5-2019

Motor Control Systems Analysis, Design, and Optimization Strategies for a Lightweight Excavation Robot

Austin Jerold Crawford
University of Arkansas, Fayetteville

Motor Control Systems Analysis, Design, and Optimization Strategies for a Lightweight Excavation Robot

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering

by

Austin J. Crawford
University of Arkansas
Bachelor of Science in Mechanical Engineering, 2017

May 2019
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Uche Wejinya, Ph.D.
Thesis Director

Yue Chen, Ph.D.
Committee Member

Mark Arnold, Ph.D.
Committee Member

ABSTRACT

This thesis entails motor control system analysis, design, and optimization for the University of Arkansas NASA Robotic Mining Competition robot. The open-loop system is to be modeled and simulated in order to achieve a desired rapid, yet smooth response to a change in input. The initial goal of this work is to find a repeatable, generalized step-by-step process that can be used to tune the gains of a PID controller for multiple different operating points. Then, sensors are to be modeled onto the robot within a feedback loop to develop an error signal and to make the control system self-corrective to account for slippage upon the Martian terrain with unknown soil parameters. Then, the closed loop system will be simulated again subject to an input disturbance that would account for the undulations and inconsistencies of the Martian terrain.

Using the analysis techniques established in the first two phases of this thesis, methods of immediate optimization with regards to motor output performance and wheel slip correction are presented for the purpose of implementation upon the next iteration of the robot build. This work also presents a general algorithm for robot autonomy in competition runs, which comes along with specific sensor configuration and pseudocode for the basic commands that the algorithm is built upon.

Future work for the analysis and design phases of this work would involve several iterations of custom motor control boards to be manufactured and tested on the robot build to verify the proposed generalized process of the PID tuning method. Future work for the automation phase of this work would involve the construction of a practice pit for the robot to build upon the primary automation strategy presented in the latter sections of this thesis.

ACKNOWLEDGEMENTS

The author would like to thank and recognize the University of Arkansas for providing the opportunity for research and involvement in the NASA RMC. Thank you to the faculty lead advisor for the competition and for robotics research for the department, Dr. Uche Wejinya. Thank you to the members of the University of Arkansas undergraduate RMC design-build team. Thank you to the author's summer 2018 understudy Barrett Loseke, who was a primary contributor to this report. Thank you to the author's research partners pertaining to the work on the excavation system, Ryan Watson and Chris Taylor.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: GENERAL DESIGN PROBLEM AND OVERVIEW

The general design problem at hand is per the requirements of the annual NASA RMC rules and rubrics. The objectives of the undergraduate design team are to design a robot that can traverse the challenging simulated chaotic Martian terrain. The mining robot must then excavate the ice simulant (gravel) and return the excavated mass for deposit into the collector bin to simulate a Martian resource mining mission. The complexities of the challenge include the abrasive characteristics of the regolith simulant, the weight and size limitations of the mining robot, and the ability to tele-operate it from a remote Mission Control Center. The on-site mining category will require teams to consider a number of design and operation factors such as dust tolerance and dust projection, communications, vehicle mass, energy/power requirements, and autonomy [1]. Specific rules, regulations, and information pertinent to the competition can be found at www.nasa.gov/offices/education/centers/kennedy/technology/nasarmc/about.

The University of Arkansas undergraduate team is divided into multiple subcategories in order to achieve all the design objectives and comply with the constraints provided by NASA. The subgroup focuses for this year's undergraduate team are as follows:

- Wheel Design (Mechanical)
- Excavation Design (Mechanical)
- Drivetrain and Mobility (Mechanical)
- Electrical Systems and Controls (Mechanical/Electrical)
- Communication and Software (Computer Science)

These teams have worked throughout the 2017-2018 academic year to develop this robot to perform a successful outing at this year's competition (which will begin next week). A picture of the completed robot is shown below:

Figure 1. Picture of University of Arkansas NASA RMC Robot

(Any specific questions about the design and build process for this year's robot can be directed towards the undergraduate RMC team leader. Contact information can be provided to the reader upon request from the author)

Next month, the team will transport this robot down to the NASA Kennedy space center and use the robot to participate in two 10-minute competition runs. During these runs the robot will be placed in the starting zone competition pit, which is designed to simulate the Martian terrain, **navigate through the obstacle area to the excavation site**, excavate icy regolith simulant material, **return through the obstacle area to the starting zone**, and deposit the icy regolith simulant into a collection bin. Two of these stated processes are emphasized because

they are the specific processes that pertain to the research work that will be presented later in this report. Research work regarding the other two processes are being addressed by my partner, Ryan Watson (rbwatson@uark.edu) whose research is oriented towards the excavation system controls and operation.

A top view dimensional layout of the competition pit is shown in the image below:

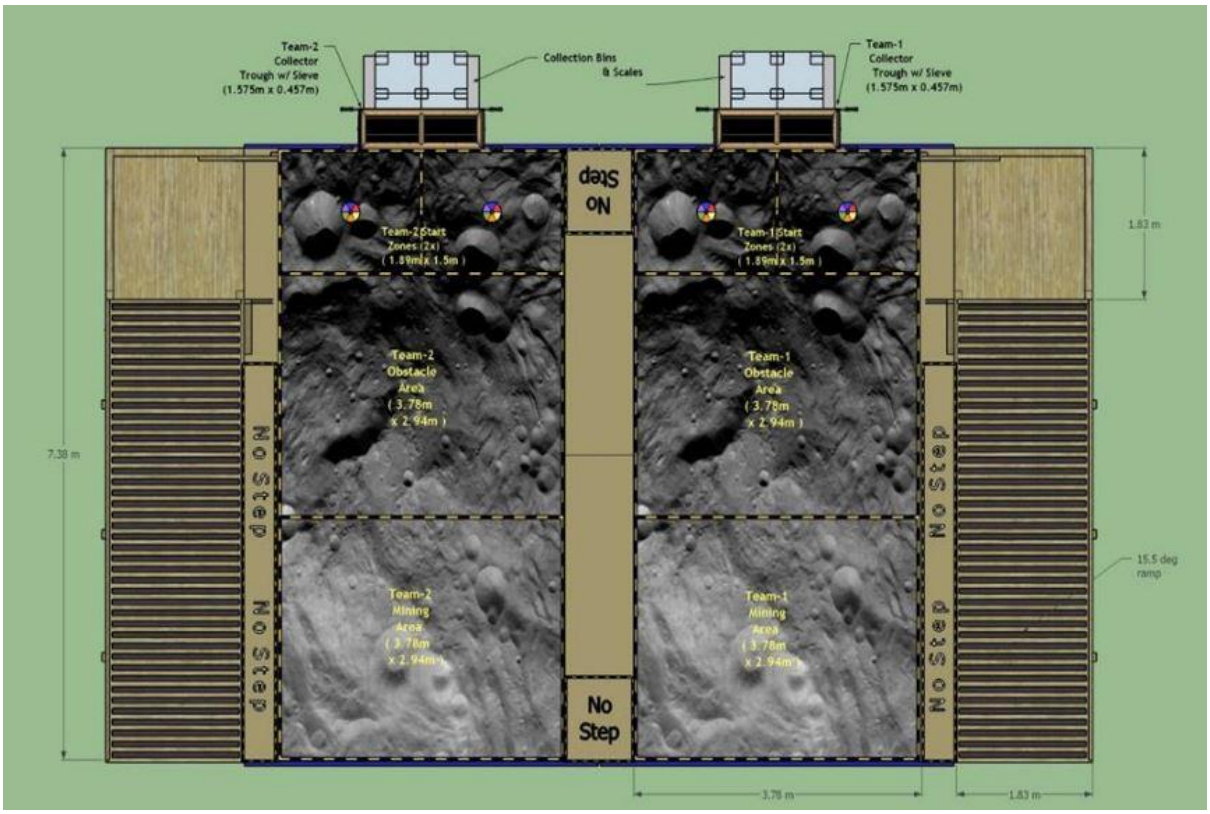

Figure 2. Competition Pit for NASA RMC Runs. [1]

This diagram is important to consider not only for practice and testing purposes of the undergraduate team for this year's build, but also for the purpose of developing an automation algorithm which will be discussed in the latter stages of this report. As previously stated, the robot will be placed in one of the two halves of the starting zone (top of the image) oriented at a randomly selected angle. The robot is to rotate and navigate its way through the obstacle area, which is significantly undulated and contains large rocks, and unto the mining area (bottom of the image). Once it has reached the mining area, it is free to dig up a payload and deliver it back

to the collection bin, which is located on the back wall just behind the starting area. The robot can make as many trips back and forth from the starting area to the mining area as the team pleases, as long as it is done within the 10-minute time constraint of the run. Now that we have established the general scope of the RMC project, we can discuss the main topic of this report: the analysis, design, optimization, and automation of the motor control systems for the mobility of the robot through the terrain.

It is desired by the author that the reader understand that the main strategies pertaining to the information used in this report are not necessarily considered as improvements and additions to a current system in place. More specifically, the current and recent years' teams have implemented aftermarket motor controllers on the robot and have not yet attempted to implement an automation process for the robot to complete its runs. The information in this report would pertain to future teams' efforts to design and manufacture their own motor controllers, as well as establish a general plan for attempting a fully autonomous run.

CHAPTER 2: OBJECTIVES

The main objective of this research is to provide an iterative process that will aid future teams in the design and optimization of the motor control systems for their RMC robots. The main control objective is to obtain a smooth, responsive, and self-correcting current-controlled speed system that can be manufactured for use on future iterations of the robot. In order to accomplish these objectives, we have decided to break our process down into four phases:

- Phase 1 Analysis: Develop a generic system for analysis of the performance and response of the robot to changes in input. First, an open loop system will be considered to identify characteristics of the response time. Then, sensors will be modeled into the system in a feedback loop to ensure that, when implemented, the steady state error and response of the system is within our (to be) specified design criterion. Lastly, the system will be modeled and analyzed with a quasi-random disturbance signal that will be used to account for the "random" terrain the robot will encounter during the passthrough of the obstacle area.
- Phase 2 Design: Once it is established that the techniques used in phase 1 are sufficient in modeling and analyzing the response of the generic case of a DC motor control system, we then model the motor control system (aftermarket parts) that is being used for this year's competition robot. Once the system is modeled, we will then enter the parameters of this year's robot into our analysis techniques in order to develop a design for a custom motor controller that will deliver optimal performance that accounts for the robot's weight, speed, and torque requirements

- Phase 3 Optimization: After using the simulation techniques from MATLAB and Simulink on the parameters for the current robot, we wish to identify the limitations of the current hardware and to see if it is desirable to augment any of these parameters. This phase would lead into recommendation for selection of new hardware or a design revision to put on a future iteration of the robot.
- Phase 4 Automation: It is desirable, for competition purposes, that the robot to be able to perform multiple cycles of excavation autonomously. We wish to develop a physical pseudocode for the robot to operate by to be sent to a computer science specialist for interpretation into proper syntax. An efficient and detail-oriented code would increase the probability of a fully autonomous competition run, which would give future teams a significant advantage in the competition.

CHAPTER 3: PHASE I – ANALYSIS

3.1 Model Representation of the D.C. Motor Control System

The first model system that we would like to analyze is the general open loop control system shown below:

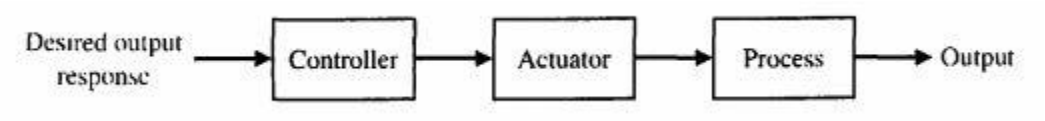

Figure 3. Open Loop Control System Block Diagram. [2]

For this phase 1 analysis, we can set the desired output response to be in terms of the motor shaft angular velocity $\dot{\theta}(s)$. It is understood that the quantity that we are looking to achieve is linear speed of the entire robot. Considering that the relationship between the linear speed of the bot and the angular speed of the motor shaft should be proportional, if we obtain the response for the angular shaft speed then we have done all the "heavy-lifting" of the control system response analysis. The gearboxes and wheels which would be added to the end of this diagram would only need to be modeled when running the simulations. In general, the actuator and process blocks can be combined into a singular block often referred to as the plant, which we will denote as P(s). The plant for our system is the commonly used brushed DC motor. For the time being, we will just leave the controller as C(s) until we reach the point where we will come up with a controller design.

The first challenge that we are faced with is producing an s-domain function for the plant P(s), which is a model representation of the DC motor. The DC motor is a power actuator device that delivers energy to a load. A figure of both the electrical and mechanical diagrams of a brushed DC motor is shown below:

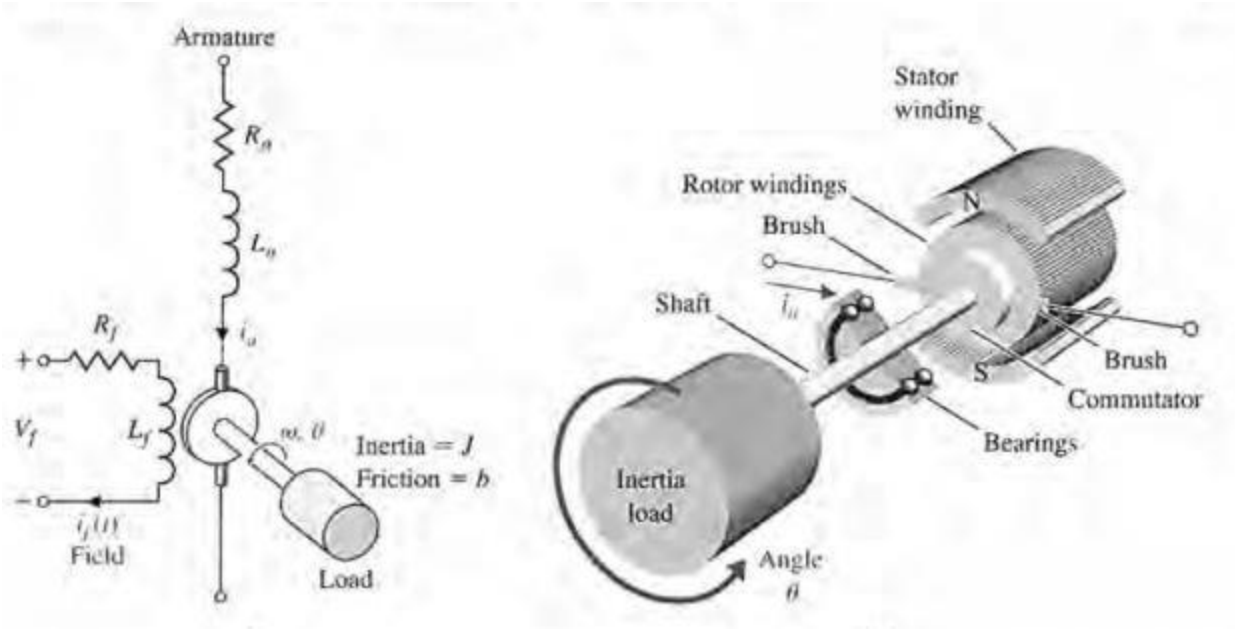

Figure 4. Brush D.C. Motor Electrical and Mechanical Diagram. [2]

The DC motor converts direct current electrical energy into rotational mechanical energy. A major fraction of the torque generated in the rotor of the motor is available to drive an external load [2]. The transfer function of the DC motor will be developed for a linear approximation to an actual motor. The input voltage may be applied to the field or to the armature terminals. The air-gap flux of the motor is proportional to the field current, provided that the field is unsaturated, so that:

$$\emptyset = K_f i_f \qquad \text{Eq. (1)}$$

The torque developed by the motor is assumed to be related linearly to the flux and the armature current as follows:

$$T_m = K_1 \emptyset i_a(t) = K_1 K_f i_f(t) i_a(t) \qquad \text{Eq. (2)}$$

It is clear from this equation that, to have a linear system, one current must be maintained constant while the other current becomes the input current. The armature-controlled DC motor uses the armature current as the control variable. The stator field can be established by a field

coil and a current or a permanent magnet. When a constant field current is established in a field coil, the motor torque is:

$$T_m(s) = \left(K_1 K_f I_f\right) I_a(s) = K_t I_a(s) \quad \text{Eq. (3)}$$

where K_t is a function of the permeability of the magnetic material. Using Kirchhoff's voltage law on the armature circuit, the armature current is related to the input voltage applied to the armature by:

$$L_a \frac{di_a}{dt} + R_a i_a = V_a - e \qquad \text{Eq. (4)}$$

$$V_a(s) = (R_a + sL_a) I_a(s) + e(s) \qquad \text{Eq. (5)}$$

Where $e(s)$ is the back electromotive-force voltage proportional to the motor speed. Therefore, we have:

$$e(s) = K_e \omega(s) = K_e \dot{\theta}(s) = K_e s\theta(s) \qquad \text{Eq. (6)}$$

where $\omega(s) = s\theta(s)$ is the transform of the angular speed and the armature current is [2]:

$$I_a(s) = \frac{V_a(s) - K_e \omega(s)}{R_a + sL_a} \quad \text{Eq. (7)}$$

Another figure is depicted below to more simply explain the armature circuit and free-body diagram of the rotor for an armature-current controlled DC motor:

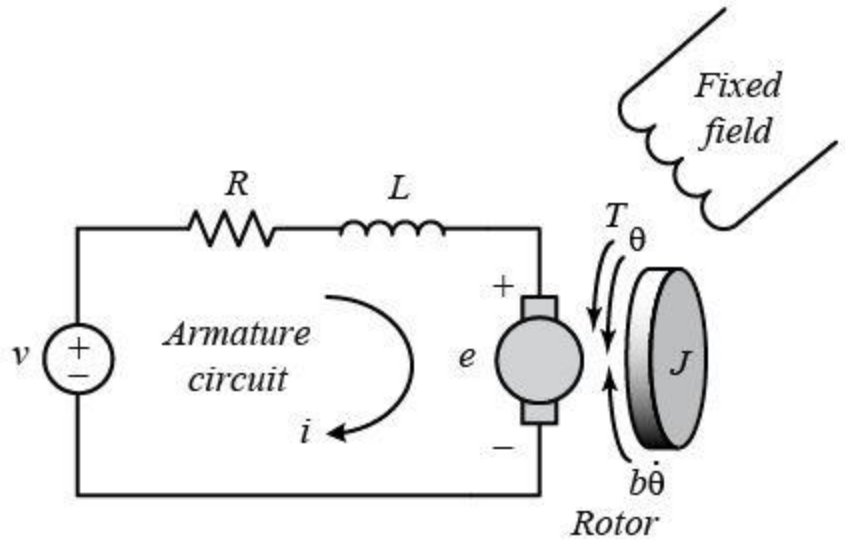

Figure 5. Equivalent Electric Circuit and Free-Body Diagram of the Rotor for an Armature-Controlled D.C. Motor. [6]

From this figure, and using Newton's 2nd Law we can derive the governing equation for the rotor as:

$$J\ddot{\theta} + b\dot{\theta} = T_L \qquad \text{Eq. (8)}$$

$$T_L(s) = T_m(s) - T_d(s) \qquad \text{Eq. (9)}$$

$$s(Js + b)\theta(s) = T_L(s) \qquad \text{Eq. (10)}$$

Where J is the polar moment of inertia for the rotor, b is the motor viscous friction constant, $T_L(s)$ is the load torque, and $T_d(s)$ the disturbance torque, which is often negligible [2]. However, the disturbance torque may indeed need to be considered for systems such as ours to account for the inconsistencies of the Martian terrain.

Condensing the governing armature circuit equation for the armature circuit into a similar format:

$$(R_a + sL_a)I_a(s) = V_a(s) - sK_e\theta(s) \qquad \text{Eq. (11)}$$

A representative block diagram of the plant (DC motor) is shown below:

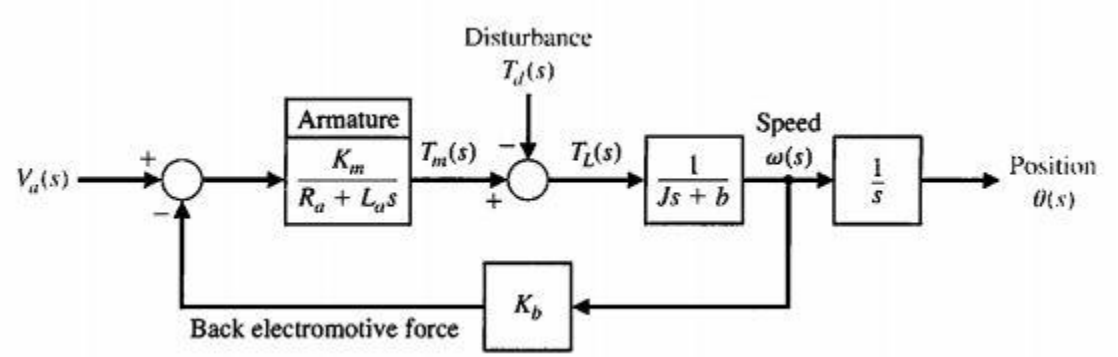

Figure 6. Block Diagram Representation of an Armature Controlled D.C. Motor. [2]

Using the previously established equations, which coincide with the block diagram, and letting $T_d(s) = 0$, we can solve to obtain the transfer function for speed as:

$$\boldsymbol{P(s) = \frac{\omega(s)}{V_a(s)} = \frac{K_t}{(R_a + sL_a)(sJ + b) + K_e K_t}} \qquad \text{Eq. (12)}$$

And the transfer function for position as:

$$P(s) = \frac{\theta(s)}{V_a(s)} = \frac{K_t}{s[(R_a + sL_a)(sJ + b) + K_e K_t]} \qquad \text{Eq. (13)}$$

Which fits in to the standard form of:

$$\frac{K_t}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \qquad \text{Eq. (14)}$$

Note that K_t is equal to K_e. This equality may be shown by considering the steady-state motor operation and the power balance when the rotor resistance is neglected. The power input to the rotor is $(K_e\omega)i_a$, and the power delivered to the shaft is $T\omega$. In the steady-state condition, the power input is equal to the power delivered to the shaft so that $(K_e\omega)i_a = (K_t i_a)\omega$, and we find that $K_t = K_e$ [2].

3.2 Open Loop Response of the Plant

Now that we have obtained a transfer function P(s) which models the DC Motor in the s-domain, we can obtain a simulated motor response to a change in input V(s). We can use an

analysis of the response so that we can design a controller that will move our response closer to a desired response, to which a desired response is one that complies with all of the design requirements. For now, we do not need to concern ourselves with the facets of controller design, but only with the analysis of the response of the plant.

Using typical constants for a DC Motor:

$$K_t = 0.05\ \frac{Nm}{A} \qquad \text{Eq. (15)}$$

$$J = 0.001\ \frac{Nms^2}{rad} \qquad \text{Eq. (16)}$$

$$\tau = \frac{L_a}{R_a} = 0.1\ s \qquad \text{Eq. (17)}$$

$$R = 2.5\ \Omega\ , L_a = 250\ mH \qquad \text{Eq. (18)}$$

$$b = 0.01\ Nms \qquad \text{Eq. (19)}$$

We can use MATLAB's step() function to generate a plot of the response of our system to a unitary change in input. The step response of the system defined above is shown below:

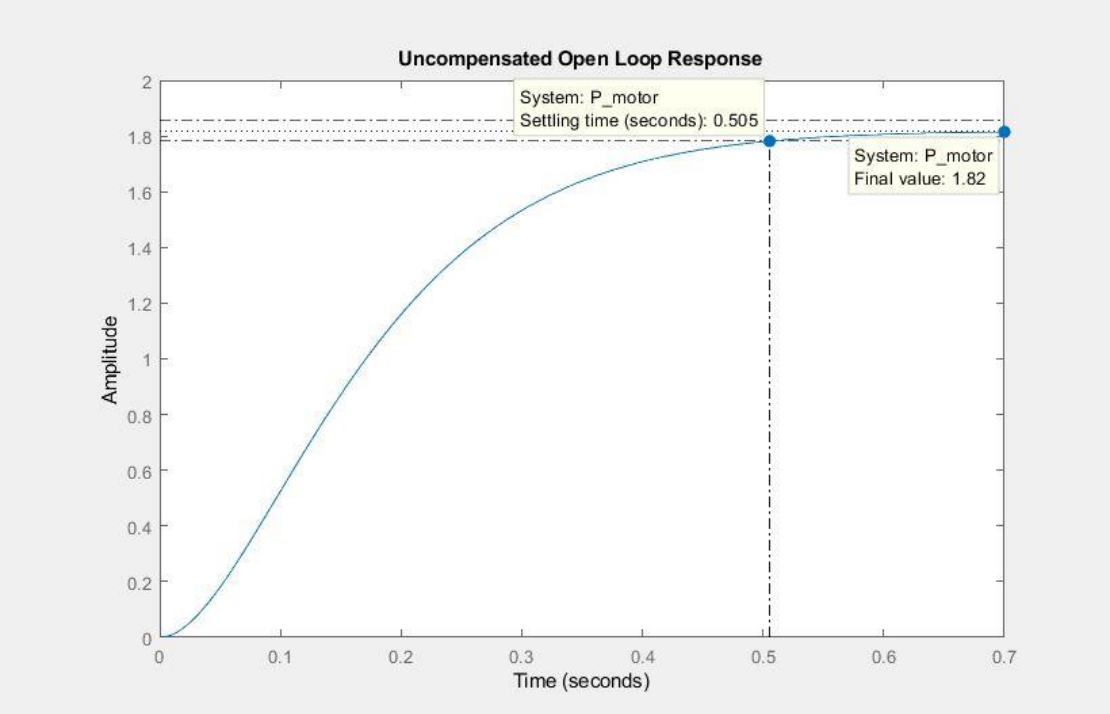

Figure 7. Step Response of the Plant in an Open Loop System.

From this plot (and/or the transfer function of the plant), we can define some specific response characteristics that we will use to better design a suitable controller. These three characteristics are the settling time, percent overshoot, and steady state error. These characteristics are related to the transfer function of any standard form system by the governing equations:

$$t_s(2\%) \approx \frac{4}{\zeta\omega_n} \qquad \text{Eq. (20)}$$

$$PO\% = 100e^{-\zeta\pi/\sqrt{1-\zeta^2}} \quad \text{Eq. (21)}$$

$$e_{ss}\% = \frac{|S.S.\ Value - Desired\ Value|}{|Desired\ Value|} \qquad \text{Eq. (22)}$$

MATLAB's plot tools automatically generate the settling time and percent overshoot for us and gives the steady-state value as well. For any response analysis, we want the desired output to be equal to the desired input. For a step response, the desired value is 1. However, the physical interpretation of this response is that our uncompensated motor will rotate at 1.82 rad/s in steady-state for an input voltage of 1 Volt.

3.3 Unity Feedback Controller Design

We now wish to create a closed-loop system and implement a controller in order to achieve a more desired response. Consider the following block diagram:

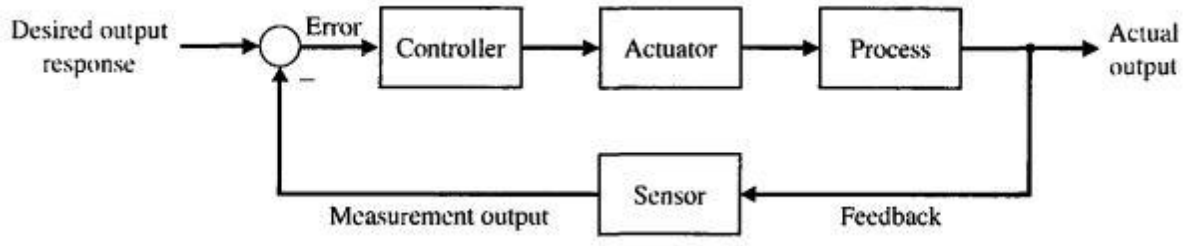

Figure 8. Block Diagram for Typical Closed Loop Control System. [2]

We have already modeled the actuator-process blocks as the plant, now we wish to model the controller C(s) by creating an error signal through a feedback loop. For now, we will leave the sensor block as a "to be determined" H(s). For a unitary H(s), the following diagram is appropriate:

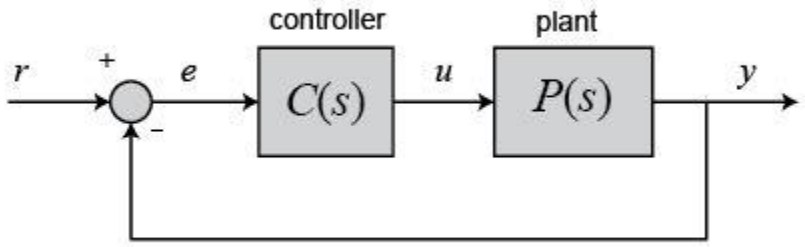

Figure 9. Block Diagram for Unity Feedback Control System. [3]

The controller design which we will use is the widely used three-term PID controller. This controller has a transfer function:

$$C(s) = K_p + \frac{K_I}{s} + K_D s \qquad \text{Eq. (23)}$$

The equation for the output in the time domain is

$$u(t) = K_p e(t) + K_I \int e(t)dt + K_D \frac{de}{dt} \qquad \text{Eq. (24)}$$

Where the error signal e(t) is the difference between input signal r(t) and output signal y(t) [2]. This error signal is fed to the PID controller, and the controller computes both the derivative and the integral of this error signal with respect to time. The control signal to the plant (u) is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_I) times the integral of the error plus the derivative gain (K_D) times the derivative of the error.

This control signal is fed to the plant and the new output is obtained. The new output is then fed back and compared to the reference input to find the new error signal. The controller

takes this new error signal and computes an update of the control input. This process repeats itself while the controller is in effect [3]. We can rearrange the transfer function of the PID controller to have a more suitable form of:

$$C(s) = \frac{K_D s^2 + K_P s + K_I}{s} \quad \text{Eq. (25)}$$

There are many methods available to determine acceptable values of the PID gains. A common approach to tuning is to use manual PID tuning method, whereby the PID control gains are obtained by trial-and-error with minimal analytic analysis using step responses obtained via simulation [2]. With powerful software like MATLAB, it is very easy to obtain step responses, and this makes the manual tuning process less time-inefficient. A chart displaying the effects of changing the control gains to the response characteristics is shown below:

PID Gain	Percent Overshoot	Settling Time	Steady-State Error
Increasing K_P	Increases	Minimal impact	Decreases
Increasing K_I	Increases	Increases	Zero steady-state error
Increasing K_D	Decreases	Decreases	No impact

Figure 10. Effects of Control Gains on the Response Characteristics

Now that we have established a means of tuning the controller, we must decide the response characteristic requirements that we wish to achieve. These criteria will be up to the designers to choose, but for our purposes we can set the requirements of:

$$t_s \leq 0.75\ s \quad \text{Eq. (26)}$$

$$PO\% < 5\% \quad \text{Eq. (27)}$$

$$e_{ss} < 1\% \quad \text{Eq. (28)}$$

We have seen from our previously uncompensated response that the settling time and overshoot already meet these criteria. However, the steady-state error comes in at a whopping 82% which must be dealt with. We will start our manual tuning process by increasing our

proportional gain until we reach the steady-state error criterion, while leaving the other gains at a unitary value.

After trial-and-error manual tuning in MATLAB we were able to meet our steady state error requirement by tuning our proportional gain up to a value of $K_P = 58$. The new response with this P-only controller is shown below:

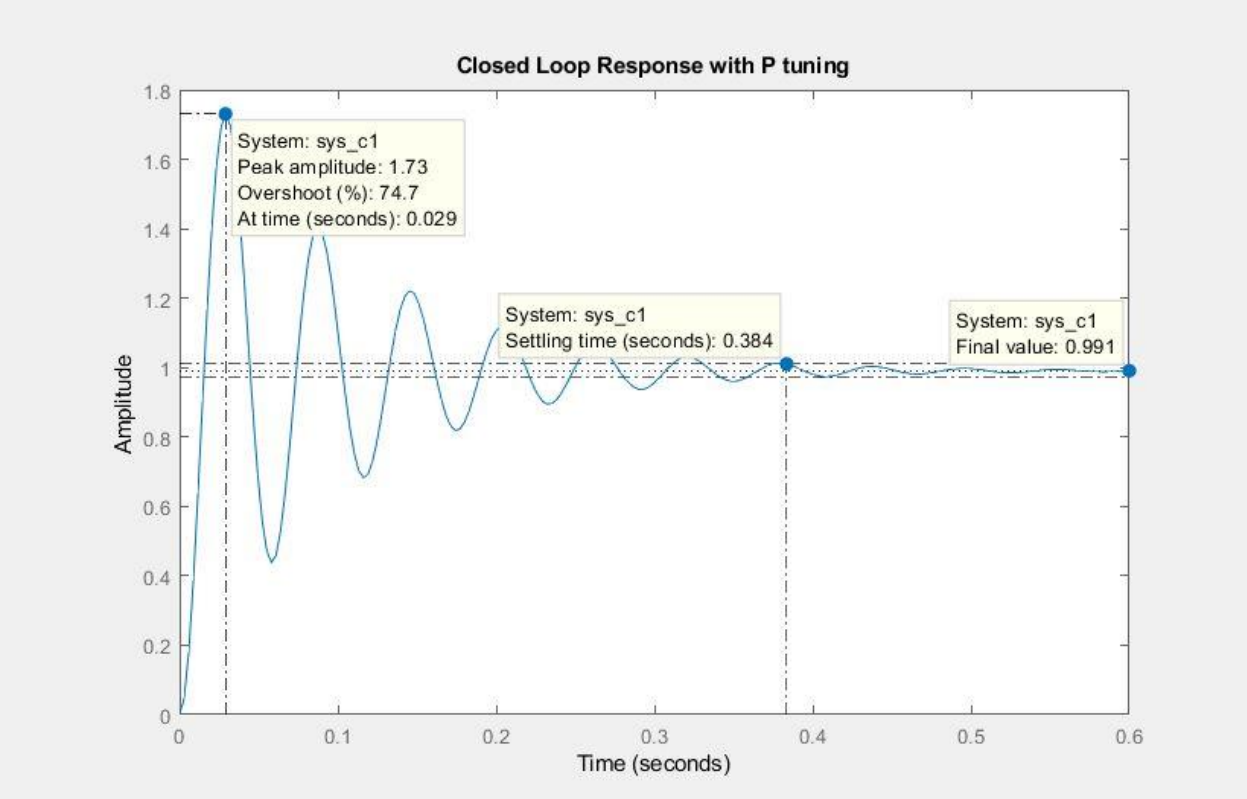

Figure 11. Closed Loop System Response with Proportional Tuning.

We can see from the response that we have achieved less than a 1% steady-state error, and our rise time is still within the design requirement. However, we now have a massive overshoot of 74.7%. To fix this, we now need to increase our derivative gain to counter this massive overshoot.

After trial-and-error manual tuning in MATLAB we were able to meet our percent overshoot requirement by tuning our derivative gain up to a value of $K_D = 1.4$. The new response with this PD controller is shown below:

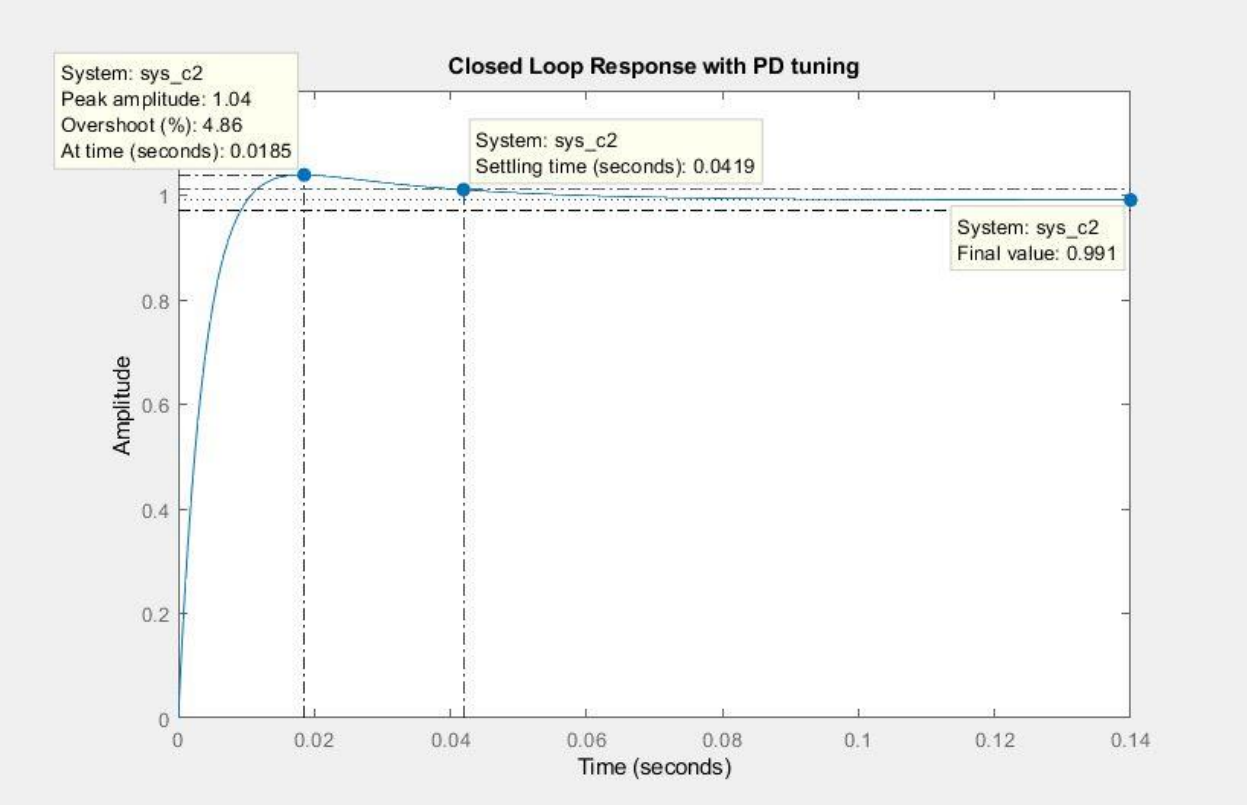

Figure 12. Closed Loop System Response with Proportional and Derivative Tuning.

This response meets all of our hypothetical design requirements without need of any integral gain tuning. For our typical problem, we were able to achieve a desired response by implementing a controller with the transfer function of:

$$C(s) = 58 + 1.4s \qquad \text{Eq. (29)}$$

To which our entire system has the new transfer function:

$$T(s) = \frac{\omega(s)}{V(s)} = \frac{0.07s+2.9}{0.00025s^2+0.075s+2.928} \quad \text{Eq. (30)}$$

This is vitally important, because now we have an analytical system that will allow us to design a PID controller for *any* unity feedback DC motor control system. All we would have to do is calculate the parameters of the hardware involved and calculate the control design objectives that we desire to accomplish.

CHAPTER 4: PHASE II - DESIGN

4.1 Mechanical Design Parameters

The mechanical design build of the RMC robot was performed by the University of Arkansas undergraduate robotics team. It is intended for this new build to last and perform for a multitude of future competitions, and not just this year's. It is because of this that analysis and optimization of the current build is necessary. In the field of the robot's motor control systems, we are required to base our design off the desired design objectives which were specified by the undergraduate competition team. It was advised to the team to pick design objectives that would be conservative, so that there was still adequate operation in the event that the robot would underperform its design objectives. The pertinent design objectives that the undergraduate team wanted to aim for are as follows:

- The weight of the robot when fully loaded with the BP-1 aggregate is estimated to be 200 lb
- The fully loaded robot should be able to navigate the length of the competition pit in around 10 seconds while at top speed
- The fully loaded robot should be able to reach top speed, from starting at rest, in 0.75 seconds

These objectives are indeed conservative in that for the first, the estimated fully loaded weight of the robot accounts for more than double the unloaded robot weight, and more than the unloaded robot weight plus the weight of a 100% volume load inside the excavation drum. The second is conservative in that a 10 second trip across the pit, assumed with a 20 second excavation cycle time, would allow for up to 15 cycles of operation for the robot to excavate and deposit. This

would drastically outperform many of the best robots in the competition, to which many complete an average of between one and two cycles. The third is conservative in that a 0.75 second acceleration/deceleration is extremely responsive and implies great mobility of the robot as compared to other robots in the competition.

Understanding torque and speed outputs for a motor used in acceleration is an important part of maximizing the efficiency of the machine. In the case of the robot used for the NASA RMC, optimal torque and speed outputs are necessary to understand in order to keep the robot accelerating appropriately, while also keeping consistent with effective output rotation of the motor. Given the constraints of the competition, a gearing ratio was added to all four motors in order to increase the torque output. Torque is the most crucial motor output quantity that will assist in acceleration on the loose Martian surface terrain, and to reach top speed quickly. Linear velocity is defined as

$$v = \frac{L}{t} \quad \text{Eq. (31)}$$

So, we wish for the robot to reach the desired linear speed of 2.4 ft/s in after a time period of 0.75 s

The impulsive force needed to move the robot at this speed (assuming no slip) is calculated using the principle of impulse and momentum

$$mv_1 + \int_{t1}^{t2} \Sigma F \partial t = mv_2 \qquad \text{Eq. (32)}$$

Where the initial momentum and initial time are both zero since the robot is starting from rest. Over a span of 0.75 seconds, the force F is applied to the robot in order to get the robot up to top speed. From this, we determine that the force needed to accelerate the robot in the desired manner is F = 640 lbf.

The minimum shaft torque that is necessary to produce this force can be obtained from the relation

$$T = F * r \qquad \text{Eq. (33)}$$

Where r, the wheel radius, is 0.5 ft. The required torque is T = 320 ft-lb.

With regard to the motors, the undergraduate team was tasked with identifying an acceptable aftermarket motor to use for the robot's drivetrain. They selected the AndyMark am-0255 2.5" CIM motor, which are very commonly used in personal recreational robotics projects. The main reasons that the team selected this model was because they are inexpensive, small, lightweight, 12V DC brushed motors that are capable of reaching much higher torques at moderate current draws. The main competitor in the selection process was the Vex Robotics 775 Pro motor, which was designed to operate at much higher speed and much lower torques. The undergraduate team wanted to match up the desired design operating point for the robot with the peak power point from the motor's performance specifications. It was assumed that a gearing ratio would have to be implemented on the motor shaft in order to reach the torque requirement. The peak power torque given by the motor performance specifications is 171.7 oz-in. They chose to implement the gear reductions of 7:1, followed by another 7:1, followed by a 2:1, which results in a total 98:1 gear reduction of the motor shaft. The required torque of 320 ft-lb is achieved by four equally distributed motors is achieved by each wheel shaft supplying 80 ft-lb of torque. The relationship to the individual motor shaft torque and the wheel torque is

$$motor\ torque * 98 = calculated\ torque\ per\ wheel \qquad \text{Eq. (34)}$$

From which we find that the required torque supplied by each motor is 0.816 ft-lb or 156.67 oz-in, which is very close to the peak power point for the AndyMark motor performance curve, which is shown below:

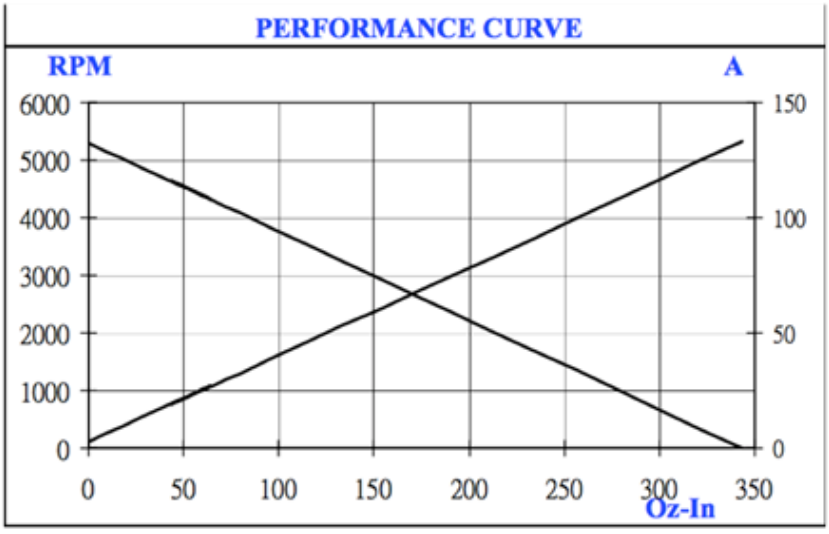

Figure 13. Typical Performance at 12V D.C. for the AndyMark 2.5" CIM Motor. [11]

TYPICAL PERFORMANCE @ 12 Vdc					
	TORQUE	SPEED	CURRENT	POWER	EFF'CY
	Oz-In	RPM (±10%)	A MAX	Wo	%
FREE LOAD	0	5310	2.7	0	0%
NORMAL LOAD	64.0	4320	27	205	63%
@MAX EFFICIENCY	45.0	4614	19.8	154	65%
@MAX POWER	171.7	2655	67.9	337	41%
@STALL	343.4	0	133.0	0	0%
HIPOT: 600 Vac/0.5 mA/1 sec					
INSULATION RESISTANCE: > 10.0 M Ohm MIN					
INSULATION CLASS: B					

Figure 14. Commonly Referenced Operating Points for AndyMark 2.5" CIM Motor. [11]

At this operating point of 156.67 oz-in, the motor is operating at around a 43% efficiency ratio and the output shaft velocity is around 2400 rpm [4]. This shaft velocity will supply the required torque but only result in a linear speed of 1.3 ft/s due to the gear reduction, which is only a little more than half of the original desired speed. However, this operating speed will still give the possibility of performing over 10 excavation and deposition cycles. The undergraduate team deemed this acceptable and proceeded forward with the build. It is noted in this work, that better motor performance (speed, efficiency) would be achieved by simply implementing a greater gear

reduction ratio. This is implied by the slope of the performance curve. For every 50 oz-in of torque that you save from not having the motor supply it, you stand to gain nearly 1000 rpm of motor shaft speed.

4.2 Modeling the D.C. Motor

Referring to the theory that was outlined in Phase I, we first needed to identify the key parameters that are necessary to model our DC motor. The pertinent parameters are:

- J – Polar moment of inertia of the rotor (Nms^2)
- b – Motor viscous friction coefficient (Nms)
- K – EMF/Torque constant ($\frac{Nm}{A}$) **In S.I. units, these two are one in the same**
- R – Internal resistance of the motor (Ω)
- L – Internal inductance of the motor (H)

We need all of these parameters in order to develop the output speed transfer function of the DC motor, mentioned in Phase I, as:

$$P(s) = \frac{\omega(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad ({}^{rad}/_{Vs}) \quad \text{Eq. (35)}$$

An engineering representative from AndyMark, the company who manufactures the motors which we are using, provided the author with the approximate dimensions and weight of the rotor through a personal communication [5]. Using this provided information, a simplistic model of the rotor geometry was created using SolidWorks in order to provide a good estimate for the polar moment of inertia. An image of the created SolidWorks model is shown below:

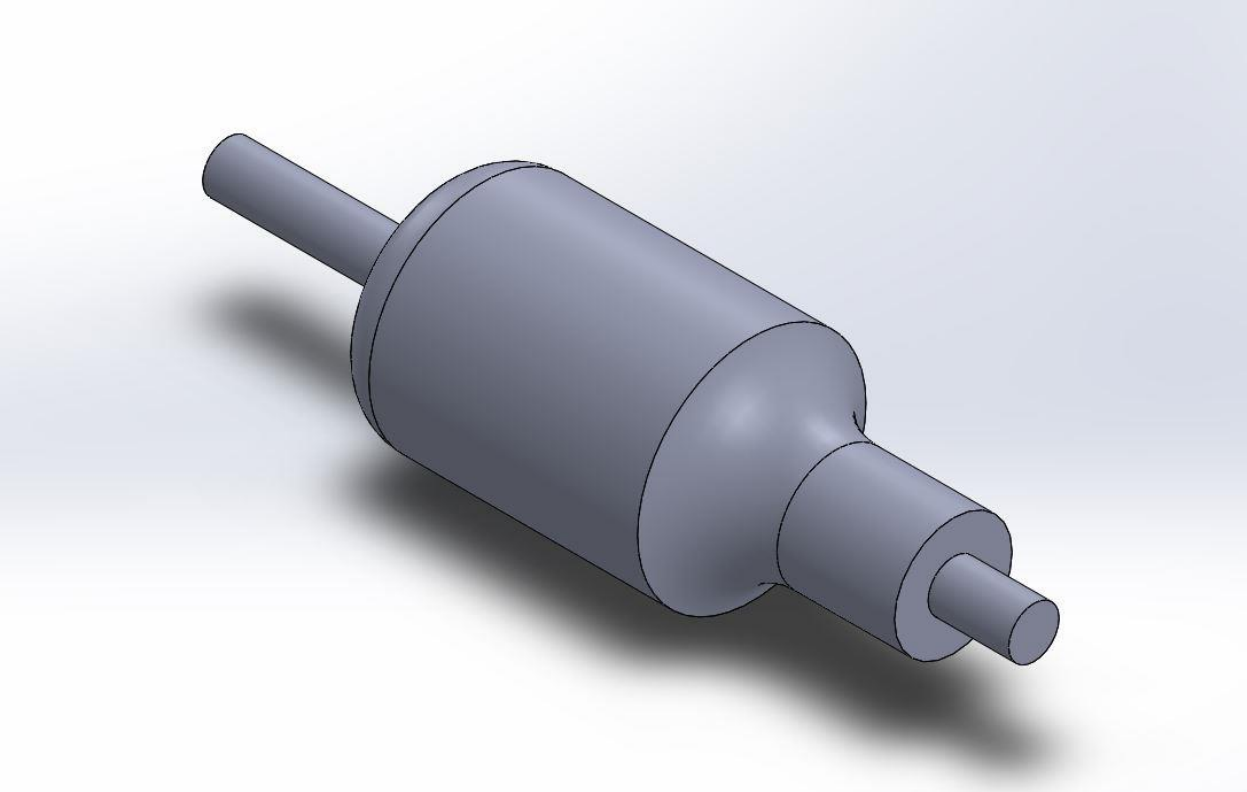

Figure 15. SolidWorks Model of AndyMark 2.5" CIM Rotor Geometry

In SolidWorks, we were able to remove an automatically assigned material density and override it to align with the provided weight estimate for the rotor. Using the mass properties analysis function, SolidWorks generated a polar moment of inertia of:

$$J = 0.29\ lb * in^2 = 0.003341\ Nms^2 \qquad \text{Eq. (36)}$$

We chose to use a SolidWorks model rather than calculating the polar moment of inertia analytically because it was much easier to do so, and we wanted to avoid risk of computation error due to the moderately complex geometry of the actual rotor. In all reality, when it comes to motor control design, it is not paramount for this parameter to be extraordinarily exact. In practice, the amount of energy and control effort that is needed to overcome the inertial effects of the motor itself are insignificant as compared to the amount of energy and control effort that is needed to move the actual load in question. Considering that this value is similar to a typical

value for J as stated in [2], we deemed this approximation acceptable and continued on with our efforts.

As previously mentioned in Phase I, the torque of a DC motor is proportional to the current by the torque constant. This relationship is formulated as:

$$T = K_t i \qquad \text{Eq. (37)}$$

And this relationship holds true for any point throughout the motor's operational envelope. In S.I. units, this torque constant is equivalent to the proportional constant that relates the back EMF to the angular velocity of the rotor. Using this relationship, we were able to obtain a torque constant of:

$$K = 0.0170 \ \frac{Nm}{A} \qquad \text{Eq. (38)}$$

When no external load is applied on the motor, any torque that the motor is producing is doing so to overcome frictional (and inertial) effects. We know from [2], and from any dynamic system that is subject to damping, that the frictional effects of the system are related to the velocity. In the case of the DC motor, the motor viscous friction coefficient is a parameter that linearly relates the no-load torque to the no-load angular velocity. The relationship is formulated as:

$$T_{NL} = b\omega_{NL} \quad \text{Eq. (39)}$$

Where

$$T_{NL} = K_t i_{NL} \quad \text{Eq. (40)}$$

And the subscript NL refers to the no-load current, torque, and angular velocity. From the performance specifications provided by AndyMark, the no-load current is 2.7 A and the no-load angular velocity is 5310 rpm. Using this data point, we were able to determine the motor viscous friction coefficient for our motor:

$$b = 8.25 * 10^{-5} \ Nms \qquad \text{Eq. (41)}$$

The internal resistance and internal inductance of the motor had to be evaluated at our desired operating point on the AM-0255 performance curve. The relationship for inductance and electrical work is:

$$W_{op,elec} = \frac{1}{2} L_{int} i_{op}^2 \quad \text{Eq. (42)}$$

The relationship of electrical work and mechanical work is defined by the motor efficiency, η, and the relationship is as follows:

$$W_{op,mech} = \eta_{op} W_{op,elec} \qquad \text{Eq. (43)}$$

At our operating point from the AM-0255 performance specifications, the mechanical work is 156.67 oz-in, or 1.1603 Nm, and our motor efficiency is approximately 0.42. Our operating current is approximately 65 A. Using these parameters, we were able to determine the internal inductance as:

$$L_{int} = \frac{2W_{op,elec}}{i_{op}^2} = 1.247\ mH \quad \text{Eq. (44)}$$

Using Kirchhoff's voltage law for the armature circuit, we established the relationship:

$$V_s = e_{op} + i_{op} R_{int} \quad \text{Eq. (45)}$$

Where e is the back electromotive force from the motor. As previously established:

$$e_{op} = K_e \omega_{op} \qquad \text{Eq. (46)}$$

Rearranging this to solve for the internal resistance:

$$R_{int} = \frac{V_s - K_e \omega_{op}}{i_{op}} \qquad \text{Eq. (47)}$$

The source voltage (nominal) is 12V, the operating angular velocity is 2400 rpm (251.33 rad/s), and the operating current is approximately 65A. We obtained an internal resistance value of:

$$R_{int} = 0.119\ \Omega \qquad \text{Eq. (48)}$$

This puts our RL time constant at approximately 10ms, which is not uncommon among small DC motor parameters.

We have now obtained all the necessary previously mentioned parameters in order to make a plant model for the DC motor. Considering that the system in question requires model-based design and simulation, we chose to create Simulink block models for this system. Using the theory behind commonly used DC motor models as discussed in the Model Representation section of Phase I, along with helpful information that can be found in [6], we created a single-input voltage to single-output angular speed model for our DC motor as shown in the diagram below.

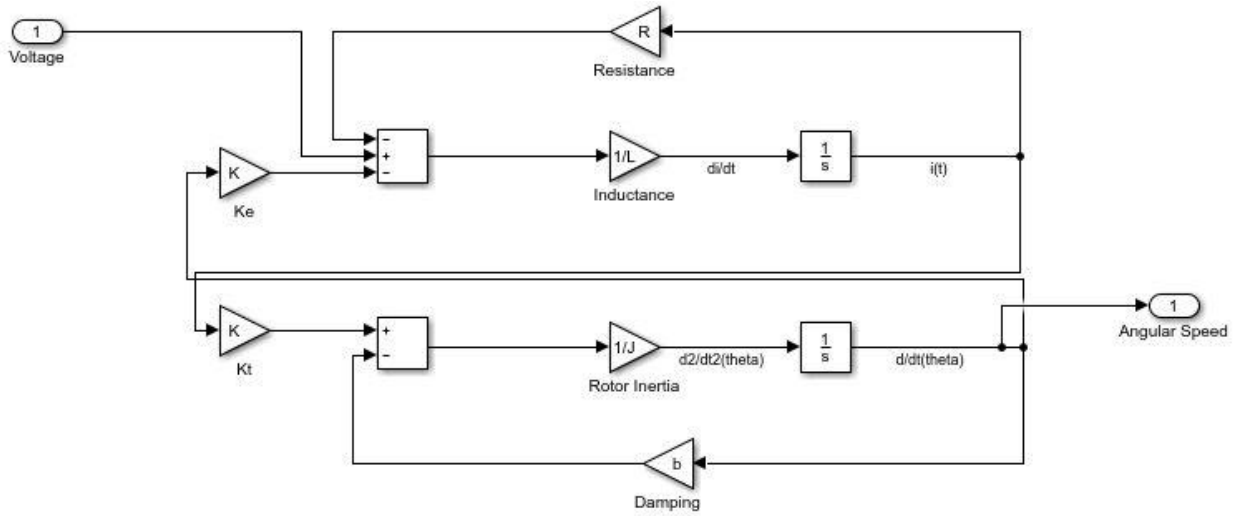

Figure 16. Simulink Model Representation of D.C. Motor

The transfer function of this model was converted into a zero-pole-gain model transfer function by using MATLAB. The transfer function of the plant (our DC motor) which was utilized in all subsequent simulations was:

$$P(s) = \frac{4080.4}{(s+94.7)(s+0.7574)} \qquad \text{Eq. (49)}$$

We can now use this Simulink model of the motor to run simulations of the response for the purpose of designing a PID controller that will adequately serve our operating purposes. We

also intend to develop a repeatable process for tuning the gains of a PID controller that would work for almost any DC motor control application. Referring back to the techniques established in Phase I, we created an uncompensated open-loop response model for our DC motor as shown below:

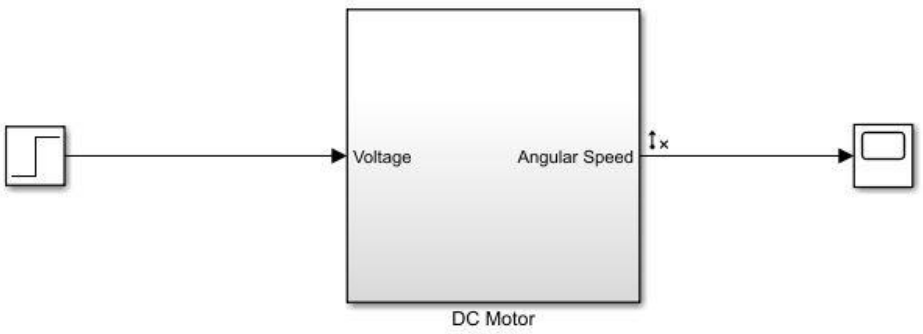

Figure 17. Simulink Uncompensated Open Loop Response Model.

where the contents of the large DC Motor block are those displayed in Figure 16. The block on the left-hand side represents a generic 1V step input applied at time t=0, and the block on the right-hand side represents a scope that would display the output response of the model. We ran the simulation of this uncompensated system and obtained the response shown below:

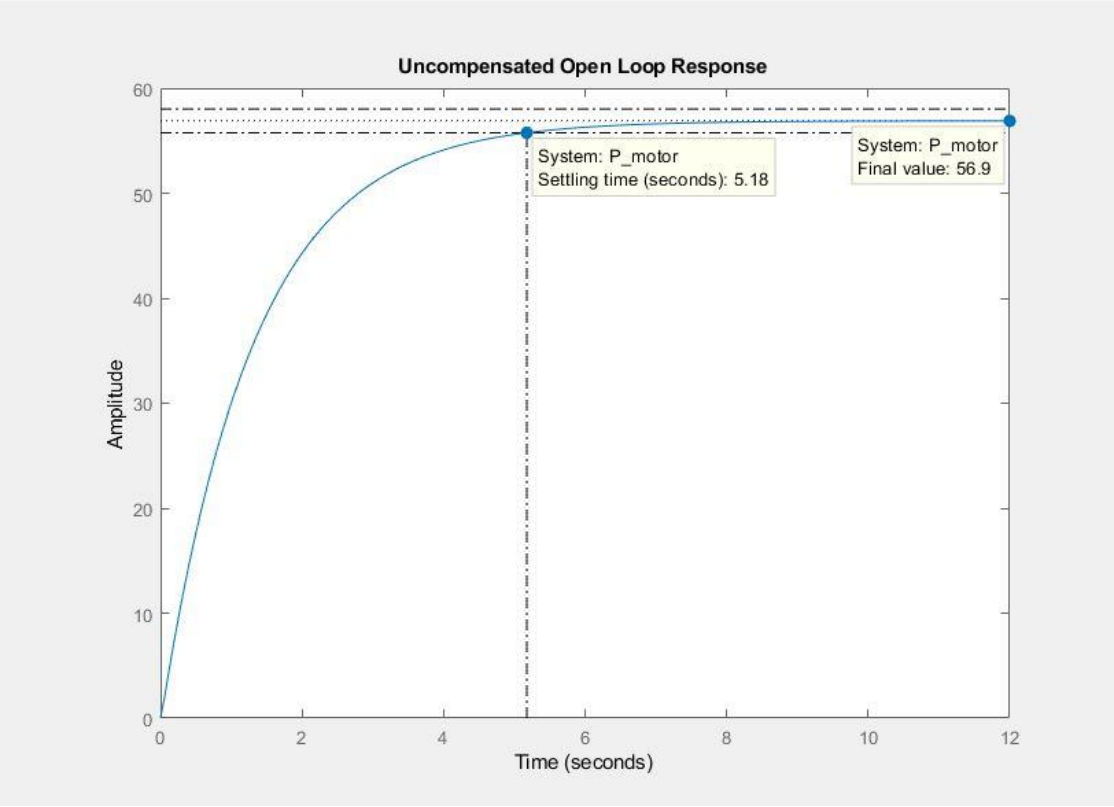

Figure 18. Uncompensated Open Loop Response of D.C. Motor

4.3 Controller Design

The primary goal of this work is to generate a repeatable method for PID controller design for systems which utilize DC motors. The first step that is necessary to begin this process is to identify which response characteristics that we wish to design around. We keep consistent with the previously used design constraints from Phase I which are:

$$t_s \leq 0.75\ s \quad \text{Eq. (50)}$$

$$PO\% < 5\% \quad \text{Eq. (51)}$$

$$e_{ss} < 1\% \quad \text{Eq. (52)}$$

However, now that we have encountered a real-world physical system, we have to go through the process of actually defining the nominal steady-state value of the desired response for the open loop system. The physical interpretation of the uncompensated response shown in Figure 18

states that for a 1V step increase of the input (or, in our case, an increase in current draw from the motor that corresponds with a 1V step increase), the response of the output angular speed of the motor shaft will increase by 56.9 rad/s. It can be seen from the performance specifications for the motor that the relationship between current draw from the motor and output angular speed is linearly proportional and positively correlated. This condition also holds true for all operating points within the motor's operating envelope. Additionally, for these systems, an increase of the magnitude of the step only significantly affects the steady-state value of the response and does not significantly affect the settling time and overshoot characteristics of the response. It is because of these system behaviors that we can accurately predict response behavior from larger input steps, such as the 12V operating point step, according to a unitary step response. As previously determined, the output shaft angular speed at the desired operating torque is around 2400rpm. If we need the motor shaft speed to increase 2400rpm over a 12V step at operating conditions, then this is the same as needing a 200rpm increase over a 1V step given that we have established that the system performance constitutes linearity between current and angular speed, and therefore voltage input (again, which coincides with a change in current draw to the motor) and angular speed. *We wish to change the proportionality of the response so that every input single step increase/decrease coincides with an output increase/decrease of 200rpm, which is approximately 21 rad/s.* So, we need to add proportional control to reduce our steady state output from 56.9 rad/s to 21 rad/s.

The method for determining what a proportional gain value on a PID controller should be can simply be described by the following relationship:

$$K_P = \frac{Desired\ Response\ Value}{Uncompensated\ Response\ Value} \quad \text{Eq. (53)}$$

To which we determined our desired response to be 21 rad/s and our uncompensated response to be 56.9 rad/s. Using this relationship, we obtain

$$K_P = 0.37 \qquad \text{Eq. (54)}$$

In order to verify the results of this proportional gain relationship, we added a PID controller block in series with our plant model in Simulink. The resulting system is shown below:

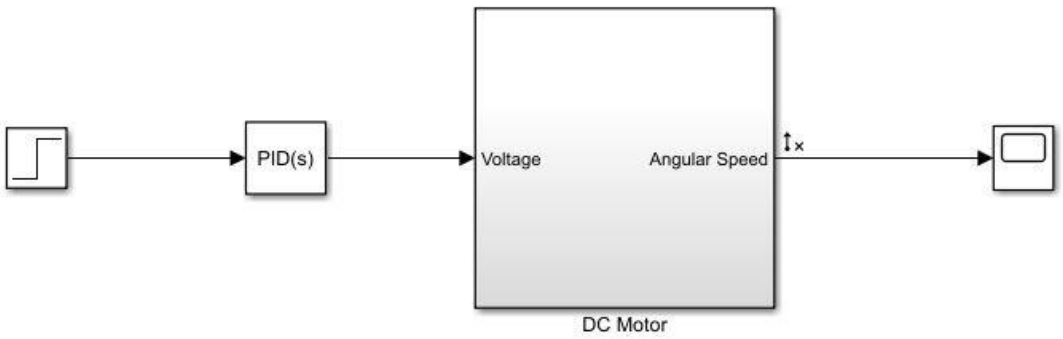

Figure 19. Open Loop Simulink Model with Proportional Control.

Attributing a P value of 0.37 to our PID block and running the simulation, we obtain the response shown below:

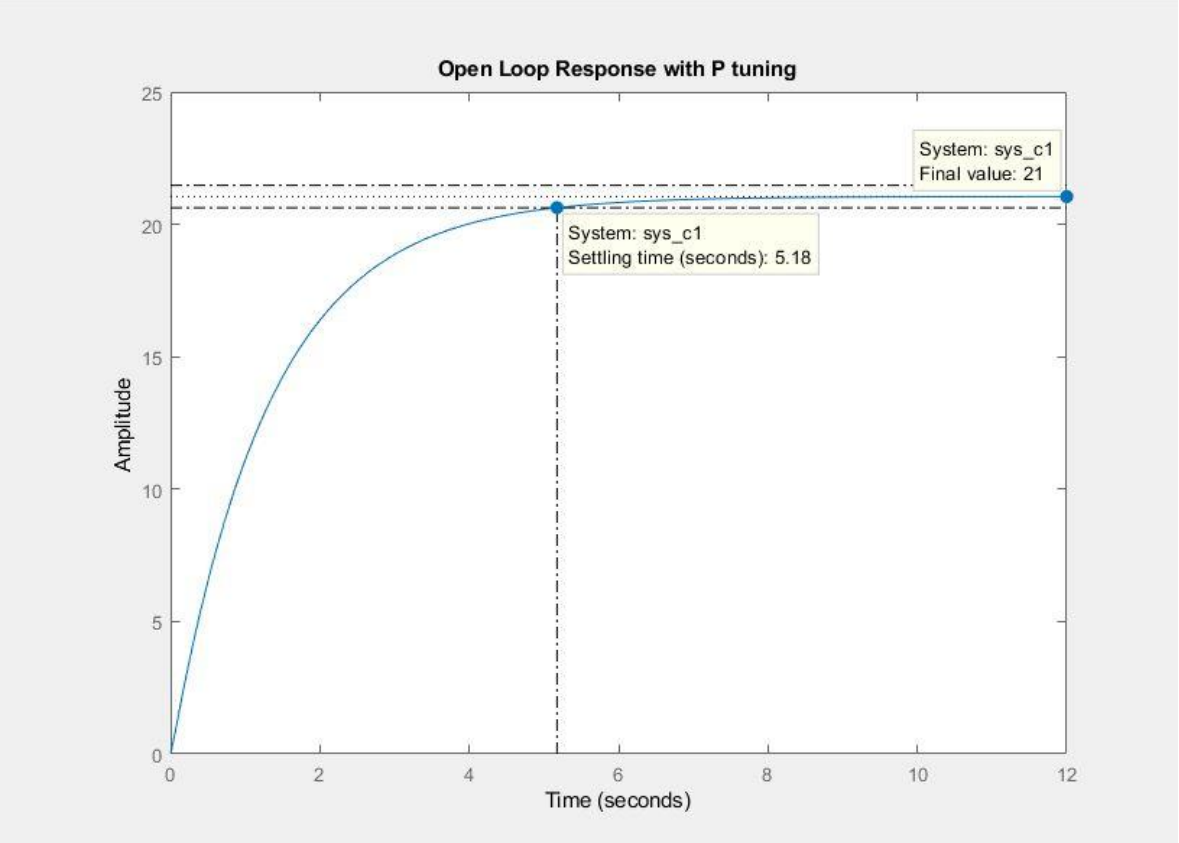

Figure 20. Open Loop Response with Proportional Control.

Which you can see that now, the steady-state value of the open loop response has converged to our desired output response value. The takeaway steps from this portion of the controller design into the general procedure that we wish to establish are:

- ***Determine the desired steady-state open loop output value for a unit step that coincides linearly with the desired steady-state value at the operating step magnitude***
- ***Tune your proportional gain to the value that caused convergence between the open loop steady-state response, with proportional control, to the desired steady-state output value***

At this point in the controller design process, we have met the steady state error criterion of 1%. However, we encounter two problems with the system when evaluating the other two design criterion. As you can see the settling time obtained with only proportional gain is 5.18 seconds

when we would like to achieve a settling time of 0.75s. If we were to follow the characteristics shown in Figure 10 for how PID gain tuning affects response characteristics, we would do so by adding derivative gain to the system. Many simulations were performed at this point and the results of these simulations showed that adding derivative and integral gains to the open loop system caused undesirable results. The addition and increasement of derivative gain in the open loop system did reduce the settling time, however never to achieve an adequate settling time. The smallest settling time achieved by adding derivative gain was about 2.2 seconds. In this process, we also observed that increasing the derivative gain of an open loop system causes the overshoot of the system to increase without bound. This is highly problematic and undesirable. We also observed that adding any sort of integral gain to the open loop system caused the response to not follow a step path but now a ramp path, which means the response increased with time and without bound. Any increase in integral gain added to the system only resulted in an increase of the slope of the ramp response.

With regards to the overshoot criterion, our open loop response does not exhibit more than a 5% overshoot which by first glance would imply that we have achieved a good result with respect to this criterion. Our open loop response actually never displays *any* overshoot, and this implies that the system experiences overdamping. In theory, one may think that overdamping is not that big of an issue because the response of the system indicates stability. However, in practice, overdamping can have highly adverse effects to the systems mechanical components and have a rougher time handling a smooth ride at lower operating speeds. In control design we don't want an exceedingly overdamped system, nor an exceedingly underdamped system, we want a "well" damped system. In essence, a "well" damped system can be defined in a multitude

of ways, but for this work we will choose to define our damping terms based on the number of displayed overshooting peaks that exceed the 2% settling criterion.

- Underdamped – The response of a system to a step input displays multiple overshooting peaks which exceed the 2% settling criterion
- Overdamped – The response of a system to a step input displays no overshooting peaks which exceed the 2% settling criterion
- "Well" damped – The response of a system to a step input displays exactly one overshooting peak which exceeds the 2% settling criterion

In order to fulfill the response design criterion by implementing derivative and integral gains, it is necessary to now utilize a closed loop unity feedback system. The characteristics of unity feedback control systems have been previously discussed in Phase I and a sample figure of a closed loop unity feedback system is shown in Figure 9. This constitutes another takeaway step to add to the general procedure as:

- ***Close the loop (unity feedback) in order to get desired response characteristics utilizing derivative and integral gains when proportional control alone is not sufficient.***

It is to be noted, upon converting an open loop system to a closed loop system, that the steady-state response value actually changes from a specific magnitude (21 rad/s, as we found before) to a unitary magnitude. This is representative of the difference in the desired step response with respect to the error signal, to which we wish to always minimize the error signal in control design. So, keeping in mind a certain proportional gain already applied, achieving a steady-state value of 1 in a unity feedback system represents achieving the desired steady-state value in the open loop. This tends to an important corollary of our control strategy: Use the open loop response to tune the proportional gain to reach the desired output value. Once you have arrived at

this value, you do not alter the proportional gain after closing the loop since this alteration will serve no significant effect on the closed loop response characteristics. Changing the proportional gain will, however, affect the actual magnitude of the response but will not be represented appropriately in the error signal. Conversely, you only tune derivative and integral gains after utilizing a feedback loop in order to achieve certain response time and steady-state error criterion. Adding derivative and integral gains to an open loop system will yield undesired response characteristics.

The closed loop system was modeled in Simulink in order to analyze the new response. A block diagram model of the Simulink closed loop system is shown below:

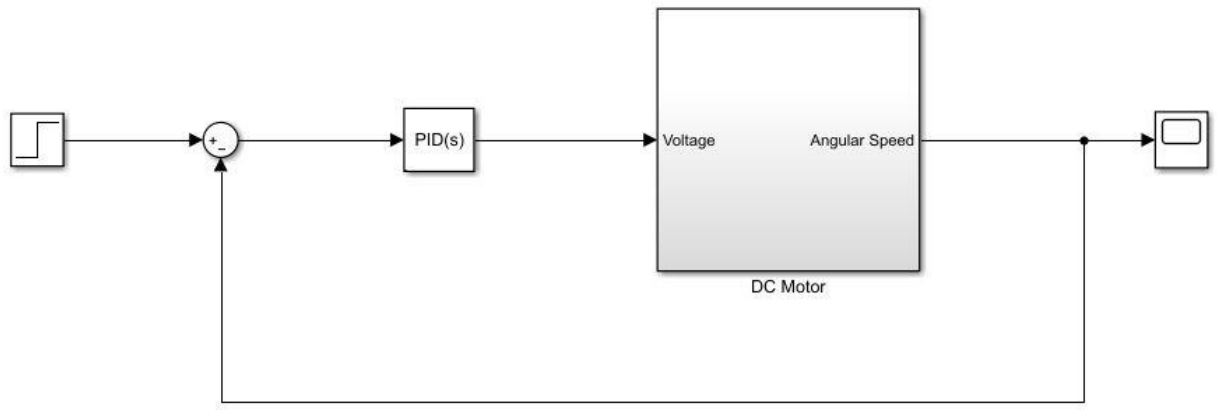

Figure 21. Simulink Unity Feedback System Model.

Simply closing off the loop allows for some changes in the response time characteristics, because the system now is acting to correct an error signal. Keeping our previous proportional gain of 0.37 and running the simulation without adding derivative or integral gains, we obtained a response as shown below:

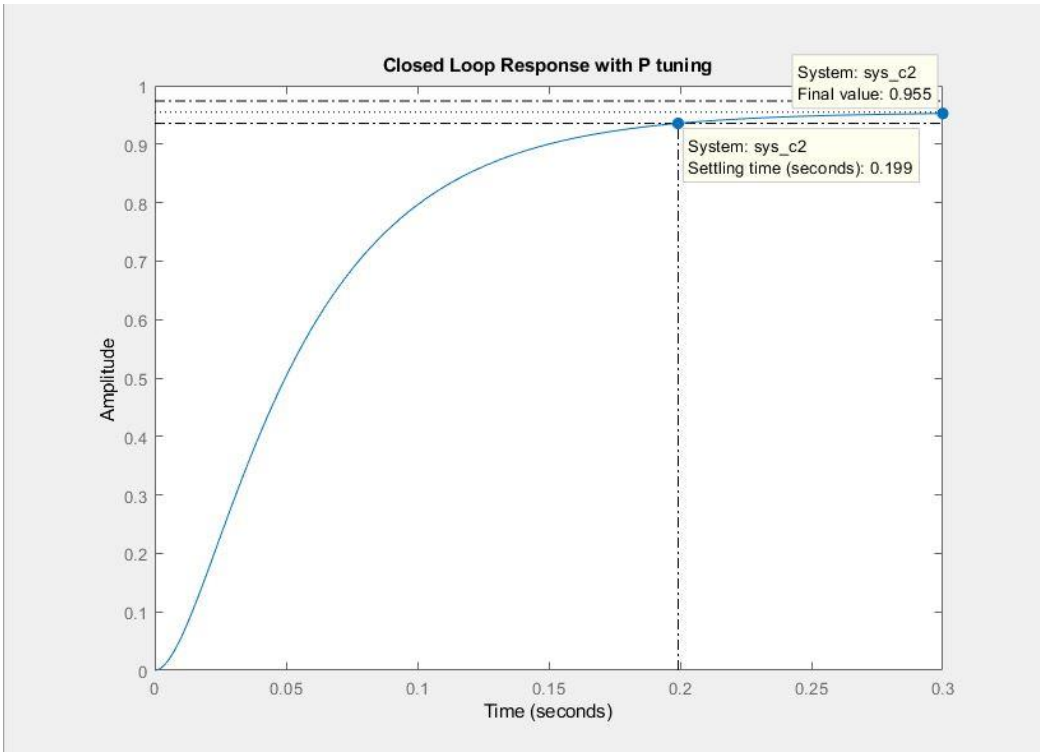

Figure 22. Closed Loop Response of D.C. Motor with P Tuning.

We can see that closing the loop without adding any additional gains does provide for a much more rapid response of the system. Where we previously failed to achieve a settling time below the design criterion, we now have easily achieved by closing the loop without adding any additional control gain. Our steady-state error, however has increased from zero to 4.5% which we will have to rectify by introducing integral gain. Additionally, we are still faced with the problem of an overdamped system response. This can also be rectified by introducing an integral gain, as following the response characteristics shown in Figure 10.

After many trial-and-error simulations, we were able to achieve a desired "well" damped response while fitting into our settling time requirement by implementing an integral gain of 0.61. Adding an integral gain (of 1 or higher) causes an elimination of steady-state error at the cost of an increased settling time and an increased overshoot. Ideally, we would use integral gain solely to eliminate steady state error and then rectify the settling time requirement by adding a

derivative gain. However, when it comes to this particular system, it was observed through many trials that adding any form of derivative gain resulted in a system response that was underdamped. To fix this problem, we started with a unitary integral gain and decreased it below 1, which in turn reduces the settling time and overshoot with respect to the integral gain unitary system. After this tuning was performed, we arrived at our final desired closed loop response:

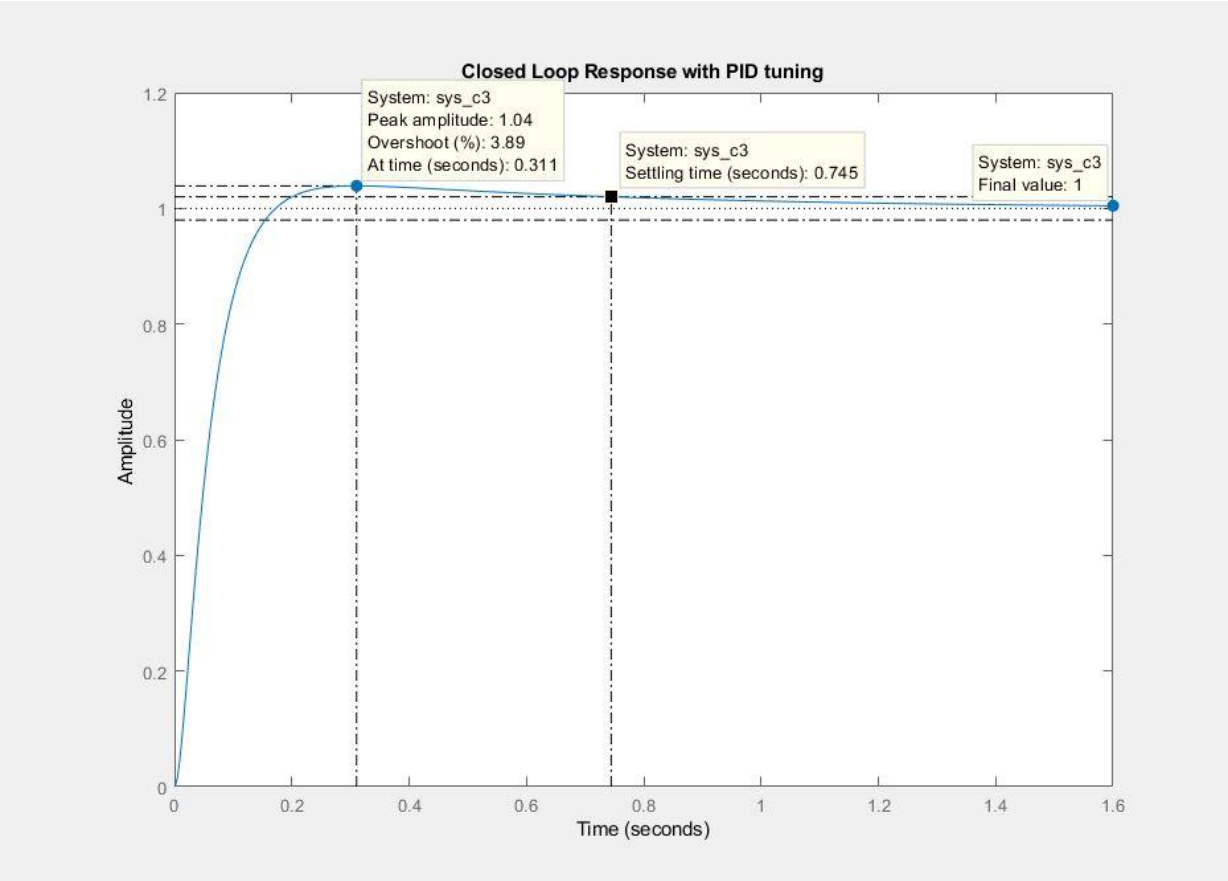

Figure 23. Closed Loop Response with PID Tuning.

As you can see here, this response is "well" damped as it has exactly one overshooting peak above the 2% settling criterion (the two symmetrical dotted lines about 1). The overshoot is 3.89% which is less than our 5% criterion, the settling time is 0.745s which is below our 0.75s criterion, and our steady state error is zero. We were able to obtain this desired response by using the PID gains of:

$$K_P = 0.37 \qquad \text{Eq. (55)}$$

$$K_I = 0.61 \qquad \text{Eq. (56)}$$

$$K_D = 0 \qquad \text{Eq. (57)}$$

Which results in the transfer function of the controller to be (in zero-pole-gain format):

$$C(s) = \frac{0.37(s+1.649)}{s} \qquad \text{Eq. (58)}$$

And the resulting transfer function of the entire closed loop system to be (in zero-pole-gain format):

$$T(s) = \frac{1509.8(s+1.649)}{(s+74.74)(s+18.96)(s+1.757)} \qquad \text{Eq. (59)}$$

The content of the final steps of the PID tuning process was combined because the derivative tuning process was evaluated and analyzed, but eventually excluded from this specific case because derivative tuning achieved undesirable results. The final takeaway steps have been added to the end of the accumulation of our established general procedure, which is shown below:

- ***Determine the desired steady-state open loop output value for a unit step that coincides linearly with the desired steady-state value at the operating step magnitude***
- ***Tune your proportional gain to the value that caused convergence between the open loop steady-state response, with proportional control, to the desired steady-state output value***
- ***Close the loop (unity feedback) in order to get desired response characteristics utilizing derivative and integral gains when proportional control alone is not sufficient.***
- ***Implement a derivative gain to the closed loop system to reduce the response time characteristics to meet the design criterion***
- ***Implement an integral gain to the closed loop system to eliminate the steady state error (this will increase the overshoot and settling time)***

- ***Readjust the derivative and integral gains iteratively, until an ID combination is found that will achieve all of the required design criterion.***

We also plotted the control effort to verify that the requirements of the system would stay beneath the allowances of the utilized hardware. The plot of the control effort for our closed loop PID system is shown below:

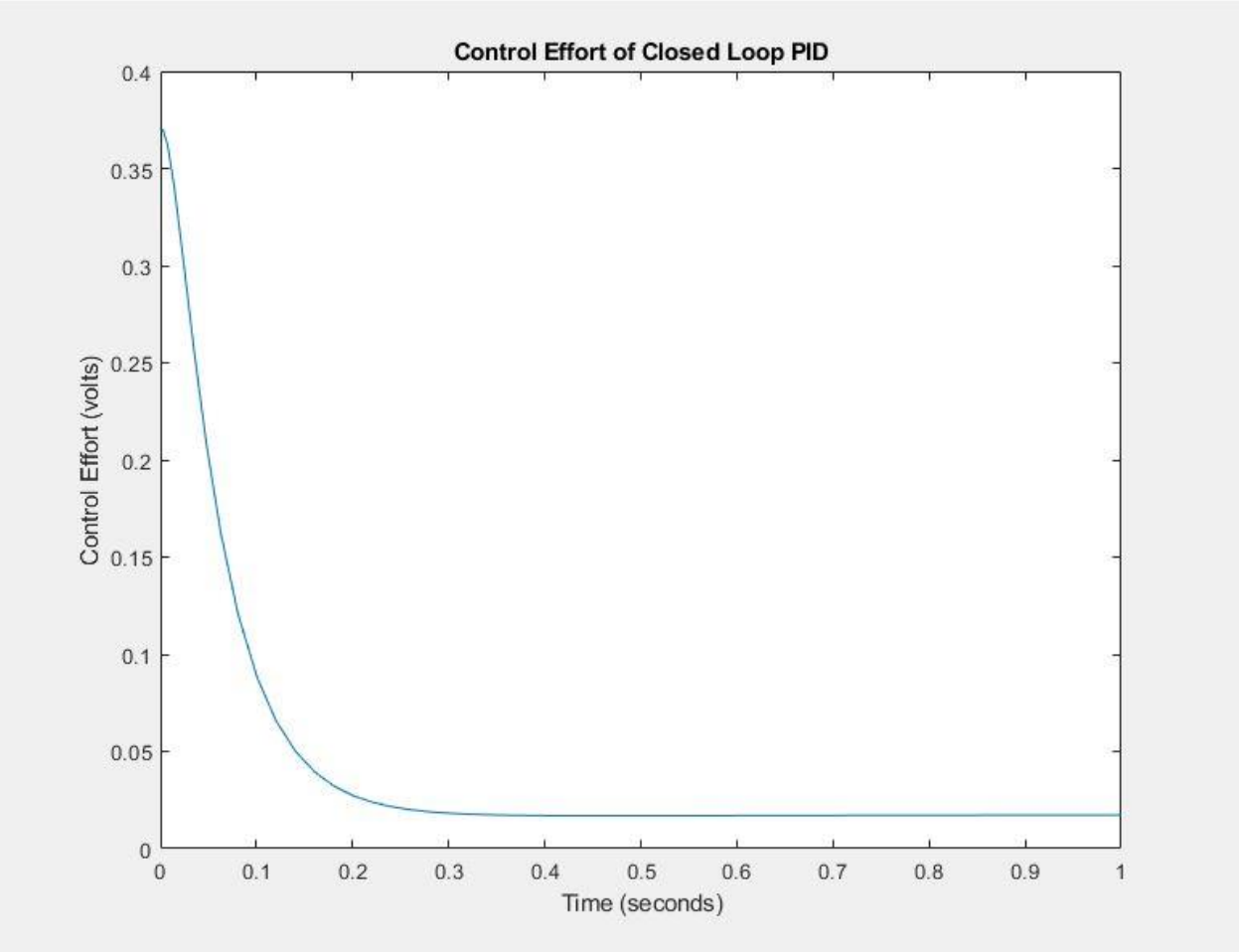

Figure 24. Control Effort of Closed Loop PID System.

4.4 Control Strategy

The analysis and processes up until this point have only dealt with a subsystem of the RMC robot that follow the flow of operation from a single output from the microcontroller (as input to the motor controller) to the output shaft of the motor. The block diagram model of this

subsystem has been previously presented in Figure 21, with the step signal representing the input from the microcontroller and the scope would represent the scope connection to the two motor terminals. It is of importance to understand that the entire driving system involves four of these subsystems running in parallel, each receiving individual inputs from the microcontroller, as well as the gear reductions following the motor shafts and rotary encoder feedback loops to the microcontroller from each individual subsystem. A block diagram of the entire robotic drivetrain control system is shown below:

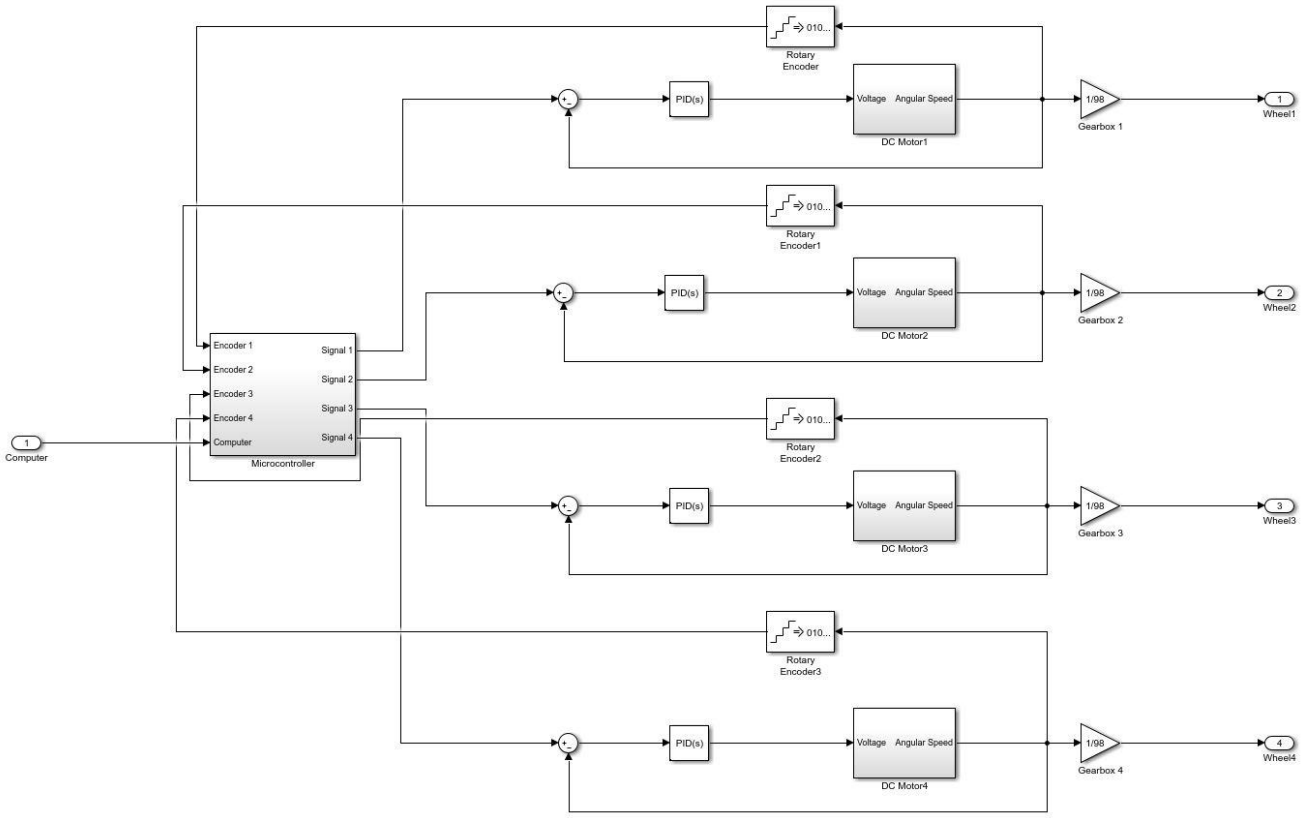

Figure 25. Block Diagram of Entire RMC Drivetrain Control System.

The previous year's robot utilized aftermarket components (Arduino MEGA, Sabertooth 2x32) with serial data transmission to adequately perform the desired tasks of the robot. A lot of the optimization for control strategy using aftermarket components deals with efficient programming execution and provided manufacturer software optimization, which falls more in the field of

computer science and is not of the author's field of expertise. Considering that this report pertains to the mechanical performance design of a "custom" controller, we will discuss this portion of control strategy in very general terms of how the controller operates whether or not it is taking digital microcontroller input or analog input.

The commonly used signal range to motor controllers of this size is 0-5V, whereas the main power of the motor controller, in our case, will be operating at 12V. Since we wish for the robot motors to operate in both forward and reverse, we wish to set up a symmetrical motor operation range about the midpoint of the signal range. This means that with an input signal of 0V the motor will be running at full speed in reverse, and at 5V the motor will be running at full speed forward, with the "stop" signal set at 2.5V. A gradual increase of motor forward operation speeds will be aligned to the range of 2.5V – 5V of the signal, with the reverse speeds identically aligned to the range of 2.5V – 0V of the signal.

The actual alignment of the two sets is utilized in a simple linear conversion parameter we denote as:

$$\emptyset = \frac{Current\ Output\ to\ Motor\ (A)}{Signal\ Input\ to\ Controller\ (V)} \qquad \text{Eq. (60)}$$

Which is essentially the slope of the linear relation. However, we need to align the relationship with more than one operating point, since the signal input range is not 0V – 5V, but rather two symmetrical 2.5V ranges. For our system, we wish to determine the actual parameters we would be utilizing. Our previous analysis and design sections have assumed the robot operating at full speed and fully-loaded, with conservative estimations taken. It is then desired that our maximum forward/backward current output magnitude will be the current at this operating point, 65 A. We wish for this value to correspond to the maximum signal input of 5V. We encounter a slight problem when applying this to the "zero" condition versus the first forward operating point. The

problem is that we wish for the "zero" condition to not supply any current to the motor to ensure a full stoppage. However, from the performance specifications of the motor, we also wish that our first forward operating point to be the no-load operating current of 2.7A. This would induce a discontinuity in our linearization if we so wished to include the "zero" condition within it. However, we simply will allocate two bit resolution ranges around the point 2.5V to correspond with the separate "zero" condition and linearize the system beginning with the point after. For an 8-bit system, our resolution for a 2.5V input signal range would be around 10mV. We will allocate the space from 2.49V – 2.51V from the signal to set the motor to the "zero" condition. Now we can define that we want the no load current of 2.7A to align with the signal voltage of 2.51V. In doing this, we obtain our linear conversion parameter to be:

$$\emptyset = \frac{65A - 2.7A}{5V - 2.51V} \approx 25\,{}^{A}/_{V} \qquad \text{Eq. (61)}$$

Or, every full Volt of signal input change should correspond to a 25A change in current output to the motor. This also gives us an incremental output resolution of 0.25A to the motors. Given a constant torque, we know that an increase in current supplied to the motors will result in an increase in shaft speed from the motor. We also know from our previous analysis that a change in proportional gain of the controller directly constitutes a change in the steady-state speed of the response at that input value. The main control strategy that we wish to implement in future iterations of the robot is that:

- ***We can achieve speed control through proportional gain scheduling on the controller, so we wish to use our input control signal to change the proportional gain on the controller.***

CHAPTER 5: PHASE III – OPTIMIZATION

5.1 Gear Reduction Ratio

It was previously stated in the system modeling section of Phase II that better motor performance (speed, efficiency) would be achieved by simply implementing a greater gear reduction ratio. The problem with the undergraduate team's design decision is that they chose to design the system around the max-power operating point of the motor, instead of choosing a point closer to the max-efficiency operating point of the motor. The author hypothesizes that if we implement a change to the gear reduction ratio, we will have more efficient motor performance and obtain a fully-loaded full speed that is closer to the original design objective than that previously observed.

It was also previously determined that at the fully-loaded operating condition we need each wheel shaft to supply 80 ft-lb (15,360 oz-in) of torque to accelerate in the desired manner. If we design around the max efficiency operating point, we would need a 240:1 gear reduction ratio, which would necessitate an implementation of a completely new gearbox and shafts and would require a full mechanical redesign of the robot. However, the current two first stage gearboxes came in variable ratio planetary gear kits, which each individual box is currently set to 7:1 but can be set to any ratio from 3:1 to 10:1. The final stage gearbox is a fixed 2:1 gear ratio. We could alter the current hardware to achieve the greater gear reduction ratios of 128:1, 162:1, and 200:1. If we were to set the gearboxes to the maximum reduction of 200:1, we would obtain a new torque requirement for the motor of 76.8 oz-in. At this point, the performance speed of the motor shaft is approximately 4200rpm and the efficiency is approximately 60%. With the new reduction ratio, the operating linear speed is reduced to 1.10 ft/s as opposed to the previous 1.28

ft/s, rejecting part of the author's initial hypothesis of increased speed. However, this operating point achieves a full pit crossing time of 22s, which still allows enough time for approximately 9 excavation trips in a single run. This information will be presented to the undergraduate mechanical design team for consideration. It is up to them to decide whether this loss of speed is an acceptable sacrifice to undergo in order to increase the efficiency ratio by 17%. Another advantage to inducing a more radical gear reduction on the robot is that doing so will reduce the limits of the current flow that the electrical components will be subjected to during operation. The new max operating current would be reduced from approximately 65A to approximately 33A. This would reduce the risk of electronic device failure due to thermal stresses and resistive heating.

5.2 Combating Slip Through Control Strategies

The slip correcting strategies in this report are based on previous work from the Skonieczny Thesis [7]. The underlying traction model behind this work is that of Bekker [8], which utilizes a model that states that net traction (or drawbar pull) is

$$DP = N_w(H_i - R_i) \quad \text{Eq. (62)}$$

Where N_w is the number of wheels of the robot, H_i is the individual wheel thrust, and R_i is the compaction resistance of a single wheel. In Bekker's work, he determined a good estimation for both wheel thrust and compaction resistance as follows

$$R_i = b\left[\left(\frac{k_c}{b} + k_\emptyset\right)\frac{{z_i}^2}{2}\right] \qquad \text{Eq. (63)}$$

Where b is the wheel width, k_c and $k_\emptyset$ are soil pressure-sinkage parameter values (which Bekker used estimates for lunar regolith), and z_i is the sinkage of the wheel. The sinkage is also estimated as

$$z_i = \left[\frac{3N_i}{2b(k_c/b+k_\emptyset)\sqrt{2r}}\right]^{2/3} \qquad \text{Eq. (64)}$$

And N_i is the normal load on a given wheel. The wheel thrust is also estimated as

$$H_i = rb\int_0^{\theta_0}\left(c + \left(\left(\frac{k_c}{b} + k_\emptyset\right)\left(r(cos\theta - cos\theta_0)\right)^n\right)tan\emptyset\right) \times \left(1 - exp(-r/K[\theta_0 - \theta - (1-j)(sin\theta_0 - sin\theta)])\right)cos\theta d\theta \qquad \text{Eq.(65)}$$

Where c and ϕ are the soil cohesion and internal friction angle, K is the shear deformation constant, j is wheel slip, and $\theta_o$is the angle from vertical to where the wheel rim contacts the level terrain [7].

These estimation methods are mainly used for predictive analysis and design of excavation robots for operating on lunar surface conditions. These methods may possibly also provide valid estimates for robots that would operate on Martian surface conditions as well. However, we may be able to utilize feedback control to combat slip issues without knowing these important soil parameters or gravity estimations. In other words, predictive analysis is good for design when sufficient information is present in order to predict how the system will perform, which is good for robotic operation on Earth, Mars, or on the Moon. However, if we utilize correction through control strategies, we could develop a system that would be able to operate adequately on any soil surface with little known or unknown soil parameters. As the purview of space exploration grows, this concept will be extremely important for future celestial body exploration.

We will look at control strategies that involve accounting and correcting for slip. Slip is defined as

$$Slip(\%) = \frac{r_{eff}\omega - v}{r_{eff}\omega} * 100\% \quad \text{Eq. (66)}$$

Where v is the vehicles linear velocity, r_{eff} is the effective wheel radius, and ω is the angular velocity of the wheel shaft. The effective radius is an equivalent radius where shear occurs between moving soil and static soil. The estimation of the effective wheel radius does not have a well-known consensus and does not have properly defined precedent. Because of this, another parameter known as travel reduction is used instead of slip

$$Travel\ Reduction(\%) = \frac{v_o - v}{v_o} * 100\% \qquad \text{Eq. (67)}$$

Where v_o would be the vehicles baseline speed on flat ground with no drawbar load applied. The work done by Skonieczny implies that it is of utmost importance to keep the drawbar pull (normalized by weight) ratio below about 0.24 for lightweight excavators. At this point, excavator performance crosses a "lightweight threshold" where travel reduction spikes from 20% to 80%.

Our strategy to combat slip is going to be based on this concept of travel reduction through using sensors on the robot. The robot will continue to perform well as long as the travel reduction is held below 20%. We can apply rotary encoders to the wheel shafts to keep a constant measurement of the wheel shaft angular velocity to be sent back to the motor controller. If applying an encoder to each of the four shafts, we can estimate that our baseline speed on flat ground is

$$v_o = r\omega_{avg} \qquad \text{Eq. (68)}$$

Where ω_{avg} is the average encoder measurement of the motor shaft speeds at a given time interval of measurement. The author currently is considering two different methods of using sensors to determine the actual linear speed v.

5.2.1 Method 1

It is a common simple practice to estimate traction through shaft speed as though the actual speed of the robot corresponds with the minimum measured shaft speed of the wheels. That is, the slowest moving wheel is the one which has traction and therefore that speed corresponds to the vehicle's propulsion speed. The measurements could be directly fed back to the microcontroller and computer to run through a program and compute a value for travel reduction as

$$Travel\ Reduction(\%) = \frac{\omega_{avg} - \omega_{min}}{\omega_{min}} * 100\% \qquad \text{Eq. (69)}$$

Then, the strategy taken to combat this slip would be to amplify the proportional gain on the controller, to supply more power to the motors, that corresponds 80% of the current experienced travel reduction. The new proportional gain value to be used would come from

$$K_{p_{i+1}} = 0.8\left(\frac{Travel\ Reduction(\%)}{100\%}\right)\left(K_{p_{max}} - K_{p_i}\right) + K_{p_i} \qquad \text{Eq. (70)}$$

Where the subscript *i* is the uncompensated, subscript *max* is the maximum operating condition as previously discussed, and subscript *i+1* is the adjusted value.

5.2.2 Method 2

We can actually measure the straight-line distance traveled by the robot over a given period of time by utilizing either ultrasonic or image sensors (which will be included on the robot regardless of use for this task, for strategies that will be discussed in Phase IV). These sensors would return a simple distance magnitude to the nearest object within its range. In our case, the object in question would be either the forward wall or back wall of the competition pit, depending on whether the robot is in a departure or return path. Once a straight-line travel path is initiated, we would record the initial distance away from the reference object. After a certain

period of time, which will also be measured, we would record a new distance measurement from the ultrasonic sensor. Linear speed over that period of time can be estimated as

$$v_i = \frac{d_{i+1} - d_i}{\Delta t} \quad \text{Eq. (71)}$$

Where d_{i+1} is the secondary ultrasonic sensor distance measurement, d_i is the primary distance measurement, and Δt is the time interval recorded between when the two measurements were taken. Then, we can use this to estimate a value for travel reduction as

$$Travel\ Reduction(\%) = \frac{r\omega_{avg} - v_i}{r\omega_{avg}} * 100\% \quad \text{Eq. (72)}$$

And we would implement a similar proportional gain adjusting strategy as that shown in method 1.

CHAPTER 6: PHASE IV - AUTOMATION

NASA has repeatedly identified robotic, autonomous, and sensing systems as the enabling technologies over the course of history. For space excavation applications, the capability does not yet exist for traversing extreme lunar, Martian, or dusty terrains, including the lunar poles, high-grade surfaces, and microgravity environments. The advancement of robotics will be central to the transition of space missions from geocentric architectures to self-sustainable, autonomous systems, which is vital for outer-planet exploration and for overcoming the many difficulties of planetary travel. Much advancement is needed in the subject of robotic autonomy in order to broaden our capabilities for space exploration and expand human presence in the solar system [9].

For the purposes of this work, the objective is to develop an algorithm that would enable the RMC excavation robot to complete fully autonomous competition runs and develop a configuration of sensors on the robot that would be necessary to implement said algorithm. In this case, many parameters and conditions for the algorithm apply to a known, fixed operating space. The operating space in question is the competition pit displayed in Fig. 1 and the dimensional parameters for this space are stated in [1]. This section of the report will only discuss the automated processes and sensor configurations associated with the robot's mobile navigation of the operating space and not the automated processes of excavation and deposition. Additionally, this report will not discuss in depth the commands used for actions such as "turn right", "turn left", and "reverse" because those are simply a matter of applying specific motor speeds and directions to the motors on each side of the robot. Since each motor is capable of

acting independently, pure rotation about the center of mass of the robot is possible, also referred to as a “zero-point turn”.

The general algorithm proposed for operating autonomous competition runs utilizes the boundaries (walls) of the competition pit as points of reference, along with a single target indicator placed on the wall with the collector bin. The proposed sensor configuration would utilize eight ultrasonic proximity sensors placed in specific locations on each side the robot frame, along with two image sensors (cameras) on the front and back sides of the robot. A simplistic diagram which displays a top view of the sensor configuration on the robot is shown below:

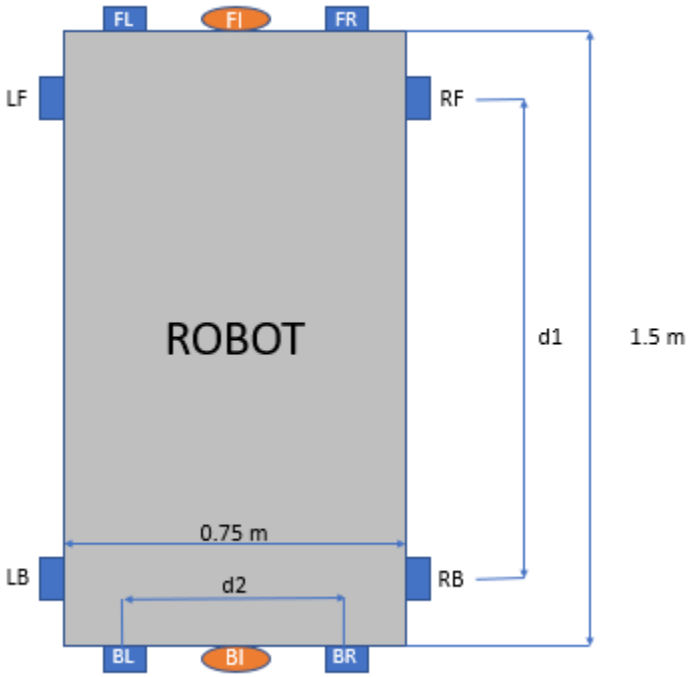

Figure 26. Sensor Configuration of RMC Robot.

Each of the blue rectangles represents the placement of an ultrasonic sensor on the robot and the orange ovals represent the placement of a camera. The two-letter nomenclature shown is “sensor IJ” where I indicates the side of the robot that the sensor is placed on and J indicates the

placement of the sensor on specified side I. The letters F, B, L, and R stand for front, back, left, and right respectively. Considering there are only two cameras, the only necessary identifier is whether it is the front camera or the back camera, so the I in the secondary symbol is just an identifier that it is an image sensor. The maximum length and width dimensions of the robot are specified in the diagram, along with the important parameters d1 and d2 which are the lengths between sensors on the same side of the robot. These lengths are fixed and are vitally important in using triangulation techniques in order to identify the both the robot position and orientation in the operation space. This is the crux of our automation strategy, if the position and orientation of the robot within the boundaries of the operating space can be obtained through triangulation of this sensor data, then the rest of the automation process is left to running previously established action commands. It needs to be noted that the primary purpose of the image sensors is to detect the target indicator placed on the collector bin, which is manufactured purposefully to be of a different color (orange) than the surrounding environment so that it is easily discernable for these sensors.

The overall task flow of the competition run is that the robot will be placed in one of the two 1.89 x 1.5m starting zones, at some multiple of 60-degree angular orientation with respect to the back wall, both of which are randomly selected prior to the run start. The robot must navigate through the 3.78 x 2.94m obstacle area, which contains more exaggerated undulation (< 30cm) and at least 3 randomly placed obstacles (boulders, 10-30cm diameter) onto the mining area. The mining area is the only allowable area for excavation, excavation that occurs outside of this area will result in a disqualification of the run. Once excavation is complete, the robot is required to navigate back through the obstacle area to the center of the back wall, where the load is to be

deposited into the collector bin. The robot is not constrained to a single trip, rather it can make as many trips as desired as long as it is within the ten-minute timeframe.

The general algorithm for a single automated run process is outlined as follows:

1. Identify collector bin location (sensors)
2. Identify orientation to back wall (compute)
3. Re-orient and reposition to the start of the obstacle area (command)
4. Identify obstacles and plan path (sensors, compute)
5. Navigate obstacle area to start of excavation area (command)
6. Identify desired excavation location (predetermined order)
7. Navigate to desired excavation location (command)
8. Run excavation program
9. Identify collector bin location (sensors)
10. Identify orientation to front wall (compute)
11. Re-orient and reposition to start of obstacle area (command)
12. Identify obstacles and plan path (sensors, compute)
13. Navigate obstacle area to start of starting area (command)
14. Re-identify more specific collector bin location (sensors)
15. Navigate and orient to prepare for deposition (compute, command)
16. Run deposition program
17. Repeat

It is to be noted that this process is a repetition of identification, orientation, and navigation steps. The three of these will be discussed in the following subsections, as well as a set of overlying conditions that the robot should always be conforming to. It also should be restated

that the following information is a starting point for trying to implement an automation process for the robot, and not an improvement or addition to a previously implemented process. There will be much testing and expansion necessary for the algorithm to encompass all possible scenarios that could occur during a competition run.

6.1 Conditions

The following conditions are desired to be maintained on the robot (running in outer loops in the algorithm) for the duration of the automated runs:

- Due to the specified robot size limitations, a circle that safely circumscribes the robot (top view, 2D) is roughly 1.7m in diameter. We will set a condition that if any of the eight proximity sensors detects a distance of less than 1 meter, the robot must not rotate in such a way that it will bring a corner of two of its sides towards that initial measured distance point (the corners are the farthest points from the robot's center of mass). Instead the robot, under this circumstance, will always be working to:
    - If necessary, reduce the magnitude of the angle measured between a robot side and the closest proximity wall.
    - Drive towards the direction of largest measured distance to a wall.
- As previously discussed in Phase III, we can use the proximity sensors to have a more exact means of measuring travel reduction. When performing a drive forward command, the motors can run at a constant angular velocity for a discrete time interval. We can measure the actual straight-line distance travelled over this time interval using a triangulation method. Referring to equation (72), v_i will be replaced with $\frac{\sqrt{(\Delta x)^2+(\Delta y)^2}}{\Delta t}$.

- The algorithm should always identify and default to using the two adjacent robot sides which have the least proximity distance measurements to the certain walls in question when running an orientation command. This ensures that the robot will not be orienting itself with respect to an obstacle under most circumstances, and we can have a higher level of confidence in these measurements based on how ultrasonic sensors work.

6.2 Identification of Collector Bin Location

The first task of the run is using the image sensors to identify the collector bin location and orientation to the back wall. The very first bit of code in the algorithm would be simply to identify if either of the image sensors detects the orange targeting beacon, just to save some time. Detailed explanation about image sensing and color detection will not be discussed in this work, considering basic image sensing using aftermarket cameras and OpenCV code will be used. The basic commands (in pseudocode) for each of the two sensors would go as follows:

```
[R,G,B]=imageread("front image sensor");


if (R~=255)
        conditionF=0;                       %% 0 means 'false'
elseif (G>69) && (G<215) && (B<80)
        conditionF=1;                       %% limits of orange shades in RGB
                                            %% unsigned 8-bit integer type.
else
        conditionF=0;
end
```

(run the same for back image sensor) and then:

```
if  (conditionF=0) && (conditionB=1)          %% orange image detected by rear
                                              %% camera only.
    **run orientation program with rear sensors**
elseif (conditionF=1) &&    (conditionB=0)    %% orange image detected
                                              %% by front camera only
    analogWrite("motorPin RF", 0);
    analogWrite("motorPin RF", 0);      %% These four lines are the write
                                        %% commands for a clockwise
                                        %% rotation of the robot (excluding setup
    analogWrite("motorPin LF", 255);    %%  code, 8-bit), will refer to as "CW
                                        %% Rotation" (This would be full speed in
    analogWrite("motorPin LB", 255);    %% 8-bit).
else                                    %% else - this would mean both
                                        %% conditions are false or both true,
    while (timeElapsed < timeLimit)     %% set to break.
        **run CW Rotation**
    end
    break
end
```

The premise around this small portion of (very simplistic) code is to simply perform a zero-point clockwise rotation of the robot until the rear camera detects the orange targeting beacon. Once the rear camera detects the orange targeting beacon, we know that the rear of the robot is

oriented (generally) towards the back wall and we can move on to the first orientation program. It is a possibility for the cameras to be both functional and the robot oriented such that neither camera detects the orange target, such as the rear camera being pointed towards one of the corners of the pit. However, because of the competition pit operating conditions defined by NASA, if both image sensors return “true” for orange we know that it must be a hardware issue and we call to a break command. We set a time limit such that if the robot performs multiple full rotations and neither sensor returns “true” we must call to a break command as there must be a hardware issue as well.

6.3 Orientation Process

This section of the report overviews the process of a simple yet important task in our automation algorithm, using triangulation to orient a certain side of the robot to align parallel (within a specified tolerance) to a certain wall. Ideally, there would be no impeding obstacles in the robot’s straight-line path to the excavation site, and we can simply run the orientation program until the back side of the robot is oriented parallel to the back wall and may drive straight to the excavation site. Figure 27 shows a geometric schematic of an arbitrary orientation of the robot’s back side with respect to a boundary wall:

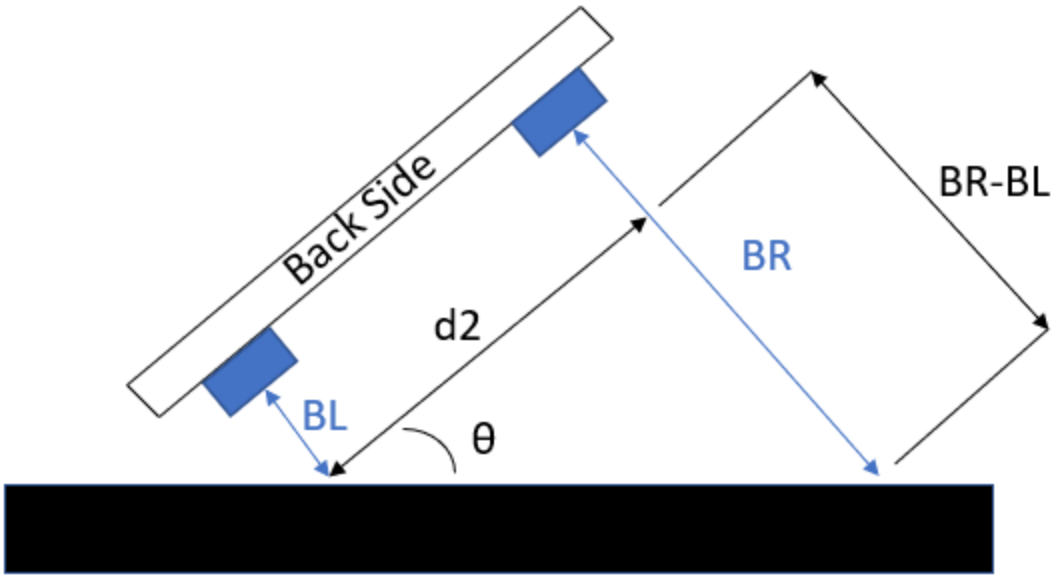

Figure 27. Geometric Schematic of Back Side with Respect to a Wall.

If the robot was oriented this way, such that the distance measured by the sensor BR was greater than the distance measured by sensor BL, then we would want to rotate the robot clockwise until our specified parallel tolerance was met. Pseudocode for this situation would go as follows:

```
BR=digitalRead("sensor BR");
BL=digitalRead("sensor BL");
tol= "specified distance tolerance"        %% conversion from digital to
                                           %% physical distance necessary.
angtol= "specified angle tolerance"

if (BR>BL)
    theta=atan((BR-BL)/d2);
    while (|BR-BL|>tol) || (|theta|>angtol)
        **run CW Rotation**
    end
```

```
elseif (BR<BL)
        theta=atan((BL-BR)/d2);
        while (|BR-BL|>tol) || (|theta|>angtol)
                **run CCW Rotation**        %% CCW Rotation would be the
                                            %% same as CW Rotation with the high and
        end                                 %% low PWM output numbers switched.
else                                        %% Somehow BR==BL, likeliness depends
                                            %% on precision of sensor used.
end
**run Drive Forward**                       %% Drive Forward is all motor pins set to
                                            %% PWM high.
```

It is noted that the code above accounts for the mirrored orientation of Figure 27 such that the distance measured by sensor BL is greater than the distance measured by sensor BR. This very simple bit of code premises the concept of using a form of triangulation to identify an objects position and orientation with respect to another object. This bit of code can be edited and reused in order to orient *any* side of the robot to *any* particular angle of orientation with respect to a wall. This is very enabling for the robot to follow specific paths to any point in the operating space using sequences of orientation and drive commands. It is noted that this is probably not the most optimal means of traveling from point-to-point, but it does indeed manifest a probability of success. The topics outlined in this subsection are the cornerstones of steps 1-3, 9-11, and 14-15 in the general algorithm.

6.4 Navigation Process

It has been stated in the previous subsection that ideally the robot would not have any obstacles to overcome, and the simplest means of moving to the desired (and predetermined) excavation site would be to orient towards that site location and drive in a straight-line path to said location. The techniques that could be used for this process have actually already been laid out in the previous section. Once the robot has oriented itself such that the back side of the robot is parallel to the back wall (and thus the other corresponding sides are also parallel to their respective walls) the task of straight-line navigation is yet again just reorienting to a certain angle which intersects the desired excavation location and running the drive forward command. A schematic of the "simplest case" scenario is shown below:

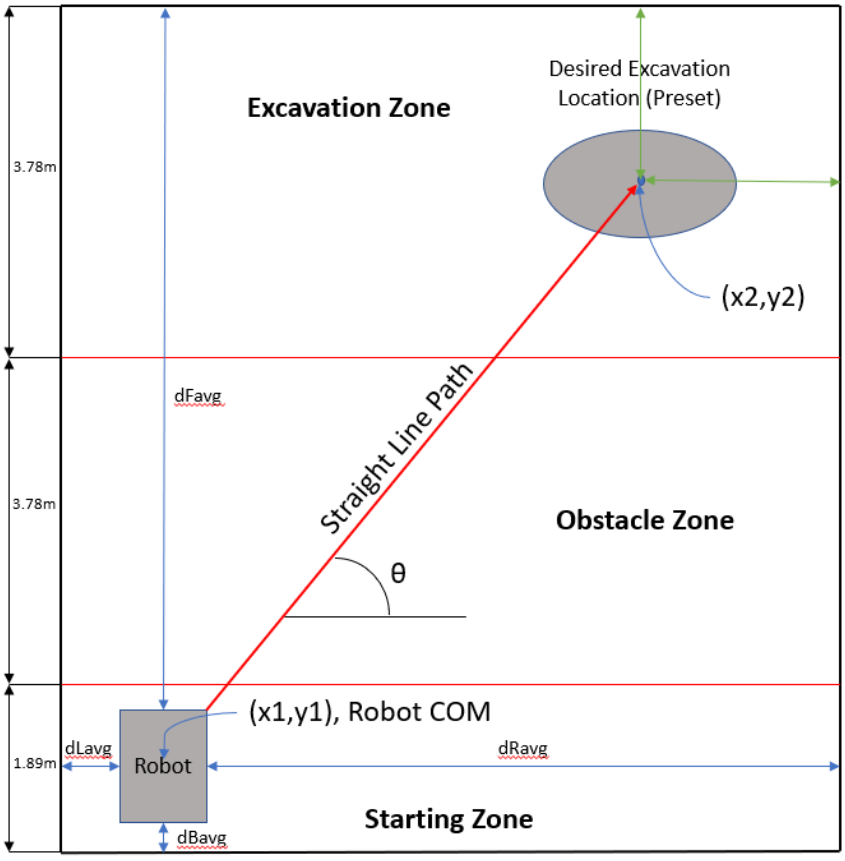

Figure 28. Schematic of Straight-Line Path to Desired Excavation Site (Excluded: Width of Pit is also 3.78m).

The programming necessary to plan this straight-line path is almost entirely based upon fixed geometry. The robot's center of mass (in this 2D plane) is desired to lie coincident to the geometric center. This is entirely possible due to the excavation system design, where a significant portion of the robot's weight lies in the excavation drum. This drum was designed to have a very wide range of motion and one of the objectives of the design was to be able to adjust the location of the robot's center of mass before transport (again, for more information regarding the excavation system control strategies contact Ryan Watson). The location of the geometric center of the robot from the center of the sides of the robot will always be a fixed distance. The means to calculate the average distance from a given wall to a robot side is a simple matter of taking the average of the distance measurements from the two proximity sensors on that side. For example, and referring to the schematic above:

$$d_{B,avg} = \frac{(BLcos\theta_B + BRcos\theta_B)}{2} \quad \text{Eq. (73)}$$

$$d_{L,avg} = \frac{(LFcos\theta_L + LBcos\theta_L)}{2} \quad \text{Eq. (74)}$$

$$x_1 = \sqrt{d_{L,avg}^2 + l_{L,COM}^2 + d_{L,avg} l_{L,COM} cos\emptyset} \quad \text{Eq. (75)}$$

$$y_1 = \sqrt{d_{B,avg}^2 + l_{B,COM}^2 + d_{B,avg} l_{B,COM} cos\emptyset} \quad \text{Eq. (76)}$$

$$\theta = tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad \text{Eq. (77)}$$

Where $l_{i,COM}$ is the length from the robot center of mass to desired side *i*, and the coordinates (x2, y2) of the first excavation location come from a predetermined order of desired excavation locations. The algorithm to perform this task could be as simple as orienting the right side of the robot to meet the calculated angle θ with respect to the back wall, with the condition that the distance measured by sensor RB is less than that measured by sensor RF to ensure the solution is

at the correct orientation of the path to the front of the robot and not to the orthogonal. Once the robot is oriented to the correct angle, the next step is to run the drive forward command until the center of mass coordinates of the robot are coincident (within tolerance) to the desired excavation site coordinates (x2, y2). This could also be done by orienting the back side of the robot to the same angle θ with respect to the back wall, with the condition that the distance measured by sensor BR is less than that measured by sensor BL. Additionally, it is desired that the robot reorient itself such that the back side of the robot is aligned parallel to the front wall prior to excavation. Since the excavation drum was designed to fit within the interior of the spacing between the wheels, this allows the robot to run a drive forward command after excavation to avoid a wheel getting stuck in the digging site.

Where the navigation planning of the robot becomes more complicated is in the extremely likely scenario that a straight-line path is impeded upon by an obstacle. This requires the robot to step away from using predetermined paths and actually become more reactionary to the surrounding environment. As mentioned before, a key design point in this strategy is that the ultrasonic sensors have very specific placement on the robot frame. The main conditions pertaining to the sensor placement are as follows:

1) The two sensors on a given side must be symmetric about the robot's geometrical center with respect to that side. This serves for purposes of ensuring that the orientation algorithms are both able to adequately ensure a parallel relationship of the robot's side to a wall, and to ensure that the average distance measurement of the two sensors can be used to create an accurate coordinate point for the robot's center of mass.

2) The sensors need to be mounted low enough to the ground to be able to detect obstacles that we are not confident that the robot can run over without getting stuck. They also need to be mounted far enough **off the ground** such that they are not constantly measuring the distance to the small undulations (<10cm) in the surface. This provides a means for the sensor measurements to be used in navigation algorithms designed to maneuver around obstacles.

The first condition plays a vital role to the strategy laid out in the orientation subsection, and the second condition plays a vital role to the strategy that is to be laid out in this subsection.

A schematic showing the scenario of impeding obstacles to the robot's calculated straight-line path is shown below:

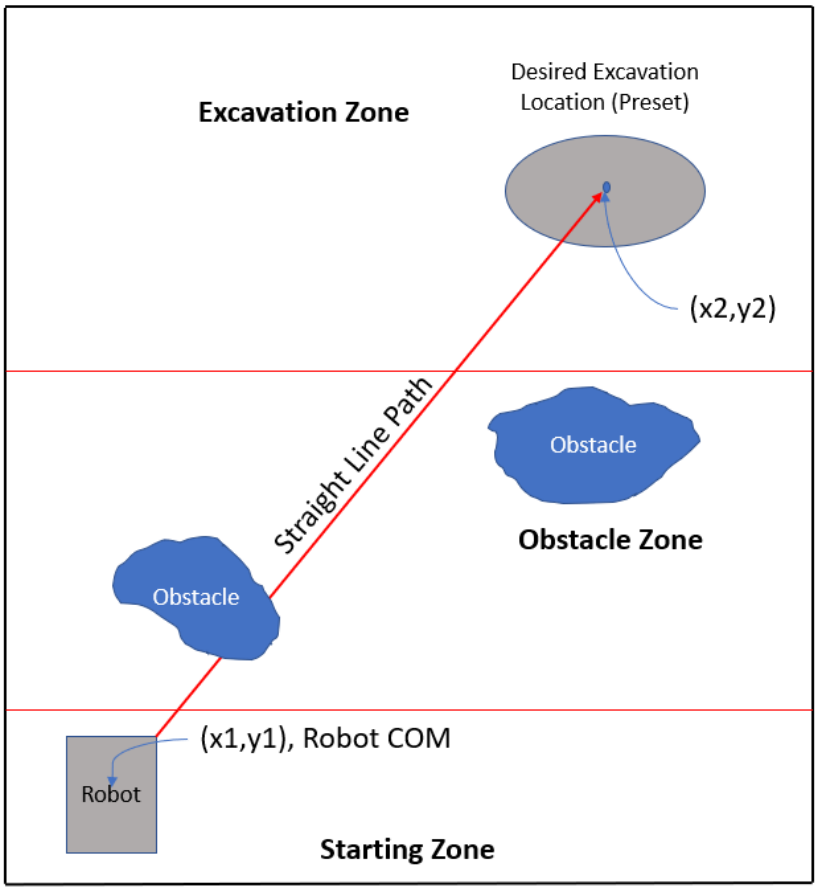

Figure 29. Scenario of Obstacle Impeding Straight-Line Path.

For this specific scenario, we can see that after the robot has performed its initial orientation command and then computed the straight-line path to the desired excavation site that the ultrasonic sensor FR will detect a shorter distance measurement than that expected of the front wall. More generally, the algorithm should first check to see that the front sensors detect the expected distance to the pit walls. If not, then the robot should be able to discern that there is an obstacle impeding the path, and it should then run a command to navigate around said obstacle to the desired location. It is to be noted, that we desire for the robot to have a preference of initially moving towards the center of the pit when circumventing an obstacle. A schematic of the adjusted path is shown below:

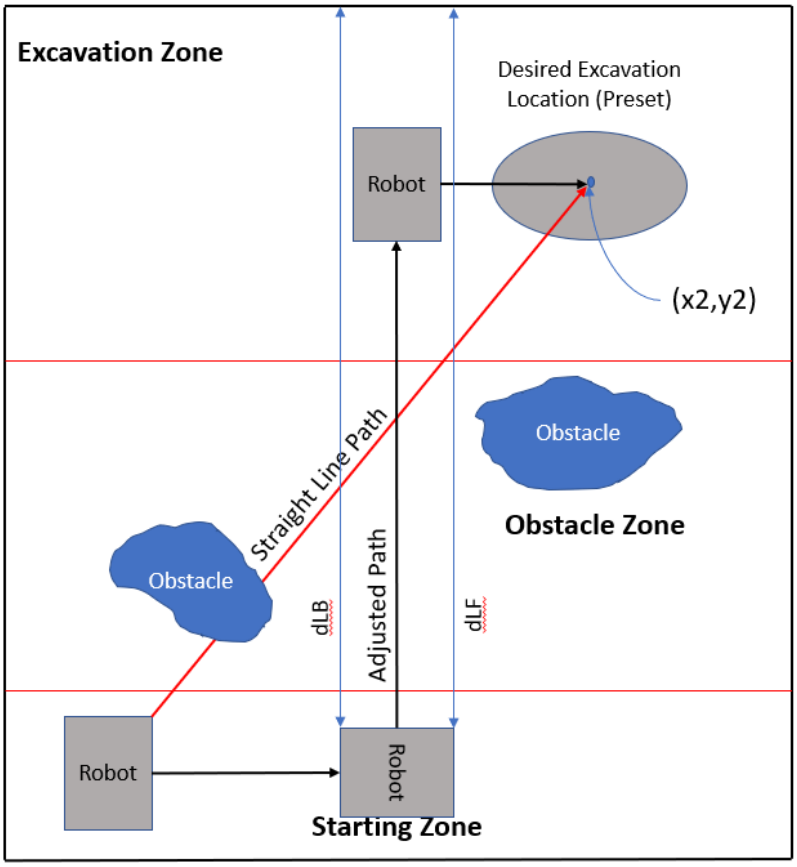

Figure 30. Adjusted Path Around Impeding Obstacles

Using this scenario as an example, the process for navigating this adjusted path is outlined as follows:

1) Sensor FR (FL, both) reads a distance measurement less than that expected for the front wall distance. The robot has encountered an obstacle.
2) Rotate the robot such that its right (left) side is parallel to the back wall, and its front side has a greater average distance to the opposing wall than the back side (i.e. if it drives forward it moves towards the x-center of the pit).
3) Now, the sensor LF (RF) and possibly LB (RB) will read a distance measurement less than that expected of the front wall (the obstacle). Run drive forward command. At some point **both** LF and LB (RF and RB) will drive past the obstacle and detect the distance to the obstacle. Once sensor LF (RF) passes the obstacle it will read the expected distance measurement to the front wall.
4) Keep driving forward until **both** LF and LB (RF and RB) detect the expected distance to the front wall, (or at least a distance to a possible second obstacle that is further away). Stop. If neither side sensor detects an obstacle then the robot has clearance to pass through, since the robot geometry is of greater length than width.
5) Rotate the robot and reorient such that the back side is again parallel to the back wall. Drive forward until robot's (center of mass) y1 coordinate is equal to the desired site coordinate y2. Stop. (If the robot encounters another obstacle instead, repeat steps 1-5 around the secondary obstacle).
6) Rotate the robot such that its right (left) side is now parallel to the front wall, and that its front side is pointing in the direction of the desired excavation site.
7) Drive forward until (x1, y1) = (x2, y2). (Within tolerance).

This process is an outline for an algorithm that is just a combination of the simple commands that were previously seen in the identification and orientation subsections. Yet this process can

even be applied to the robot when it is not travelling parallel to any reference wall but at any arbitrary angle θ with respect to the coordinate system that is established during the beginning of the navigation process. This process is very simple, yet it can be successfully applied to a very wide variety of scenarios that the robot may encounter and that is a very powerful asset for the robot to have. Yes, there needs to be rigorous testing applied to this strategy such that it can cover even more difficult scenarios and it needs to be built upon and improved, but keep in mind that the purpose of this information is to serve as a starting point for developing autonomy for the current and future robots used in the RMC. The topics outlined in this subsection are the cornerstones of the remaining steps (4-7, 11-13, 15) of the general algorithm presented.

CHAPTER 7: CONCLUSION AND RECOMMENDED FUTURE WORK

The primary objective of the first two phases of this work was to obtain a rapid yet smooth response of the RMC motor control system by utilizing the established control tuning strategy. The overlaying objective is to verify that this control tuning strategy will prove to be a consistent and effective way of determining PID gains for a variety of different DC motor applications. This objective is considered to be a work in progress, and the utilization of this strategy upon many more systems in order to verify its effectiveness. A proposed testing method which we wish to perform in the coming year is to build a practice run pit so that the actual number of excavation trips per 10-minute run that can be performed by the robot can be determined. This would greatly help the team in more rapidly determining what areas of the robot need to be optimized in order to achieve better competition performance.

The primary objective of Phase III was to provide insight for improvement that can be immediately implemented on the current and future robot builds. A change to the gear ratios on the motor shafts would result in a significant improvement in power efficiency with minimal losses in torque at the operating points. Implementing a slip correction strategy such as the ones proposed would help identify design problems that could be improved in the next iterations of the robot and greatly improve performance in the competition runs.

Future work would also involve several iterations of custom motor control boards to be manufactured and tested on the robot build to verify the proposed generalized process of the PID tuning method. Future work for the automation phase of this work would involve the

construction of a practice pit for the robot to perform runs in, which would lead to a verification or rejection of the proposed strategy as well as lend more insight to how the strategy can be improved.

REFERENCES

[1] Heiney, A. (Ed.). (2018, April 27). NASA's Ninth Annual Robotic Mining Competition: Rules and Rubrics. Retrieved May 10, 2018, from https://www.nasa.gov/sites/default/files/atoms/files/2018_rules_rubrics_parti.pdf

[2] Bishop, R. H., & Dorf, R. C. (2011). *Modern Control Systems* (12th ed.). Upper Saddle River, NJ: Pearson Education.

[3] Introduction: PID Controller Design. (n.d.). Retrieved May 10, 2018, from http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=ControlPID

[4] Loseke, Barrett (2018). RMC Independent Study: Summer 2018. Submitted August 5, 2018.

[5] Massouda, N. (2018, July 11). Re: 2.5 CIM Motor Question: 4244 [E-mail to the author]. AndyMark Inc.

[6] DC Motor Speed: Simulink Modeling. (n.d.). Retrieved May 12, 2018, from http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SimulinkModeling

[7] Skonieczny, K. (2013). Lightweight Robotic Excavation. Carnegie Mellon University. 51-71.

[8] Mieczyslaw Gregory Bekker (1969). *Introduction to terrain-vehicle systems*. Ann Arbor, MI: University of Michigan Press.

[9] Pavone, M., Acikmese, B., Nesnas, I. A., & Starek, J. (2014). *Spacecraft Autonomy Challenges for Next Generation Space Missions* [Scholarly project]. Retrieved April 8, 2019, from https://web.stanford.edu/~pavone/papers/Pavone.Acikmese.ea.LNS14.pdf

[10] Parrish, L., "Regolith Advanced Surface Systems Operations Robot (RASSOR) Excavator," NASA Technology Transfer Program, pp. 1-2, Mar. 2013

[11] Motor Specifications FR801-001. (n.d.). Retrieved May, 2017, from files.andymark.com/CIM-motor-curve.pdf. CCL Industrial Motor Limited (CIM).

[12] Technical Specifications. (n.d.). Retrieved August, 2018, from https://www.themartiangarden.com/tech-specs. MMS-1 Specifications