimizes energy use by minimizing a cost function which describes energy use. Both the energy used and the cumulative cost function can therefore be used to evaluate each controller performance. The energy demand profile and the corresponding non-cumulative cost function for each step in time is shown in Figure A.5-16. Using the rule-based controller, the non-cumulative cost function will be greater than the demanded energy from the profile.

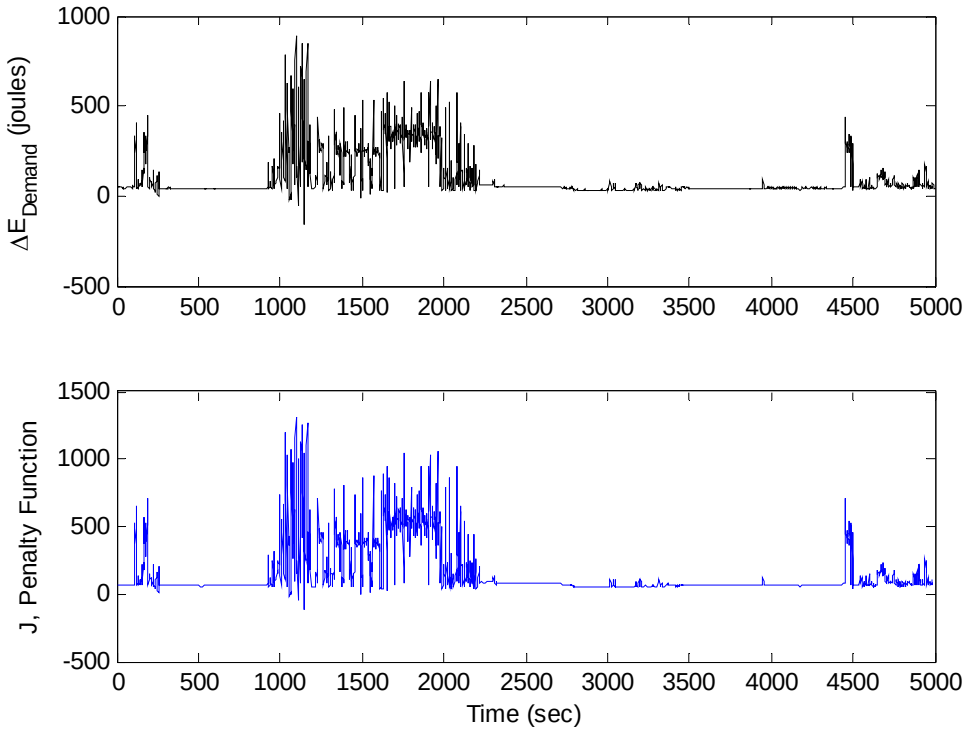

Figure A.5-16. Rule Based Controller Results Cost Function: Battery and Ultracapacitor System

Table A.5-5 shows that at the end of the 83.3 minute profile, a total of 0 joules remained in the battery and 0 joules remained in the ultracapacitor because the power source was unable to complete the robot power profile. The demanded profile requires a total of 487,263 joules while 746,496 joules left the battery to meet the power demand profile. The rule-based controller was unable to meet the demands of the profile and subsequently stopped the profile prematurely at 4987 seconds rather than 5000.

Table A.5-5: Battery and Ultracapacitor Rule-Based 10min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Rule-Based	65	746496	1200	0	0	0.00%

A.5.6.3 Relative Performance of Battery and Ultracapacitor System Controllers

The results over a longer profile show similar results in Table A.5-6. The Dynamic Programming controller requires 99.56% of the total energy to meet the 83.3 minute profile. The

rule-based Controller requires over 100% of the available energy. While Dynamic Programming yields a more optimal solution than the Rule Based method, the benefit is that the Dynamic Programming algorithm completes the profile but the rule-cased controller does not. The conclusion which can therefore be drawn from this comparison is that the rule-based controller yields nearly the same results as Dynamic Programming for the profiles that have been considered. The rule-based solution only missed completing the profile by 13 seconds which on a 5000 second profile is within the tolerance of this work.

Table A.5-6: Battery and Ultracapacitor 65min Profile Results

Power Allocation	Time (min)	Initial Battery Energy (joules)	Initial Ultracapacitor Energy (joules)	Battery Energy Remaining (joules)	Ultracapacitor Energy Remaining (joules)	Percent Energy Remaining
Dynamic Programming	65	746496	1200	2119.1	0	0.44%
Rule-Based	65	746496	1200	0	0	0.00%

A.5.7 Rule Based Controller MATLAB® Script (Varied Total Mass and Hybrid Topology)

```
%% 2010_09_10
%% HyPER
%% DREW LOGAN
%% Rule Based Controller: Vary Hybrid Composition and Determien Relative
%% Performance
    cd;
    close all
    clear all
    clc
for profile=1:3
    close all;
    keep profile


%% 1.0 Load Talon Profile
% LOAD PROFILE
    if profile==1
        load('E_Demand12Hrs');
    elseif profile==2
        load('E_Demand30');
    elseif profile==3
        load('E_Demand_Climb100');
    end
    E_Demand=E_Demand';
    for index_E_Demand=1:length(E_Demand);
```

```
        if index_E_Demand==1;
            E_Demand_Cumulative(1,index_E_Demand)=E_Demand(1,index_E_Demand);
        else

E_Demand_Cumulative(1,index_E_Demand)=E_Demand_Cumulative(1,index_E_Demand-
1)+E_Demand(1,index_E_Demand);
        end
    end
% TIME INTERVAL
    deltaT=1;
% CHANGE MASS FROM 0kg TO 6 BB2590 MASS (8.4KG)
    Num_BB2590_Equivalent= 6;
    Mtot= 1.4*Num_BB2590_Equivalent;       %kg


%% 2.0 BATTERY AND ULTRACAP Characteristics
% 2.1 BATTERY (GENERALIZED BB2590)
        Mbatt_BB2590= 1.4;                                  %kg
        Ebatt_BB2590= 746469;                               %joules
        Emaxbatt_BB2590= 23*28.8*1;                         %25amp*28.8V*1sec
        Vbatt_BB2590=28.8;                                  %V
        N= 1.1;                                             % Peukert Number
        Rbatt=20;                                           % Amp-hr
        DensityBattery= Ebatt_BB2590/Mbatt_BB2590;          %joule/kg
        DensityBatteryMax= Emaxbatt_BB2590/Mbatt_BB2590;%joule/kg
% 2.2 ULTRACAP (GENERALIZED MAXWELL ULTRACAP)
        DensityCap= 0.0199*1000000;                         %joule/kg
        DensityCapMax= 10028;                               %joule/kg
% 2.3 GENERATOR
        % DensityGen= 404930*(x^1.3852);
        % DensityGenMax= 12.959*(x^1.3769);


%% 3.0 DC/DC Converter Parameters
% DCDC=1 if the system is using converters
% DCDC=0 if we do not include losses due to the converters
        DCDC=1;
        if DCDC==1; % YES USE DC/DC CONVERTERS
            b= 0.92694;
            m= 3.27768;
        else
            b=1;
            m=0;
        end


%% 4.0 Power Source Properties
    indexGG=1;
    resolution=0.1;
for GG= 0:resolution:1
    disp(GG)
    indexFF=1;
% Initialize Power Source Values
% Generator Characteristics
```

```
Mgen=(1-GG)*Mtot;
    GenMAXint= 12.959*(Mgen^1.3769);
    if Mgen<=2*m/GenMAXint
        m_gen=0;
        b_gen=1;
    else
        m_gen=m;
        b_gen=b;
    end
        DeltaEgenMAX= 12.959*(Mgen^1.3769);
        DeltaEgenMAX=DeltaEgenMAX*b_gen-m_gen;
        EgenInt= 404930*(Mgen^1.3852);
for FF=0:resolution:1;
% BB2590 Battery Characteristics
Mbatt= (GG)*(1-FF)*Mtot;
    BattMAXint=Mbatt*DensityBatteryMax;
    if Mbatt<=2*m/BattMAXint;
        m_batt=0;
        b_batt=1;
    else
        m_batt=m;
        b_batt=b;
    end
        DeltaEbattMAX= Mbatt*DensityBatteryMax;
            DeltaEbattMAX=DeltaEbattMAX*b_batt-m_batt;
            EbattInt= Mbatt*DensityBattery;
            Cbatt=EbattInt/(Vbatt_BB2590*3600);
            kbatt=
(Vbatt_BB2590)*(Cbatt/Rbatt)*(Rbatt^(1/N))*((deltaT/3600)^(-
1/N))*(EbattInt^(-1/N))*(deltaT);
            % The characteristics of the battery vary with mass. While the
            % capacity varries, the voltage does not. The kbatt function
            % is a constant based upon the batteries capacity and is used
            % to approximate the change in capacity based upon power drawn.
            if isnan(kbatt)
                kbatt=1;
            end
% Ultracap Characteristics
Mcap= (GG)*(FF)*Mtot;
    CapMAXint=Mcap*DensityCapMax;
    if Mcap<=2*m/CapMAXint
        m_cap=0;
        b_cap=1;
    else
        m_cap=m;
        b_cap=b;
    end
            DeltaEcapMAX= Mcap*DensityCapMax;
                DeltaEcapMAX= DeltaEcapMAX*b_cap-m_cap;
            EcapInt= Mcap*DensityCap;


%% 5.0 RUN PROFILE
    deltaHEAT=0;
```

```
    flag=1;
for EE=1:length(E_Demand);
%     disp(EE)
    T(:,EE)=EE;
% STEP 1: GENERATOR MEETS DEMAND- CHARGE BATTERY AND CAP
                        deltaEgen(indexFF,EE)=DeltaEgenMAX;
                if deltaEgen(indexFF,EE)>E_Demand(:,EE);
                    % BATTERY
                        deltaEbatt_y(1,:)= DeltaEgenMAX-E_Demand(:,EE);
                            % Charge with additional available energy
                        if EE==1
                            % Initially can't charge battery
                        battery_y= EbattInt-EbattInt;
                        else
                        battery_y= EbattInt-Ebatt(indexFF,EE-1);
                        end
                            battery_y= (battery_y/kbatt)^N;
                            battery_y= (battery_y+m_batt)/b_batt;
                        deltaEbatt_y(2,:)=battery_y;
                            % How much till the battery is full.
                        deltaEbatt_y(3,:)= (DeltaEbattMAX+m_batt)/b_batt;
                            % Maximum power into or out of the battery
                            deltaEbatt_y= min(deltaEbatt_y);
                            deltaEbatt(indexFF,EE)=deltaEbatt_y;
                        deltaEbatt_DCDC= deltaEbatt(indexFF,EE)*b_batt-
m_batt;
                        if deltaEbatt_DCDC>=0 % CHARGE BATTERY
                            delta_E_battSOC(indexFF,EE)=
kbatt*(deltaEbatt_DCDC^(1/N));
                        else % DISCHARGE BATTERY
                            deltaEbatt_DCDC= -deltaEbatt_DCDC;
                            deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(indexFF,EE)= -deltaEbatt_DCDC;
                        end
                    % ULTRACAP
                        deltaEcap_y(1,:)= DeltaEgenMAX-E_Demand(:,EE)-
deltaEbatt(indexFF,EE);
                            % Charge the ultrcap with any additiona energy
                        if EE==1;
                        deltaEcap_y(2,:)= (EcapInt-EcapInt+m_cap)/b_cap;
                        else
                        deltaEcap_y(2,:)= (EcapInt-Ecap(indexFF,EE-
1)+m_cap)/b_cap;
                            % How much till the ultracap is full
                        end
                        deltaEcap_y(3,:)= (DeltaEcapMAX+m_cap)/b_cap;
                            % Maximum power into or out of the ultracap
                            deltaEcap_y= min(deltaEcap_y);
                            deltaEcap(indexFF,EE)=deltaEcap_y;
                        deltaEcap_DCDC= deltaEcap(indexFF,EE)*b_cap-m_cap;
                        if deltaEcap_DCDC<0
                            deltaEcap_DCDC=
(deltaEcap(indexFF,EE)+m_batt)/b_batt;
                            deltaEcap_DCDC= -deltaEcap_DCDC;
                        end
```

```
                            delta_E_capSOC(indexFF,EE)= deltaEcap_DCDC;
                    % HEAT
                        deltaHEAT_STEP= deltaEgen(indexFF,EE)-E_Demand(:,EE)-
deltaEbatt(indexFF,EE)-deltaEcap(indexFF,EE);
                            deltaHEAT(indexFF,EE)= deltaHEAT_STEP;
                            % Any additional energy unused goes to heat
% STEP 2: GENERATOR AND BATTERY MEETS DEMAND-CHARGE ULTRACAPACITOR-
                elseif deltaEgen(indexFF,EE)+DeltaEbattMAX>=E_Demand(:,EE)
                    % ULTRACAP
                        deltaEcap_ny(1,:)= deltaEgen(indexFF,EE) +
DeltaEbattMAX - E_Demand(:,EE);
                            % Additional energy available to charge the
                            % ultracap.
                        if EE==1
                        deltaEcap_ny(2,:)= (EcapInt-EcapInt+m_cap)/b_cap;
                        else
                        deltaEcap_ny(2,:)= (EcapInt-Ecap(indexFF,EE-
1)+m_cap)/b_cap;
                            % How much till the ultracap is full
                        end
                        deltaEcap_ny(3,:)= (DeltaEcapMAX+m_cap)/b_cap;
                            % Maximum power into or out of the ultracap
                            deltaEcap_ny= min(deltaEcap_ny);
                            deltaEcap(indexFF,EE)=deltaEcap_ny;
                        deltaEcap_DCDC= deltaEcap(indexFF,EE)*b_cap-m_cap;
                            delta_E_capSOC(indexFF,EE)= deltaEcap_DCDC;
                    % BATTERY
                        deltaEbatt_ny= deltaEgen(indexFF,EE)-E_Demand(:,EE)-
deltaEcap(indexFF,EE);
                            % Required energy from the battery
                            deltaEbatt(indexFF,EE)= deltaEbatt_ny;
                        deltaEbatt_DCDC= -deltaEbatt(indexFF,EE);
                        deltaEbatt_DCDC= (deltaEbatt_DCDC+m_batt)/b_batt;
                        deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(indexFF,EE)= -deltaEbatt_DCDC;
                        deltaHEAT(indexFF,EE)=0;
% STEP 3: DISCHARGE ALL TO MEET DEMAND-DISCHARGE ULTRACAPACITOR-
                else
                    % BATTERY
                        deltaEbatt_nn= DeltaEbattMAX;
                            % Maximum battery output
                            deltaEbatt(indexFF,EE)= -deltaEbatt_nn;
                        deltaEbatt_DCDC= -deltaEbatt(indexFF,EE);
                        deltaEbatt_DCDC= (deltaEbatt_DCDC+m_batt)/b_batt;
                        deltaEbatt_DCDC= (deltaEbatt_DCDC/kbatt)^N;
                            delta_E_battSOC(indexFF,EE)= -deltaEbatt_DCDC;
                    % ULTRACAP
                        deltaEcap_nn= -E_Demand(:,EE)+deltaEgen(indexFF,EE)-
deltaEbatt(indexFF,EE);
                            % Required additional energy from the ultracap
                            deltaEcap(indexFF,EE)= deltaEcap_nn;
                        deltaEcap_DCDC= -deltaEcap(indexFF,EE);
                        deltaEcap_DCDC= (deltaEcap_DCDC+m_cap)/b_batt;
                            delta_E_capSOC(indexFF,EE)= -deltaEcap_DCDC;
                       deltaHEAT(indexFF,EE)=0;
```

```
                end
% STEP 4: SHOW CHANGE IN SOC
                    if EE==1
                    Egen(indexFF,EE)= EgenInt - deltaEgen(indexFF,EE);
                    Ebatt(indexFF,EE)= EbattInt +
delta_E_battSOC(indexFF,EE);
                    Ecap(indexFF,EE)= EcapInt + delta_E_capSOC(indexFF,EE);
                    else
                    Egen(indexFF,EE)= Egen(indexFF,EE-1) -
deltaEgen(indexFF,EE);
                    Ebatt(indexFF,EE)= Ebatt(indexFF,EE-1) +
delta_E_battSOC(indexFF,EE);
                    Ecap(indexFF,EE)= Ecap(indexFF,EE-1) +
delta_E_capSOC(indexFF,EE);
                    end
% STEP 5: PENALTY FUNCTION CALCULATIONS

J(:,EE)=(delta_E_capSOC(indexFF,EE)+delta_E_battSOC(indexFF,EE))^2;
% STEP 6: DISP ERRORS
                % PEAK POWER CONDITION
                if E_Demand(:,EE)>DeltaEgenMAX+DeltaEbattMAX+DeltaEcapMAX
                    disp('PEAK DEMAND FAILURE');
                    flag=0;
                    break
                end
                % TOTAL POWER CONDITION
                if
E_Demand(:,EE)>DeltaEgenMAX+Ebatt(indexFF,EE)+Ecap(indexFF,EE)
                    disp('Total Energy Violation');
                    flag=0;
                    break
                end
                % TOTAL GENERATOR ENERGY
                if Egen(indexFF,EE)<0
                    disp('Generator Limited');
                    flag=0;
                    break
                end
                % TOTAL BATTERY ENERGY
                if Ebatt(indexFF,EE)<0
                    disp('Battery Limited')
                    flag=0;
                    break
                end
                % TOTAL ULTRACAP ENERGY
                if Ecap(indexFF,EE)<0
                    disp('Ultracap Limited');
                    flag=0;
                    break
                end
end
    Total_J=sum(J);
```

```
%% 6.0 RESULTS
        zEnergyRemaining(indexFF,indexGG)=
(Egen(indexFF,max(EE))+Ebatt(indexFF,max(EE))+Ecap(indexFF,max(EE)))*flag;
        zGenMass(indexFF,indexGG)= Mgen;
        zBattMass(indexFF,indexGG)= Mbatt;
        zGenDOH(indexFF,indexGG)= GG;
        zBattDOH(indexFF,indexGG)= FF;
        zJ(indexFF,indexGG)=Total_J;
        indexFF=indexFF+1;
end
        indexGG=indexGG+1;
end


%% 7.0 DISPLAY RESULTS
figure
    mesh(zGenDOH,zBattDOH,zEnergyRemaining);
    zlabel('Additional Available Energy (joules)');
    xlabel('(1-%M_T_o_t_,_G_e_n_e_r_a_t_o_r)');
    ylabel('Battery/Ultracapcitor DOH');


%% 8.0 SAVE FILE
    if profile==1
        save RESULTS_E_Demand12Hrs
    elseif profile==2
        save RESULTS_E_Demand30
    elseif profile==3
        save RESULTS_E_Demand_Climb100_small_int
    end

end
```