Development of a Tow Capacity Test Device for Small Unmanned Vehicles

Shane Barnett

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
In
Mechanical Engineering

Dr. Charles F. Reinholtz, Chairman
Dept. of Mechanical Engineering

Dr. Alfred L. Wicks
Dept. of Mechanical Engineering

Dr. Mehdi Ahmadian
Dept. of Mechanical Engineering

December 15, 2005
Blacksburg, Virginia

Keywords: Drawbar, Mobile Towed Dynamometer, Unmanned Vehicle Testing, Metric

Development of a Tow Capacity Test Device for Small Unmanned Vehicles

Shane Barnett

ABSTRACT

Unmanned ground vehicles (UGVs) will increasingly be used for tasks such as retrieving injured soldiers from a battlefield, transporting supplies, and towing other small vehicles and payloads. To date, the unmanned test community has not standardized on an apparatus or test operating procedure (TOP) specifically for evaluating the towing capacity of small unmanned ground vehicles. Draw-bar testing has been adapted by several groups to quantify small unmanned ground vehicle (SUGV) tow capacity; however, these devices are inherently limited to measuring peak static towing force. This paper describes an alternative method using a variable-resistance tow sled for quantifying the dynamic towing capacity of SUGVs. The tow sled contains a frontal skid plate and a rear axle and wheel arrangement. A weighted carriage is transferred from the rear of the sled to the front of the sled by a cable geared to the rear axle. As the sled is pulled along the ground, towing resistance increases in a controlled linear fashion. An encoder on the rear axle and a load cell in the tow chain provide motion and force data. Testing of the tow sled has been conducted on a TALON SUGV at the Southwest Research Institute (SwRI) Small Robot Test Facility and a MATILDA SUGV at the Joint Unmanned Systems Test, Experimentation, and Research (JOUSTER) site.

Acknowledgments

I owe a lifetime of gratitude to my parents, Don and Glenda. You have given me both the motivation and humility to make it this far.

I blame Dr. Reinholtz for me staying here beyond my undergraduate degree. His enthusiasm and character imparts good things upon other people.

Dr. Wicks has always kept me honest. His constant badgering keeps me guessing as to whether he is just giving me a hard time or has actually caught me doing no good.

I also have to thank all of my fellow lab members for their generosity in helping me on this project, especially Ben Hastings and Sean Baity for helping on the electrical side. They, along with Chris Terwelp and Leiann Leppin, were instrumental at Southwest Research Institute where we fielded our initial tests.

Finally, I would like to thank Bill McBride and Andrew Moore of SwRI for allowing us to come down and test the tow sled. Valuable information was gained form this trip.

This work was supported by Army Research Development Engineering Command Simulation Technology Training Center (RDECOM-STTC) under contract N61339-04-C-0062.

Table of Contents

Index of Figures v
Index of Tables vi
Chapter 1 - Introduction 1
1.1 Motivation 1
Chapter 2 – Literature Review 5
2.1 Chassis Dynamometers 5
2.2 Towable Dynamometers 7
2.3 Army TOPS 2-2-604 11
Chapter 3 – Tow Sled Development 13
Chapter 4 – Tow Sled Instrumentation 23
4.1 – Interface SM-500 Load Cell 23
4.1.1 Load Cell Calibration 24
4.2 – Tamagawa 10 bit Incremental Encoder 26
4.3 Data Acquisition 28
Chapter 5 - Test Guidelines 30
5.1 Standard Pull Test 30
5.2 – Fixed Load Use of Sled 32
Chapter 6 – Test Results 33
6.1 Evaluation of the TALON at Southwest Research Institute 33
6.1.1 Terrain Comparison: Dry Asphalt vs. Damp Soil 35
6.2 Vehicle Comparison: TALON vs. MATILDA 42
6.3 Drivetrain Efficiency 46
Chapter 7 - Conclusions 51
Appendix A – TALON Test Results 53
Appendix B – MATILDA motor specs 54
Appendix C – MATLAB Code for TALON Pulls One, Two, and Four 55
Appendix D – MATLAB Code for TALON Pulls Twelve, Thirteen, and Fifteen 63
Appendix E – MATLAB Code for TALON, MATILDA Comparison 71
Appendix F – MATLAB Code for MATILDA Theoretical vs. Actual Pulls 75
References 79

Index of Figures

Figure 1. TATRC Valkyrie Soldier Extraction. Two Packbots tow the Extraction Payload 2
Figure 2. a) Packbot set to deliver sked to an injured soldier and b)"Soldier" secured in sked ready for retrieval. www.skedco.com..... 3
Figure 3. Chassis Dynamometer produced by SuperFlow 6
Figure 4. Roller configuration for the Stryker on Yuma Proving Ground's chassis dynamometer..... 7
Figure 5. Dynamometer test at Aberdeen Proving Ground 8
Figure 6. Superflow's TD-1200 towable dynamometer 8
Figure 7. Tractor pull for remote control cars..... 9
Figure 8. Setup used to measure peak static tractive force of locomotions..... 10
Figure 9. Packbot undergoing static drawbar tests in various terrain. 11
Figure 10. Free Body Diagram Showing Relevant Forces 13
Figure 11. Free body diagram of the sled design showing significant moments about the rear axle..... 15
Figure 12. Solidworks drawing of the tow sled. 16
Figure 13. Cable winding around wheel axle. 17
Figure 14. Tow sled shown with the weight carriage at starting position. 17
Figure 15. Variable hitch height ensures tow force is horizontal. 19
Figure 16. The top diagonal tube will receive the highest bending stress. 20
Figure 17. Free body diagram of the top tube under load..... 21
Figure 18. Johnny 5 undergoing a tow test at JOUSTER Field Day, 10-30-04 22
Figure 19. Close-up view of load cell connected inline the tow chain. 23
Figure 20. Calibration curve for the Interface SM-500 load cell..... 25
Figure 21. Load cell amplifier schematic. 26
Figure 22. Encoder used on the tow sled. 27
Figure 23. Placement of the encoder to measure wheel speed and position. 27
Figure 24. Data collection user interface 29
Figure 25. Hitch point shown on the TALON 31
Figure 26. TALON undergoing pull test..... 34
Figure 27. TALON Speed vs. Time on Dry Asphalt. Average maximum speed is 1.11 mph. 36
Figure 28. TALON Speed vs. Time on Wet Grass. Average maximum speed is 1.16 mph. 37
Figure 29. Plot of three TALON pulls of Force vs. Time on Dry Asphalt. 38
Figure 30. TALON Force vs. Time on Wet Grass. 39
Figure 31. TALON Power vs. Time on Dry Asphalt. 40
Figure 32. TALON Power vs. Time on Wet Grass..... 41
Figure 33. Fourier transform of filter compared plotted in the frequency domain. 42
Figure 34. Close-up view of the competing SUGV's, a) MATILDA and b) TALON. Notice the profile and void ratio of each robots' treads..... 43
Figure 35. Velocity vs. Time Plot comparing the TALON vs. MATILDA 44
Figure 36. Tow force vs. time plot of TALON vs. MATILDA 45

Figure 37. Power vs. time for the TALON and MATILDA. 46
Figure 38. Actual vs. Theoretical performance of MATILDA..... 48
Figure 39. Theoretical vs. actual power vs. tow force of MATILDA. 49

Index of Tables

Table 1. Summary of results. Tests with the same test variables are averaged using a moving average of 25 points.. 34
Table 2. Terrain comparison of asphalt vs. wet grass with all other variables constant....35

Chapter 1 - Introduction

The JOUSTER (JOint Unmanned Systems Test, Experimentation & Research) site [1] is dedicated to creating scientifically based experimentation, evaluation, comparison, research, and development of unmanned systems. One goal of the JOUSTER site is to help the unmanned systems community and the Department of Defense (DoD) test community to create and implement metrics to quantify system characteristics. As part of this effort, we have developed a simple tow sled device for measuring the dynamic towing capacity of small unmanned ground vehicles (SUGV's).

1.1 Motivation

There is little question that unmanned vehicles will play a key role in future military operations. In an interview with the Boston Globe, Marcus Corbin, senior analyst with the Center for Defense Information, said, "The unmanned aircraft, like the Predator, got a lot of attention in Afghanistan, but, to me, they won't be as important as the ground vehicles. There are few countries that can challenge our Air Force. But anyone can challenge our ground forces in urban warfare. It's a type of combat with a lot of casualties on both sides, and the only easy answer may be robots. If we continue to occupy foreign countries that don't like us very much, the role of these robots will be key" [2].

With advances in technology, robots are becoming more intelligent and reliable. Tasks such as retrieving injured soldiers from the battlefield, transporting supplies, and towing other small vehicles and payloads are already being explored [3]. As an increasing number of robots are deployed in the battlefield, system capabilities will be

exploited and robotic systems will be implemented in additional battlefield missions. For instance, the Army Telemedicine and Advanced Technology Center (TATRC) Valkyrie program is evaluating SUGVs for use in extracting wounded soldiers from the battlefield [3,4].

Figure 1. TATRC Valkyrie Soldier Extraction. Two Packbots tow the Extraction Payload [4].

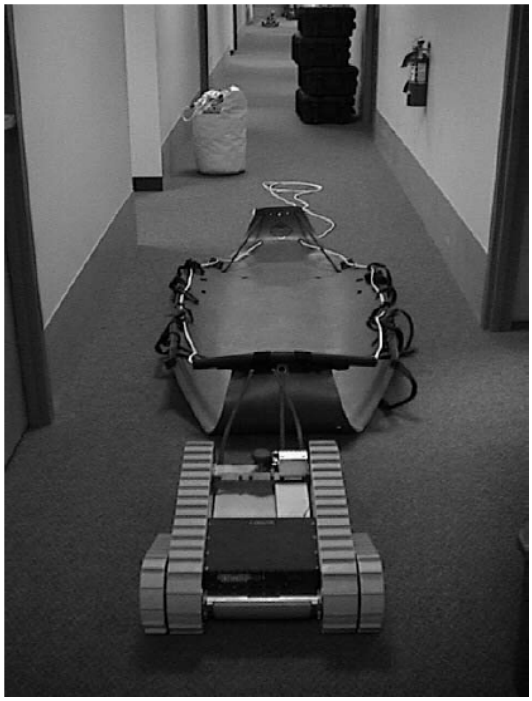
a)  b) 

Figure 2. a) Packbot set to deliver sked to an injured soldier and b)"Soldier" secured in sked ready for retrieval. www.skedco.com

It is critical to evaluate the tow capacity of SUGVs to determine the optimal configurations for successfully and quickly extracting injured soldiers from the battlefield. To date, SPAWAR (Space and Naval Warfare Systems Command) lists 80 available small unmanned ground vehicle platforms [5]. Without quantifying tow capacities of these systems, the end user may find it difficult to determine the system that is best suited for a particular mission.

As one example of the confusion that currently exists, Mesa Robotics lists the weight of their MATILDA SUGV as 61 pounds and the towing capacity at 225 pounds [6]. Interestingly, the Howstuffworks website lists the towing capacity for this same robot as 475 pounds [7]. Both of these specifications are probably accurate under particular test conditions, such as towing a wheeled trailer on a level, paved surface.

Unfortunately, these specifications are not helpful in understanding MATILDA's ability to tow an Extraction Payload such as the one described above. Likewise, Foster-Miller boasts a "drag capacity" of their TALON robot at 200 pounds [8], but makes no specifications on what surface this mass is being drug across, or exactly what constitutes this mass. Two hundred pounds of payload will drag easier over smooth tile than it will over a gravel road.

Simply specifying robots by their power output or peak drive torque is also inadequate for understanding their towing and payload carrying capacity, since ground traction may also limit performance. In many cases, the light weight of the vehicles (a positive feature for transport) will not produce sufficient traction to take full advantage of the high-torque electric drive motors. This suggests that tests should also be conducted with a payload aboard the vehicle to increase ground forces and traction.

In Mobility Analysis of Small, Lightweight Robotic Vehicles, Brooke Haueisen of Army TARDEC lists drawbar pull as one of the desired vehicle metrics. "Drawbar pull is the total thrust minus the total resistance of the vehicle. This is a simplified measure of the vehicle's ability to move. If the value is less than or equal to zero then the vehicle will not be able to move [9]".

Some desired attributes of this of this test device are that it must be cost effective, reliable, easy to use, and able to accommodate many different robotic platforms. The next section discusses available vehicle dynamometers currently on the market.

Chapter 2 – Literature Review

This section discusses the current vehicle dynamometers that exist today. There exist plenty of dynamic tow capacity test devices currently on the market, but unfortunately none exist sized for testing in the small robot community. Initial research has shown quite a few varieties of mobile dynamometers capable of taming large, military transport vehicles and tanks, passenger vehicles, and even some small enough to accommodate remote control trucks. There is little if any sized for a small unmanned ground vehicle.

2.1 Chassis Dynamometers

Superflow produces stationary chassis dynamometers suited for passenger vehicles. An example of one of their products is show in Figure 3. Although this might serve the requirement if scaled down to fit an SUGV, it still possesses some fundamental disadvantages. This dynamometer would not be able to simulate different surface mediums. The initial cost of fabrication would be significantly higher, as hydraulics or electric motors would need to be implemented to impose a load on the drivetrain. Finally, setup for each robot would be unique. Different drive scenarios arise such as how many wheels are powered, location of each powered wheel, or would this device be feasible at all if the robot used treads? Now the entire tread surface would have to situate itself on the resisting drum while the robot is constrained somehow from moving. Constraining the robot to the ground or a nearby wall add extra strain on the drivetrain introducing error in testing.

Figure 3. Chassis Dynamometer produced by SuperFlow [10]

So what if you could take this concept to accommodate various sizes and configurations of vehicles? The Combat and Automotive Systems group of U.S. Army Yuma Proving Ground in Yuma, Arizona recently developed their own stationary vehicle chassis dynamometer. Housed in the Large Multipurpose Environmental Chamber (LMPEC), this unit is capable of handling military vehicles with as many as five driven axles with a 25,000 lb vertical load capacity per axle. Ten individual rollers each 18 inches in diameter are positioned to support the vehicle weight and to impose torque on the drive wheels. This torque comes from back driving hydraulic pumps and measuring the resulting current. Top speed of the rollers is 60 mph and each can absorb up to 1980 ft-lbs of torque through this full range of speed. Encoders are placed on five of the roller axles to measure wheel speed and a load cell to this same axle measures delivered torque. Figure 4 shows the roller configuration for the M1126 Stryker infantry carrier vehicle with three driven axles. Rollers highlighted in blue are instrumented.

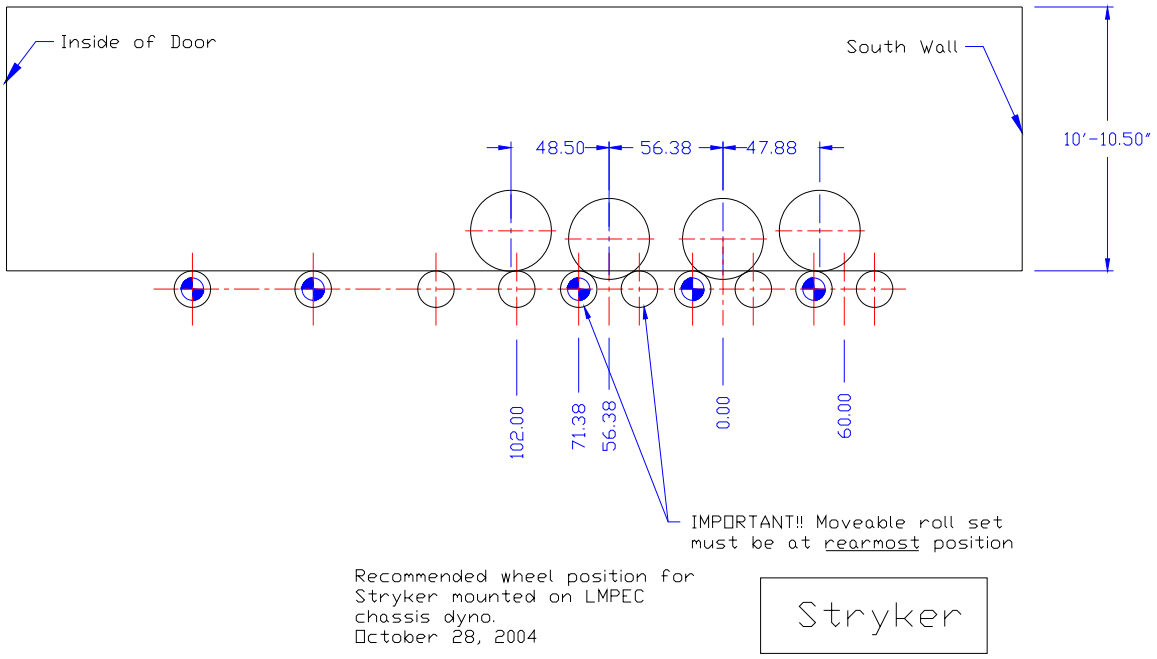


Figure 4. Roller configuration for the Stryker on Yuma Proving Ground's chassis dynamometer.

During testing, vehicles are restrained laterally to the surrounding concrete walls as well as vertically to the ground.

2.2 Towable Dynamometers

In order to test robots in varying terrain, the next logical step is to research outdoor, mobile dynamometers.

The dynamometer course at Aberdeen Proving Ground (APG), shown in Figure 5, takes place on a level, concrete oval-shaped course. The trailer is capable of imposing a 100,000 pound load on the tested vehicle. This is done through back driving large electric motors through an intricate power grid onboard the sled.

Figure 5. Dynamometer test at Aberdeen Proving Ground [11].

The TD-1200 is used by automotive manufacturers for testing passenger cars, 4WD recreational vehicles and mini-vans. This unit uses air-cooled eddy absorbers to develop up to 1,200 pounds of drawbar resistance. It can simulate grades up to 12 percent with vehicles up to 10,000 pounds GVW. Retail value of the TD-1200 is significant at $105,000, and is depicted below in Figure 6.

Figure 6. Superflow's TD-1200 towable dynamometer [12]

So far these last two examples have met the ability to traverse assorted environments, but are unattractive due to their high price tag. They rely on either

expensive, high capacity electric motors dissipating energy through a complex resistor grid or heavy hydraulic pumps and plumbing to impose a load. The goal of producing this dynamometer is to introduce a reliable, cost effective solution for developing vehicle metrics.

The most simple and time-proven concept is the mobile "tractor-pull" device. The history of these date back to 1962, when people formed two parallel lines as a tractor drug a flatbed trailer between the lane. One by one, the bystanders would leap onto the moving platform, gradually slowing down the tractor. This was given up in favor of a moving weight when the speed of tractors became too overwhelming for the warm "ballast". An example of such a varying load tractor sled, shown in figure 7, is small enough to accommodate a remote control truck.

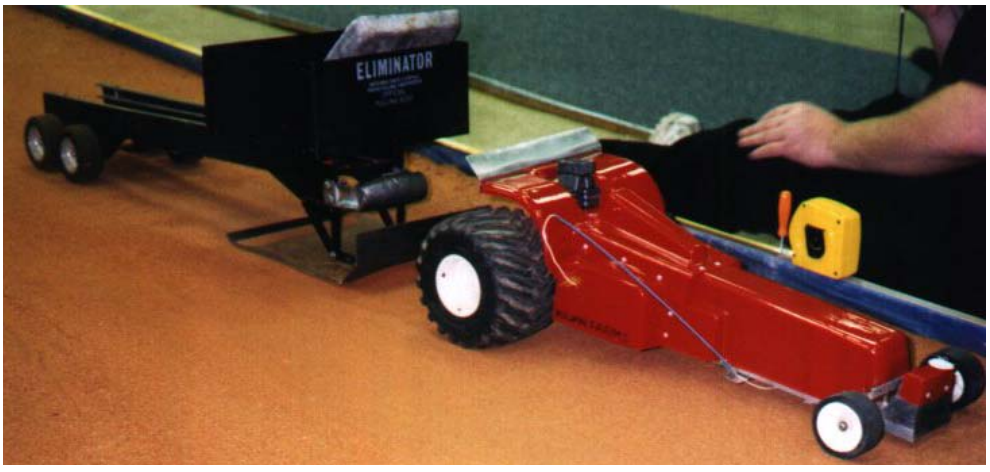


Figure 7. Tractor pull for remote control cars.[13]

One device worth mentioning is a setup to measure the starting tractive effort of model locomotives, shown in Figure 8. A rotary load scale measures the static drawbar

force by fastening to the rear hitch of the locomotive. This is actually the same method the NATO Reference Mobility Model (NRMM) uses to validate its vehicle simulations.

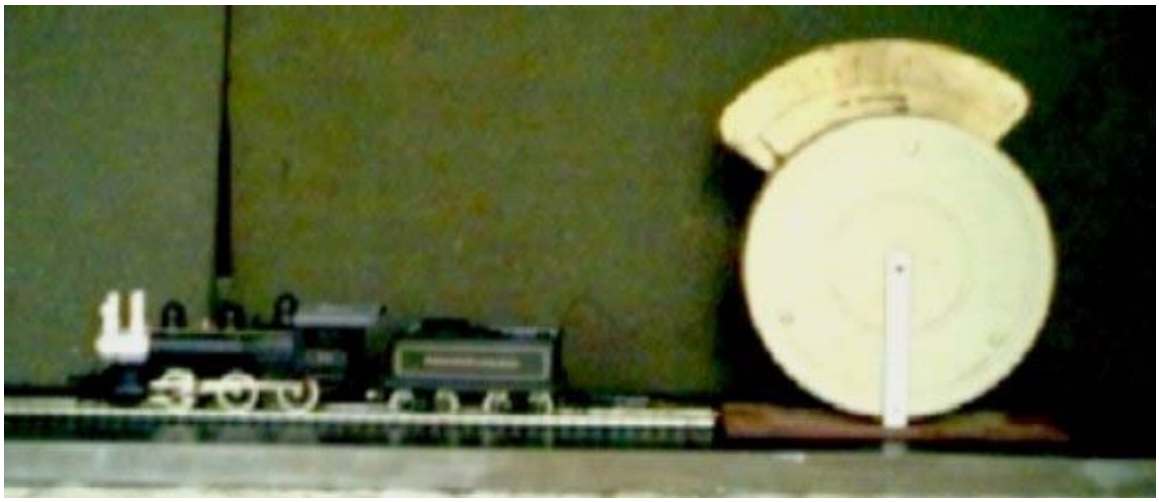


Figure 8. Setup used to measure peak static tractive force of locomotions [14]

The NATO Reference Mobility Model (NRMM) is a software mobility model the Department of Defense uses to evaluate new vehicles. This software, based on physical measurements such as tread pattern, ground clearance, power, etc, tries to predict how the vehicle will perform in different terrain. Investigators from the US Army TARDEC and ERDC used it to validate the Packbot model in the NRMM. One of the main metrics produced through the NRMM is a definitive go/no-go judgment of whether or not a given vehicle will be able to traverse a specified terrain. Traditionally, the NRMM was developed for modeling larger vehicles over 1500 pounds. Due to the military's shifting focus to smaller, lightweight vehicles, researchers are investigating the NRMM's ability to predict these vehicle characteristics. For validating the Packbot, they conducted static drawbar tests, as shown in Figure 9. There is a drawback to this method of validation

though, in that it will only provide one value of maximum tractive effort. How does one validate the model under varying resistance to the robot's motion? A more refined model can be input to the simulator by evaluating the robot's available effort under all loads and not merely peak traction.

Figure 9. Packbot undergoing static drawbar tests in various terrain. [14]

2.3 Army TOPS 2-2-604

The U.S. Army Test Operations Procedure (TOP) 2-2-605 [15] defines the procedures for determining power losses of large tracked vehicles. The Army uses a mobile field dynamometer, which imposes a variable load on the tested vehicle. Data

required from this test is towing force, vehicle speed, vehicle weight, gear range, and soil conditions.

This TOP standardizes testing methods with their mobile field dynamometer. This procedure was used as a model while creating test guidelines for the tow sled. As stated in the TOPS, the purpose of the field dynamometer is to “establish curves for comparing performance with similar vehicles and for predicting gradeability”.

Chapter 3 – Tow Sled Development

This chapter discusses development and construction of the tow sled prototype. Design criterion are introduced and explained. Figure 10 shows the preliminary free body diagram before any dimensions are established.

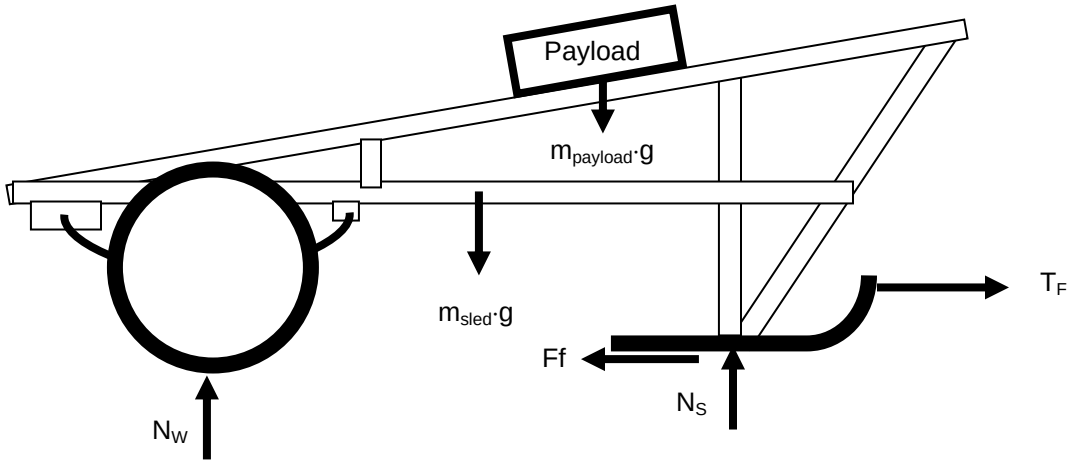


Figure 10. Free Body Diagram Showing Relevant Forces

Design of the tow sled started with the basic kinematic concept, which the payload would initially reside over the rear axle and gradually move towards the forward-seated skid plate.

The first design goal was to determine size and estimate pulling capacity of the "customer" robots. This was constrained to the small(light) vehicle classification from 31-400 pounds based on the JRP Master Plan. A broad range of this size robot was surveyed from the Navy's Spawar website along with unmanned vehicles developed here at Virginia Tech. Testing with a force gauge on the MATILDA, Johnny 5, and Gemini (The latter two are Virginia Tech's entries in AUVSI's Autonomous Vehicle Challenge)

robots gave a guideline for the goal towing capacity and weight requirements. This tow range is from 30 to 300 pounds.

The ideal test setup allows any size robot to start the pull with little initial drag. What this imposes on the design is that there must be low initial weight on the skid plate for any size robot. The sled should be able to initially balance its moment through the wheel axle. Design of the frame was an iterative process with initially little design constraints, bounded by the following moment equation and referencing Figure 11:

$$\sum M_A \sim 0 \qquad \textbf{(1)}$$

$$\sum M_A = (m_P g) X_P - (m_{CG} g) X_{CG} - (m_{SP} g) X_{SP} \qquad \textbf{(2)}$$

with m_P and X_P being the mass and offset of the payload from the rear axle, and m_{SP} and X_{SP} being the mass and offset of the skid plate, respectively. The goal of the payload weight selection is to find such a combination that the skid plate will hold little weight while still maintaining ground contact.

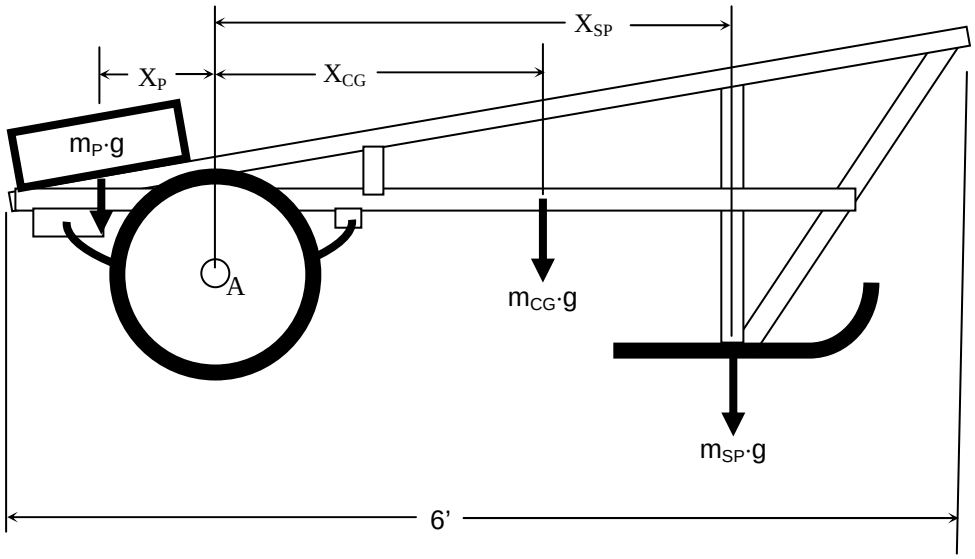


Figure 11. Free body diagram of the sled design showing significant moments about the rear axle.

Length was limited to what was feasible for transport, and in this case a pickup truck would most likely be transporting this from site to site. Width of the sled ideally matches that of the robot in test, as to not limit what passageways the robot may traverse. Two feet, while being wider than the average smaller robot, still allows this device to pass through most routes unimpeded. From all of this, an initial design was drawn in Solidworks, shown in **Figure 12**.

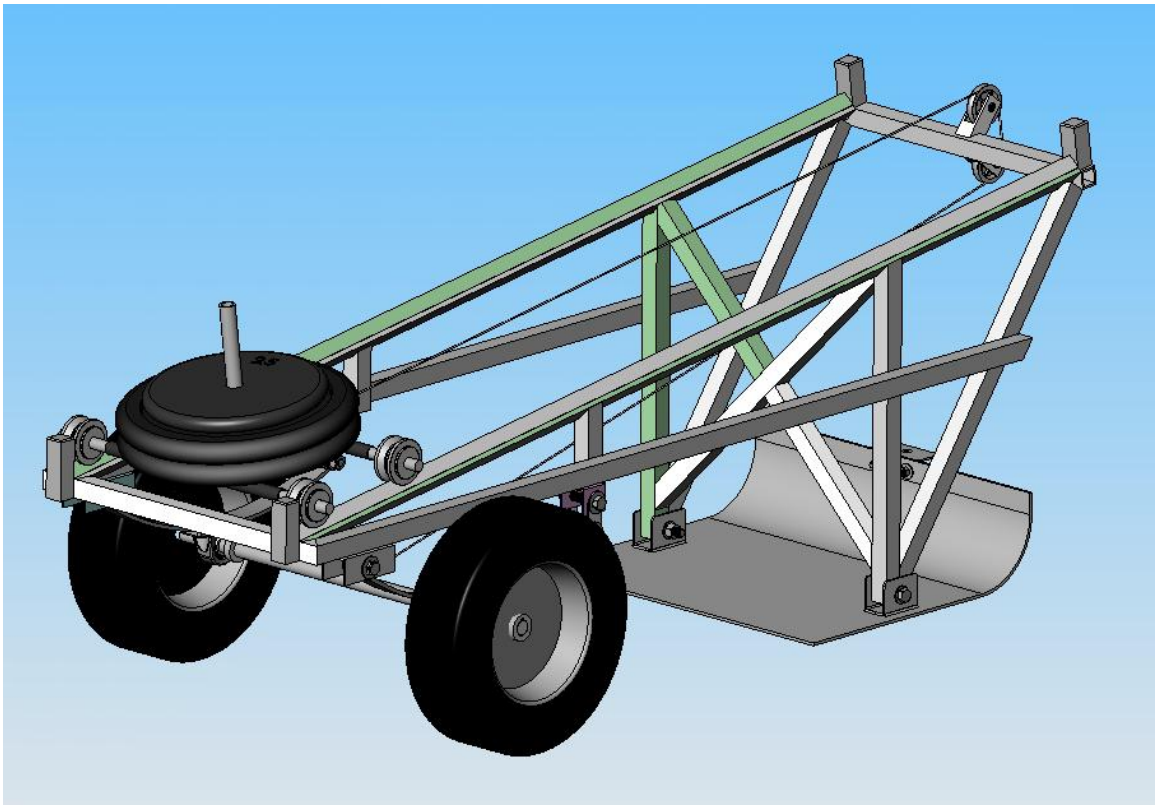


Figure 12. Solidworks drawing of the tow sled.

The sled is designed to become increasingly difficult to pull as it advances. This will test the robot through its full motor range of both high speed, low torque settings all the way through robot stall. This is accomplished by a weighted carriage that initially rests just behind the rear axle. Plate-style weights are added or removed from the carriage to adjust the maximum towing resistance. As the test vehicle advances the tow sled forward, a cable wraps around the axle, shown beautifully in Figure 13. This cable is connected to the weighted carriage through a pulley system in such a way that the carriage moves toward the front of the sled as the cable is wound, demonstrated in Figure 14. As the carriage moves forward, less weight is supported by the wheels and more weight is supported by the skid plate on the front of the sled. This increases the towing

load in a controlled fashion until the resistance becomes too great and the robot stops. This may be due either to the motors stalling or slipping of the treads or wheels.

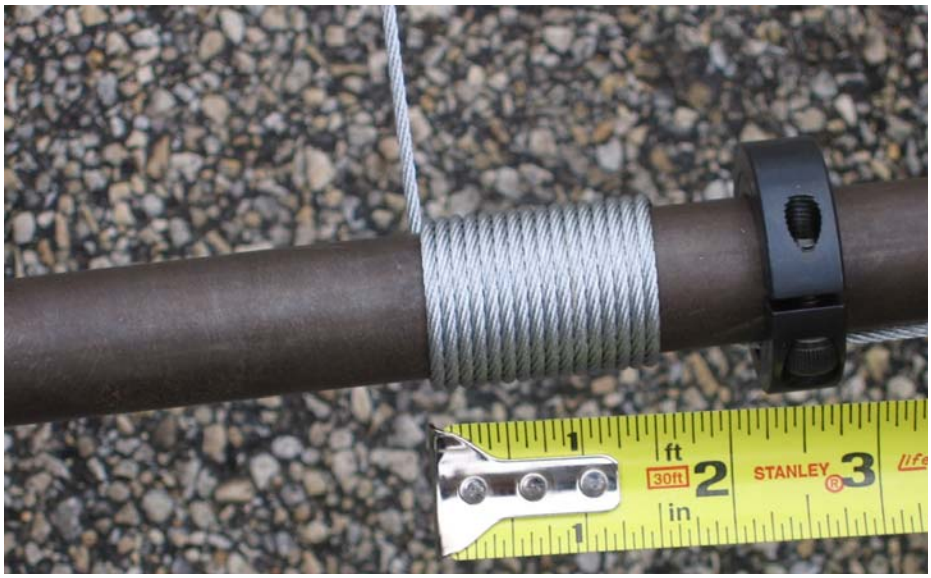


Figure 13. Cable winding around wheel axle.

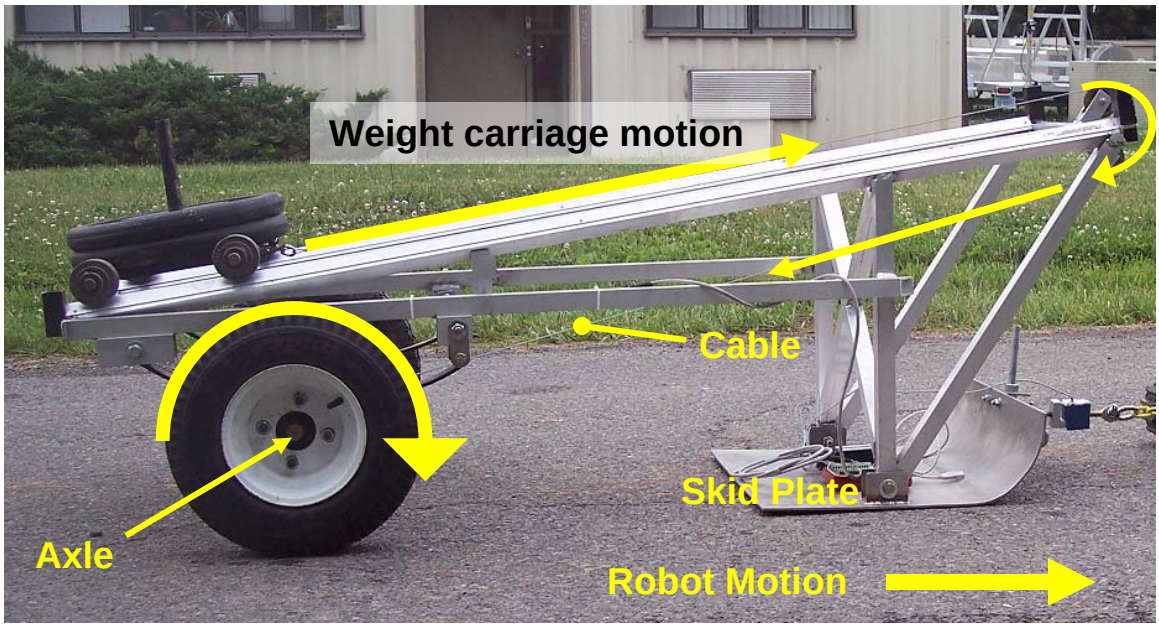


Figure 14. Tow sled shown with the weight carriage at starting position.

The skid plate material was chosen to be stainless steel for its durable characteristic and corrosion resistance to the inevitable rain. Rust developing on the underside of the skid plate would create less drag due to flaking. The asphalt medium was favored due to its repeatability from site to site, eliminating other obstreperous variables such as moisture content and surface roughness of non-paved surfaces. Trials were performed dragging a type 304 stainless steel plate over asphalt with various weights. The empirical kinetic coefficient of friction was around 0.4. Planning to bog down a robot pulling 300 lbf, this requires a 750 pound weight centered over the skid plate.

The six inch radius on the leading edge allows the sled to traverse less than ideal terrain. This bend also allows a vertical mounting point for the adjustable tow hitch, as seen in Figure 15. This adjustable tow hitch shown can provide a hitch point from zero to one foot high above ground. Ensuring the chain is horizontal eliminates vertical forces that could increase or decrease the skid plate normal force. Pick the lowest feasible hitch point on both the robot and sled. This is especially critical when operating in mediums other than asphalt. In soft soil, “one modification to the test procedure which will eliminate the shifting of the center of pressure rearward, which affects the trim angle of a test vehicle during a drawbar pull-slip test, is to locate the hitch point at ground level. When the hitch point was located at the traction-producing level, the trim attitude remained constant and level throughout most of the slip range [16].”

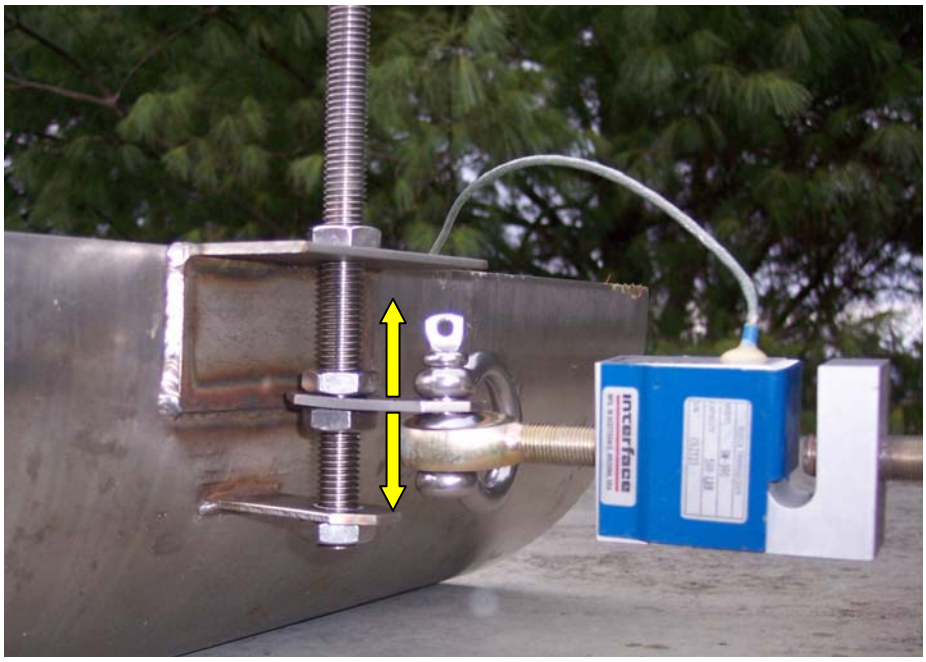


Figure 15. Variable hitch height ensures tow force is horizontal.

The size of aluminum chosen was extruded square tubing of 1.25” width and 1/8th inch wall thickness. A simple stress calculation was done to find the maximum bending stress the top rail would see under a worst case scenario. The highest stress this top diagonal tube will see when the weight carriage is positioned in Figure 16. The payload is centered over the longest gap in vertical supports of approximately 28 inches in width.

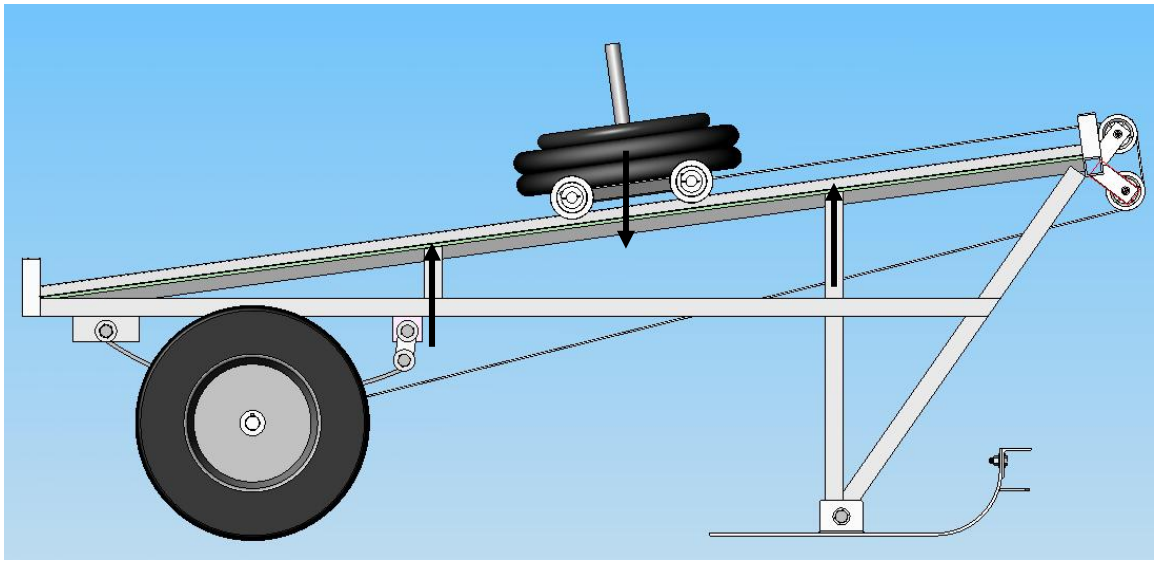


Figure 16. The top diagonal tube will receive the highest bending stress.

Assume the maximum load of 750 pounds is centered between the welded supports spanning 28 inches, shown in Figure 17. Since there are two parallel tracks, this 750 pound load should be carried evenly at 375 pounds apiece. The moment is highest at the center with 1,312.5 lb_f-in, found through the "Fixed supports – center load" example given in Shigley and Mischke [17]. The elastic flexure formula defines the maximum bending stress a member will see as

$$\sigma_m = \frac{Mc}{I} \tag{3}$$

where M is the applied moment, c is the distance from the centerline axis, and I is the area moment of inertia. The maximum stress seen by this center point under a 375 pound point load is 6,829 psi.

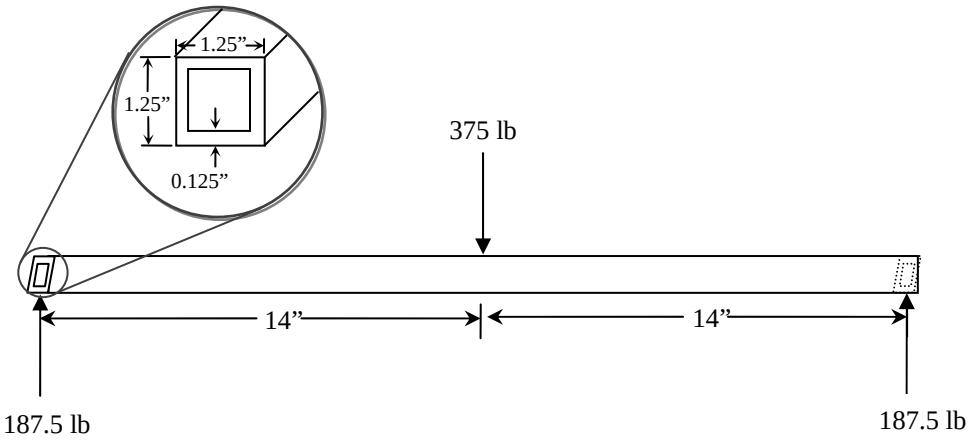


Figure 17. Free body diagram of the top tube under load.

Aluminum alloy 6063 was chosen as the frame material for its light weight and high corrosion resistance. With a yield strength of 21 ksi, this will be strong enough to use. All welding was done with a quality TIG welder and 4043 filler rod. This test device will at some point endure the elements, so this latter property was given high design priority.

A leaf-spring style rear suspension helps reduce wheel-hop if the sled is pulled over rough terrain. Each leaf spring is rated up to 1000 pounds. This suspension has minimal effect on geometry, permitting less than a one degree variation in track angle when a 150 pound carriage payload is moved to the extreme positions along the track. V-groove roller bearing wheels ensure the weight carriage moves smoothly along the tracks. To minimize rolling resistance, tire pressure is maintained at the maximum rated capacity of the tire. Figure 18 shows the completed tow sled pulled by Virginia Tech’s Johnny 5, the 1st place competitor at the 2004 Intelligent Ground Vehicle Competition.

Figure 18. Johnny 5 undergoing a tow test at JOUSTER Field Day, 10-30-04

Chapter 4 – Tow Sled Instrumentation

Desired values for analysis are wheel speed of the vehicle and the force it is exerting. This is accomplished through simple, well placed sensors. The tow sled is instrumented with an encoder to measure wheel speed/position and a load cell in the tow chain to measure force. The encoder and load cell output is read using LabJack U12, a USB-based multifunction data acquisition and control device.

4.1 – Interface SM-500 Load Cell

A 0-500 lb_f load cell, model SM-500 by Interface, is attached in series between the skid plate and tow chain, featured in Figure 19. Rod end joints thread to the load cell, allowing free articulation of the chain.

Figure 19. Close-up view of load cell connected inline the tow chain.

The weak signal from the load cell is passed through a custom instrumentation amplifier. This boosted signal is then sent to the LabJack which assigns a timestamp and sends the data to the laptop for processing.

4.1.1 Load Cell Calibration

The SM-500 load cell was calibrated in the lab on November 12, 2004 using measured blocks of iron. Weight was added initially in 2.5 pound increments up to 10 pounds, 5 pound increments from 10-20 pounds and then 50 pound increments from 50 to 400 pounds. Exceptional linearity of the signal output can be found in Figure 20. The actual signal to input ratio found through calibration was 6.19 mV/lb, but this was fine-tuned to 6.15 mV/lb to gain a closer relationship in the lower range of 0-100 pounds. This produces slightly more accurate results in the practical range of small UGV's.

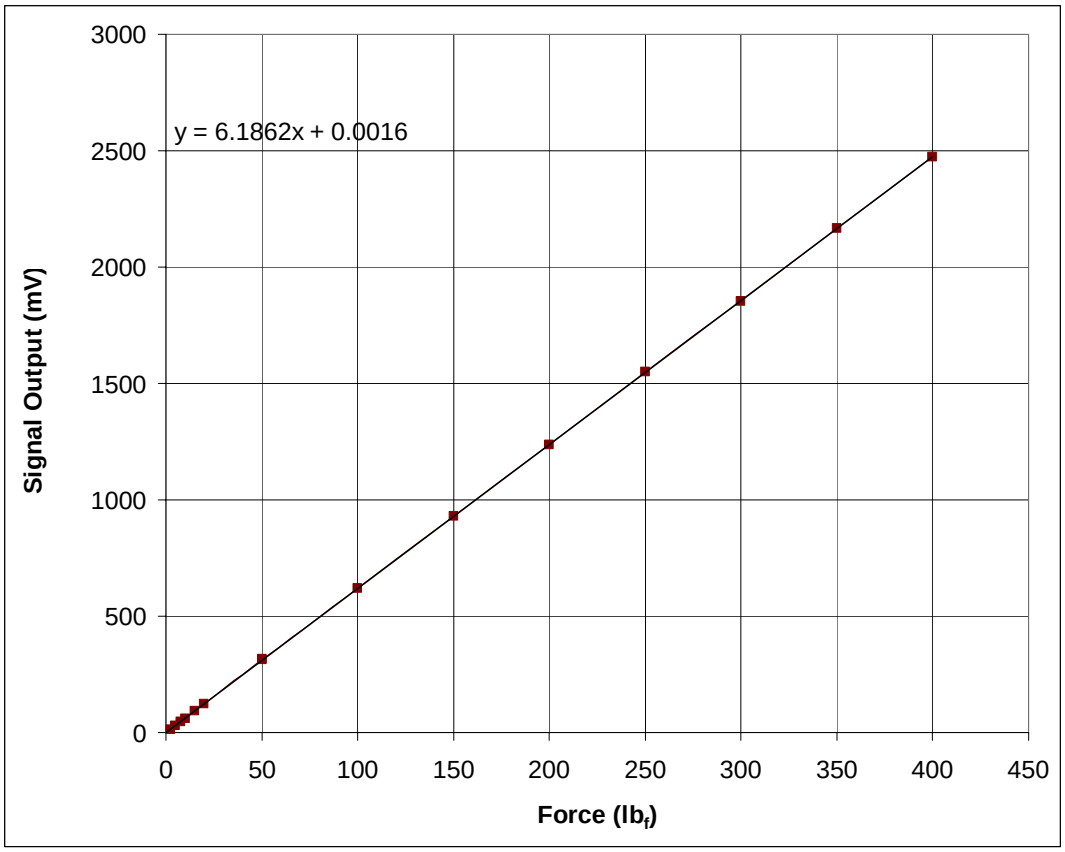


Figure 20. Calibration curve for the Interface SM-500 load cell.

The function of the load cell amplifier is to power the load cell's Wheatstone bridge as well as amplify its signal. The load cell transducer schematic is show in Figure 21. The MAXIM202 chip is intended to power serial communication when a ±10v power supply is not available. This takes the DAQ supplied +5V and through onboard charge pumps and capacitors, convert the +5V to ±10V to power the AD624 instrumentation amplifier. The AD624 is a high precision, low noise amp designed primarily for use with low level transducers; a load cell being one of these. The amplifier's variable gain was set at 200 through use of a 10kΩ/ 10 turn potentiometer.

The potentiometer was manually tuned to zero the load cell output when there was zero load present.

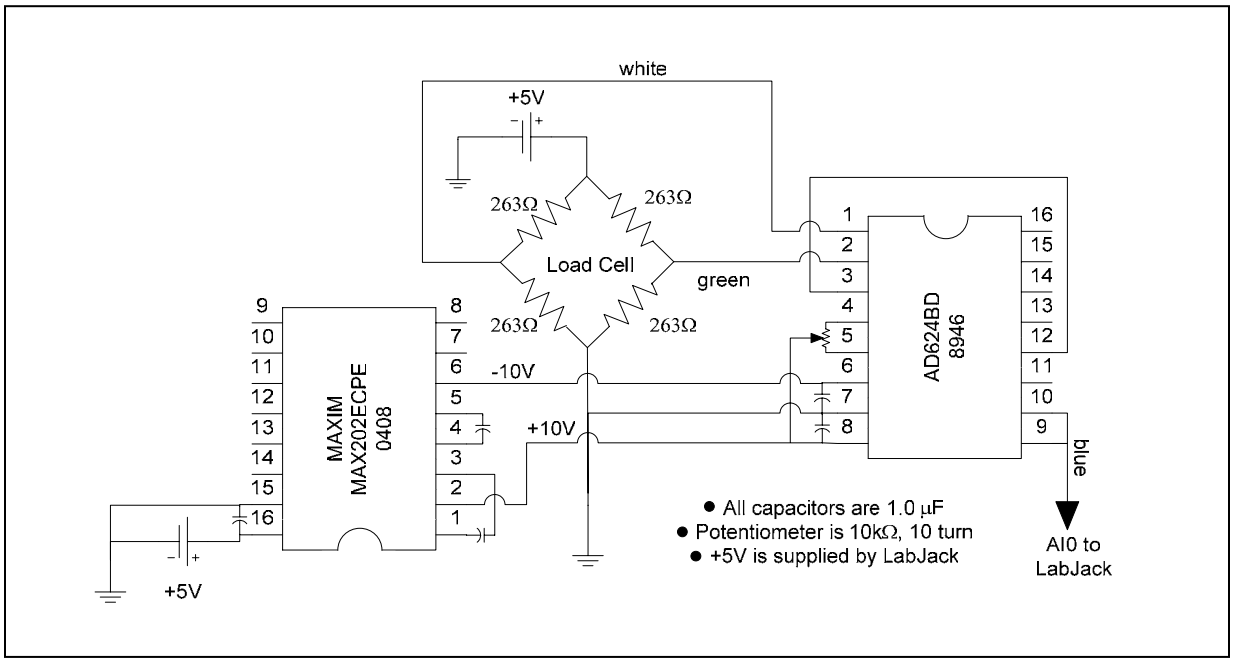


Figure 21. Load cell amplifier schematic.

4.2 – Tamagawa 10 bit Incremental Encoder

The Tamagawa 10 bit optical incremental encoder is shown in Figure 22. The encoder's purpose is to measure both sled velocity and position. This is placed inboard of the right wheel with a 1:1 gearing to the driven axle, as show in Figure 23. The resolution of this encoder is adequate at 1024 counts per revolution. With the current axle and wheel diameters, the encoder turns one revolution per 4.2 feet of forward motion. This gives an equivalent distance resolution of ±0.05 inches.

Figure 22. Encoder used on the tow sled.

Figure 23. Placement of the encoder to measure wheel speed and position.

Power to the encoder comes from the LabJack +5V port and counts are read through the "CNT" port. The encoder position is sent to the data acquisition device which passes this along with a corresponding timestamp. The software will then differentiate this into a velocity measurement.

4.3 Data Acquisition

Data is pulled from the sensors using LabJack U12, a versatile USB-based multifunction data acquisition and control device. The LabJack DAQ is a simple 12-bit measurement peripheral capable of sampling at a continuous 1.2 kHz. The software interface was developed through National Instrument's Labview program. A screenshot following a successful test is shown in Figure 24. The graphs update in real time during each run and show the test coordinator instant feedback. Variables such as data sampling rate and payload can all be changed through simple text boxes on the front panel. Ten hertz proved to be an acceptable collection rate. The virtual instrument can save each experimental run to a excel spreadsheet file for further data manipulation. The software is easily configurable to accommodate additional sensors.

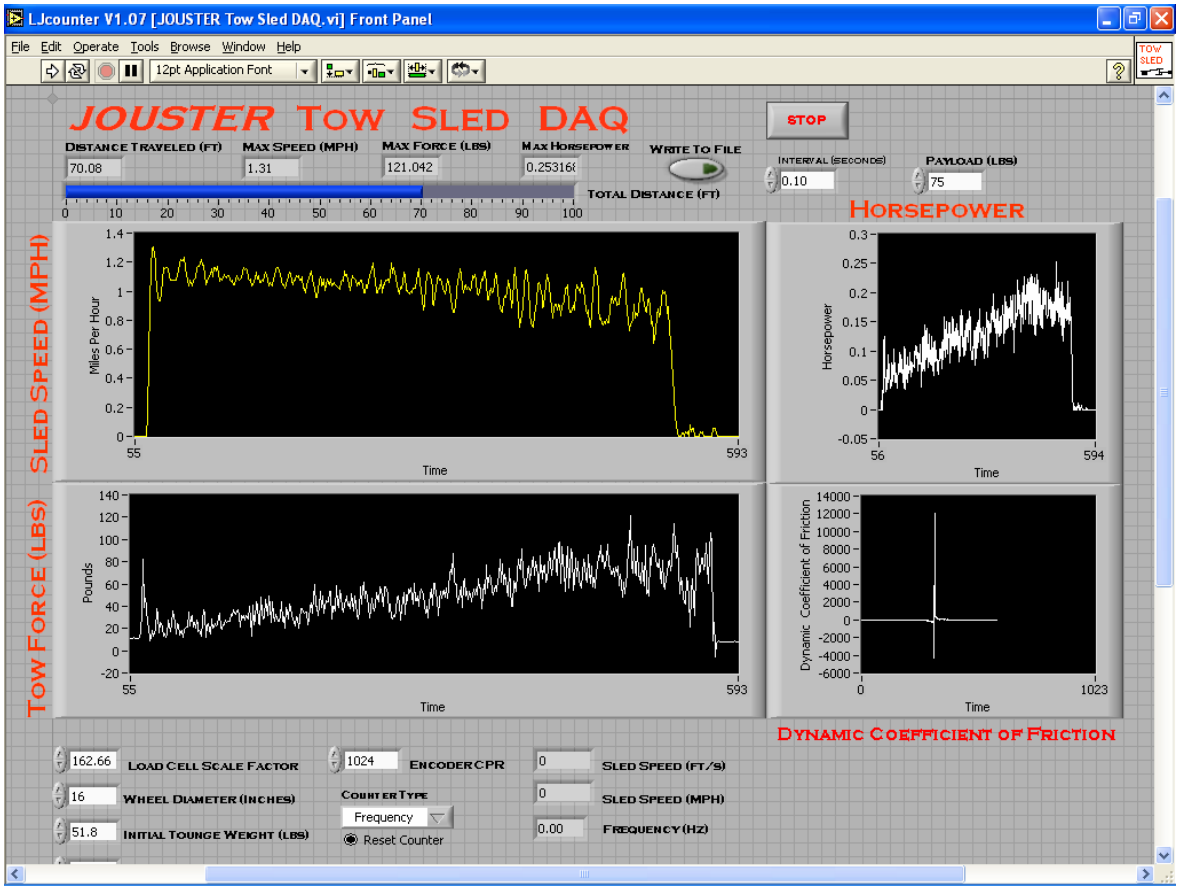


Figure 24. Data collection user interface

Chapter 5 - Test Guidelines

The tow sled presented here can be configured to undergo a variety of tests, ultimately limited to the creativity of the tester. Two uses are outlined briefly here: the standard pull on level asphalt, and the fixed load use.

5.1 Standard Pull Test

As with any testing metric, many extraneous variables must be eliminated to narrow down the metric to repeatable results. Considerations were given to allow for testing to be similar regardless of location.

The sled must be prepared by first making sure tires are at appropriate pressure for the 2 ply 16" tires (25 psi). Distance the encoder measures will be subject to the diameter of the tires under load. Make one pull with the carriage loaded with the correct weight and the DAQ running. This pull should stop exactly at 15 revolutions of the tires. Manually measure the elapsed distance with a tape measure and verify this with the "Total Distance" slider bar along the top of the Labview interface. The encoder and manually measured distances should read the same. If not, the "Wheel Diameter" field will have to be updated with the correct value of the tape measure. To find this correct diameter, divide the measured distance in inches by 15π.

First ensure that the vehicle is in good condition with fully charged batteries. On a level, asphalt test course, clear all rocks and debris from the intended vehicle path. Connect the sled tow chain to the UGV rear hard point and adjust the sled hitch point to match this height. For many robots, hitch point will not exist and must be improvised. Use sound judgment to make sure the load on the chain will not make the robot pitch

excessively when pulling. The tow chain must be perfectly horizontal. Figure 25 shows the TALON's rear hitch point. Place the desired payload on the sled weight carriage making sure that this weight does not tip the sled back, causing the skid plate to rise off the ground. Place sufficient weight behind the rear axle to allow the skid plate to only support about 20 pounds initially. This will allow even the smaller robots to execute a quick start. Take note of the UGV payload and control this variable, as this will significantly affect its traction. Sled tires must be set to the maximum rated capacity.

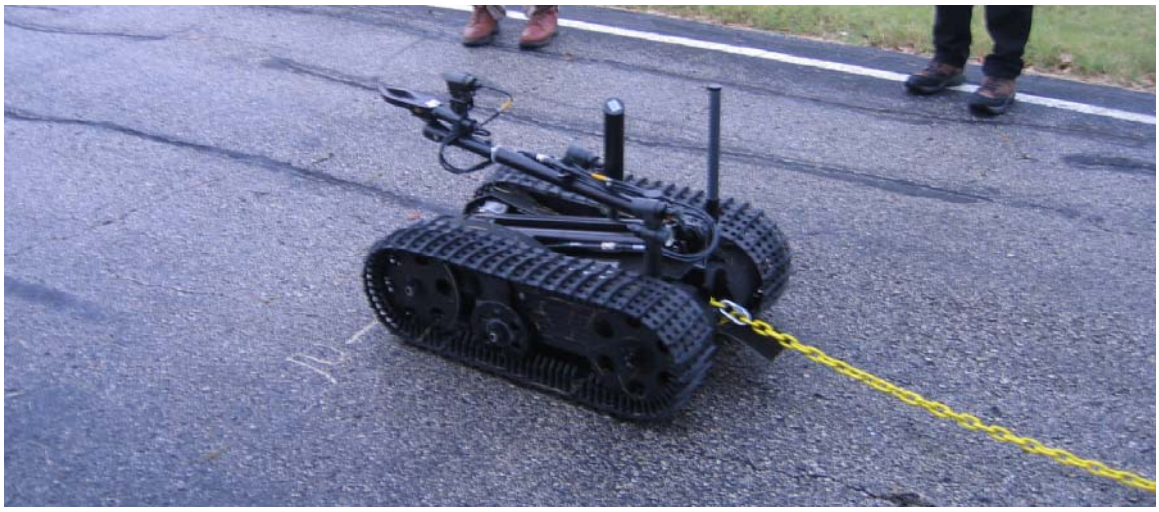


Figure 25. Hitch point shown on the TALON

When all sensors are responding properly to the data acquisition unit, begin data recording and commence testing of the UGV by steering it forward in a straight line at full power. The increasing load will allow variation over most of the UGV's speed range. The test ends when the robot's treads/wheels slip or when the motors stall. Required data are UGV variable settings, terrain, towing force, sled speed/position, time log, sled payload, and robot payload.

Data collection rates were taken at 10 Hertz and then filtered through a linear moving average of 25 points. Metrics should be found by averaging three pulls of similar variables.

Determining the benefit of changing specific UGV variables to achieve better performance, such as payload and tread material, is achieved with instant gratification simply by hooking up the sled and executing another pull.

5.2 – Fixed Load Use of Sled

The tow sled can also be used impose a fixed weight on the skid plate, producing near constant static and dynamic resistance levels. This is done by mechanically fixing the weight carriage to a calculated level on the weight track and disconnecting the cable to the rear axle. Battery life and UGV endurance while towing a load can be drawn from this fixed weight scenario. One could determine the force required to pull an injured soldier from the battlefield and recreate this in a safe environment. More desired metrics could be maximum distance until the batteries degrade, speed of rescue, maximum grade and terrain the robot/load combination can traverse, and qualitatively, how the robot controls react or differ during this weight burden. By gradually increasing starting resistance, the maximum load a robot can move from a standstill can be determined. It is up to the test proctor to decide which metrics to quantify from the tow sled use.

Chapter 6 – Test Results

This chapter encompasses the initial sled evaluation at the Southwest Research Institute's (SwRI) Small Robotics Vehicle Test Bed (SRVTB). It presents test results from the TALON robot. Pulls with varying criterion are represented in table format for quick comparison. Later, the MATILDA is tested under similar conditions as the TALON and results are compared showing how the tow sled can create an impartial test to validate competing platforms against one another.

6.1 Evaluation of the TALON at Southwest Research Institute

In November of 2004, JOUSTER members traveled to SwRI's Small Robotics Vehicle Test Bed (SRVTB) to help evaluate both Foster-Miller's TALON robot and the tow sled itself.

During the two day evaluation of TALON using the tow sled, 15 pulls were logged. Of these, 11 proved to be reliable data. Table 1 summarizes data recorded from these tests. Day two proved fortunate enough to provide us with a wet proving ground for testing different mediums. A picture from this monumental event is shown in Figure 26. This initial trial was not meant to be an exhaustive test, fulfilling a complete matrix of sled payload, speed setting, and terrain variables. Variables were selected based on time constraints, and they are presented in the service of displaying potential data collection through use of the tow sled.

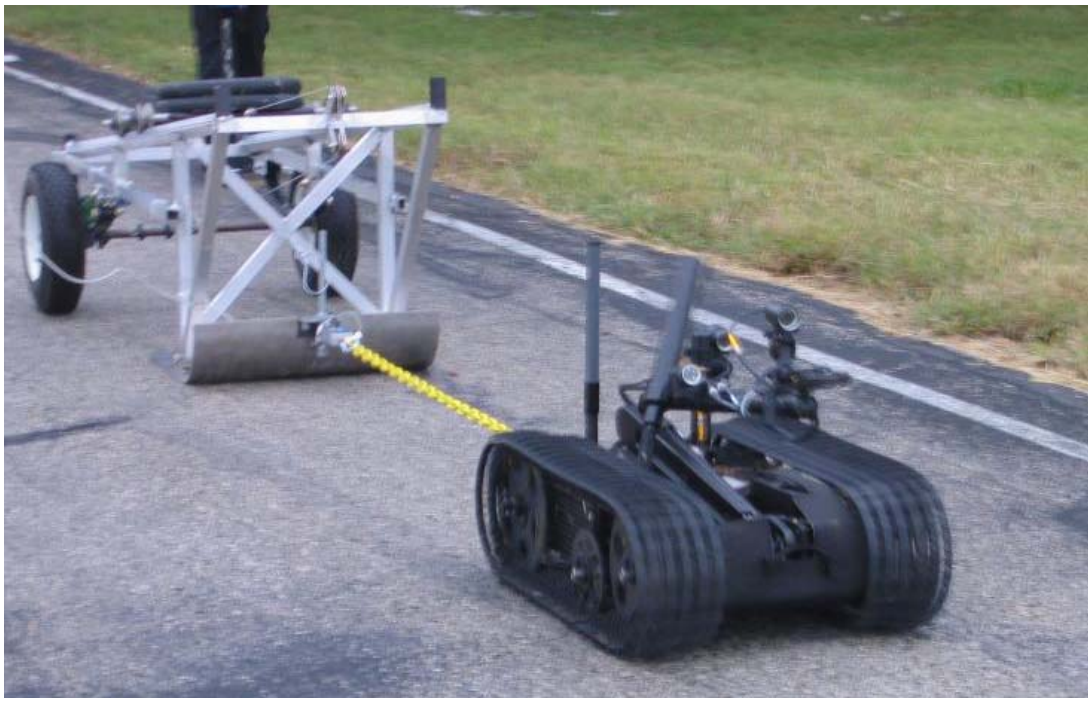


Figure 26. TALON undergoing pull test.

Table 1. Summary of results. Tests with the same test variables are averaged using a moving average of 25 points.

Terrain	TALON Speed setting	Sled Payload (lb)	Average Pull Distance (ft)	Average Max Speed (mph)	Average Max Force (lb_f)	Average Max Power (hp)
Asphalt	Half	75	70.3	1.14	80.34	0.19
Asphalt	Half	100	53.5	1.11	84.50	0.18
Asphalt	Half	125	51.3	1.14	79.59	0.17
Asphalt	Full	100	60.9	4.20	71.00	0.47
Wet Grass	Half	100	62.8	1.16	87.92	0.22

The drawback of dragging a weighted steel plate over pavement is that the data will be inherently noisy. Data appears to be a series of jerks. This is hard to avoid, but meaningful model validation can be extracted by passing data through a filter. A moving average of 25 points smoothed the data. As mentioned before, tow force will vary

linearly, also causing velocity to decrease in the same manner. Available power, being the product of these two, will follow a quadratic curve.

6.1.1 Terrain Comparison: Dry Asphalt vs. Damp Soil

The following Figures 27 through 31 display the results of six pulls. Three good runs from asphalt and wet grass are shown in Figure 27 and Figure 28, respectively. Pulls one, two, and four were performed on dry asphalt. Pulls twelve, thirteen, and fourteen were executed the following day on wet grass. All other variables were kept constant. The sled payload was 100 pounds and the TALON's speed setting remained at half of maximum speed. All data was previously run through a moving average filter of 25 points to smooth out data and eliminate jerks that would yield high outliers. The results of these pulls are summarized in Table 2. While the maximum speeds seen on both mediums are similar, the velocity profile degrades less when driving through the grass, displaying a flatter, more robust plateau.

Table 2. Terrain comparison of asphalt vs. wet grass with all other variables constant.

Terrain Comparison	**Asphalt**	**Wet Grass**
Speed Setting	Half	Half
Sled Payload (lb)	100	100
Avg. Pull Distance (ft)	53.5	62.8
Avg. Max Speed (mph)	1.11	1.16
Avg. Max Force (lb_f)	84.5	87.9
Avg. Max Power (hp)	0.18	0.22

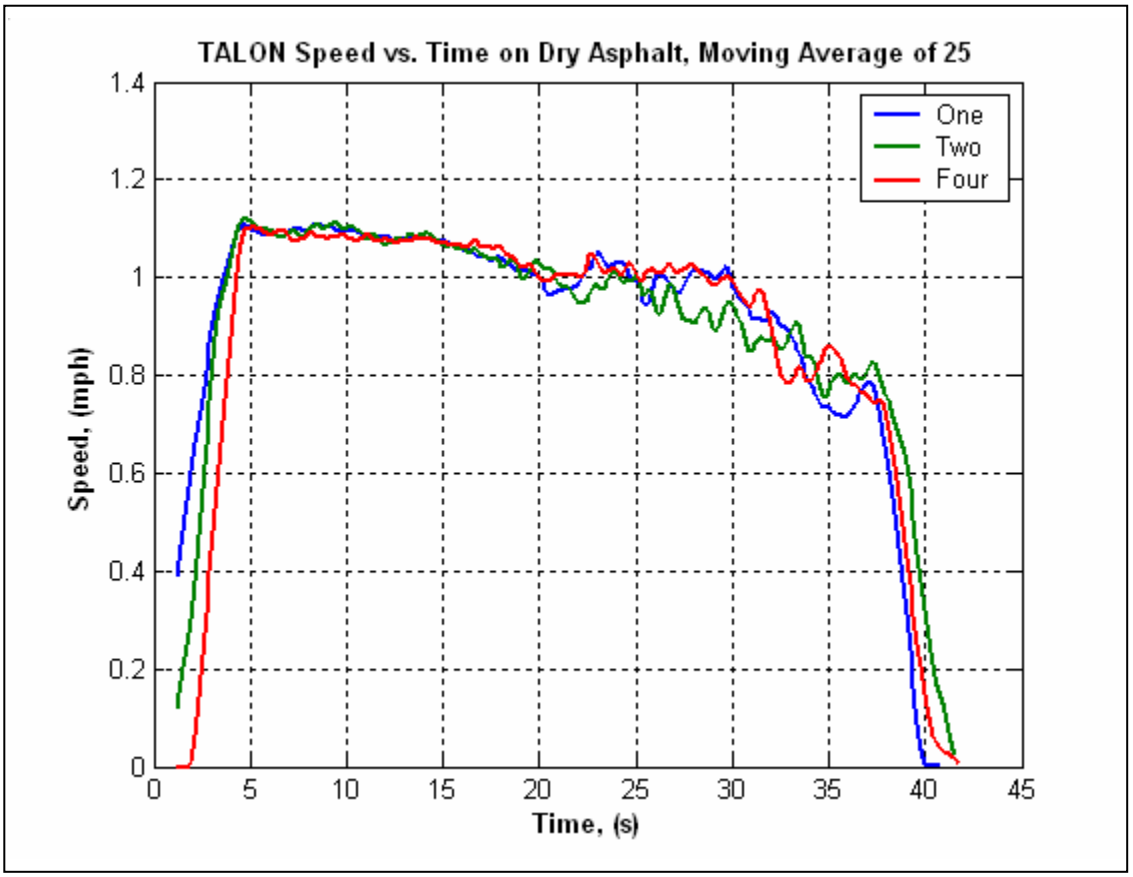


Figure 27. TALON Speed vs. Time on Dry Asphalt. Average maximum speed is 1.11 mph.

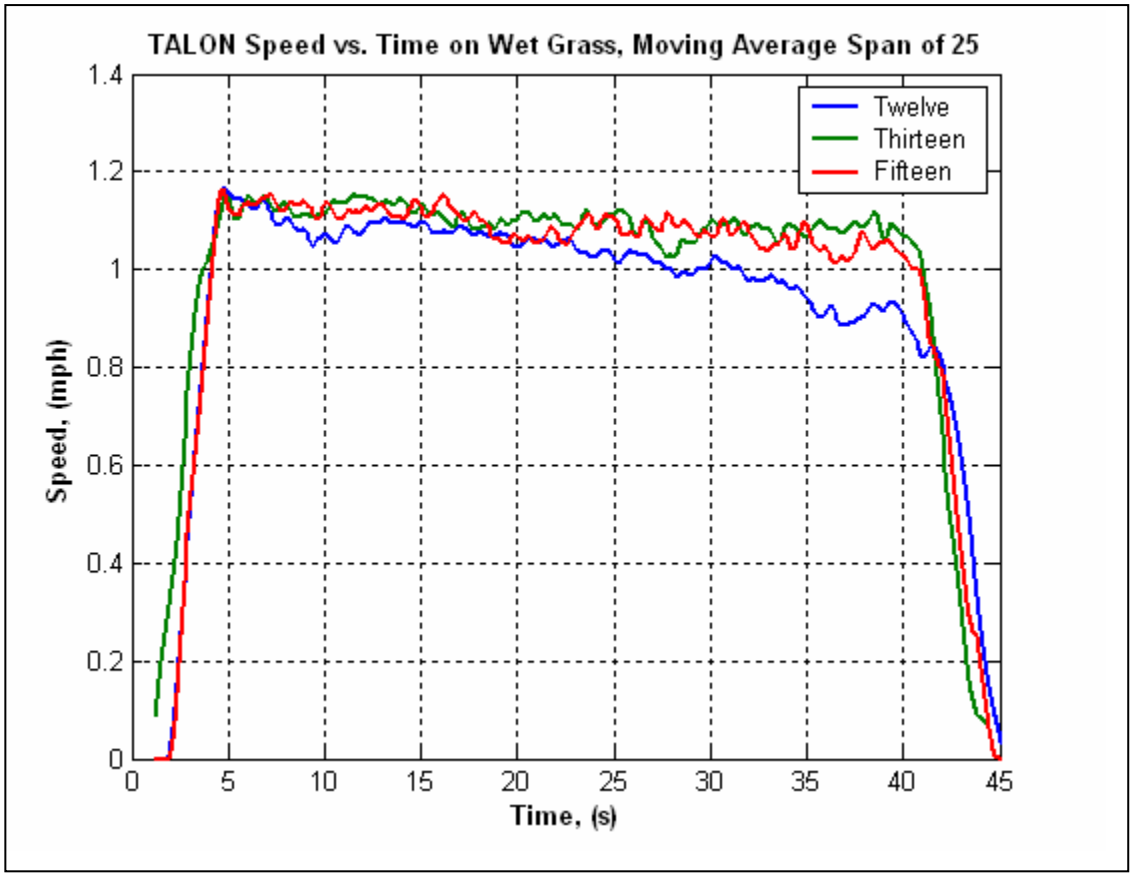


Figure 28. TALON Speed vs. Time on Wet Grass. Average maximum speed is 1.16 mph.

Force versus time plots are shown in Figures 29 and 30. Notice that the first runs across asphalt yield consistently lower tractive forces, as the TALON's rubber grousers slips sooner on asphalt than they do on grass. From three runs in each surface, the grass allowed the TALON to exert about 3.4 pounds more force than on asphalt. (Average peak of 87.9 lb_f vs. 84.5 lb_f on grass and asphalt, respectively) The fifteenth pull ended with both the drive motors stalling, meaning that the robot was at last power limited and not traction limited. This is a rare occurrence, as many small robots' treads slip before stalling due to their light weight (without a payload) and oversized motors.

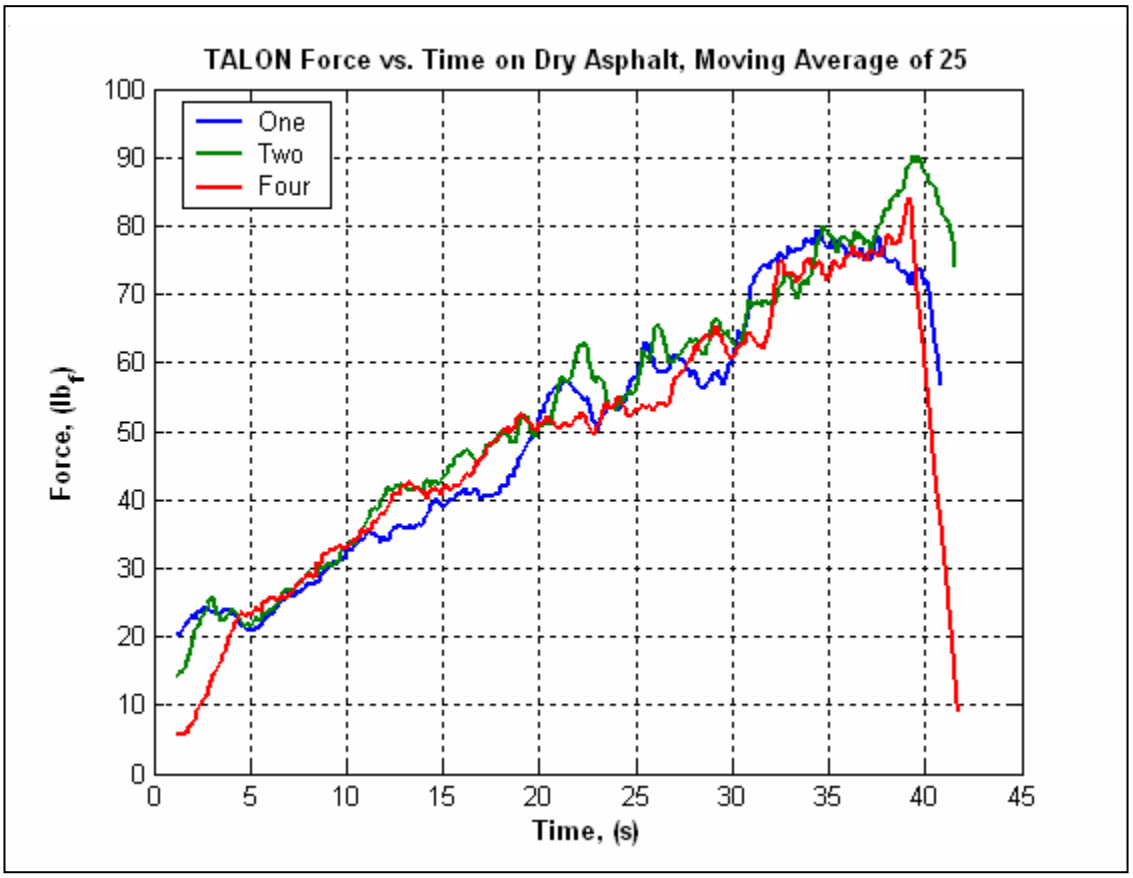


Figure 29. Plot of three TALON pulls of Force vs. Time on Dry Asphalt.

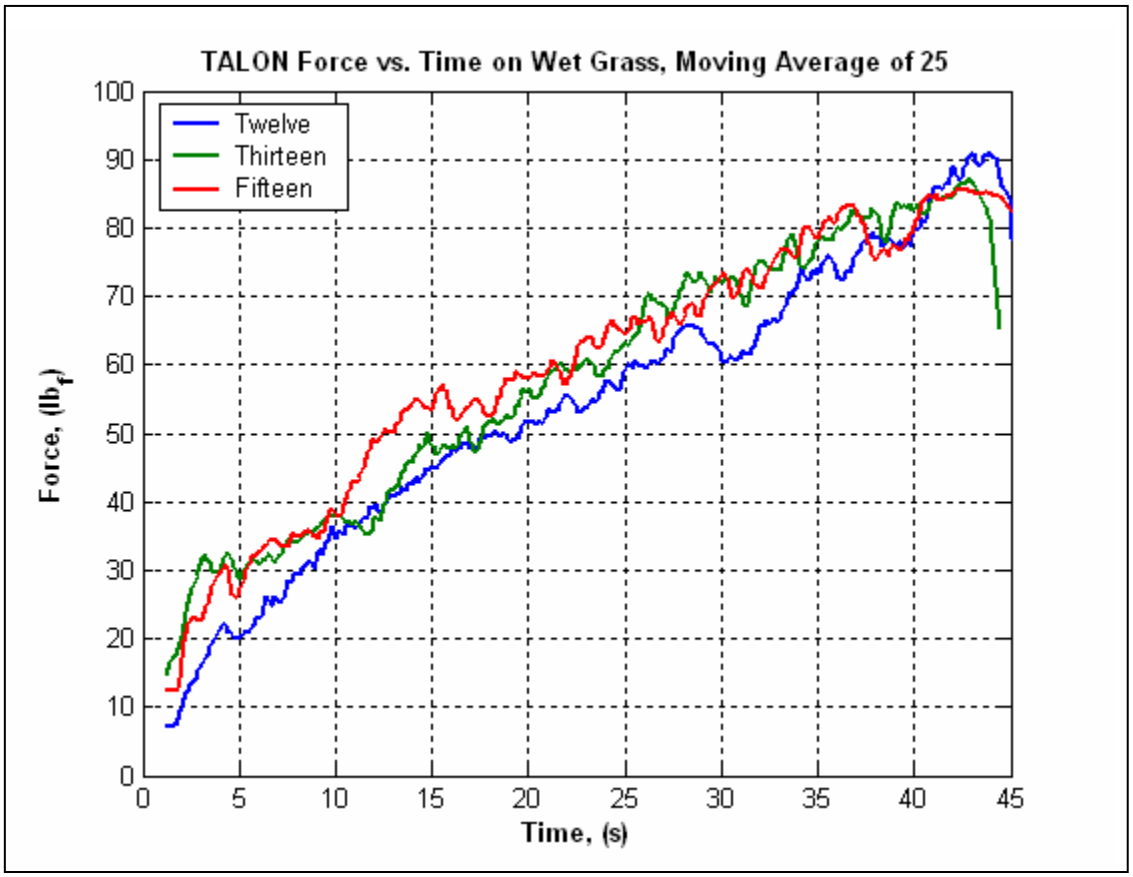


Figure 30. TALON Force vs. Time on Wet Grass.

Once again, TALON proves to display better characteristics in the softer terrain, as shown comparing power plots in Figures 31 and 32. Overall, the TALON showed better towing ability in wet grass than on asphalt. This is probably due to the aggressive track design using stiff rubber grousers that command a deeper grasp in softer terrain.

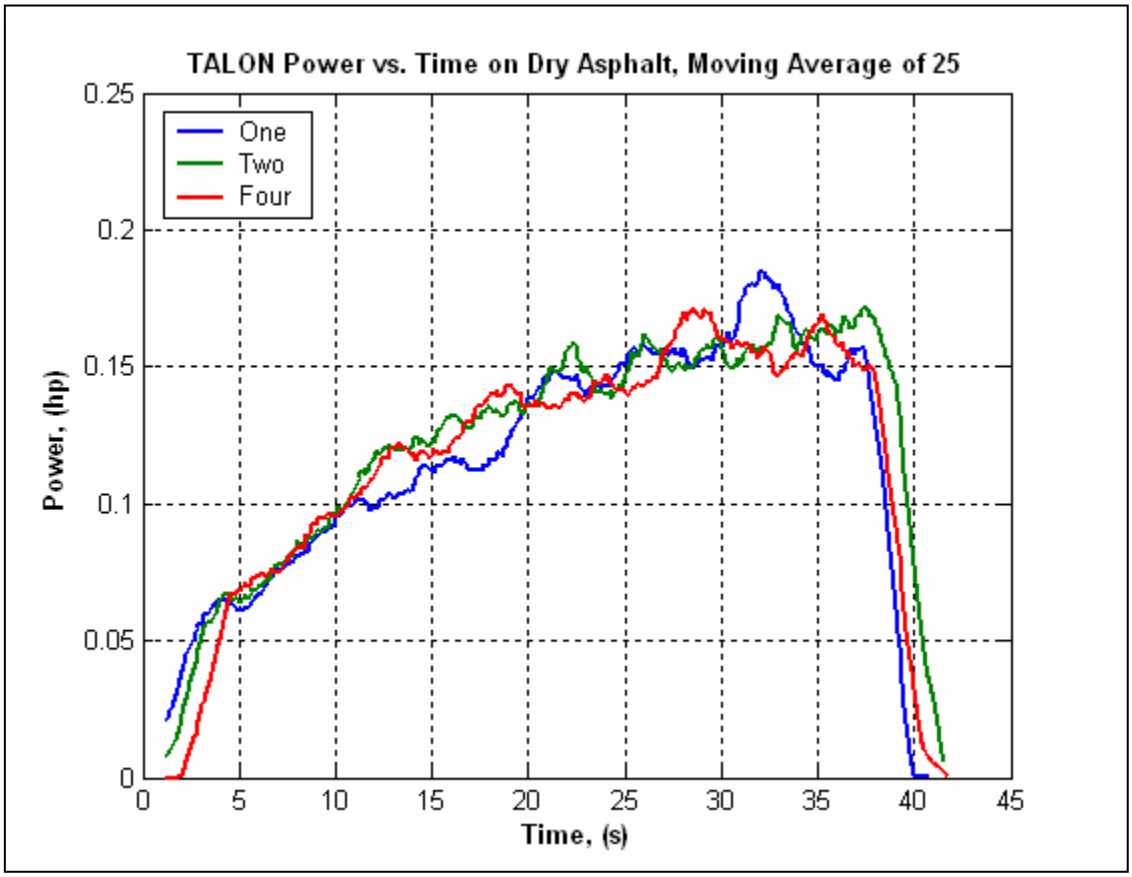


Figure 31. TALON Power vs. Time on Dry Asphalt.

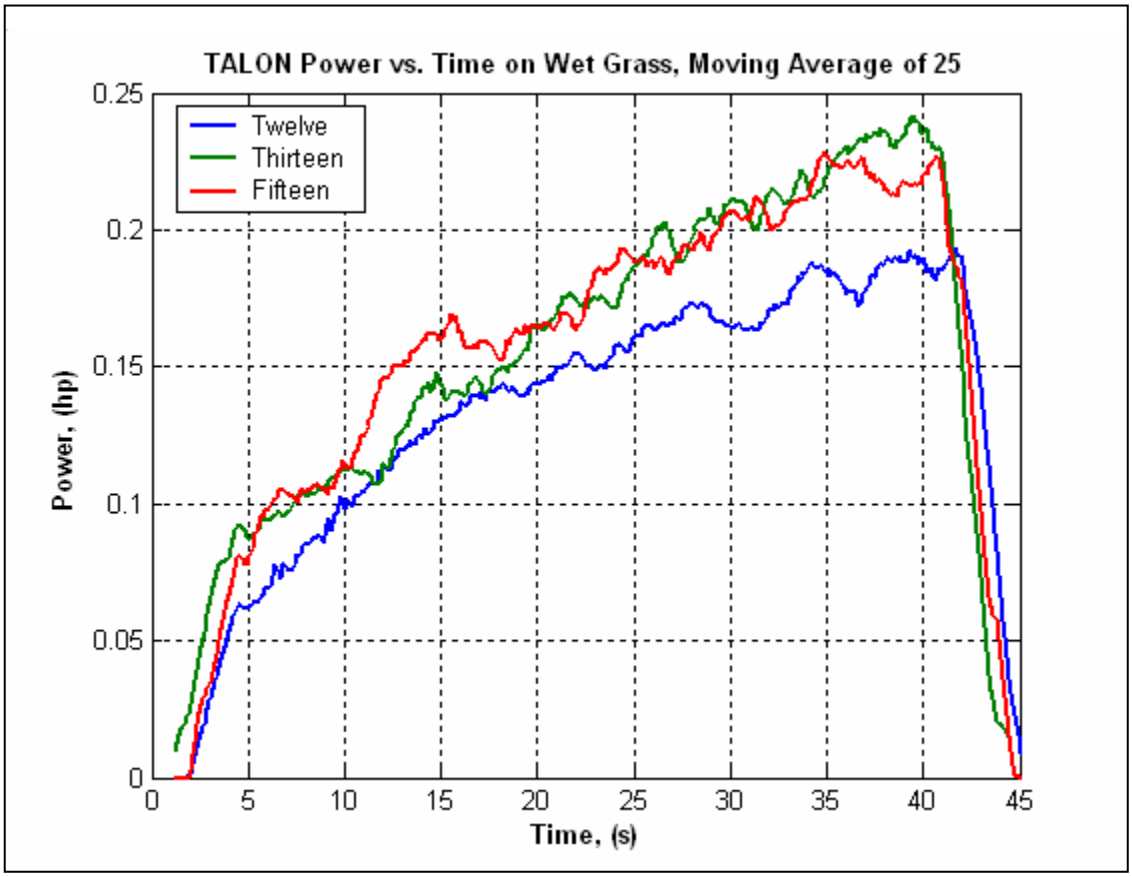


Figure 32. TALON Power vs. Time on Wet Grass.

As with any low pass filter, such as a moving average, there will be a range of data that will be removed. To find this frequency range, the following test was performed to find the cutoff frequency of the filter. White Gaussian noise was generated and passed through the moving average filter. The Fourier transform of the output was taken and averaged 100,000 times to produce concise data. This Fourier transform of the filter was plotted against the Fourier transform of the noise, shown in

Figure 33. The cutoff frequency producing a magnitude of 3 dB lower than unity is around 175 hertz. This means that any input to the system above such frequency will be eliminated by this simple moving average of 25 points.

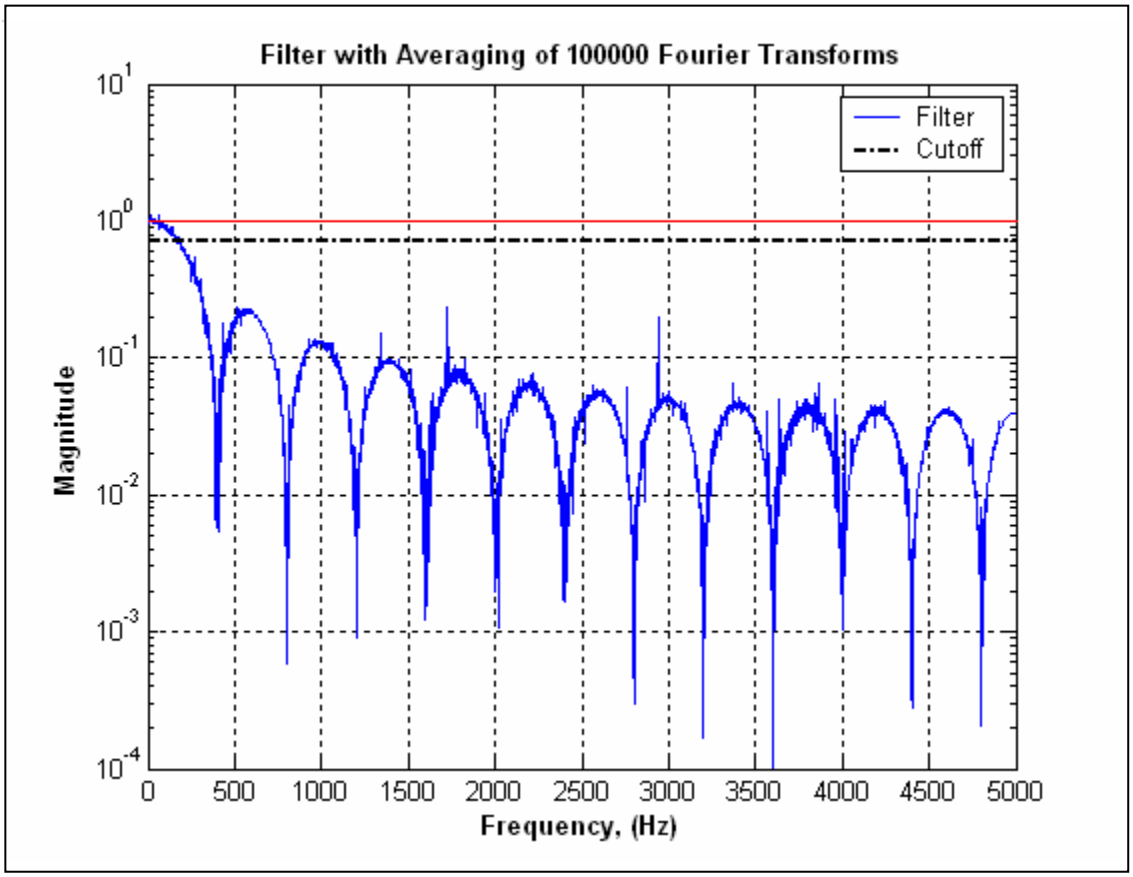


Figure 33. Fourier transform of filter compared plotted in the frequency domain.

6.2 Vehicle Comparison: TALON vs. MATILDA

The purpose of this tow sled is to be able to benchmark one competing SUGV versus another. The two robots referenced here were chosen because they are similar in size and were readily available. This section will discuss performance comparisons between two platforms: Foster-Miller's TALON and Mesa Robotics' MATILDA, shown in Figure 34. Note that the TALON weighed 83 pounds and the MATILDA weighed 64 pounds as tested. For its power supply, the TALON was equipped with a 36V 5500mAh

lead acid battery and the Matilda was equipped with four 12V 8000mAh NiMH battery packs.

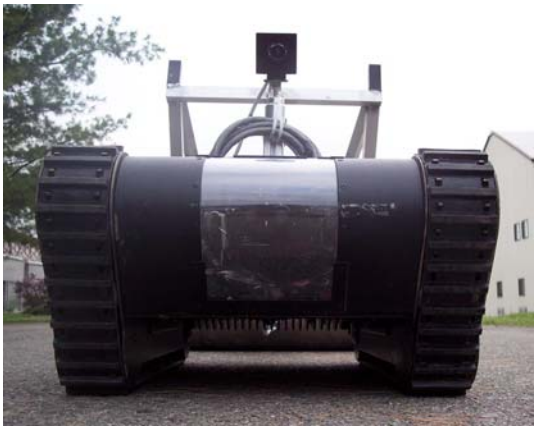
a)  b) 

Figure 34. Close-up view of the competing SUGV's, a) MATILDA and b) TALON. Notice the profile and void ratio of each robots' treads.

Test variables were kept as consistent as possible. Both platforms drove over clear, level, and dry asphalt. Speed settings on both robots were at half of their maximum setting. One hundred pounds of payload was loaded onto the weight carriage. Tire pressures of the sled tires were both at 25 psi. Both robots carried a minimal payload in their cargo bays (MATILDA held a seven pound IBM laptop for communication with the OCU). Finally, both tests were conducted immediately after a battery change with fully charged batteries.

Figures Figure 35, Figure 36, and Figure 37 present overlaid comparisons of a typical TALON and MATILDA pull.

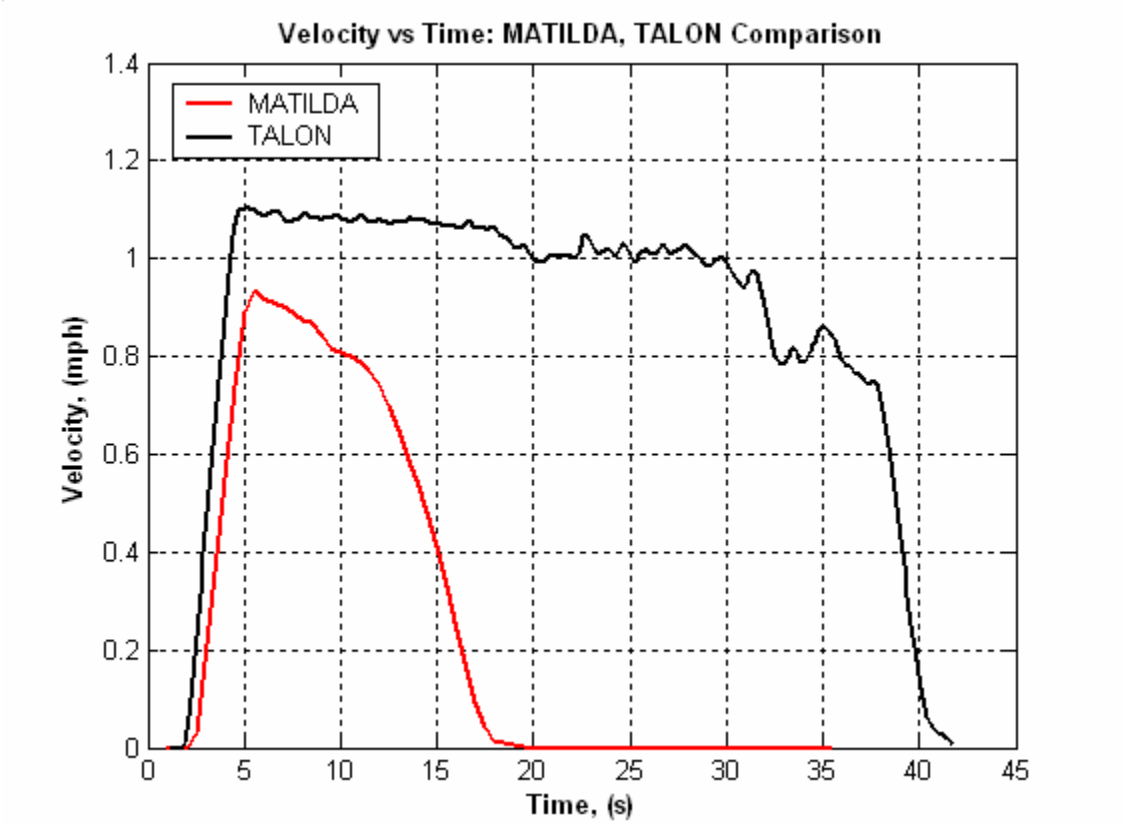


Figure 35. Velocity vs. Time Plot comparing the TALON vs. MATILDA . Speed = Half, Sled Payload = 100 lb. Clear, level asphalt surface and fresh batteries.

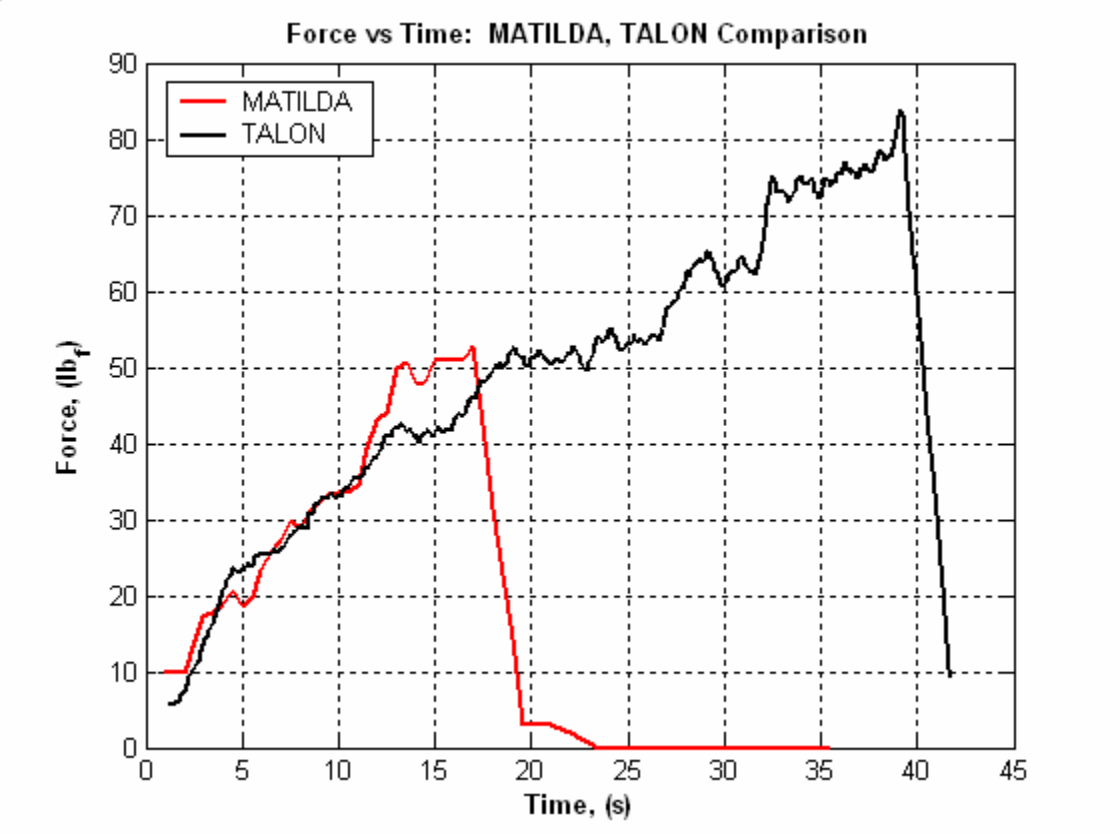


Figure 36. Tow force vs. time plot of TALON vs. MATILDA .
Speed = Half, Sled Payload = 100 lb. Clear, level asphalt surface and fresh batteries.

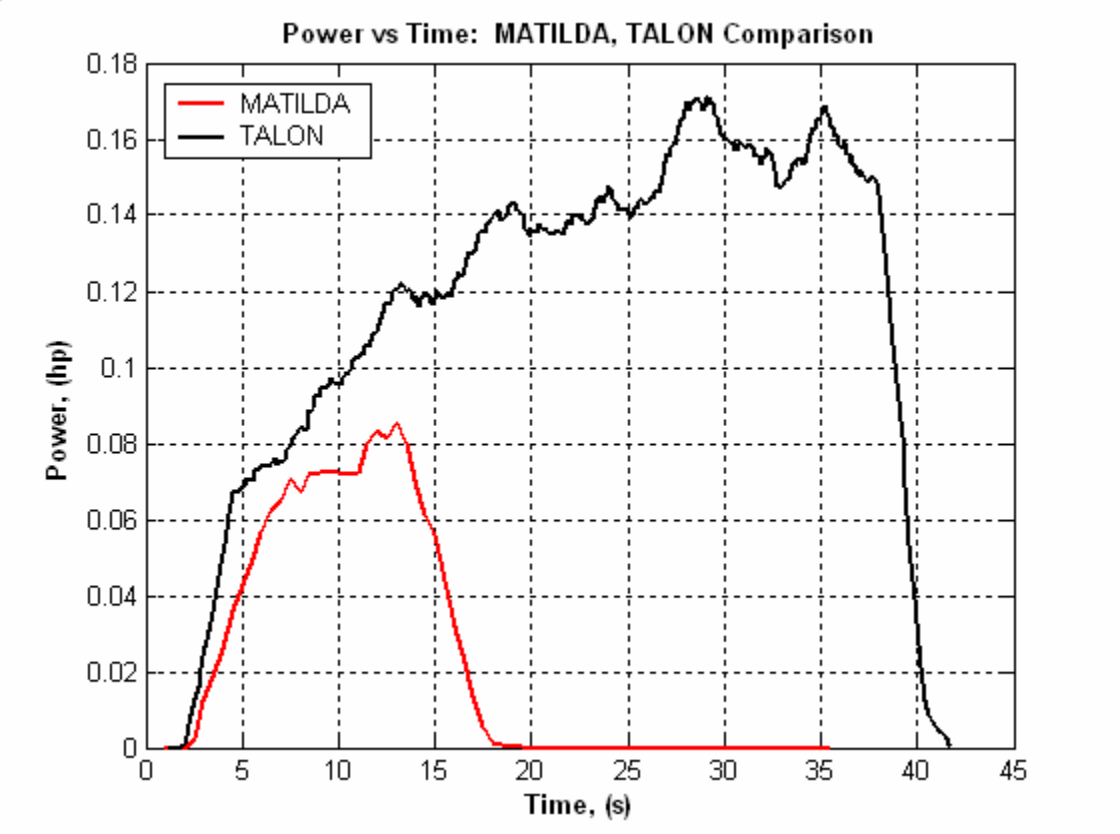


Figure 37. Power vs. time for the TALON and MATILDA.
Speed = Half, Sled Payload = 100 lb. Clear, level, asphalt surface and fresh batteries.

6.3 Drivetrain Efficiency

Talking to dozens of UGV developers, none could quantify the efficiency of their drivetrain propulsion. This is an area widely regarded as left to general intuition. Treads are not as efficient as wheels and the narrower the wheel, the less energy required is a general rule of thumb. This tow sled lets us delve deeper than these proven anecdotes, and actually quantify the drivetrain's level of efficiency.

This velocity vs. force curve can be compared to the original torque speed curve of the motor. Differences between the two will show how much power is required to drive the robot. This is analogous to comparing the brake horsepower of an engine to the actual power delivered to the wheels. Power delivered to the ground will be less due to drivetrain losses (gearbox, differential, and other auxiliary power draws).

The MATILDA robot employs the Mabuchi motor model RS-775SF-7513. The motor torque/speed curve was obtained from the manufacturer and is contained in Appendix B – MATILDA motor specs. From Mesa Associates, gear reduction to provide necessary torque is 134.17:1. The theoretical no load gearbox output is 144 revolutions per minute, resulting in a tread velocity of 3.62 feet per second. The kinematic tread motion to gearbox relation is necessary, and results in 1.51 linear feet for every revolution of the gearbox output shaft. This result also provides the moment arm from the gearbox to drive treads. This is simply the radius of the recently discovered pulley circumference, and is 2.88 inches.

These are the required robot characteristics necessary to numerically convert motor capabilities to theoretical robot potential. Motor angular velocity is reduced through the gearbox and then translated to linear motion. Equivalent tow force is found starting with motor torque, increasing this through the gearbox ratio, and then dividing by the moment arm to find the theoretical force the treads can deliver. Figure 38 provides the useful theoretical versus actual robot performance.

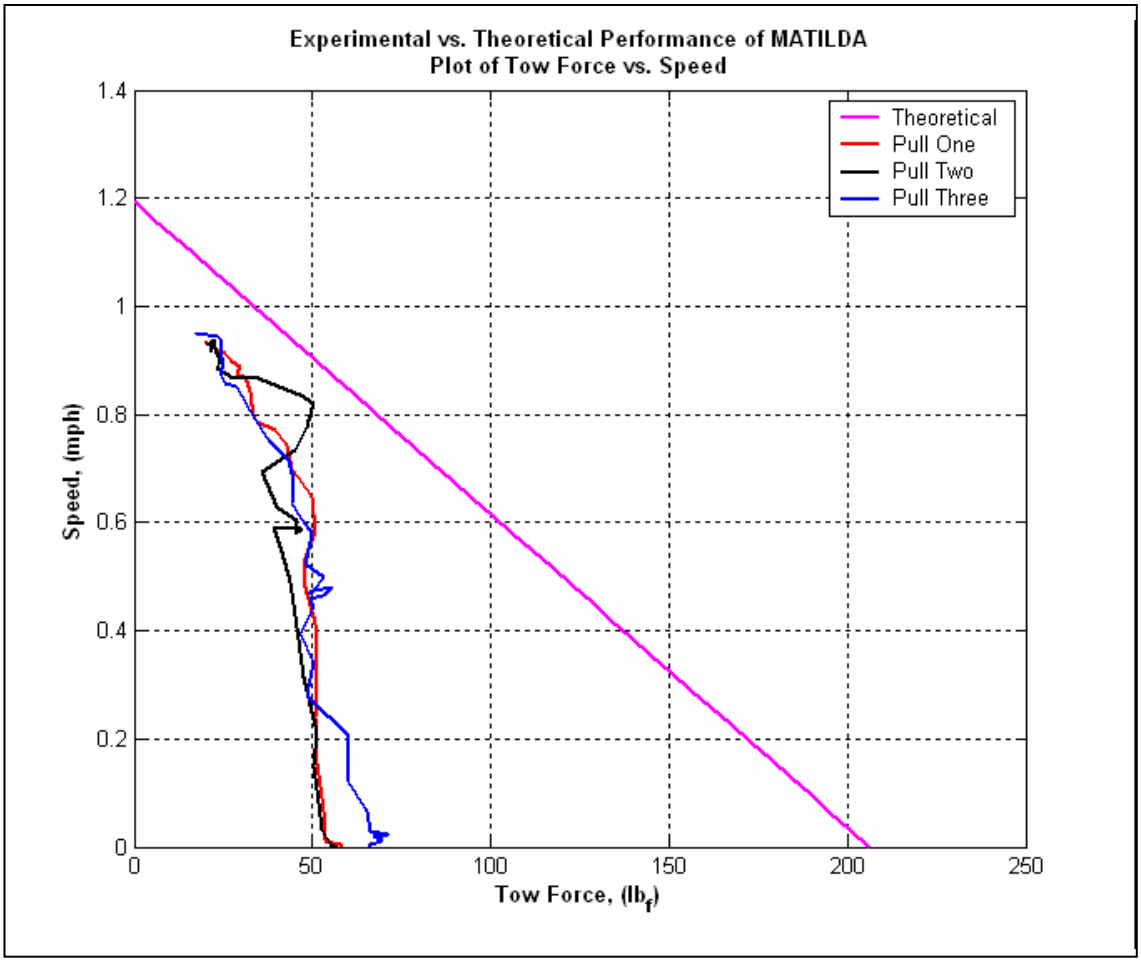


Figure 38. Actual vs. Theoretical performance of MATILDA

It is important to note in all of the actual test lines, that the tow force and velocity from the tested MATILDA do not actually meet zero. Speed will not match the no-load motor equivalent speed since the sled imposes a load at all times. This explains the gap in pull data around 0.95 mph along the ordinate. Figure 39 shows MATILDA's power plotted against tow force. The actual output found through testing is represented by the blue arc, while the theoretical power assuming an ideal drivetrain and no losses is represented by the larger, green arc. Note that these results display the performance of an unloaded MATILDA. Only a seven pound laptop computer resided in its payload bay.

More impressive tow force figures can be realized by adding a more significant payload, allowing this higher traction to deliver more towing capacity without slip.

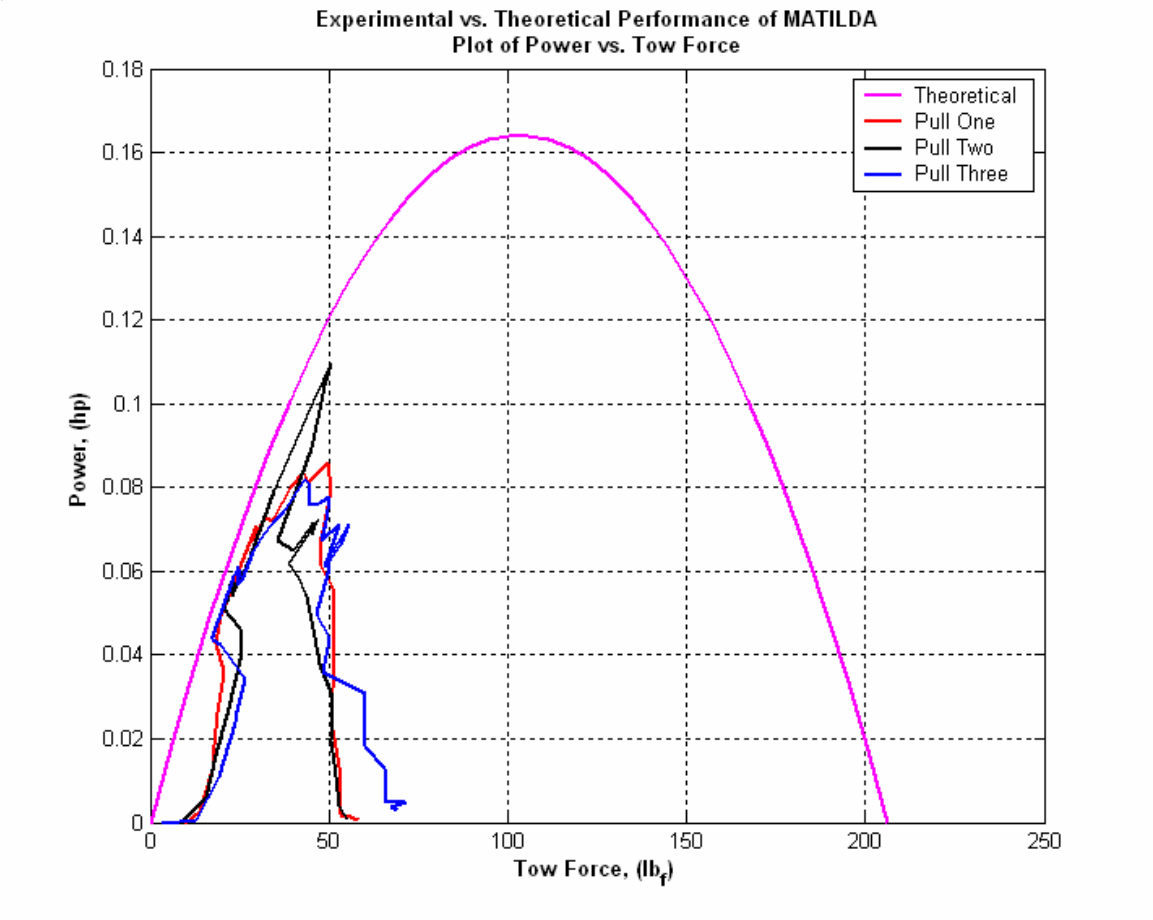


Figure 39. Theoretical vs. actual power vs. tow force of MATILDA.

What this translates to in practical use, is that in order to use this robot for towing applications, the robot would benefit from sharing all or a portion of the payload on top of the platform as opposed to a pure towing configuration. This is similar to a fifth wheel truck, which supports a significant portion of the trailer weight. Adding more payload will push MATILDA's 'Actual' performance curve in Figure 47 and Figure 39 more to

the right. By doubling the weight, one should expect the MATILDA to tow around 100 pounds instead of slipping at 50 pounds.

Chapter 7 - Conclusions

With the military shifting its focus to lighter and a more agile fighting force, small ground vehicles are proving more and more to be the answer. Test centers such as Aberdeen and Yuma must be able to transition to accepting lightweight vehicles for evaluation. This requires new facilities and tests customized for these smaller vehicles.

The NRMM traditionally has been used to model large vehicles over 1500 pounds. Untested in the realm of small, lightweight robots, the software must be validated with real world situations. The variable-load tow sled is a useful validation tool.

An important finding is that robot payloads must be increased in order to take full advantage of their high performance motors. Motor/gearbox combinations are often selected for their high speed capabilities, but this also offers a fair amount of torque not being used at low speeds. When called upon for towing tasks, these lighter robots will benefit by sharing some portion of the payload on the actual robot platform, and not entirely on a separate towed trailer. This introduces the value of creating fifth wheel trailers for towing, such as a revised sked stretcher that lies partially across the robot platform.

An increasing number of small unmanned vehicle platforms are being introduced to the market with widely varying claims regarding their performance. It is important to quantify performance in ways that are consistent with the expected uses of these systems. The TALON vs. MATILDA head-to-head comparison proves that this piece of equipment is crucial in educating the customer of a vehicle's ability. Similar robot platforms can all be tested to mate the best vehicle to a specific mission. This tow sled

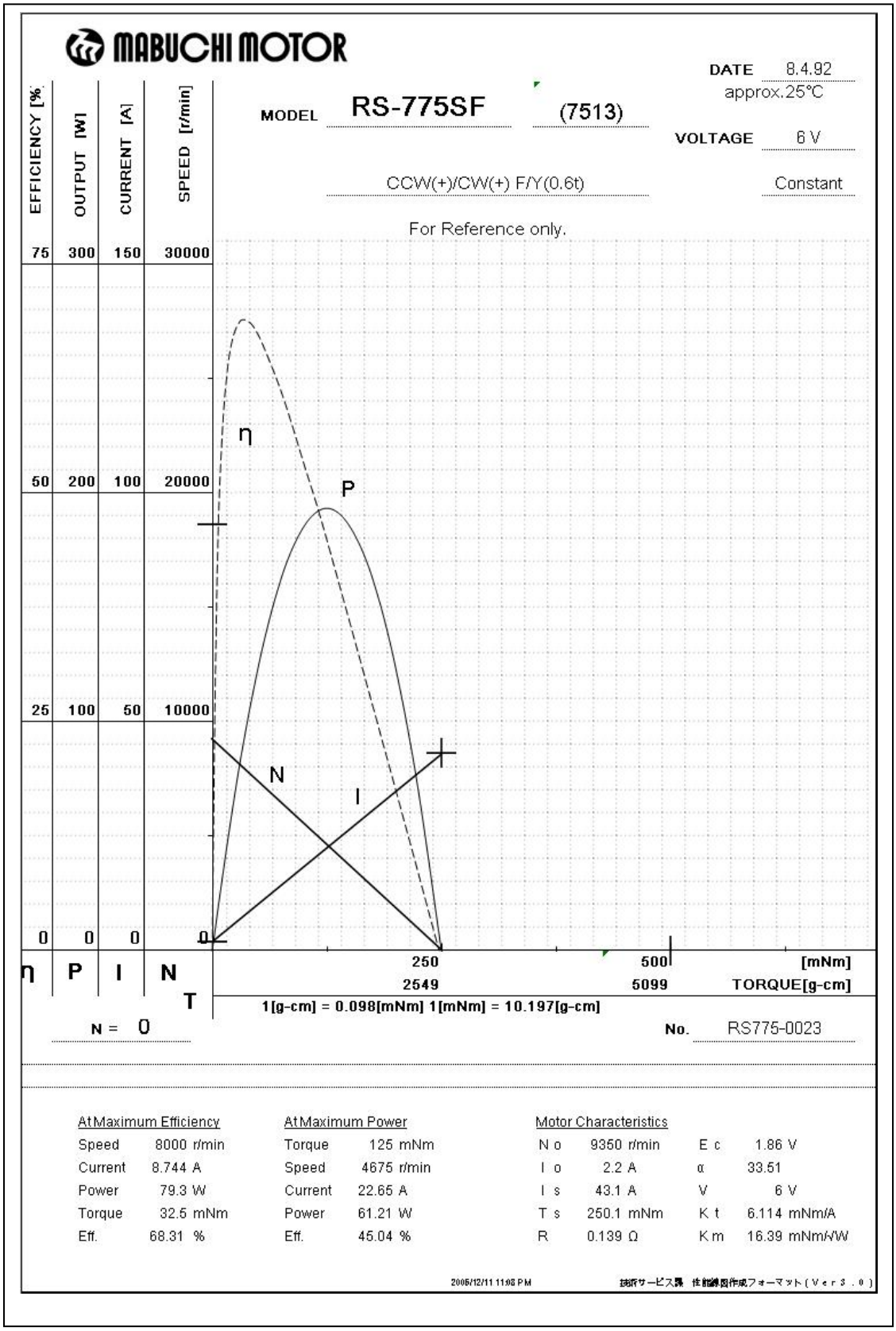
provides a more cost effective method when compared to the expensive, more complicated mobile dynamometers in use by automotive and military proving grounds. This sled is lightweight and transportable to any site. Using a consistent ground medium of asphalt, tests can be accurately reproduced. The variable weight tow sled described here is a simple but potentially valuable tool for evaluating the towing capacity of emerging unmanned vehicles.

Appendix A – TALON Test Results

Pull summaries filtered through a moving average of 25 points. Inconclusive tests are grayed out and not tallied for averages and other data.

Trial	Speed Setting	Sled payload (lbs)	Total Distance (ft)	Max Speed (mph)	Max Force (lbs)	Max Power (hp)	Notes - grayed out = inconclusive test.
Day 1	**15-Nov-04**						
1	1/2	100	54' 2"	1.11	79.41	0.18	
2	1/2	100	53' 11"	1.12	90.16	0.17	
3	1/2	100	50' 10"	1.13	80.48	0.16	Stopped prematurely
4	1/2	100	52' 5.5"	1.10	83.94	0.17	
5	Full	100	52' 9"	3.61	93.08	0.49	Control lost - suspect low battery
6	Full	100		3.49	160.10	0.44	Low Battery - incomplete run
7	Full	100	63' 2"	4.18	70.88	0.50	Fresh Battery, First complete run at full speed, drifted to left
8	Full	100	58' 8"	4.16	71.11	0.45	drifted to left off the road
9	1/2	125		1.14	79.59	0.17	
10	1/2	75	70' 3"	1.14	80.34	0.19	
Day 2	**16-Nov-04**						
11	1/2	100					overcast day, drizzle - TALON ran over wet paint and lost traction, inconclusive
12	1/2	100	60'3"	1.17	90.82	0.21	wet asphalt
13	1/2	100		1.15	87.07	0.24	grass - wet and level, couple of small rocks ran over by skid plate
14	1/2	100	46'2"	1.16	75.62	0.20	grass - wet and level - hit large rock to stop test premature
15	1/2	100		1.16	85.87	0.23	

Appendix B – MATILDA motor specs



Appendix C – MATLAB Code for TALON Pulls One, Two, and Four

```
%Created 1-4-06
%TALON Test
%Test Date:  11-15-05
%Comparison of all pulls done on dry, level asphalt
%with speed setting of half, and a sled payload of 100 pounds
%pulls one, two, three, and four were all done with half speed setting
%and 100 pounds sled payload

close all
clear all
clc

n = input('What size of moving average? (must be an odd number)\n');


%_____________________________________________________________
%PULL ONE

timeone = [0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
        1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1
        2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2
        3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
        4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
        5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5
        6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6
        7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7
        8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8
        9.9  10.0  10.1  10.2  10.3  10.4  10.5  10.6  10.7  10.8  10.9
        11.0  11.1  11.2  11.3  11.4  11.5  11.6  11.7  11.8  11.9  12.0
        12.1  12.2  12.3  12.4  12.5  12.6  12.7  12.8  12.9  13.0  13.1
        13.2  13.3  13.4  13.5  13.6  13.7  13.8  13.9  14.0  14.1  14.2
        14.3  14.4  14.5  14.6  14.7  14.8  14.9  15.0  15.1  15.2  15.3
        15.4  15.5  15.6  15.7  15.8  15.9  16.0  16.1  16.2  16.3  16.4
        16.5  16.6  16.7  16.8  16.9  17.0  17.1  17.2  17.3  17.4  17.5
        17.6  17.7  17.8  17.9  18.0  18.1  18.2  18.3  18.4  18.5  18.6
        18.7  18.8  18.9  19.0  19.1  19.2  19.3  19.4  19.5  19.6  19.7
        19.8  19.9  20.0  20.1  20.2  20.3  20.4  20.5  20.6  20.7  20.8
        20.9  21.0  21.1  21.2  21.3  21.4  21.5  21.6  21.7  21.8  21.9
        22.0  22.1  22.2  22.3  22.4  22.5  22.6  22.7  22.8  22.9  23.0
        23.1  23.2  23.3  23.4  23.5  23.6  23.7  23.8  23.9  24.0  24.1
        24.2  24.3  24.4  24.5  24.6  24.7  24.8  24.9 25.0  25.1  25.2  25.3
        25.4  25.5  25.6  25.7  25.8  25.9  26.0  26.1  26.2  26.3  26.4
        26.5  26.6  26.7  26.8  26.9  27.0  27.1  27.2  27.3  27.4  27.5
        27.6  27.7  27.8  27.9  28.0  28.1  28.2  28.3  28.4  28.5  28.6
        28.7  28.8  28.9  29.0  29.1  29.2  29.3  29.4  29.5  29.6  29.7
        29.8  29.9  30.0  30.1  30.2  30.3  30.4  30.5  30.6  30.7  30.8
        30.9  31.0  31.1  31.2  31.3  31.4  31.5  31.6  31.7  31.8  31.9
        32.0  32.1  32.2  32.3  32.4  32.5  32.6  32.7  32.8  32.9  33.0
        33.1  33.2  33.3  33.4  33.5  33.6  33.7  33.8  33.9  34.0  34.1
        34.2  34.3  34.4  34.5  34.6  34.7  34.8  34.9  35.0  35.1  35.2
        35.3  35.4  35.5  35.6  35.7  35.8  35.9  36.0  36.1  36.2  36.3
        36.4  36.5  36.6  36.7  36.8  36.9  37.0  37.1  37.2  37.3  37.4
        37.5  37.6  37.7  37.8  37.9  38.0  38.1  38.2  38.3  38.4  38.5
        38.6  38.7  38.8  38.9  39.0  39.1  39.2  39.3  39.4  39.5  39.6
        39.7  39.8  39.9  40.0  40.1  40.2  40.3  40.4  40.5  40.6  40.7
        40.8  40.9  41.0  41.1  41.2  41.3  41.4  41.5  41.6  41.7  41.8
        41.9  42.0];
speedone = [0  0  0  0  0  0  0.053636  0.176151  0.292187  0.493895  0.595721
        0.632569  0.563175  0.528453  0.424999  0.377685  0.406173  0.46005  0.536357  0.58717  0.637499  0.668211
        0.731112  0.726317  0.770311  0.792679  0.804535  0.776335  0.75819  0.755369  0.770311  0.822038  0.911807
        0.977607  1.05605  1.133054  1.168747  1.203698  1.179985  1.150126  1.137285  1.104001  1.062498  1.086264
        1.072714  1.074949  1.072714  1.104001  1.12635  1.104001  1.12635  1.162107  1.153167  1.133054  1.072714
        1.074949  1.045896  1.016843  1.019078  1.074949  1.072714  1.104001  1.12635  1.133054  1.099532  1.104001
        1.072714  1.027547  1.035935  1.045896  1.110207  1.09262  1.137285  1.133054  1.062498  1.133054  1.05605
        1.09262  1.110207  1.104001  1.115622  1.133054  1.137285  1.09262  1.110207  1.133054  1.115622  1.144981
        1.072714  1.035935  1.056905  1.072714  1.133054  1.104001  1.153167  1.12635  1.133054  1.12635  1.104001
        1.045896  1.072714  1.072714  1.074949  1.072714  1.104001  1.133054  1.072714  1.045896  1.104001  1.099532
```

```
1.104001 1.12635 1.133054 1.072714 1.086264 1.062498 1.045896 1.110207 1.035114 1.072714 1.086264
1.035935 1.045896 1.110207 1.063867 1.045896 1.115622 1.062498 1.133054 1.083129 1.09262 1.072714
1.115622 1.062498 1.045896 1.001894 1.016843 1.035935 1.144981 1.153167 1.121373 1.110207 1.074949
1.035935 1.086264 1.072714 1.121373 1.110207 1.133054 1.08906 1.027547 1.045896 1.00636 1.110207
1.104001 1.08906 1.056905 1.019078 1.00636 1.028972 1.074949 1.045896 1.074949 1.045896 1.045896
1.099532 1.074949 1.072714 1.016843 1.045896 1.045896 0.99226 1.074949 1.045896 1.074949 1.019078
1.016843 1.019078 1.016843 0.965442 1.045896 1.045896 1.104001 1.028972 1.00636 0.965442 1.027547
0.98281 1.074949 1.110207 1.150126 1.072714 1.027547 0.98281 0.958738 0.974816 0.920101 0.965442
0.96883 0.929685 0.987791 1.001894 1.00636 1.019078 0.998188 1.035935 0.958738 1.028972 0.987791
1.019078 0.987791 0.965442 0.958738 0.911807 0.87158 0.858171 0.900633 0.938625 0.987791 0.99226
1.016843 1.019078 0.987791 0.938625 0.987791 0.99226 1.016843 1.019078 1.074949 1.045896 1.086264
1.062498 1.133054 1.191441 1.150126 1.153167 1.086264 0.956248 0.929685 1.05605 1.09262 1.179985
1.086264 1.009373 0.87158 0.812346 0.805088 0.858171 0.880755 1.035935 1.045896 1.083129 1.074949
1.062498 1.115622 0.99226 0.977607 0.920659 0.929685 0.956248 1.056905 1.019078 0.920101 0.920659
0.813475 0.796873 0.763321 0.911807 0.977607 1.083129 1.133054 1.221872 1.262415 1.153167 1.00636
0.974816 0.987791 1.045896 0.987791 0.911807 0.842527 0.804535 0.813475 0.831353 0.987791 1.153167
1.191159 1.179985 1.045896 1.045896 0.900633 0.938625 0.87158 0.965442 1.074949 1.179985 1.191159
1.206803 1.104001 1.083129 0.891348 0.911807 0.910113 0.929685 0.958738 0.974816 0.891348 0.938625
0.998188 0.956248 0.87158 0.893581 0.805088 0.831353 0.851396 0.770311 0.813475 0.785268 0.833841
0.965442 1.027547 1.009373 0.987791 1.001894 1.019078 0.977607 0.939471 0.831353 0.862595 0.920659
1.045896 1.121373 1.115622 0.99226 0.833841 0.731112 0.643628 0.661323 0.763321 0.777718 0.718829
0.649877 0.61681 0.661323 0.733962 0.724082 0.7509 0.813475 0.724082 0.668211 0.726317 0.724082
0.75819 0.661323 0.7509 0.755369 0.900633 0.884989 0.812346 0.718829 0.7509 0.675245 0.690076
0.61681 0.622799 0.661323 0.509539 0.675245 0.610936 0.697264 0.731112 0.842527 0.849998 0.880755
0.858171 0.862595 0.893581 0.813475 0.903123 0.939471 1.019078 0.948854 0.974816 0.813475 0.743748
0.616528 0.429086 0.258778 0.108313 0 0 0 0 0 0 0
0 0.029053 0 0 0 0 0 0 0.026818 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0];
forceone = [11.754727 11.754727 11.437031 12.707812 17.155547 23.509453 25.73332 33.040312 40.347305 25.73332
21.603281 13.978594 11.119336 19.697109 10.483945 22.874063 26.368711 30.181055 22.874063 12.707812 27.321797
22.556367 14.613984 19.061719 18.108633 14.296289 20.967891 22.874063 26.051016 25.73332 32.087227 39.711914
35.26418 33.675703 41.618086 25.73332 22.238672 15.56707 20.967891 18.426328 22.238672 18.744023 12.072422
22.556367 31.769531 16.202461 13.343203 21.920977 24.780234 18.108633 16.520156 17.790937 17.790937 17.790937
20.967891 26.686406 20.650195 23.827148 32.087227 39.711914 12.390117 20.014805 18.108633 20.014805 24.780234
23.827148 20.3325 21.285586 26.051016 35.89957 29.863359 22.556367 26.368711 26.368711 23.509453 26.051016
31.451836 27.639492 30.181055 21.603281 35.89957 34.628789 21.285586 35.26418 31.769531 21.603281 26.686406
18.108633 24.780234 27.957187 33.358008 30.181055 23.509453 28.592578 33.993398 36.534961 21.603281 24.462539
31.769531 47.654297 35.26418 36.852656 35.89957 31.451836 33.040312 48.607383 26.051016 27.321797 41.618086
31.451836 30.816445 33.675703 38.441133 32.087227 22.874063 41.935781 38.123438 35.89957 33.358008 43.524258
28.592578 35.581875 37.170352 37.488047 35.89957 40.982695 33.358008 35.26418 25.09793 29.863359 31.451836
33.993398 31.451836 34.311094 37.805742 53.055117 47.018906 37.805742 40.347305 24.462539 47.018906 27.639492
30.181055 39.711914 44.795039 34.311094 27.957187 43.206562 37.170352 47.018906 27.957187 47.971992 35.26418
40.029609 48.289687 44.159648 51.784336 42.888867 36.534961 38.123438 47.971992 37.805742 32.722617 39.394219
43.206562 51.784336 29.227969 26.051016 55.914375 51.784336 38.758828 33.040312 46.06582 50.83125 25.415625
33.993398 38.441133 51.466641 43.524258 40.029609 45.43043 32.404922 37.170352 38.758828 41.618086 46.701211
41.618086 32.087227 38.123438 58.773633 44.795039 37.805742 45.112734 54.008203 51.784336 48.289687 62.585977
51.466641 42.253477 56.549766 43.841953 64.174453 60.044414 59.726719 51.148945 59.091328 47.018906 46.383516
55.914375 43.841953 50.195859 53.690508 67.669102 59.726719 65.127539 67.669102 55.278984 70.210664 54.008203
64.174453 64.809844 53.372813 44.795039 64.492148 63.539062 47.971992 56.23207 60.9975 51.148945 41.935781
50.83125 41.935781 42.888867 41.618086 37.805742 44.795039 61.315195 56.867461 46.06582 37.488047 38.123438
42.253477 45.748125 71.481445 88.636992 73.387617 67.669102 73.069922 59.091328 51.784336 50.513555 54.643594
59.726719 49.242773 53.055117 59.726719 57.502852 62.585977 68.304492 65.76293 51.148945 62.903672 64.492148
75.293789 58.773633 69.892969 60.9975 69.257578 59.726719 64.174453 63.539062 52.102031 46.06582 44.795039
42.888867 52.419727 57.185156 53.690508 55.278984 62.903672 81.965391 75.92918 74.976094 58.455937 54.008203
55.278984 58.773633 57.185156 48.925078 50.83125 47.018906 80.059219 66.080625 57.820547 47.336602 40.982695
35.26418 48.925078 47.971992 49.242773 68.939883 61.950586 74.658398 51.148945 77.199961 64.492148 65.445234
56.867461 77.835352 75.611484 82.918477 67.033711 75.293789 80.059219 83.236172 84.189258 62.268281 70.846055
84.506953 72.434531 70.846055 85.460039 91.178555 63.539062 69.575273 78.153047 75.92918 77.835352 66.39832
69.257578 65.76293 58.455937 75.92918 80.059219 87.683906 71.481445 94.037813 79.741523 74.340703 71.799141
94.990898 94.037813 54.961289 78.470742 79.423828 74.658398 85.142344 90.860859 80.694609 86.413125 77.835352
77.517656 68.304492 87.048516 44.477344 84.506953 90.860859 68.622187 72.752227 61.632891 58.138242 82.600781
80.376914 81.012305 103.568672 57.820547 100.391719 75.293789 61.315195 73.705313 65.445234
68.304492 91.813945 90.543164 73.705313 72.116836 81.33 81.012305 87.683906 52.419727 66.080625 64.809844
83.236172 63.856758 66.716016 89.907773 106.745625 102.933281 82.600781 45.748125 68.622187
71.16375 65.127539 52.419727 73.387617 67.351406 83.236172 78.153047 67.351406 57.185156 74.658398 76.56457
74.658398 56.549766 51.784336 81.647695 92.131641 72.434531 72.752227 81.012305 83.871562 75.611484 82.283086
93.402422 13.343203 1.906172 5.718516 6.671602 11.754727 9.848555 8.260078];
powerone = speedone.*forceone./375;  %converts speed and force to horsepower
```

```
%simple moving average of speed
speedonefilter = filter(ones(1,n)/n,1,speedone);
%delete first set of data to make correct matrix
speedonefilter(1:n-1)=[];

%simple moving average of force
forceonefilter = filter(ones(1,n)/n,1,forceone);
%delete first set of data to make correct matrix
forceonefilter(1:n-1)=[];

%simple moving average of power
poweronefilter = filter(ones(1,n)/n,1,powerone);
%delete first set of date to make correct matrix
poweronefilter(1:n-1)=[];

%reduce length of timeone to match that of speedone filtered
timeoneavg = timeone;
%delete end of matrix
i = 1;
for i = 1:(n-1)/2
   t = length(timeoneavg);
   timeoneavg(t)=[];
end
%delete beginning of matrix
timeoneavg(1:(n-1)/2)=[];

%_____________________________________________________________
%PULL TWO

timetwo = [0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
           1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1
           2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2
           3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
           4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
           5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5
           6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6
           7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7
           8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8
           9.9  10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
           11.0 11.1 11.2 11.3 11.4 11.5 11.6 11.7 11.8 11.9 12.0
           12.1 12.2 12.3 12.4 12.5 12.6 12.7 12.8 12.9 13.0 13.1
           13.2 13.3 13.4 13.5 13.6 13.7 13.8 13.9 14.0 14.1 14.2
           14.3 14.4 14.5 14.6 14.7 14.8 14.9 15.0 15.1 15.2 15.3
           15.4 15.5 15.6 15.7 15.8 15.9 16.0 16.1 16.2 16.3 16.4
           16.5 16.6 16.7 16.8 16.9 17.0 17.1 17.2 17.3 17.4 17.5
           17.6 17.7 17.8 17.9 18.0 18.1 18.2 18.3 18.4 18.5 18.6
           18.7 18.8 18.9 19.0 19.1 19.2 19.3 19.4 19.5 19.6 19.7
           19.8 19.9 20.0 20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8
           20.9 21.0 21.1 21.2 21.3 21.4 21.5 21.6 21.7 21.8 21.9
           22.0 22.1 22.2 22.3 22.4 22.5 22.6 22.7 22.8 22.9 23.0
           23.1 23.2 23.3 23.4 23.5 23.6 23.7 23.8 23.9 24.0 24.1
           24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9 25.0 25.1 25.2 25.3
           25.4 25.5 25.6 25.7 25.8 25.9 26.0 26.1 26.2 26.3 26.4
           26.5 26.6 26.7 26.8 26.9 27.0 27.1 27.2 27.3 27.4 27.5
           27.6 27.7 27.8 27.9 28.0 28.1 28.2 28.3 28.4 28.5 28.6
           28.7 28.8 28.9 29.0 29.1 29.2 29.3 29.4 29.5 29.6 29.7
           29.8 29.9 30.0 30.1 30.2 30.3 30.4 30.5 30.6 30.7 30.8
           30.9 31.0 31.1 31.2 31.3 31.4 31.5 31.6 31.7 31.8 31.9
           32.0 32.1 32.2 32.3 32.4 32.5 32.6 32.7 32.8 32.9 33.0
           33.1 33.2 33.3 33.4 33.5 33.6 33.7 33.8 33.9 34.0 34.1
           34.2 34.3 34.4 34.5 34.6 34.7 34.8 34.9 35.0 35.1 35.2
           35.3 35.4 35.5 35.6 35.7 35.8 35.9 36.0 36.1 36.2 36.3
           36.4 36.5 36.6 36.7 36.8 36.9 37.0 37.1 37.2 37.3 37.4
           37.5 37.6 37.7 37.8 37.9 38.0 38.1 38.2 38.3 38.4 38.5
           38.6 38.7 38.8 38.9 39.0 39.1 39.2 39.3 39.4 39.5 39.6
           39.7 39.8 39.9 40.0 40.1 40.2 40.3 40.4 40.5 40.6 40.7
           40.8 40.9 41.0 41.1 41.2 41.3 41.4 41.5 41.6 41.7 41.8
           41.9 42.0 42.1 42.2 42.3 42.4 42.5 42.6 42.7];
speedtwo = [0  0  0  0  0  0  0  0  0  0  0
           0  0  0  0  0  0  0  0  0.029053  0.287532  0.46033
```

0.639159 0.747582 0.75819 0.777718 0.690076 0.645887 0.563175 0.54631 0.649877 0.7509 0.929685
1.019078 1.133054 1.153167 1.162107 1.206803 1.203698 1.121373 1.099532 1.110207 1.063867 1.072714
1.115622 1.115622 1.133054 1.137285 1.121373 1.12635 1.174339 1.09262 1.099532 1.110207 1.063867
1.072714 1.086264 1.09262 1.12635 1.164363 1.121373 1.099532 1.115622 1.099532 1.035114 1.001894
1.016843 1.035935 1.086264 1.12635 1.121373 1.110207 1.104001 1.115622 1.086264 1.045896 0.977607
1.05605 1.072714 1.09262 1.144981 1.179985 1.150126 1.21852 1.162107 1.099532 1.016843 0.99226
1.016843 1.045896 1.162107 1.153167 1.191159 1.099532 1.162107 1.179985 1.191159 1.099532 1.045896
0.99226 0.987791 1.045896 1.104001 1.12635 1.162107 1.179985 1.191159 1.153167 1.074949 1.072714
1.027547 1.035935 1.045896 1.05605 1.035114 1.12635 1.144981 1.062498 1.104001 1.083129 1.035114
1.045896 1.056905 1.035935 1.104001 1.110207 1.035114 1.045896 1.056905 1.035935 1.045896 1.137285
1.074949 1.115622 1.115622 1.099532 1.09262 1.083129 1.074949 1.062498 1.056905 1.12635 1.150126
1.245598 1.133054 1.009373 1.027547 1.019078 1.035114 1.137285 1.133054 1.08906 1.115622 1.045896
1.045896 1.045896 1.045896 1.045896 0.965442 1.016843 1.016843 1.045896 0.987791 1.019078 1.016843
1.099532 1.133054 1.153167 1.104001 1.072714 1.074949 1.072714 0.958738 0.99226 1.016843 1.072714
1.191159 1.179985 1.121373 1.011716 0.977607 0.920659 0.929685 0.929685 0.96883 0.965442 0.977607
1.001894 1.045896 1.062498 1.027547 0.965442 0.920101 1.028972 1.133054 1.221872 1.203698 1.12635
0.948854 0.920659 0.842527 0.884989 0.900633 0.99226 1.104001 1.072714 1.104001 1.099532 1.133054
1.099532 1.016843 0.965442 0.900633 0.938625 0.958738 0.965442 0.987791 0.938625 1.016843 0.965442
0.958738 0.965442 0.929685 0.911807 0.900633 0.831353 0.880755 0.823436 0.87158 0.893581 0.920101
0.911807 1.027547 1.08906 1.133054 1.083129 0.977607 1.045896 1.115622 1.168747 1.191159 1.083129
0.948854 0.884989 0.851396 0.929685 1.104001 1.21852 1.179985 1.063867 0.939471 0.831353 0.747582
0.731112 0.7509 0.747582 0.96883 1.019078 1.063867 1.164363 1.153167 1.121373 0.998188 0.911807
0.862595 0.893581 0.99226 0.948854 1.115622 0.884989 0.849998 0.822038 0.831353 0.842527 0.929685
1.072714 1.12635 1.162107 1.099532 1.016843 0.842527 0.7509 0.7509 0.668211 0.831353 0.842527
0.87158 0.911807 0.884989 0.900633 0.831353 0.87158 0.965442 1.074949 1.099532 1.056905 1.035935
0.958738 0.920659 0.833841 0.831353 0.792679 0.823436 0.929685 1.028972 1.063867 1.099532 1.056905
1.009373 0.958738 0.920659 0.833841 0.785268 0.726317 0.610936 0.616528 0.589993 0.668211 0.664061
0.704604 0.804535 0.929685 1.019078 1.074949 1.072714 0.987791 0.911807 0.87158 0.938625 0.842527
1.019078 1.133054 1.153167 0.987791 0.911807 0.755369 0.670446 0.639159 0.724082 0.813475 0.858171
0.87158 0.920659 0.862595 0.884989 0.851396 0.743748 0.639159 0.622799 0.603816 0.7509 0.880755
0.849998 0.900633 0.812346 0.776335 0.670446 0.616528 0.610936 0.639159 0.676955 0.805088 0.938625
0.998188 0.99226 1.00636 1.001894 0.929685 0.796873 0.733962 0.697264 0.661323 0.75819 0.755369
0.796873 0.792679 0.804535 0.776335 0.785268 0.755369 0.823436 0.822038 0.884989 0.833841 0.75819
0.755369 0.717186 0.704604 0.697264 0.668211 0.724082 0.697264 0.670446 0.552001 0.37545 0.319579
0.348632 0.406737 0.482721 0.493895 0.509539 0.406737 0.321814 0.203369 0.053636 0.058105 0
0 0 0 0.026818 0 0 0 0 0 0.026818 0.058717
0.053125 0.029053 0.054156 0 0 0 0 0 0 0];

forcetwo = [6.989297 7.624688 7.942383 7.942383 7.624688 7.624688 7.306992 7.306992 7.306992 6.989297 6.989297
6.989297 6.671602 6.671602 6.671602 6.671602 7.306992 8.895469 16.837852 68.304492 47.654297 22.238672
32.087227 19.697109 13.343203 22.874063 7.942383 15.249375 25.73332 24.462539 32.722617 31.451836 42.888867
33.040312 20.650195 16.520156 31.769531 19.379414 15.884766 18.744023 24.462539 20.650195 17.473242 24.144844
26.686406 20.3325 20.3325 14.296289 15.249375 24.462539 28.274883 23.509453 26.051016 22.556367 21.603281
18.744023 27.004102 25.415625 23.509453 13.343203 28.910273 22.238672 16.202461 24.462539 33.358008 25.09793
17.790937 25.09793 34.946484 34.946484 16.520156 23.509453 24.780234 15.249375 29.545664 39.076523 35.581875
26.051016 33.358008 44.159648 29.545664 26.051016 27.004102 27.004102 9.848555 33.675703 37.805742 22.874063
27.004102 35.26418 40.347305 27.957187 36.217266 22.874063 24.144844 30.181055 21.285586 27.321797 34.628789
35.581875 34.946484 40.665 32.087227 45.748125 30.49875 26.686406 37.805742 30.49875 27.004102 42.888867
47.654297 38.441133 31.134141 41.935781 33.675703 33.993398 39.711914 43.841953 36.852656 36.534961 43.841953
34.628789 43.841953 45.43043 48.607383 45.748125 42.571172 49.242773 31.134141 44.159648 49.560469 52.737422
37.488047 43.524258 43.206562 40.347305 44.159648 40.982695 41.618086 37.488047 38.123438 38.758828 28.910273
39.076523 31.134141 40.982695 38.441133 50.513555 47.018906 39.711914 46.383516 47.971992 43.524258 58.773633
50.83125 47.018906 35.26418 49.878164 52.737422 35.89957 32.722617 54.643594 55.59668 43.524258 55.59668
53.690508 43.206562 37.488047 58.138242 47.971992 32.722617 44.477344 58.455937 44.795039 53.055117 38.123438
38.441133 35.89957 46.383516 42.571172 43.206562 44.795039 56.867461 60.362109 56.867461 53.372813 49.560469
55.59668 60.9975 53.690508 63.539062 60.9975 51.148945 49.878164 39.711914 33.040312 38.441133 38.123438
43.206562 55.59668 62.903672 75.293789 60.044414 49.878164 45.43043 44.159648 23.191758 29.863359 48.289687
42.888867 54.008203 73.069922 71.799141 62.903672 53.372813 72.116836 59.726719 45.112734 78.788438 64.809844
63.539062 64.492148 74.976094 65.127539 60.679805 56.23207 65.445234 68.939883 64.809844 63.539062 79.423828
55.914375 50.513555 44.795039 55.278984 60.9975 54.643594 35.26418 41.300391 31.451836 34.311094 49.242773
64.809844 71.16375 63.221367 52.737422 44.159648 26.051016 49.242773 52.737422 59.409023 55.914375 71.799141
77.835352 94.673203 59.726719 69.257578 64.174453 54.008203 54.643594 55.914375 67.351406 60.044414 65.445234
67.351406 68.304492 73.705313 44.795039 69.257578 51.784336 76.882266 85.142344 78.788438 65.445234 43.841953
44.795039 33.358008 38.123438 49.242773 61.632891 65.127539 64.174453 67.669102 72.752227 53.055117 72.116836
70.210664 77.199961 75.293789 81.33 67.033711 66.080625 58.138242 55.914375 57.502852 47.336602 69.257578
73.387617 67.033711 74.023008 58.138242 77.199961 64.809844 57.502852 68.939883 50.83125 43.524258 50.83125
67.351406 65.127539 55.278984 53.372813 67.033711 71.16375 65.76293 79.106133 74.340703 89.590078 95.626289
90.860859 95.308594 60.679805 63.856758 72.116836 73.387617 64.809844 81.965391 51.784336 73.387617 61.950586
53.372813 39.076523 54.008203 70.210664 66.716016 71.481445 84.189258 84.189258 76.882266 90.543164 79.423828
88.954687 67.033711 60.044414 72.116836 75.92918 83.871562 87.048516 83.871562 72.116836 65.127539 71.16375
77.517656 83.553867 91.178555 86.73082 86.09543 82.918477 79.106133 97.532461 62.585977 74.658398 72.116836

```
        70.846055 81.012305 63.221367 60.362109 89.272383 85.777734 74.976094 81.33     71.16375  84.189258 64.809844
        92.131641 90.860859 70.846055 60.679805 105.474844           71.799141 80.694609 63.539062 99.756328 72.752227
        70.846055 87.683906 99.756328 87.683906 90.543164 95.943984 81.965391 81.647695 92.131641 93.720117
        104.204062          86.413125 87.048516 78.153047 81.965391 86.413125 109.287188          99.120937
        105.157148          107.698711          92.131641 86.09543  74.658398 78.470742 77.199961 88.954687 72.116836
        61.632891 75.611484 94.037813 63.856758 74.976094 84.824648 76.882266 62.268281 76.56457  73.387617 63.856758
        63.221367 74.976094 75.293789 74.658398 67.986797 5.083125];
powertwo = speedtwo.*forcetwo./375;  %converts speed and force to horsepower

%simple moving average of speed
speedtwofilter = filter(ones(1,n)/n,1,speedtwo);
%delete first set of data to make correct matrix
speedtwofilter(1:n-1)=[];

%simple moving average of force
forcetwofilter = filter(ones(1,n)/n,1,forcetwo);
%delete first set of data to make correct matrix
forcetwofilter(1:n-1)=[];

%simple moving average of power
powertwofilter = filter(ones(1,n)/n,1,powertwo);
%delete first set of date to make correct matrix
powertwofilter(1:n-1)=[];

%reduce length of timetwo to match that of speedtwo filtered
timetwoavg = timetwo;
%delete end of matrix
i=1;
for i = 1:(n-1)/2
   t = length(timetwoavg);
   timetwoavg(t)=[];
end
%delete beginning of matrix
timetwoavg(1:(n-1)/2)=[];

%_____________________________________________________________
%PULL FOUR
timefour = [0.0     0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9     1.0
        1.1     1.2     1.3     1.4     1.5     1.6     1.7     1.8     1.9     2.0     2.1
        2.2     2.3     2.4     2.5     2.6     2.7     2.8     2.9     3.0     3.1     3.2
        3.3     3.4     3.5     3.6     3.7     3.8     3.9     4.0     4.1     4.2     4.3
        4.4     4.5     4.6     4.7     4.8     4.9     5.0     5.1     5.2     5.3     5.4
        5.5     5.6     5.7     5.8     5.9     6.0     6.1     6.2     6.3     6.4     6.5
        6.6     6.7     6.8     6.9     7.0     7.1     7.2     7.3     7.4     7.5     7.6
        7.7     7.8     7.9     8.0     8.1     8.2     8.3     8.4     8.5     8.6     8.7
        8.8     8.9     9.0     9.1     9.2     9.3     9.4     9.5     9.6     9.7     9.8
        9.9     10.0    10.1    10.2    10.3    10.4    10.5    10.6    10.7    10.8    10.9
        11.0    11.1    11.2    11.3    11.4    11.5    11.6    11.7    11.8    11.9    12.0
        12.1    12.2    12.3    12.4    12.5    12.6    12.7    12.8    12.9    13.0    13.1
        13.2    13.3    13.4    13.5    13.6    13.7    13.8    13.9    14.0    14.1    14.2
        14.3    14.4    14.5    14.6    14.7    14.8    14.9    15.0    15.1    15.2    15.3
        15.4    15.5    15.6    15.7    15.8    15.9    16.0    16.1    16.2    16.3    16.4
        16.5    16.6    16.7    16.8    16.9    17.0    17.1    17.2    17.3    17.4    17.5
        17.6    17.7    17.8    17.9    18.0    18.1    18.2    18.3    18.4    18.5    18.6
        18.7    18.8    18.9    19.0    19.1    19.2    19.3    19.4    19.5    19.6    19.7
        19.8    19.9    20.0    20.1    20.2    20.3    20.4    20.5    20.6    20.7    20.8
        20.9    21.0    21.1    21.2    21.3    21.4    21.5    21.6    21.7    21.8    21.9
        22.0    22.1    22.2    22.3    22.4    22.5    22.6    22.7    22.8    22.9    23.0
        23.1    23.2    23.3    23.4    23.5    23.6    23.7    23.8    23.9    24.0    24.1
        24.2    24.3    24.4    24.5    24.6    24.7    24.8    24.9 25.0   25.1    25.2    25.3
        25.4    25.5    25.6    25.7    25.8    25.9    26.0    26.1    26.2    26.3    26.4
        26.5    26.6    26.7    26.8    26.9    27.0    27.1    27.2    27.3    27.4    27.5
        27.6    27.7    27.8    27.9    28.0    28.1    28.2    28.3    28.4    28.5    28.6
        28.7    28.8    28.9    29.0    29.1    29.2    29.3    29.4    29.5    29.6    29.7
        29.8    29.9    30.0    30.1    30.2    30.3    30.4    30.5    30.6    30.7    30.8
        30.9    31.0    31.1    31.2    31.3    31.4    31.5    31.6    31.7    31.8    31.9
        32.0    32.1    32.2    32.3    32.4    32.5    32.6    32.7    32.8    32.9    33.0
        33.1    33.2    33.3    33.4    33.5    33.6    33.7    33.8    33.9    34.0    34.1
        34.2    34.3    34.4    34.5    34.6    34.7    34.8    34.9    35.0    35.1    35.2
        35.3    35.4    35.5    35.6    35.7    35.8    35.9    36.0    36.1    36.2    36.3
```

```
        36.4    36.5    36.6    36.7    36.8    36.9    37.0    37.1    37.2    37.3    37.4
        37.5    37.6    37.7    37.8    37.9    38.0    38.1    38.2    38.3    38.4    38.5
        38.6    38.7    38.8    38.9    39.0    39.1    39.2    39.3    39.4    39.5    39.6
        39.7    39.8    39.9    40.0    40.1    40.2    40.3    40.4    40.5    40.6    40.7
        40.8    40.9    41.0    41.1    41.2    41.3    41.4    41.5    41.6    41.7    41.8
        41.9    42.0    42.1    42.2    42.3    42.4    42.5    42.6    42.7    42.8    42.9];
speedfour = [0  0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0.028753 0.241361 0.557811
        0.743748 0.900633 1.083129 1.09262 1.072714 1.086264 1.062498 1.074949 1.028972 0.987791 1.062498
        1.115622 1.12635 1.121373 1.137285 1.133054 1.099532 1.045896 1.072714 1.104001 1.099532 1.074949
        1.099532 1.133054 1.179985 1.191159 1.153167 1.133054 1.12635 1.104001 1.099532 1.086264 1.009373
        1.016843 1.001894 1.00636 1.019078 1.056905 1.062498 1.104001 1.137285 1.121373 1.12635 1.144981
        1.142185 1.074949 1.083129 1.035114 1.045896 1.027547 1.035935 1.074949 1.110207 1.133054 1.168747
        1.174339 1.099532 1.09262 1.083129 1.074949 1.009373 1.056905 0.99226 1.00636 1.05605 1.074949
        1.142185 1.144981 1.12635 1.09262 1.110207 1.074949 1.062498 1.056905 1.045896 1.104001 1.099532
        1.133054 1.099532 1.104001 1.072714 1.045896 1.072714 1.045896 1.045896 1.074949 1.072714 1.104001
        1.072714 1.104001 1.099532 1.104001 1.072714 1.016843 1.019078 1.045896 1.072714 1.104001 1.099532
        1.104001 1.05605 1.063867 1.072714 1.086264 1.062498 1.074949 1.083129 1.063867 1.072714 1.086264
        1.062498 1.104001 1.110207 1.150126 1.153167 1.115622 1.062498 1.016843 1.028972 1.035114 1.072714
        1.056905 1.08906 1.074949 1.05605 0.987791 1.009373 1.086264 1.072714 1.035114 1.110207 1.016843
        1.035935 1.056905 1.072714 1.09262 1.110207 1.074949 1.08906 1.115622 1.072714 1.121373 1.083129
        1.133054 1.08906 1.086264 1.045896 0.987791 0.965442 1.016843 1.019078 1.045896 1.045896 1.045896
        1.019078 1.045896 1.072714 1.045896 1.045896 1.045896 0.99226 0.958738 1.001894 1.00636 1.072714
        1.056905 1.009373 0.958738 0.974816 1.009373 1.074949 1.056905 1.045896 0.977607 1.001894 0.99226
        0.920101 0.880755 0.884989 0.891348 0.974816 0.965442 1.00636 1.027547 1.072714 1.063867 1.12635
        1.110207 1.063867 1.027547 0.99226 0.98281 0.998188 0.99226 0.987791 1.074949 1.045896 1.045896
        0.987791 0.938625 0.958738 1.016843 1.12635 1.179985 1.162107 1.153167 1.074949 0.987791 0.884989
        0.831353 0.87158 0.938625 0.958738 1.019078 1.074949 1.072714 1.086264 1.009373 0.987791 0.920659
        0.891348 0.965442 1.115622 1.168747 1.133054 1.110207 0.977607 0.99226 0.96883 0.929685 0.958738
        0.974816 0.977607 1.019078 1.056905 1.009373 0.987791 1.001894 1.016843 0.98281 1.027547 1.045896
        1.035114 1.083129 1.045896 1.035935 1.056905 1.019078 0.977607 1.001894 1.045896 1.035935 1.056905
        1.019078 1.00636 1.05605 1.045896 1.009373 0.998188 0.938625 0.920101 0.920659 0.929685 0.884989
        0.958738 0.965442 0.87158 0.938625 0.929685 0.965442 1.016843 1.12635 1.162107 1.099532 1.045896
        1.072714 1.045896 1.045896 0.958738 0.911807 0.842527 0.804535 0.842527 0.858171 0.784422 0.812346
        0.776335 0.804535 0.910113 1.009373 1.133054 1.191441 1.121373 1.028972 0.929685 0.849998 0.784422
        0.75819 0.632569 0.61681 0.557811 0.531249 0.440377 0.509539 0.517557 0.649877 0.639159 0.796873
        0.822038 0.858171 0.958738 0.99226 1.045896 1.045896 1.016843 0.965442 0.900633 0.938625 0.900633
        0.938625 0.929685 0.965442 0.929685 0.884989 0.842527 0.7509 0.777718 0.726317 0.668211 0.589993
        0.552001 0.61681 0.645887 0.690623 0.726317 0.75819 0.747582 0.7509 0.733962 0.823436 0.842527
        0.812346 0.862595 0.884989 0.792679 0.87656 0.813475 0.920659 0.862595 0.7509 0.675245 0.637499
        0.522948 0.731112 0.639159 0.637499 0.58717 0.482721 0.431297 0.216626 0.203369 0.132812 0.176151
        0.080454 0.115013 0.027078 0.058105 0.026562 0.088075 0.026818 0.172519 0.162469 0.116211 0.026562
        0       0       0       0       0       0       0       0       0       0       0
        0       0       0       0       0       0       0       0       0       0       0
        0];
forcefour = [5.40082 5.083125 5.40082 5.718516 5.40082 5.718516 5.40082 5.718516 6.353906 6.353906 6.036211
        6.036211 6.036211 6.036211 6.036211 6.036211 6.036211 5.40082 5.40082 5.083125 5.40082 5.718516
        5.718516 6.036211 5.718516 5.718516 5.718516 6.036211 7.306992 10.483945 26.051016 15.56707 10.16625
        24.144844 32.404922 22.556367 15.56707 19.379414 16.202461 11.754727 15.56707 28.592578 26.686406 25.415625
        13.343203 21.920977 19.379414 20.650195 25.415625 26.368711 26.686406 30.816445 20.967891 24.144844 30.49875
        35.26418 28.274883 27.957187 20.014805 19.697109 27.004102 19.697109 21.920977 20.3325 25.73332 13.978594
        24.780234 28.910273 32.722617 32.087227 24.144844 24.780234 24.462539 27.639492 24.780234 28.274883 24.462539
        26.051016 34.946484 31.134141 26.686406 34.628789 35.581875 21.285586 30.49875 41.935781 29.227969 23.191758
        33.358008 28.592578 25.09793 25.415625 35.581875 31.134141 26.368711 28.274883 47.971992 48.289687 21.285586
        35.26418 51.466641 27.957187 27.957187 48.925078 38.123438 20.014805 37.170352 41.618086 33.358008 24.144844
        32.087227 32.404922 26.368711 33.993398 44.159648 32.722617 27.957187 40.029609 41.618086 41.618086 29.863359
        45.112734 50.513555 27.004102 38.441133 53.690508 46.701211 30.181055 54.008203 52.737422 33.675703 33.675703
        49.878164 47.654297 36.534961 50.513555 51.466641 29.227969 33.040312 43.841953 55.59668 39.394219 34.946484
        47.336602 47.336602 27.639492 37.170352 36.217266 38.758828 28.274883 43.524258 41.618086 38.123438 33.993398
        47.018906 53.372813 40.347305 49.878164 55.59668 37.170352 36.852656 46.383516 44.795039 47.336602 37.805742
        53.690508 32.722617 42.571172 47.654297 49.878164 24.780234 56.867461 52.102031 46.383516 38.758828 33.040312
        42.888867 47.654297 59.091328 62.903672 44.795039 54.008203 52.737422 57.820547 55.59668 57.820547 60.679805
        40.029609 49.242773 51.784336 59.726719 39.076523 54.008203 60.044414 48.925078 41.300391 38.123438 54.008203
        56.23207 47.971992 58.138242 50.513555 57.502852 58.138242 46.701211 39.394219 44.477344 56.867461 39.394219
        42.888867 58.138242 63.539062 66.080625 44.159648 60.679805 59.726719 58.455937 45.112734 47.336602 47.971992
        31.769531 28.910273 52.419727 61.632891 54.961289 56.549766 53.055117 46.701211 59.726719 60.044414 54.325898
        44.795039 58.455937 59.091328 38.758828 47.336602 47.018906 42.253477 49.878164 46.383516 49.242773 59.091328
        69.257578 61.315195 58.138242 67.033711 51.148945 57.185156 45.43043 49.242773 68.304492 57.502852 72.116836
        62.268281 45.112734 43.524258 29.863359 37.488047 48.925078 41.618086 56.549766 57.185156 47.654297 44.477344
```

```
        60.679805 67.033711 48.925078 55.278984 63.856758 68.622187 51.148945 46.383516 62.585977 59.726719 53.055117
        69.257578 66.080625 60.9975   54.008203 60.9975   68.939883 67.351406 62.268281 49.560469 55.59668  62.903672
        56.23207  63.221367 62.903672 69.892969 72.752227 68.622187 83.553867 64.174453 54.961289 72.434531 61.950586
        69.257578 67.669102 55.278984 68.304492 59.409023 73.387617 79.741523 56.549766 48.925078 61.315195 33.675703
        38.123438 35.26418  44.159648 53.690508 67.351406 74.023008 92.131641 75.293789 80.694609 77.517656 74.340703
        83.553867 82.283086 62.903672 61.632891 52.419727 44.477344 56.867461 53.372813 68.939883 73.387617 81.012305
        72.434531 79.106133 78.788438 80.376914 114.370312           73.705313 96.89707  85.142344 63.539062 53.055117
        77.517656 75.293789 85.460039 65.445234 77.199961 60.362109 77.199961 73.069922 67.033711 49.560469 84.824648
        60.362109 73.705313 53.372813 71.16375  76.56457  82.918477 91.49625  77.199961 72.434531 76.56457  73.705313
        96.579375 88.001602 83.871562 72.116836 73.387617 67.351406 75.92918  77.517656 86.09543  81.965391 74.340703
        82.600781 68.939883 57.820547 63.539062 57.820547 76.882266 59.091328 72.752227 65.76293  90.860859
        105.792539           71.481445 90.860859 55.278984 98.167852 82.918477 89.272383 90.543164 93.084727 90.860859
        75.611484 69.575273 67.033711 84.506953 81.33       84.506953 97.850156 84.506953 96.26168  103.250977
        97.214766 43.206562 0          11.437031 2.223867  4.447734  8.895469  11.754727 9.848555  8.577773  12.072422
        9.848555  9.530859  10.801641 9.848555  9.530859  9.530859  9.530859  10.483945 9.530859  8.895469  10.16625
        9.848555  8.895469  9.530859  9.848555  9.213164];
powerfour = speedfour.*forcefour./375;  %converts speed and force to horsepower

%simple moving average of speed
speedfourfilter = filter(ones(1,n)/n,1,speedfour);
%delete first set of data to make correct matrix
speedfourfilter(1:n-1)=[];

%simple moving average of force
forcefourfilter = filter(ones(1,n)/n,1,forcefour);
%delete first set of data to make correct matrix
forcefourfilter(1:n-1)=[];

%simple moving average of power
powerfourfilter = filter(ones(1,n)/n,1,powerfour);
%delete first set of date to make correct matrix
powerfourfilter(1:n-1)=[];

%reduce length of timefour to match that of speedfour filtered
timefouravg = timefour;
%delete end of matrix
for i = 1:(n-1)/2
   t = length(timefouravg);
   timefouravg(t)=[];
end
%delete beginning of matrix
timefouravg(1:(n-1)/2)=[];


%_____________________________________________________________________________
%FIGURES

figure(1)
h = plot(timeoneavg,speedonefilter,timetwoavg,speedtwofilter,timefouravg,speedfourfilter);
legend('One','Two','Four')
title(['TALON Speed vs. Time on Dry Asphalt, Moving Average of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Speed, (mph)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)

figure(2)
h = plot(timeoneavg,forceonefilter,timetwoavg,forcetwofilter,timefouravg,forcefourfilter);
legend('One','Two','Four')
title(['TALON Force vs. Time on Dry Asphalt, Moving Average of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Force, (lb_f)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
```

```
figure(3)
h = plot(timeoneavg,poweronefilter,timetwoavg,powertwofilter,timefouravg,powerfourfilter);
legend('One','Two','Four')
title(['TALON Power vs. Time on Dry Asphalt, Moving Average of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Power, (hp)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
ylim([0 0.25])

figure(4)
h = plot(forceonefilter,speedonefilter,forcetwofilter,speedtwofilter,forcefourfilter,speedfourfilter);
legend('One','Two','Four')
title(['TALON Speed vs. Force on Dry Asphalt, Moving Average of ',num2str(n)])
xlabel('Force, (lb_f)')
ylabel('Speed, (mph)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
```

Appendix D – MATLAB Code for TALON Pulls Twelve, Thirteen, and Fifteen

```
%Created 1-4-06
%TALON Test
%Test Date:  11-16-05
%Comparison of all pulls done on wet, level grass
%with speed setting of half, and a sled payload of 100 pounds
%pulls twelve, thirteen, and fifteen were all done with half speed setting
%and 100 pounds sled payload

close all
clear all
clc

n = input('What size of moving average? (must be an odd number)\n');

%_____________________________________________________________________
%PULL TWELVE
timetwelve = [0.0   0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9   1.0
        1.1   1.2   1.3   1.4   1.5   1.6   1.7   1.8   1.9   2.0   2.1
        2.2   2.3   2.4   2.5   2.6   2.7   2.8   2.9   3.0   3.1   3.2
        3.3   3.4   3.5   3.6   3.7   3.8   3.9   4.0   4.1   4.2   4.3
        4.4   4.5   4.6   4.7   4.8   4.9   5.0   5.1   5.2   5.3   5.4
        5.5   5.6   5.7   5.8   5.9   6.0   6.1   6.2   6.3   6.4   6.5
        6.6   6.7   6.8   6.9   7.0   7.1   7.2   7.3   7.4   7.5   7.6
        7.7   7.8   7.9   8.0   8.1   8.2   8.3   8.4   8.5   8.6   8.7
        8.8   8.9   9.0   9.1   9.2   9.3   9.4   9.5   9.6   9.7   9.8
        9.9   10.0  10.1  10.2  10.3  10.4  10.5  10.6  10.7  10.8  10.9
        11.0  11.1  11.2  11.3  11.4  11.5  11.6  11.7  11.8  11.9  12.0
        12.1  12.2  12.3  12.4  12.5  12.6  12.7  12.8  12.9  13.0  13.1
        13.2  13.3  13.4  13.5  13.6  13.7  13.8  13.9  14.0  14.1  14.2
        14.3  14.4  14.5  14.6  14.7  14.8  14.9  15.0  15.1  15.2  15.3
        15.4  15.5  15.6  15.7  15.8  15.9  16.0  16.1  16.2  16.3  16.4
        16.5  16.6  16.7  16.8  16.9  17.0  17.1  17.2  17.3  17.4  17.5
        17.6  17.7  17.8  17.9  18.0  18.1  18.2  18.3  18.4  18.5  18.6
        18.7  18.8  18.9  19.0  19.1  19.2  19.3  19.4  19.5  19.6  19.7
        19.8  19.9  20.0  20.1  20.2  20.3  20.4  20.5  20.6  20.7  20.8
        20.9  21.0  21.1  21.2  21.3  21.4  21.5  21.6  21.7  21.8  21.9
        22.0  22.1  22.2  22.3  22.4  22.5  22.6  22.7  22.8  22.9  23.0
        23.1  23.2  23.3  23.4  23.5  23.6  23.7  23.8  23.9  24.0  24.1
        24.2  24.3  24.4  24.5  24.6  24.7  24.8  24.9 25.0  25.1  25.2  25.3
        25.4  25.5  25.6  25.7  25.8  25.9  26.0  26.1  26.2  26.3  26.4
        26.5  26.6  26.7  26.8  26.9  27.0  27.1  27.2  27.3  27.4  27.5
        27.6  27.7  27.8  27.9  28.0  28.1  28.2  28.3  28.4  28.5  28.6
        28.7  28.8  28.9  29.0  29.1  29.2  29.3  29.4  29.5  29.6  29.7
        29.8  29.9  30.0  30.1  30.2  30.3  30.4  30.5  30.6  30.7  30.8
        30.9  31.0  31.1  31.2  31.3  31.4  31.5  31.6  31.7  31.8  31.9
        32.0  32.1  32.2  32.3  32.4  32.5  32.6  32.7  32.8  32.9  33.0
        33.1  33.2  33.3  33.4  33.5  33.6  33.7  33.8  33.9  34.0  34.1
        34.2  34.3  34.4  34.5  34.6  34.7  34.8  34.9  35.0  35.1  35.2
        35.3  35.4  35.5  35.6  35.7  35.8  35.9  36.0  36.1  36.2  36.3
        36.4  36.5  36.6  36.7  36.8  36.9  37.0  37.1  37.2  37.3  37.4
        37.5  37.6  37.7  37.8  37.9  38.0  38.1  38.2  38.3  38.4  38.5
        38.6  38.7  38.8  38.9  39.0  39.1  39.2  39.3  39.4  39.5  39.6
        39.7  39.8  39.9  40.0  40.1  40.2  40.3  40.4  40.5  40.6  40.7
        40.8  40.9  41.0  41.1  41.2  41.3  41.4  41.5  41.6  41.7  41.8
        41.9  42.0  42.1  42.2  42.3  42.4  42.5  42.6  42.7  42.8  42.9
        43.0  43.1  43.2  43.3  43.4  43.5  43.6  43.7  43.8  43.9  44.0
        44.1  44.2  44.3  44.4  44.5  44.6  44.7  44.8  44.9  45.0  45.1
        45.2  45.3  45.4  45.5  45.6  45.7  45.8  45.9  46.0  46.1  46.2
        46.3  46.4  46.5];
speedtwelve = [0   0   0   0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0   0.029358  0.239062  0.643628
        0.645887  1.00636  1.233621  1.35049  1.301559  1.249265  1.21852  1.121373  1.099532  1.115622  1.115622
        1.133054  1.164363  1.121373  1.153167  1.144981  1.142185  1.133054  1.164363  1.162107  1.142185  1.174339
        1.153167  1.150126  1.164363  1.104001  1.142185  1.203698  1.233621  1.178879  1.164363  1.104001  1.08906
        1.086264  1.099532  1.09262  1.110207  1.074949  1.062498  1.086264  1.099532  1.063867  1.028972  1.104001
        1.153167  1.365475  1.340892  1.249265  1.287257  0.842527  0.839425  0.948854  0.911807  0.958738  1.072714
```

```
1.162107 1.179985 1.220212 1.179985 1.162107 1.045896 0.987791 0.947737 0.977607 0.965442 1.027547
1.009373 1.074949 1.137285 1.162107 1.142185 1.074949 1.083129 1.063867 1.028972 1.016843 0.98281
1.027547 1.045896 1.09262 1.110207 1.104001 1.062498 1.074949 1.083129 1.09262 1.164363 1.191159
1.142185 1.144981 1.072714 1.045896 1.072714 1.045896 1.019078 1.016843 1.045896 1.045896 1.12635
1.133054 1.153167 1.162107 1.099532 1.104001 1.12635 1.074949 1.072714 1.074949 1.153167 1.162107
1.153167 1.104001 1.099532 1.115622 1.062498 1.074949 1.083129 1.063867 1.045896 1.056905 1.035935
1.016843 1.028972 1.063867 1.045896 1.115622 1.142185 1.162107 1.191441 1.150126 1.12635 1.086264
1.009373 0.958738 0.974816 1.009373 1.104001 1.115622 1.12635 1.121373 1.110207 1.072714 1.035114
1.027547 1.045896 1.063867 1.137285 1.099532 1.09262 1.115622 1.099532 1.121373 1.110207 1.153167
1.035114 1.027547 0.99226 1.009373 1.027547 1.045896 1.016843 1.074949 1.045896 1.072714 1.074949
1.099532 1.074949 1.074949 1.099532 1.099532 1.074949 1.072714 1.045896 1.045896 1.019078 0.99226
0.987791 0.99226 1.016843 1.045896 1.074949 1.072714 1.115622 1.115622 1.133054 1.110207 1.035114
1.045896 0.998188 1.009373 1.045896 1.164363 1.121373 1.072714 0.998188 0.98281 0.958738 1.028972
1.063867 1.083129 1.074949 1.035935 1.016843 1.05605 1.121373 1.137285 1.104001 0.99226 0.929685
0.911807 0.958738 0.98281 0.998188 1.019078 1.016843 1.045896 1.045896 1.045896 1.045896 1.019078
1.045896 1.072714 1.104001 1.12635 1.104001 1.045896 0.987791 0.911807 0.900633 0.947737 1.00636
1.019078 1.027547 1.062498 1.074949 1.083129 1.09262 1.045896 1.027547 0.98281 0.987791 1.028972
1.00636 1.045896 1.056905 1.035935 1.045896 1.028972 0.977607 0.99226 0.998188 0.929685 0.938625
0.880755 0.842527 0.87656 0.96883 0.99226 1.035114 1.028972 0.987791 0.929685 0.998188 1.072714
1.09262 1.05605 1.016843 0.956248 0.96883 1.019078 1.121373 1.179985 1.137285 1.035114 0.998188
0.938625 0.958738 0.99226 1.045896 1.045896 1.016843 0.99226 0.987791 0.965442 0.900633 0.911807
0.929685 1.019078 1.045896 1.019078 0.900633 0.884989 0.958738 1.072714 1.074949 1.045896 0.958738
0.911807 0.910113 0.929685 0.987791 1.001894 0.977607 0.99226 0.998188 1.009373 1.045896 1.083129
0.977607 0.99226 0.939471 0.929685 0.900633 0.920659 0.948854 0.938625 0.96883 0.929685 0.929685
0.947737 0.862595 0.938625 0.947737 0.958738 0.977607 0.947737 0.842527 0.823436 0.880755 0.911807
0.862595 0.893581 0.813475 0.849998 0.822038 0.858171 0.862595 0.974816 1.045896 1.115622 1.144981
0.99226 0.862595 0.785268 0.668211 0.704034 0.747582 0.831353 0.939471 0.98281 0.958738 0.920659
0.862595 0.884989 0.96883 0.956248 0.900633 0.893581 0.891348 0.938625 0.998188 1.009373 1.133054
1.137285 1.121373 1.019078 0.939471 0.743748 0.755369 0.785268 0.842527 0.903123 0.939471 0.911807
0.891348 0.893581 0.842527 0.743748 0.763321 0.697264 0.690076 0.731112 0.724082 0.718829 0.792679
0.831353 0.747582 0.75819 0.858171 0.977607 1.115622 1.072714 0.965442 0.900633 0.858171 0.813475
0.726317 0.670446 0.589993 0.581053 0.509539 0.493895 0.429086 0.406737 0.321814 0.261474 0.214543
0.174316 0.116211 0.134089 0.053636 0.029053 0.026818 0.029053 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0];
forcetwelve = [7.306992 7.306992 7.306992 7.306992 7.306992 7.306992 7.306992 7.306992 7.306992 7.306992
7.942383 7.624688 7.624688 7.306992 7.306992 6.989297 6.989297 7.306992 6.989297 7.306992 6.671602
6.989297 6.671602 6.989297 7.306992 7.624688 7.624688 7.942383 8.577773 10.483945 27.321797 38.441133
14.613984 42.571172 31.769531 17.155547 25.73332 12.390117 17.473242 13.343203 17.473242 26.051016 21.285586
17.790937 21.920977 20.014805 16.837852 31.134141 23.827148 15.56707 20.650195 22.238672 22.238672 17.790937
22.238672 16.520156 15.56707 22.238672 24.144844 18.426328 20.014805 25.09793 15.884766 21.603281 24.780234
20.3325 22.556367 32.087227 27.321797 23.191758 30.181055 37.488047 30.816445 22.874063 35.581875 43.524258
61.315195 14.296289 2.223867 6.671602 52.419727 15.249375 11.119336 20.650195 24.780234 35.26418 44.159648
49.878164 17.473242 32.404922 37.805742 31.134141 25.09793 27.639492 43.524258 38.123438 39.076523 39.076523
30.816445 42.253477 29.863359 44.795039 34.946484 37.488047 26.686406 25.09793 42.571172 34.311094 25.415625
49.560469 58.455937 26.368711 27.639492 38.123438 38.441133 26.368711 30.181055 56.549766 34.946484 32.722617
43.206562 27.957187 41.300391 43.524258 39.394219 35.581875 56.549766 46.701211 29.545664 37.170352 57.820547
37.170352 43.524258 30.816445 29.545664 50.83125 33.040312 47.336602 47.654297 46.383516 43.524258 36.534961
46.06582 39.711914 38.441133 48.289687 47.654297 32.404922 42.253477 59.409023 44.159648 50.513555 40.665
46.06582 54.643594 36.534961 44.795039 58.138242 36.217266 52.102031 45.112734 38.758828 43.206562 52.419727
47.018906 48.289687 53.055117 48.289687 48.925078 44.159648 51.148945 57.185156 52.419727 39.076523 48.607383
54.325898 53.055117 64.809844 40.982695 48.607383 39.394219 48.289687 40.665 56.549766 40.665 35.581875
55.914375 62.268281 52.102031 50.513555 55.278984 58.455937 41.935781 52.737422 61.632891 42.253477 51.466641
43.206562 48.925078 51.148945 42.888867 45.748125 46.701211 48.289687 49.560469 45.43043 53.372813 60.679805
57.820547 58.138242 53.372813 60.362109 55.914375 51.148945 54.008203 54.008203 52.737422 51.466641 55.278984
47.018906 60.362109 62.585977 67.986797 45.748125 44.477344 47.971992 48.925078 50.195859 65.445234 67.986797
59.726719 61.315195 52.102031 56.549766 45.748125 45.748125 42.888867 38.441133 47.971992 57.185156 55.59668
60.679805 54.008203 61.632891 63.539062 66.080625 62.585977 59.409023 64.809844 62.268281 49.878164 67.033711
74.340703 59.409023 66.080625 55.914375 48.925078 47.018906 47.654297 53.055117 76.56457 75.611484 53.055117
56.23207 59.409023 52.419727 53.690508 69.257578 64.174453 55.914375 63.539062 61.950586 54.325898 58.455937
71.799141 60.362109 69.892969 79.423828 51.784336 64.174453 67.986797 52.419727 55.278984 69.257578 65.127539
61.950586 79.741523 95.308594 75.611484 61.950586 61.632891 66.080625 62.585977 73.705313 64.809844 54.325898
63.856758 60.044414 49.242773 67.986797 63.221367 67.351406 57.820547 40.982695 48.289687 54.643594 52.419727
57.185156 66.716016 61.315195 51.784336 70.528359 66.080625 61.950586 67.986797 65.127539 81.012305 61.632891
63.856758 63.539062 62.585977 61.315195 56.23207 74.340703 68.622187 58.455937 63.539062 61.632891 59.091328
55.59668 78.153047 76.246875 86.413125 67.669102 54.961289 73.387617 65.445234 60.044414 79.423828 64.174453
62.903672 74.658398 79.423828 86.413125 77.199961 78.788438 82.283086 79.741523 60.679805 72.116836 75.293789
67.986797 82.600781 89.590078 74.976094 61.950586 61.632891 77.835352 71.481445 81.012305 70.210664 61.632891
78.788438 66.080625 92.449336 67.351406 91.49625 88.001602 86.09543 64.174453 61.632891 63.856758 46.701211
62.585977 63.856758 79.106133 94.355508 93.720117 83.871562 67.033711 82.600781 93.402422 89.907773 80.694609
```

```
        86.413125 97.532461 70.528359 65.127539 92.131641 99.438633 77.517656 71.481445 89.272383 91.49625  55.914375
        37.170352 64.174453 58.455937 59.409023 78.470742 65.445234 95.943984 81.012305 78.788438 88.319297 81.33
        81.012305 80.376914 95.308594 93.720117 82.600781 98.485547 98.803242 93.402422 91.49625  97.214766 97.532461
        75.92918  94.990898 99.120937 86.09543  73.387617 75.293789 60.9975   79.741523 68.939883 87.683906 92.767031
        94.673203 76.882266 106.745625          119.771133          86.73082  78.788438 78.470742 66.39832  93.720117
        109.604883          123.265781          109.604883          119.771133          102.615586          82.918477
        95.626289 69.892969 78.470742 68.622187 90.225469 80.376914 77.835352 79.106133 83.871562 91.813945 85.777734
        76.56457  76.882266 73.069922 60.679805 68.622187 66.080625 60.679805 74.976094 61.950586 19.697109 7.942383
        8.577773];
powertwelve = speedtwelve.*forcetwelve./375;  %converts speed and force to horsepower

%simple moving average of speed
speedtwelvefilter = filter(ones(1,n)/n,1,speedtwelve);
%delete first set of data to make correct matrix
speedtwelvefilter(1:n-1)=[];

%simple moving average of force
forcetwelvefilter = filter(ones(1,n)/n,1,forcetwelve);
%delete first set of data to make correct matrix
forcetwelvefilter(1:n-1)=[];

%simple moving average of power
powertwelvefilter = filter(ones(1,n)/n,1,powertwelve);
%delete first set of date to make correct matrix
powertwelvefilter(1:n-1)=[];

%reduce length of timetwelve to match that of speedtwelve filtered
timetwelveavg = timetwelve;
%delete end of matrix
i = 1;
for i = 1:(n-1)/2
   t = length(timetwelveavg);
   timetwelveavg(t)=[];
end
%delete beginning of matrix
timetwelveavg(1:(n-1)/2)=[];

%_____________________________________________________________________
%PULL THIRTEEN

timethirteen = [0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0
        1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1
        2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2
        3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
        4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
        5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5
        6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6
        7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7
        8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8
        9.9  10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
        11.0 11.1 11.2 11.3 11.4 11.5 11.6 11.7 11.8 11.9 12.0
        12.1 12.2 12.3 12.4 12.5 12.6 12.7 12.8 12.9 13.0 13.1
        13.2 13.3 13.4 13.5 13.6 13.7 13.8 13.9 14.0 14.1 14.2
        14.3 14.4 14.5 14.6 14.7 14.8 14.9 15.0 15.1 15.2 15.3
        15.4 15.5 15.6 15.7 15.8 15.9 16.0 16.1 16.2 16.3 16.4
        16.5 16.6 16.7 16.8 16.9 17.0 17.1 17.2 17.3 17.4 17.5
        17.6 17.7 17.8 17.9 18.0 18.1 18.2 18.3 18.4 18.5 18.6
        18.7 18.8 18.9 19.0 19.1 19.2 19.3 19.4 19.5 19.6 19.7
        19.8 19.9 20.0 20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8
        20.9 21.0 21.1 21.2 21.3 21.4 21.5 21.6 21.7 21.8 21.9
        22.0 22.1 22.2 22.3 22.4 22.5 22.6 22.7 22.8 22.9 23.0
        23.1 23.2 23.3 23.4 23.5 23.6 23.7 23.8 23.9 24.0 24.1
        24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9 25.0 25.1 25.2 25.3
        25.4 25.5 25.6 25.7 25.8 25.9 26.0 26.1 26.2 26.3 26.4
        26.5 26.6 26.7 26.8 26.9 27.0 27.1 27.2 27.3 27.4 27.5
        27.6 27.7 27.8 27.9 28.0 28.1 28.2 28.3 28.4 28.5 28.6
        28.7 28.8 28.9 29.0 29.1 29.2 29.3 29.4 29.5 29.6 29.7
        29.8 29.9 30.0 30.1 30.2 30.3 30.4 30.5 30.6 30.7 30.8
        30.9 31.0 31.1 31.2 31.3 31.4 31.5 31.6 31.7 31.8 31.9
        32.0 32.1 32.2 32.3 32.4 32.5 32.6 32.7 32.8 32.9 33.0
```

33.1 33.2 33.3 33.4 33.5 33.6 33.7 33.8 33.9 34.0 34.1
34.2 34.3 34.4 34.5 34.6 34.7 34.8 34.9 35.0 35.1 35.2
35.3 35.4 35.5 35.6 35.7 35.8 35.9 36.0 36.1 36.2 36.3
36.4 36.5 36.6 36.7 36.8 36.9 37.0 37.1 37.2 37.3 37.4
37.5 37.6 37.7 37.8 37.9 38.0 38.1 38.2 38.3 38.4 38.5
38.6 38.7 38.8 38.9 39.0 39.1 39.2 39.3 39.4 39.5 39.6
39.7 39.8 39.9 40.0 40.1 40.2 40.3 40.4 40.5 40.6 40.7
40.8 40.9 41.0 41.1 41.2 41.3 41.4 41.5 41.6 41.7 41.8
41.9 42.0 42.1 42.2 42.3 42.4 42.5 42.6 42.7 42.8 42.9
43.0 43.1 43.2 43.3 43.4 43.5 43.6 43.7 43.8 43.9 44.0
44.1 44.2 44.3 44.4 44.5 44.6 44.7 44.8 44.9 45.0 45.1
45.2 45.3 45.4 45.5 45.6];

speedthirteen = [0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0.053636 0.261474
0.348632 0.581053 0.831353 0.987791 0.99226 0.87158 0.777718 0.726317 0.697264 0.726317 0.7509
0.784422 0.804535 0.929685 1.179985 1.30737 1.394528 1.336423 1.245598 1.150126 1.110207 0.987791
0.956248 0.987791 1.05605 1.00636 1.083129 1.191159 1.19531 1.174339 1.153167 1.063867 1.05605
1.045896 1.142185 1.191159 1.21852 1.207632 1.272676 1.249265 1.12635 0.987791 1.019078 0.987791
1.072714 1.162107 1.179985 1.191159 1.233621 1.278317 1.206803 1.162107 1.045896 1.045896 1.099532
1.104001 1.12635 1.191159 1.206803 1.162107 1.099532 1.045896 1.045896 1.074949 1.12635 1.174339
1.115622 1.133054 1.164363 1.063867 1.045896 1.056905 0.98281 1.045896 1.191441 1.207632 1.12635
1.086264 1.062498 1.104001 1.137285 1.150126 1.153167 1.174339 1.115622 1.104001 1.110207 1.074949
1.115622 1.174339 1.179985 1.236386 1.299754 1.365475 1.168747 1.086264 0.965442 0.977607 1.083129
1.162107 1.168747 1.233056 1.153167 1.150126 1.191441 1.220212 1.115622 1.086264 1.153167 1.133054
1.12635 1.045896 1.12635 1.045896 1.12635 1.133054 1.206803 1.278317 1.206803 1.191159 1.12635
1.074949 0.965442 1.016843 1.099532 1.133054 1.206803 1.30737 1.314074 1.191159 1.153167 1.086264
1.035935 0.987791 0.974816 0.891348 0.99226 1.115622 1.19531 1.30737 1.272676 1.236386 1.072714
1.056905 1.035935 1.074949 1.083129 1.063867 1.099532 1.027547 0.98281 1.104001 1.245598 1.221872
1.074949 0.998188 1.045896 1.09262 1.110207 1.153167 1.09262 1.144981 1.099532 1.121373 1.083129
1.019078 1.016843 1.016843 1.045896 1.074949 1.099532 1.099532 1.133054 1.153167 1.045896 1.104001
1.072714 1.072714 1.133054 1.206803 1.220212 1.191441 1.063867 0.938625 1.027547 1.121373 1.206803
1.191441 1.121373 1.12635 1.115622 1.121373 1.153167 1.137285 1.063867 1.099532 1.086264 0.98281
0.987791 0.974816 1.035114 1.153167 1.233056 1.233621 1.178879 1.137285 1.045896 1.062498 1.144981
1.099532 1.035114 1.028972 0.987791 1.009373 1.144981 1.12635 1.121373 1.164363 1.133054 1.035935
0.96883 1.019078 1.121373 1.21852 1.30737 1.274997 1.233056 1.099532 0.958738 0.938625 0.987791
1.019078 1.133054 1.206803 1.249265 1.099532 1.074949 1.099532 1.133054 1.099532 1.191159 1.233621
1.220212 1.045896 1.016843 0.965442 0.987791 1.045896 1.074949 0.99226 0.96883 1.009373 1.045896
1.110207 1.150126 1.12635 1.174339 1.142185 1.016843 0.893581 0.833841 0.884989 1.027547 1.009373
0.958738 1.028972 1.035114 1.12635 1.115622 0.98281 0.929685 1.028972 1.104001 1.19531 1.262415
1.233621 1.063867 1.001894 1.045896 1.142185 1.144981 1.179985 1.121373 1.191441 1.133054 1.009373
0.96883 1.099532 1.09262 1.137285 1.104001 1.062498 1.086264 0.99226 0.929685 0.938625 1.016843
1.179985 1.249265 1.260439 1.191159 1.153167 1.162107 1.083129 0.948854 0.911807 0.958738 0.99226
1.133054 1.164363 1.121373 1.153167 1.174339 1.19531 1.133054 1.05605 0.977607 0.99226 0.96883
0.938625 0.948854 0.920659 0.958738 1.035935 1.086264 1.206803 1.322645 1.272676 1.30737 1.19531
1.086264 0.965442 0.977607 1.05605 1.074949 1.08906 1.056905 1.072714 1.035114 1.083129 1.074949
1.115622 1.115622 1.153167 1.074949 1.072714 0.987791 0.99226 0.929685 0.99226 1.104001 1.206803
1.30737 1.287257 1.220212 1.099532 0.929685 0.884989 0.900633 1.045896 1.191159 1.206803 1.162107
1.153167 1.162107 1.099532 1.086264 1.035935 1.074949 1.083129 1.09262 1.099532 1.144981 1.142185
1.162107 1.137285 1.063867 0.911807 0.880755 0.929685 0.987791 1.164363 1.121373 1.179985 1.086264
1.062498 1.104001 1.05605 1.062498 1.016843 1.027547 1.072714 1.00636 1.028972 0.938625 0.977607
1.027547 0.965442 0.948854 0.893581 0.563175 0.373791 0.440377 0.37545 0.143766 0.027078 0.053636
0.143766 0.176151 0.134089 0.10625 0.088075 0.053636 0.029053 0.145263 0.321814 0.348632 0.232421
0.160907 0.058105 0.029053 0 0 0 0 0 0 0 0
0 0 0 0.026818 0.029053 0.026818];

forcethirteen = [6.353906 6.036211 6.353906 6.353906 6.353906 6.353906 6.036211 6.671602 6.036211 6.353906
6.036211 6.036211 6.353906 6.036211 6.036211 6.036211 6.353906 6.353906 9.530859 14.93168 49.878164
47.018906 39.394219 59.091328 37.170352 26.686406 26.686406 21.285586 16.202461 16.837852 22.556367 27.321797
30.181055 51.784336 51.466641 37.805742 35.26418 34.311094 16.520156 18.426328 24.144844 27.321797 30.181055
28.274883 30.181055 31.451836 42.253477 33.993398 32.722617 27.639492 29.545664 22.556367 29.227969 36.852656
40.029609 41.935781 29.227969 20.967891 30.181055 29.545664 26.051016 16.520156 23.827148 36.852656 31.134141
34.628789 37.170352 40.029609 26.368711 38.758828 32.722617 31.451836 28.910273 30.816445 37.805742 35.89957
28.910273 38.441133 42.571172 23.191758 29.863359 34.946484 30.49875 34.311094 45.748125 44.477344 30.816445
37.805742 34.946484 27.321797 31.451836 35.89957 38.123438 37.170352 47.654297 42.571172 22.874063 34.946484
42.571172 38.123438 36.217266 36.534961 40.982695 42.888867 39.711914 37.488047 40.029609 35.581875 42.571172
47.018906 44.795039 37.488047 28.910273 34.946484 28.592578 32.722617 24.462539 27.321797 42.571172 45.112734
34.311094 28.910273 41.300391 26.686406 39.711914 32.404922 25.73332 23.509453 37.805742 38.441133 44.477344
44.159648 84.824648 47.018906 36.217266 42.888867 71.16375 55.278984 44.159648 60.362109 42.571172 26.686406
31.769531 51.784336 44.477344 60.362109 53.055117 50.195859 50.195859 45.748125 27.957187 30.816445 44.795039
40.029609 41.935781 60.9975 65.445234 68.939883 62.585977 60.9975 57.820547 38.441133 40.347305 42.888867
33.675703 41.935781 42.253477 44.795039 53.055117 42.571172 53.372813 55.59668 52.102031 50.83125 42.571172

```
        41.300391 62.268281 53.055117 53.372813 51.784336 45.43043  60.362109 38.758828 48.289687 43.206562 50.513555
        53.055117 60.679805 65.76293  65.76293  55.278984 46.06582  43.524258 62.585977 42.571172 45.748125 45.748125
        59.409023 54.325898 57.820547 48.289687 57.185156 51.466641 52.737422 66.716016 69.257578 65.127539 61.950586
        62.585977 65.127539 48.925078 47.971992 70.528359 59.726719 43.524258 47.018906 49.878164 49.242773 67.033711
        64.492148 55.59668  65.445234 74.023008 68.304492 57.820547 63.221367 54.008203 62.585977 66.39832  65.76293
        52.102031 54.325898 62.585977 59.091328 69.892969 63.539062 52.419727 53.055117 59.091328 57.820547 52.419727
        62.268281 59.409023 62.268281 58.773633 53.372813 47.336602 48.289687 57.820547 58.138242 67.669102 71.16375
        74.023008 64.174453 93.402422 67.669102 70.210664 65.76293  57.185156 81.965391 69.575273 63.221367 59.409023
        46.701211 70.210664 58.455937 71.481445 65.76293  68.939883 64.492148 75.611484 80.059219 91.813945 82.918477
        79.423828 62.268281 50.513555 81.012305 61.315195 74.023008 68.622187 74.340703 53.055117 58.138242 45.43043
        81.012305 71.16375  68.622187 67.351406 87.683906 87.048516 82.918477 87.683906 79.423828 81.33       76.56457
        68.939883 75.92918  75.92918  83.553867 75.611484 73.387617 59.409023 57.820547 61.950586 51.466641 57.502852
        68.304492 65.445234 74.340703 67.033711 70.846055 67.669102 69.257578 87.366211 92.767031 86.09543  91.49625
        77.517656 69.892969 55.914375 53.690508 60.679805 70.210664 58.138242 68.304492 81.33       99.120937 101.6625
        85.460039 80.694609 72.116836 68.939883 67.033711 70.846055 59.091328 61.315195 68.939883 66.080625 80.059219
        88.319297 87.366211 94.990898 97.214766 103.250977           85.142344 81.965391 69.892969 81.965391 49.878164
        54.008203 55.914375 64.809844 70.846055 72.434531 77.199961 79.741523 83.236172 74.340703 85.142344 83.553867
        85.142344 81.965391 74.976094 81.012305 85.777734 81.012305 94.355508 97.532461 101.344805
        108.651797           96.26168  84.824648 65.127539 63.539062 60.044414 57.502852 70.846055 79.106133 93.084727
        94.673203 74.976094 70.846055 89.272383 65.127539 74.340703 92.131641 79.423828 76.56457  78.788438
        120.088828           68.304492 93.720117 84.824648 79.423828 57.185156 81.647695 74.658398 99.438633 84.506953
        100.391719           84.824648 89.907773 84.824648 78.788438 88.954687 74.340703 82.283086 79.741523 72.116836
        80.376914 72.116836 79.106133 89.907773 86.73082  78.788438 80.694609 85.777734 88.319297 87.048516 84.824648
        92.767031 89.272383 86.73082  91.178555 92.767031 88.954687 88.954687 84.824648 82.600781 87.048516 85.142344
        88.636992 88.319297 84.189258 85.460039 83.236172 85.460039 83.871562 90.543164 88.954687 81.965391 78.153047
        75.611484 75.293789 74.023008 73.705313 73.705313 74.023008 73.387617 73.387617 73.069922 6.353906  3.812344
        1.270781  4.76543   4.130039];
powerthirteen = speedthirteen.*forcethirteen./375;  %converts speed and force to horsepower

%simple moving average of speed
speedthirteenfilter = filter(ones(1,n)/n,1,speedthirteen);
%delete first set of data to make correct matrix
speedthirteenfilter(1:n-1)=[];

%simple moving average of force
forcethirteenfilter = filter(ones(1,n)/n,1,forcethirteen);
%delete first set of data to make correct matrix
forcethirteenfilter(1:n-1)=[];

%simple moving average of power
powerthirteenfilter = filter(ones(1,n)/n,1,powerthirteen);
%delete first set of date to make correct matrix
powerthirteenfilter(1:n-1)=[];

%reduce length of timethirteen to match that of speedthirteen filtered
timethirteenavg = timethirteen;
%delete end of matrix
for i = 1:(n-1)/2
   t = length(timethirteenavg);
   timethirteenavg(t)=[];
end
%delete beginning of matrix
timethirteenavg(1:(n-1)/2)=[];

%_____________________________________________________________________
%PULL FIFTEEN

timefifteen = [0    0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9   1
        1.1   1.2   1.3   1.4   1.5   1.6   1.7   1.8   1.9   2     2.1
        2.2   2.3   2.4   2.5   2.6   2.7   2.8   2.9   3     3.1   3.2
        3.3   3.4   3.5   3.6   3.7   3.8   3.9   4     4.1   4.2   4.3
        4.4   4.5   4.6   4.7   4.8   4.9   5     5.1   5.2   5.3   5.4
        5.5   5.6   5.7   5.8   5.9   6     6.1   6.2   6.3   6.4   6.5
        6.6   6.7   6.8   6.9   7     7.1   7.2   7.3   7.4   7.5   7.6
        7.7   7.8   7.9   8     8.1   8.2   8.3   8.4   8.5   8.6   8.7
        8.8   8.9   9     9.1   9.2   9.3   9.4   9.5   9.6   9.7   9.8
        9.9   10    10.1  10.2  10.3  10.4  10.5  10.6  10.7  10.8  10.9
        11    11.1  11.2  11.3  11.4  11.5  11.6  11.7  11.8  11.9  12
        12.1  12.2  12.3  12.4  12.5  12.6  12.7  12.8  12.9  13    13.1
        13.2  13.3  13.4  13.5  13.6  13.7  13.8  13.9  14    14.1  14.2
```

14.3 14.4 14.5 14.6 14.7 14.8 14.9 15 15.1 15.2 15.3
15.4 15.5 15.6 15.7 15.8 15.9 16 16.1 16.2 16.3 16.4
16.5 16.6 16.7 16.8 16.9 17 17.1 17.2 17.3 17.4 17.5
17.6 17.7 17.8 17.9 18 18.1 18.2 18.3 18.4 18.5 18.6
18.7 18.8 18.9 19 19.1 19.2 19.3 19.4 19.5 19.6 19.7
19.8 19.9 20 20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8
20.9 21 21.1 21.2 21.3 21.4 21.5 21.6 21.7 21.8 21.9
22 22.1 22.2 22.3 22.4 22.5 22.6 22.7 22.8 22.9 23
23.1 23.2 23.3 23.4 23.5 23.6 23.7 23.8 23.9 24 24.1
24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9 25 25.1 25.2 25.3
25.4 25.5 25.6 25.7 25.8 25.9 26 26.1 26.2 26.3 26.4
26.5 26.6 26.7 26.8 26.9 27 27.1 27.2 27.3 27.4 27.5
27.6 27.7 27.8 27.9 28 28.1 28.2 28.3 28.4 28.5 28.6
28.7 28.8 28.9 29 29.1 29.2 29.3 29.4 29.5 29.6 29.7
29.8 29.9 30 30.1 30.2 30.3 30.4 30.5 30.6 30.7 30.8
30.9 31 31.1 31.2 31.3 31.4 31.5 31.6 31.7 31.8 31.9
32 32.1 32.2 32.3 32.4 32.5 32.6 32.7 32.8 32.9 33
33.1 33.2 33.3 33.4 33.5 33.6 33.7 33.8 33.9 34 34.1
34.2 34.3 34.4 34.5 34.6 34.7 34.8 34.9 35 35.1 35.2
35.3 35.4 35.5 35.6 35.7 35.8 35.9 36 36.1 36.2 36.3
36.4 36.5 36.6 36.7 36.8 36.9 37 37.1 37.2 37.3 37.4
37.5 37.6 37.7 37.8 37.9 38 38.1 38.2 38.3 38.4 38.5
38.6 38.7 38.8 38.9 39 39.1 39.2 39.3 39.4 39.5 39.6
39.7 39.8 39.9 40 40.1 40.2 40.3 40.4 40.5 40.6 40.7
40.8 40.9 41 41.1 41.2 41.3 41.4 41.5 41.6 41.7 41.8
41.9 42 42.1 42.2 42.3 42.4 42.5 42.6 42.7 42.8 42.9
43 43.1 43.2 43.3 43.4 43.5 43.6 43.7 43.8 43.9 44
44.1 44.2 44.3 44.4 44.5 44.6 44.7 44.8 44.9 45 45.1
45.2 45.3 45.4 45.5 45.6 45.7 45.8 45.9 46 46.1 46.2
46.3 46.4 46.5 46.6 46.7 46.8 46.9 47 47.1 47.2 47.3
47.4 47.5 47.6 47.7 47.8 47.9 48 48.1 48.2 48.3 48.4
48.5 48.6 48.7 48.8 48.9 49 49.1 49.2];

speedfifteen = [0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0.029053 0.294996
0.610106 0.965442 1.423581 1.528617 1.452633 1.421346 1.30737 1.287257 1.220212 1.12635 0.958738
0.884989 0.958738 1.045896 1.074949 1.083129 1.150126 1.179985 1.203698 1.153167 1.150126 1.110207
1.104001 1.115622 1.115622 0.99226 1.035114 1.110207 1.133054 1.19531 1.262415 1.206803 1.207632
1.21852 1.104001 1.08906 1.115622 1.099532 1.09262 1.137285 1.104001 1.08906 1.174339 1.179985
1.260439 1.249265 1.162107 1.153167 1.220212 1.153167 1.099532 1.045896 0.958738 0.911807 0.987791
1.099532 1.12635 1.191159 1.260439 1.249265 1.278317 1.099532 1.099532 1.074949 1.072714 1.074949
1.137285 1.236386 1.314074 1.233056 1.08906 1.045896 1.110207 0.920101 0.965442 0.96883 1.062498
1.220212 1.326832 1.150126 1.12635 1.115622 1.009373 1.045896 1.028972 1.035114 1.072714 1.174339
1.142185 1.220212 1.191441 1.220212 1.221872 1.245598 1.220212 1.150126 1.083129 1.016843 0.98281
1.056905 1.045896 1.09262 1.164363 1.191159 1.221872 1.291773 1.179985 1.035114 1.028972 1.016843
0.956248 0.880755 0.99226 1.104001 1.12635 1.191159 1.233621 1.249265 1.206803 1.133054 1.12635
1.133054 1.179985 1.220212 1.233621 1.278317 1.233621 1.104001 1.019078 1.045896 0.965442 0.987791
1.019078 1.099532 1.191159 1.086264 1.08906 1.162107 1.299754 1.207632 1.314074 1.203698 1.035935
0.958738 1.001894 1.00636 1.045896 1.115622 1.142185 1.162107 1.137285 1.121373 1.05605 1.099532
1.063867 1.203698 1.099532 1.063867 0.920659 0.958738 0.99226 1.104001 1.153167 1.133054 1.045896
0.958738 0.99226 1.016843 0.95551 0.970106 0.938625 0.987791 1.045896 1.162107 1.179985 1.336423
1.233621 1.162107 1.099532 1.045896 1.045896 1.016843 1.001894 1.035114 1.045896 1.056905 0.98281
1.016843 1.028972 1.00636 1.045896 1.027547 1.062498 1.162107 1.137285 1.121373 1.099532 1.056905
0.98281 1.045896 1.083129 1.063867 1.072714 1.056905 1.009373 1.074949 1.164363 1.220212 1.274997
1.233056 1.153167 1.09262 1.110207 1.162107 1.142185 1.203698 1.072714 0.920101 0.866503 0.87158
1.035935 1.233056 1.314074 1.236386 1.110207 1.045896 0.956248 1.086264 1.072714 0.929685 0.938625
0.929685 1.019078 1.104001 1.153167 1.133054 1.153167 1.191159 1.233621 1.074949 1.019078 1.016843
1.072714 1.191159 1.233621 1.278317 1.179985 1.133054 1.12635 1.016843 0.858171 0.939471 0.956248
1.12635 1.262415 1.178879 1.206803 1.056905 0.903123 0.87158 1.001894 1.121373 1.153167 1.233056
1.221872 1.162107 1.028972 0.977607 1.045896 1.115622 1.062498 1.045896 1.05605 1.062498 1.104001
1.027547 0.938625 0.920101 1.05605 1.153167 1.178879 1.203698 1.206803 1.236386 1.191441 1.12635
1.035114 0.998188 0.99226 0.977607 1.001894 1.072714 1.121373 1.137285 1.191159 1.221872 1.233056
1.019078 0.939471 0.833841 0.777718 0.812346 0.747582 0.894603 1.008807 1.074949 1.179985 1.191441
1.207632 1.233621 1.233056 1.121373 0.99226 1.001894 1.09262 1.179985 1.233056 1.207632 1.12635
1.05605 0.958738 0.929685 0.998188 1.045896 1.121373 1.164363 1.074949 1.009373 0.96883 0.884989
0.776335 0.839425 0.900633 1.009373 1.174339 1.12635 1.121373 1.137285 1.016843 0.98281 0.939471
0.99226 1.035114 1.028972 1.045896 1.019078 1.074949 1.099532 1.074949 1.072714 1.074949 1.099532
1.074949 1.179985 1.074949 1.072714 1.133054 1.206803 1.104001 0.99226 0.900633 0.884989 0.929685
0.965442 1.074949 1.072714 1.104001 1.019078 1.045896 1.099532 1.086264 1.035935 1.045896 1.001894
0.920101 0.965442 0.998188 0.98281 1.016843 1.05605 1.035114 1.019078 0.998188 0.878036 0.929685

```
        0.947737  0.833841  0.858171  0.733962  0.478124  0.348632  0.135391  0.058105  0.823436  0.792679  0.804535
        0.747582  0.75819   0.755369  0.610936  0.557811  0.321814  0.086259  0         0.058105  0         0
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0 ];
forcefifteen = [13.978594 13.343203 13.343203 13.025508 13.978594 13.660898 13.343203 13.343203 13.343203 13.978594
        13.025508 13.660898 12.072422 12.072422 12.072422 12.390117 11.754727 11.754727 11.437031 11.119336 11.119336
        10.801641 11.437031 11.754727 11.754727 11.437031 13.660898 13.660898 13.025508 14.613984 20.014805 50.513555
        81.965391 85.142344 43.206562 41.618086 26.686406 20.014805 13.978594 10.801641 0.953086  15.884766 12.390117
        17.473242 29.545664 35.581875 26.051016 42.571172 37.170352 27.004102 26.686406 30.816445 20.014805 20.967891
        27.004102 27.639492 32.722617 32.404922 42.888867 40.347305 32.404922 38.758828 43.206562 26.051016 27.321797
        36.852656 27.004102 31.134141 38.123438 29.545664 32.404922 38.441133 45.43043  42.571172 38.441133 36.534961
        31.451836 30.181055 36.217266 24.144844 30.181055 27.004102 23.827148 34.311094 34.628789 36.852656 40.347305
        41.300391 47.654297 45.748125 39.076523 20.014805 33.993398 33.675703 36.217266 37.805742 38.441133 48.289687
        47.654297 36.534961 26.686406 25.09793  26.686406 31.134141 44.795039 29.863359 33.993398 54.008203 59.726719
        50.83125  37.488047 45.112734 25.73332  38.441133 42.253477 53.372813 53.055117 56.549766 52.419727 54.643594
        56.549766 38.441133 49.242773 51.148945 44.795039 43.841953 39.394219 42.253477 61.315195 63.539062 47.018906
        61.315195 58.138242 53.055117 52.737422 47.018906 58.773633 40.029609 38.441133 35.581875 53.372813 57.820547
        50.513555 81.33    57.502852 67.669102 72.434531 53.055117 52.419727 52.737422 48.925078 50.513555 56.867461
        59.726719 60.679805 53.372813 60.362109 42.888867 41.618086 45.43043  54.325898 45.43043  50.83125  60.362109
        64.492148 66.716016 67.033711 65.127539 56.23207  54.643594 45.112734 30.181055 59.091328 22.556367 33.993398
        50.83125  63.221367 68.622187 72.752227 66.716016 45.43043  68.622187 59.726719 46.701211 57.185156 50.195859
        42.571172 42.888867 47.971992 45.43043  54.008203 64.174453 65.76293  60.044414 63.539062 64.809844 67.033711
        55.914375 54.008203 53.055117 67.351406 59.726719 67.986797 70.210664 73.069922 73.387617 63.221367 45.748125
        55.278984 48.925078 47.654297 39.394219 45.43043  47.336602 55.914375 62.903672 60.679805 57.185156 57.502852
        67.986797 61.315195 79.106133 68.304492 65.445234 77.199961 58.138242 53.055117 55.278984 56.549766 51.148945
        48.925078 63.221367 69.257578 61.315195 68.622187 68.622187 67.351406 68.304492 63.539062 72.752227 58.455937
        68.939883 68.304492 58.773633 69.892969 56.549766 57.820547 54.643594 70.210664 68.939883 74.340703 77.199961
        79.106133 72.434531 68.304492 54.008203 44.477344 66.080625 56.23207  59.091328 67.986797 55.914375 67.669102
        77.517656 81.012305 82.918477 83.236172 70.846055 56.23207  48.289687 59.091328 63.539062 52.737422 66.080625
        85.777734 72.434531 69.257578 44.159648 42.253477 52.419727 61.315195 65.445234 88.954687 77.835352 79.423828
        69.257578 71.799141 66.080625 60.044414 69.892969 70.846055 89.590078 87.683906 73.705313 64.174453 67.351406
        66.39832  66.080625 51.466641 55.914375 67.669102 71.481445 73.069922 69.892969 63.539062 74.658398 80.694609
        89.907773 91.178555 81.965391 82.918477 81.647695 76.246875 64.492148 64.174453 53.372813 60.362109 55.278984
        67.351406 78.153047 75.92918  80.059219 75.611484 81.012305 77.199961 81.647695 67.033711 64.174453 54.325898
        45.112734 60.362109 75.92918  83.236172 93.402422 103.568672 106.42793 96.26168  91.49625  78.788438
        74.658398 72.752227 67.986797 66.080625 69.892969 74.976094 85.142344 72.116836 70.210664 73.069922 70.528359
        68.622187 68.622187 77.835352 87.683906 86.413125 81.965391 77.199961 94.355508 85.142344 84.506953 91.813945
        88.319297 94.990898 99.756328 91.178555 89.907773 73.069922 75.293789 70.210664 70.210664 73.387617 83.236172
        93.084727 86.413125 77.199961 69.575273 74.023008 98.485547 75.92918  82.918477 85.777734 74.023008 73.705313
        64.809844 89.272383 68.622187 80.059219 63.221367 66.080625 78.788438 55.278984 61.950586 68.304492 84.824648
        81.647695 75.611484 98.803242 73.387617 74.976094 95.626289 84.506953 87.048516 80.059219 77.199961 74.023008
        84.506953 86.09543  84.189258 86.413125 85.142344 91.49625  82.600781 89.907773 84.506953 80.376914 92.767031
        89.272383 77.835352 92.449336 86.09543  83.553867 87.048516 79.423828 78.788438 72.752227 91.813945 84.189258
        80.694609 91.178555 81.012305 79.106133 92.449336 82.600781 91.813945 96.579375 89.907773 89.272383 86.09543
        84.506953 85.460039 83.871562 84.824648 83.553867 82.918477 82.918477 82.283086 82.600781 82.283086 81.965391
        81.965391 82.283086 83.236172 82.283086 82.283086 81.965391 82.283086 81.647695 81.965391 81.647695 81.647695
        81.647695 81.012305 81.33    81.012305 80.694609 81.33    80.376914 81.012305 80.694609 80.376914 80.694609
        80.694609 80.059219 80.376914 81.012305 80.059219 80.059219 80.376914 80.376914 80.059219 79.741523 79.741523
        80.059219 80.059219 79.741523 79.741523 80.059219 79.423828 79.741523 79.423828 80.376914 79.423828
        79.423828];
powerfifteen = speedfifteen.*forcefifteen./375;  %converts speed and force to horsepower

%simple moving average of speed
speedfifteenfilter = filter(ones(1,n)/n,1,speedfifteen);
%delete first set of data to make correct matrix
speedfifteenfilter(1:n-1)=[];

%simple moving average of force
forcefifteenfilter = filter(ones(1,n)/n,1,forcefifteen);
%delete first set of data to make correct matrix
forcefifteenfilter(1:n-1)=[];

%simple moving average of power
powerfifteenfilter = filter(ones(1,n)/n,1,powerfifteen);
%delete first set of date to make correct matrix
powerfifteenfilter(1:n-1)=[];
```

```
%reduce length of timefifteen to match that of speedfifteen filtered
timefifteenavg = timefifteen;
%delete end of matrix
for i = 1:(n-1)/2
   t = length(timefifteenavg);
   timefifteenavg(t)=[];
end
%delete beginning of matrix
timefifteenavg(1:(n-1)/2)=[];

%_____________________________________________________________________
%FIGURES

figure(1)
h = plot(timetwelveavg,speedtwelvefilter,timethirteenavg,speedthirteenfilter,timefifteenavg,speedfifteenfilter);
legend('Twelve','Thirteen','Fifteen')
title(['TALON Speed vs. Time on Wet Grass, Moving Average Span of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Speed, (mph)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
xlim([0 45])

figure(2)
h = plot(timetwelveavg,forcetwelvefilter,timethirteenavg,forcethirteenfilter,timefifteenavg,forcefifteenfilter);
legend('Twelve','Thirteen','Fifteen')
title(['TALON Force vs. Time on Wet Grass, Moving Average of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Force, (lb_f)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
xlim([0 45])

figure(3)
h = plot(timetwelveavg,powertwelvefilter,timethirteenavg,powerthirteenfilter,timefifteenavg,powerfifteenfilter);
legend('Twelve','Thirteen','Fifteen')
title(['TALON Power vs. Time on Wet Grass, Moving Average of ',num2str(n)])
xlabel('Time, (s)')
ylabel('Power, (hp)')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
set(h,'Linewidth',2)
xlim([0 45])
```

Appendix E – MATLAB Code for TALON, MATILDA Comparison

```
%Created 1-9-06
%Pull One, MATILDA,
%Speed setting:  Half
%Sled Payload:  100 lbs
%tape measure distance:  ???"
%TALON moving average of 25, sampled at 10 Hz
%MATILDA  moving average of 5, sampled at 2 Hz due to equipment

close all
clear all
clc

n = input('What size of moving average for TALON? (must be an odd number)\n\n');
m = input('\nWhat size of moving average for MATILDA (must be an odd number)\n');
%___________________________________________________________________________
%MATILDA DATA

timeoneM = [0 0.5   1.0   1.5   2.0   2.5   3.0   3.5   4.0   4.5   5.0   5.5
        6.0   6.5   7.0   7.5   8.0   8.5   9.1   9.5   10.0  10.5  11.0
        11.5  12.0  12.5  13.0  13.5  14.0  14.5  15.0  15.5  16.0  16.5
        17.0  17.5  18.0  18.5  19.0  19.5  20.0  20.5  21.0  21.5  22.0
        22.5  23.0  23.5  24.0  24.5  25  25.5  26  26.5  27  27.5  28  28.5  29  29.5  30  30.5  31  31.5  32  32.5
33  33.5  34  34.5  35  35.5  36  36.5];
speedoneM = [0   0   0   0   0   0   0   0.3067   1.436266   1.857296   1.897796
        1.878048  1.85035   1.85579   1.711466  1.832058  1.760732  1.732784  1.700902  1.675716  1.541486  1.509702
        1.639484  1.618648  1.568844  1.377268  1.228296  1.193234  1.112318  0.946088  0.854008  0.684812  0.521894
        0.295714  0.03163   0.036316  0.020152  0.027124  0   0   0   0   0   0
        0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0].2;
forceoneM = [9.848555   9.530859  9.848555  9.848555  9.848555  9.848555  9.848555  30.816445 26.051016 12.072422
        15.249375 19.697109 19.061719 33.358008 28.910273 27.004102 27.321797 32.722617 28.274883 39.711914 34.311094
        31.134141 34.946484 28.274883 44.159648 58.455937 48.925078 40.982695 56.549766 48.289687 44.477344 49.878164
        55.278984 58.455937 46.06582  46.06582  58.455937 3.812344  3.176953  2.859258  2.859258  2.859258  3.176953
        2.859258  3.176953  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0];
poweroneM = speedoneM.*forceoneM./375;  %converts speed and force to horsepower

%simple moving average of speed
speedonefilterM = filter(ones(1,m)/m,1,speedoneM);
%delete first set of data to make correct matrix
speedonefilterM(1:m-1)=[];

%simple moving average of force
forceonefilterM = filter(ones(1,m)/m,1,forceoneM);
%delete first set of data to make correct matrix
forceonefilterM(1:m-1)=[];

%simple moving average of power
poweronefilterM = filter(ones(1,m)/m,1,poweroneM);
%delete first set of date to make correct matrix
poweronefilterM(1:m-1)=[];


%-------------------------------------------------------------------
%reduce length of timeone to match that of speedone filtered
timeoneavgM = timeoneM;
%delete end of matrix
i = 1;
for i = 1:(m-1)/2
   t = length(timeoneavgM);
   timeoneavgM(t)=[];
end
%delete beginning of matrix
timeoneavgM(1:(m-1)/2)=[];
%___________________________________________________________________________

%___________________________________________________________________________
%TALON DATA
```

```
timefourT = [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1
2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2
3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3
4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4
5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5
6.6 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6
7.7 7.8 7.9 8.0 8.1 8.2 8.3 8.4 8.5 8.6 8.7
8.8 8.9 9.0 9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.8
9.9 10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
11.0 11.1 11.2 11.3 11.4 11.5 11.6 11.7 11.8 11.9 12.0
12.1 12.2 12.3 12.4 12.5 12.6 12.7 12.8 12.9 13.0 13.1
13.2 13.3 13.4 13.5 13.6 13.7 13.8 13.9 14.0 14.1 14.2
14.3 14.4 14.5 14.6 14.7 14.8 14.9 15.0 15.1 15.2 15.3
15.4 15.5 15.6 15.7 15.8 15.9 16.0 16.1 16.2 16.3 16.4
16.5 16.6 16.7 16.8 16.9 17.0 17.1 17.2 17.3 17.4 17.5
17.6 17.7 17.8 17.9 18.0 18.1 18.2 18.3 18.4 18.5 18.6
18.7 18.8 18.9 19.0 19.1 19.2 19.3 19.4 19.5 19.6 19.7
19.8 19.9 20.0 20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8
20.9 21.0 21.1 21.2 21.3 21.4 21.5 21.6 21.7 21.8 21.9
22.0 22.1 22.2 22.3 22.4 22.5 22.6 22.7 22.8 22.9 23.0
23.1 23.2 23.3 23.4 23.5 23.6 23.7 23.8 23.9 24.0 24.1
24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9 25.0 25.1 25.2 25.3
25.4 25.5 25.6 25.7 25.8 25.9 26.0 26.1 26.2 26.3 26.4
26.5 26.6 26.7 26.8 26.9 27.0 27.1 27.2 27.3 27.4 27.5
27.6 27.7 27.8 27.9 28.0 28.1 28.2 28.3 28.4 28.5 28.6
28.7 28.8 28.9 29.0 29.1 29.2 29.3 29.4 29.5 29.6 29.7
29.8 29.9 30.0 30.1 30.2 30.3 30.4 30.5 30.6 30.7 30.8
30.9 31.0 31.1 31.2 31.3 31.4 31.5 31.6 31.7 31.8 31.9
32.0 32.1 32.2 32.3 32.4 32.5 32.6 32.7 32.8 32.9 33.0
33.1 33.2 33.3 33.4 33.5 33.6 33.7 33.8 33.9 34.0 34.1
34.2 34.3 34.4 34.5 34.6 34.7 34.8 34.9 35.0 35.1 35.2
35.3 35.4 35.5 35.6 35.7 35.8 35.9 36.0 36.1 36.2 36.3
36.4 36.5 36.6 36.7 36.8 36.9 37.0 37.1 37.2 37.3 37.4
37.5 37.6 37.7 37.8 37.9 38.0 38.1 38.2 38.3 38.4 38.5
38.6 38.7 38.8 38.9 39.0 39.1 39.2 39.3 39.4 39.5 39.6
39.7 39.8 39.9 40.0 40.1 40.2 40.3 40.4 40.5 40.6 40.7
40.8 40.9 41.0 41.1 41.2 41.3 41.4 41.5 41.6 41.7 41.8
41.9 42.0 42.1 42.2 42.3 42.4 42.5 42.6 42.7 42.8 42.9];
speedfourT = [0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0.028753 0.241361 0.557811
0.743748 0.900633 1.083129 1.09262 1.072714 1.086264 1.062498 1.074949 1.028972 0.987791 1.062498
1.115622 1.12635 1.121373 1.137285 1.133054 1.099532 1.045896 1.072714 1.104001 1.099532 1.074949
1.099532 1.133054 1.179985 1.191159 1.153167 1.133054 1.12635 1.104001 1.099532 1.086264 1.009373
1.016843 1.001894 1.00636 1.019078 1.056905 1.062498 1.104001 1.137285 1.121373 1.12635 1.144981
1.142185 1.074949 1.083129 1.035114 1.045896 1.027547 1.035935 1.074949 1.110207 1.133054 1.168747
1.174339 1.099532 1.09262 1.083129 1.074949 1.009373 1.056905 0.99226 1.00636 1.05605 1.074949
1.142185 1.144981 1.12635 1.09262 1.110207 1.074949 1.062498 1.056905 1.045896 1.104001 1.099532
1.133054 1.099532 1.104001 1.072714 1.045896 1.072714 1.045896 1.045896 1.074949 1.072714 1.104001
1.072714 1.104001 1.099532 1.104001 1.072714 1.016843 1.019078 1.045896 1.072714 1.104001 1.099532
1.104001 1.05605 1.063867 1.072714 1.086264 1.062498 1.074949 1.083129 1.063867 1.072714 1.086264
1.062498 1.104001 1.110207 1.150126 1.153167 1.115622 1.062498 1.016843 1.028972 1.035114 1.072714
1.056905 1.08906 1.074949 1.05605 0.987791 1.009373 1.086264 1.072714 1.035114 1.110207 1.016843
1.035935 1.056905 1.072714 1.09262 1.110207 1.074949 1.08906 1.115622 1.072714 1.121373 1.083129
1.133054 1.08906 1.086264 1.045896 0.987791 0.965442 1.016843 1.019078 1.045896 1.045896 1.045896
1.019078 1.045896 1.072714 1.045896 1.045896 1.045896 0.99226 0.958738 1.001894 1.00636 1.072714
1.056905 1.009373 0.958738 0.974816 1.009373 1.074949 1.056905 1.045896 0.977607 1.001894 0.99226
0.920101 0.880755 0.884989 0.891348 0.974816 0.965442 1.00636 1.027547 1.072714 1.063867 1.12635
1.110207 1.063867 1.027547 0.99226 0.98281 0.998188 0.99226 0.987791 1.074949 1.045896 1.045896
0.987791 0.938625 0.958738 1.016843 1.12635 1.179985 1.162107 1.153167 1.074949 0.987791 0.884989
0.831353 0.87158 0.938625 0.958738 1.019078 1.074949 1.072714 1.086264 1.009373 0.987791 0.920659
0.891348 0.965442 1.115622 1.168747 1.133054 1.110207 0.977607 0.99226 0.96883 0.929685 0.958738
0.974816 0.977607 1.019078 1.056905 1.009373 0.987791 1.001894 1.016843 0.98281 1.027547 1.045896
1.035114 1.083129 1.045896 1.035935 1.056905 1.019078 0.977607 1.001894 1.045896 1.035935 1.056905
1.019078 1.00636 1.05605 1.045896 1.009373 0.998188 0.938625 0.920101 0.920659 0.929685 0.884989
0.958738 0.965442 0.87158 0.938625 0.929685 0.965442 1.016843 1.12635 1.162107 1.099532 1.045896
1.072714 1.045896 1.045896 0.958738 0.911807 0.842527 0.804535 0.842527 0.858171 0.784422 0.812346
0.776335 0.804535 0.910113 1.009373 1.133054 1.191441 1.121373 1.028972 0.929685 0.849998 0.784422
0.75819 0.632569 0.61681 0.557811 0.531249 0.440377 0.509539 0.517557 0.649877 0.639159 0.796873
```

```
        0.822038  0.858171  0.958738  0.99226   1.045896  1.045896  1.016843  0.965442  0.900633  0.938625  0.900633
        0.938625  0.929685  0.965442  0.929685  0.884989  0.842527  0.7509    0.777718  0.726317  0.668211  0.589993
        0.552001  0.61681   0.645887  0.690623  0.726317  0.75819   0.747582  0.7509    0.733962  0.823436  0.842527
        0.812346  0.862595  0.884989  0.792679  0.87656   0.813475  0.920659  0.862595  0.7509    0.675245  0.637499
        0.522948  0.731112  0.639159  0.637499  0.58717   0.482721  0.431297  0.216626  0.203369  0.132812  0.176151
        0.080454  0.115013  0.027078  0.058105  0.026562  0.088075  0.026818  0.172519  0.162469  0.116211  0.026562
        0         0         0         0         0         0         0         0         0         0         0
        0         0         0         0         0         0         0         0         0         0         0
        0];
forcefourT = [5.400825.083125  5.40082   5.718516  5.40082   5.718516  5.40082   5.718516  6.353906  6.353906  6.036211
        6.036211  6.036211  6.036211  6.036211  6.036211  6.036211  5.40082   5.40082   5.083125  5.40082   5.718516
        5.718516  6.036211  5.718516  5.718516  5.718516  6.036211  7.306992  10.483945 26.051016 15.56707  10.16625
        24.144844 32.404922 22.556367 15.56707  19.379414 16.202461 11.754727 15.56707  28.592578 26.686406 25.415625
        13.343203 21.920977 19.379414 20.650195 25.415625 26.368711 26.686406 30.816445 20.967891 24.144844 30.49875
        35.26418  28.274883 27.957187 20.014805 19.697109 27.004102 19.697109 21.920977 20.3325   25.73332  13.978594
        24.780234 28.910273 32.722617 32.087227 24.144844 24.780234 24.462539 27.639492 24.780234 28.274883 24.462539
        26.051016 34.946484 31.134141 26.686406 34.628789 35.581875 21.285586 30.49875  41.935781 29.227969 23.191758
        33.358008 28.592578 25.09793  25.415625 35.581875 31.134141 26.368711 28.274883 47.971992 48.289687 21.285586
        35.26418  51.466641 27.957187 27.957187 48.925078 38.123438 20.014805 37.170352 41.618086 33.358008 24.144844
        32.087227 32.404922 26.368711 33.993398 44.159648 32.722617 27.957187 40.029609 41.618086 41.618086 29.863359
        45.112734 50.513555 27.004102 38.441133 53.690508 46.701211 30.181055 54.008203 52.737422 33.675703 33.675703
        49.878164 47.654297 36.534961 50.513555 51.466641 29.227969 33.040312 43.841953 55.59668  39.394219 34.946484
        47.336602 47.336602 27.639492 37.170352 36.217266 38.758828 28.274883 43.524258 41.618086 38.123438 33.993398
        47.018906 53.372813 40.347305 49.878164 55.59668  37.170352 36.852656 46.383516 44.795039 47.336602 37.805742
        53.690508 32.722617 42.571172 47.654297 49.878164 24.780234 56.867461 52.102031 46.383516 38.758828 33.040312
        42.888867 47.654297 59.091328 62.903672 44.795039 54.008203 52.737422 57.820547 55.59668  57.820547 60.679805
        40.029609 49.242773 51.784336 59.726719 39.076523 54.008203 60.044414 48.925078 41.300391 38.123438 54.008203
        56.23207  47.971992 58.138242 50.513555 57.502852 58.138242 46.701211 39.394219 44.477344 56.867461 39.394219
        42.888867 58.138242 63.539062 66.080625 44.159648 60.679805 59.726719 58.455937 45.112734 47.336602 47.971992
        31.769531 28.910273 52.419727 61.632891 54.961289 56.549766 53.055117 46.701211 59.726719 60.044414 54.325898
        44.795039 58.455937 59.091328 38.758828 47.336602 47.018906 42.253477 49.878164 46.383516 49.242773 59.091328
        69.257578 61.315195 58.138242 67.033711 51.148945 57.185156 45.43043  49.242773 68.304492 57.502852 72.116836
        62.268281 45.112734 43.524258 29.863359 37.488047 48.925078 41.618086 56.549766 57.185156 47.654297 44.477344
        60.679805 67.033711 48.925078 55.278984 63.856758 68.622187 51.148945 46.383516 62.585977 59.726719 53.055117
        69.257578 66.080625 60.9975   54.008203 60.9975   68.939883 67.351406 62.268281 49.560469 55.59668  62.903672
        56.23207  63.221367 62.903672 69.892969 72.752227 68.622187 83.553867 64.174453 54.961289 72.434531 61.950586
        69.257578 67.669102 55.278984 68.304492 59.409023 73.387617 79.741523 56.549766 48.925078 61.315195 33.675703
        38.123438 35.26418  44.159648 53.690508 67.351406 74.023008 92.131641 75.293789 80.694609 77.517656 74.340703
        83.553867 82.283086 62.903672 61.632891 52.419727 44.477344 56.867461 53.372813 68.939883 73.387617 81.012305
        72.434531 79.106133 78.788438 80.376914 114.370312          73.705313 96.89707  85.142344 63.539062 53.055117
        77.517656 75.293789 85.460039 65.445234 77.199961 60.362109 77.199961 73.069922 67.033711 49.560469 84.824648
        60.362109 73.705313 53.372813 71.16375  76.56457  82.918477 91.49625  77.199961 72.434531 76.56457  73.705313
        96.579375 88.001602 83.871562 72.116836 73.387617 67.351406 75.92918  77.517656 86.09543  81.965391 74.340703
        82.600781 68.939883 57.820547 63.539062 57.820547 76.882266 59.091328 72.752227 65.76293  90.860859
        105.792539          71.481445 90.860859 55.278984 98.167852 82.918477 89.272383 90.543164 93.084727 90.860859
        75.611484 69.575273 67.033711 84.506953 81.33     84.506953 97.850156 84.506953 96.26168  103.250977
        97.214766 43.206562 0         11.437031 2.223867  4.447734  8.895469  11.754727 9.848555  8.577773  12.072422
        9.848555  9.530859  10.801641 9.848555  9.530859  9.530859  9.530859  10.483945 9.530859  8.895469  10.16625
        9.848555  8.895469  9.530859  9.848555  9.213164];
powerfourT = speedfourT.*forcefourT./375;  %converts speed and force to horsepower


%simple moving average of speed
speedfourfilterT = filter(ones(1,n)/n,1,speedfourT);
%delete first set of data to make correct matrix
speedfourfilterT(1:n-1)=[];

%simple moving average of force
forcefourfilterT = filter(ones(1,n)/n,1,forcefourT);
%delete first set of data to make correct matrix
forcefourfilterT(1:n-1)=[];

%simple moving average of power
powerfourfilterT = filter(ones(1,n)/n,1,powerfourT);
%delete first set of date to make correct matrix
powerfourfilterT(1:n-1)=[];

%-----------------------------------------------------------------
%reduce length of timefour to match that of speedfour filtered
timefouravgT = timefourT;
```

```
%delete end of matrix
for i = 1:(n-1)/2
   t = length(timefouravgT);
   timefouravgT(t)=[];
end
%delete beginning of matrix
timefouravgT(1:(n-1)/2)=[];
%_____________________________________________________________________

%_____________________________________________________________________

figure(1)
plot(timeoneavgM,speedonefilterM,'r','Linewidth',2)
hold on
plot(timefouravgT,speedfourfilterT,'k','Linewidth',2)
title('Velocity vs Time: MATILDA, TALON Comparison')
xlabel('Time, (s)'); ylabel('Velocity, (mph)')
legend('MATILDA','TALON')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')


figure(2)
plot(timeoneavgM,forceonefilterM,'r','Linewidth',2)
hold on
plot(timefouravgT,forcefourfilterT,'k','Linewidth',2)
title('Force vs Time:  MATILDA, TALON Comparison')
xlabel('Time, (s)'); ylabel('Force, (lb_f)')
legend('MATILDA','TALON')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')


figure(3)
plot(timeoneavgM,poweronefilterM,'r','Linewidth',2)
hold on
plot(timefouravgT,powerfourfilterT,'k','Linewidth',2)
title('Power vs Time:  MATILDA, TALON Comparison')
xlabel('Time, (s)'); ylabel('Power, (hp)')
legend('MATILDA','TALON')
grid on
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')

% figure(4)
% plot(speedonefilterM,poweronefilterM,'r','Linewidth',2)
% title(['Power vs. Speed: Moving average span of ',num2str(n)])
% xlabel('Speed, (mph)');ylabel('Power, (hp)')
%
% figure(5)
% plot(forceonefilterM,poweronefilterM,'r','Linewidth',2)
% title(['Power vs. Force: Moving Average Span of ',num2str(n)])
% xlabel('Force, lb_f'); ylabel('Power, (hp)')
%
% figure(6)
% plot(forceonefilterM,speedonefilterM,'r','Linewidth',2)
% title(['Speed vs. Force: Moving Average Span of  ',num2str(n)])
% xlabel('Force, (lb_f)'); ylabel('Speed, (mph)')
```

Appendix F – MATLAB Code for MATILDA Theoretical vs. Actual Pulls

```
%12-11-05
%predict theoretical performance of MATILDA and compare it to actual tested performance

clear all
close all
clc
%Matilda data

n = input('What size of moving average? (must be an odd number)\n');

%_____________________________________________________________________
%_____________________________________________________________________
%PULL ONE

timeoneM = [0.0     0.5     1.0     1.5     2.0     2.5     3.0     3.5     4.0     4.5     5.0
            5.5     6.0     6.5     7.0     7.5     8.0     8.5     9.1     9.5     10.0    10.5
            11.0    11.5    12.0    12.5    13.0    13.5    14.0    14.5    15.0    15.5    16.0
            16.5    17.0    17.5    18.0    18.5    19.0    19.5    20.0    20.5    21.0    21.5
            22.0    22.5    23.0    23.5    24.0];
speedoneM = [0      0       0       0       0       0       0       0       0       0       0
            0.3067  1.436266 1.857296 1.897796 1.878048 1.85035 1.85579 1.711466 1.832058 1.760732 1.732784
            1.700902 1.675716 1.541486 1.509702 1.639484 1.618648 1.568844 1.377268 1.228296 1.193234 1.112318
            0.946088 0.854008 0.684812 0.521894 0.295714 0.03163 0.036316 0.020152 0.027124 0       0
            0       0       0       0       0];
forceoneM = [9.848555   9.848555 9.848555 9.848555 9.848555 9.530859 9.848555 9.848555 9.848555 9.848555
            9.848555  30.816445 26.051016 12.072422 15.249375 19.697109 19.061719 33.358008 28.910273 27.004102 27.321797
            32.722617 28.274883 39.711914 34.311094 31.134141 34.946484 28.274883 44.159648 58.455937 48.925078 40.982695
            56.549766 48.289687 44.477344 49.878164 55.278984 58.455937 46.06582  46.06582  58.455937 58.455937
58.455937  58.455937  58.455937 58.455937 58.455937 58.455937 58.455937];
poweroneM = speedoneM.*forceoneM./375;  %converts speed and force to horsepower

%simple moving average of speed
speedonefilterM = filter(ones(1,n)/n,1,speedoneM);
%delete first set of data to make correct matrix
speedonefilterM(1:n-1)=[];

%simple moving average of force
forceonefilterM = filter(ones(1,n)/n,1,forceoneM);
%delete first set of data to make correct matrix
forceonefilterM(1:n-1)=[];

%simple moving average of power
poweronefilterM = filter(ones(1,n)/n,1,poweroneM);
%delete first set of date to make correct matrix
poweronefilterM(1:n-1)=[];


%-----------------------------------------------------------------
%reduce length of timeone to match that of speedone filtered
timeoneavgM = timeoneM;
%delete end of matrix
i = 1;
for i = 1:(n-1)/2
   t = length(timeoneavgM);
   timeoneavgM(t)=[];
end
%delete beginning of matrix
timeoneavgM(1:(n-1)/2)=[];

%_____________________________________________________________________
%_____________________________________________________________________
%PULL TWO

timetwoM = [0.0     0.5     1.0     1.5     2.0     2.5     3.0     3.6     4.0     4.5     5.0
            5.6     6.0     6.5     7.0     7.6     8.0     8.5     9.0     9.6     10.0    10.5
            11.0    11.5    12.0    12.5    13.0    13.5    14.0    14.5    15.0    15.5    16.0
```

```
          16.5      17.0      17.5      18.1      18.5      19.0      19.5      20.0      20.5      21.0      21.5
          22.1      22.5];
speedtwoM = [0      0         0         0         0         0         0         0.54455   1.73484   1.845078  1.893612
          1.708122  2.108662  1.823992  1.784664  1.727342  1.863512  1.785418  1.696744  1.594246  1.734466  1.544594
          1.629024  1.388012  1.484498  1.30737   1.115414  1.006798  1.118996  1.233566  1.312558  1.228296  1.011656
          0.664682  0.669148  0.495536  0.286546  0.01257   0.020752  0.005272  0         0         0         0
          0         0];
forcetwoM = [6.036211         6.036211  6.353906  6.353906  6.036211  6.353906  15.884766 42.571172 21.603281 22.556367
          26.051016 13.025508 19.061719 32.722617 18.108633 24.462539 16.202461 27.957187 28.592578 41.935781 59.091328
          74.976094 47.654297 20.650195 39.394219 42.253477 28.592578 68.939883 47.971992 40.029609 40.982695 36.534961
          29.545664 60.679805 50.195859 51.784336 45.43043   47.971992 56.867461 56.867461   56.867461   56.867461
56.867461   56.867461   56.867461   56.867461];
powertwoM = speedtwoM.*forcetwoM./375;  %converts speed and force to horsepower

%simple moving average of speed
speedtwofilterM = filter(ones(1,n)/n,1,speedtwoM);
%delete first set of data to make correct matrix
speedtwofilterM(1:n-1)=[];

%simple moving average of force
forcetwofilterM = filter(ones(1,n)/n,1,forcetwoM);
%delete first set of data to make correct matrix
forcetwofilterM(1:n-1)=[];

%simple moving average of power
powertwofilterM = filter(ones(1,n)/n,1,powertwoM);
%delete first set of date to make correct matrix
powertwofilterM(1:n-1)=[];


%-----------------------------------------------------------------
%reduce length of timetwo to match that of speedtwo filtered
timetwoavgM = timetwoM;
%delete end of matrix
i = 1;
for i = 1:(n-1)/2
   t = length(timetwoavgM);
   timetwoavgM(t)=[];
end
%delete beginning of matrix
timetwoavgM(1:(n-1)/2)=[];
%_____________________________________________________________________________
%_____________________________________________________________________________
%PULL THREE

timethreeM = [0.0    0.4       0.9       1.4       1.9       2.4       2.9       3.4       3.9       4.4       4.9
          5.4       5.9       6.4       6.9       7.4       7.9       8.4       8.9       9.4       9.9       10.4
          10.9      11.4      11.9      12.5      12.9      13.4      13.9      14.4      14.9      15.4      15.9
          16.4      16.9      17.4      17.9      18.4      18.9      19.4      19.9      20.4      20.9      21.4
          22.0      22.4      22.9      23.4      23.9      24.4      24.9      25.4      25.9      26.4      26.9
          27.4      27.9      28.4      28.9      29.4      29.9      30.4];
speedthreeM = [0      0         0         0         0         0         0         0         0         0         0
          0         0         0         0.05188   1.11759   1.805416  1.918882  1.929928  1.91361   1.8988    1.855622
          1.857296  1.829264  1.652438  1.845032  1.748348  1.68693   1.654964  1.570952  1.509702  1.452634  1.33067
          1.296826  1.182858  1.096504  1.115414  1.119848  0.983764  0.915422  0.847322  0.892332  1.038516  1.055952
          0.964186  0.70113   0.814512  0.906724  0.53955   0.453362  0.088196  0.052716  0.041504  0.026412  0.072488
          0.04754   0         0         0         0         0         0];
forcethreeM = [3.494648         3.494648  3.494648  3.494648  3.494648  3.494648  3.494648  3.494648  3.494648  3.176953
          3.176953  3.494648  3.494648  3.494648  50.83125  34.946484 26.686406 4.76543   15.249375 17.790937 22.556367
          27.321797 34.311094 20.967891 18.108633 21.603281 30.816445 29.863359 26.686406 33.993398 36.852656 44.477344
          46.06582  55.59668  40.982695 34.311094 59.726719 59.409023 51.148945 34.311094 60.679805 45.748125 52.419727
          58.773633 60.044414 48.607383 28.274883 55.914375 39.076523 79.423828 38.441133 88.001602 54.961289 66.080625
          82.600781 66.080625   66.080625   66.080625   66.080625   66.080625   66.080625   66.080625];
powerthreeM = speedthreeM.*forcethreeM./375;  %converts speed and force to horsepower

%simple moving average of speed
speedthreefilterM = filter(ones(1,n)/n,1,speedthreeM);
%delete first set of data to make correct matrix
speedthreefilterM(1:n-1)=[];
```

```
%simple moving average of force
forcethreefilterM = filter(ones(1,n)/n,1,forcethreeM);
%delete first set of data to make correct matrix
forcethreefilterM(1:n-1)=[];

%simple moving average of power
powerthreefilterM = filter(ones(1,n)/n,1,powerthreeM);
%delete first set of date to make correct matrix
powerthreefilterM(1:n-1)=[];


%-----------------------------------------------------------------
%reduce length of timethree to match that of speedthree filtered
timethreeavgM = timethreeM;
%delete end of matrix
i = 1;
for i = 1:(n-1)/2
   t = length(timethreeavgM);
   timethreeavgM(t)=[];
end
%delete beginning of matrix
timethreeavgM(1:(n-1)/2)=[];

%_____________________________________________________________________________
%_____________________________________________________________________________




voltage = 6;

theoretical_torque = linspace(0,533.4*voltage/12.8,38); %units of mN-m, 1 motor

%gear reduction and unit changes
theoretical_force = 2.*theoretical_torque.*134.17.*.0030757; %units of lbf, including gearbox adduction and 3" moment arm

theoretical_speed = ((-19947*voltage/12.8)/(533.4*voltage/12.8)).*theoretical_torque + 19947*voltage/12.8; %rev/min
speed_mph = theoretical_speed.*1.508.*60./(134.17*5280); %units of mph, after 134.17 gear reduction and 1.508 ft/rev conversion
theoretical_power = 2.*theoretical_torque.*theoretical_speed./7120910.464; %convert to horsepower


speedonefilterM = speedonefilterM./2;
speedtwofilterM = speedtwofilterM./2;
speedthreefilterM = speedthreefilterM./2;

poweronefilterM = poweronefilterM./2;
powertwofilterM = powertwofilterM./2;
powerthreefilterM = powerthreefilterM./2;

cut1 = 14;
cut2 = 9;
cut3 = 17;

%PLOT TOW FORCE VS SPEED
figure(1)
a = plot(theoretical_force,speed_mph,'m');
set(a,'LineWidth',2)
hold on
plot(forceonefilterM(cut1:length(forceonefilterM)),speedonefilterM(cut1:length(speedonefilterM)),'r','Linewidth',2)
hold on
plot(forcetwofilterM(cut2:length(forcetwofilterM)),speedtwofilterM(cut2:length(speedtwofilterM)),'k','Linewidth',2)
hold on
plot(forcethreefilterM(cut3:length(forcethreefilterM)),speedthreefilterM(cut3:length(speedthreefilterM)),'b','Linewidth',2)
title({'Experimental vs. Theoretical Performance of MATILDA';'Plot of Tow Force vs. Speed'})
xlabel('Tow Force, (lb_f)')
ylabel('Speed, (mph)')
grid on
legend('Theoretical','Pull One','Pull Two','Pull Three')
set(get(gca, 'xlabel'),'FontWeight','bold')
```

```
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
%set(gca,'ylim',[0 4.5])


% %PLOT POWER VS SPEED
figure(2)
b = plot(speed_mph,theoretical_power,'m');
set(b,'LineWidth',2)
hold on
plot(speedonefilterM(cut1:length(speedonefilterM)),poweronefilterM(cut1:length(poweronefilterM)),'r','Linewidth',2)
hold on
plot(speedtwofilterM(cut2:length(speedtwofilterM)),powertwofilterM(cut2:length(powertwofilterM)),'k','Linewidth',2)
hold on
plot(speedthreefilterM(cut3:length(speedthreefilterM)),powerthreefilterM(cut3:length(powerthreefilterM)),'b','Linewidth',2)
title({'Experimental vs. Theoretical Performance of MATILDA';'Plot of Power vs. Speed'})
xlabel('Speed, (mph)')
ylabel('Power, (hp)')
grid on
legend('Theoretical','Pull One','Pull Two','Pull Three')
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')

end1 = 4;
end2 = 4;
end3 = 4;
%PLOT POWER VS FORCE
figure(3)
c = plot(theoretical_force,theoretical_power,'m');
set(c,'LineWidth',2)
hold on
plot(forceonefilterM(1:length(forceonefilterM)-end1),poweronefilterM(1:length(poweronefilterM)-end1),'r','Linewidth',2)
hold on
plot(forcetwofilterM(1:length(forcetwofilterM)-end2),powertwofilterM(1:length(powertwofilterM)-end2),'k','Linewidth',2)
hold on
plot(forcethreefilterM(1:length(forcethreefilterM)-end3),powerthreefilterM(1:length(powerthreefilterM)-end3),'b','Linewidth',2)
title({'Experimental vs. Theoretical Performance of MATILDA';'Plot of Power vs. Tow Force'})
xlabel('Tow Force, (lb_f)')
ylabel('Power, (hp)')
grid on
legend('Theoretical','Pull One','Pull Two','Pull Three')
set(get(gca, 'xlabel'),'FontWeight','bold')
set(get(gca, 'ylabel'),'FontWeight','bold')
set(get(gca, 'title'),'FontWeight','bold')
```

References

[1] Gombar, B., Terwelp, C., Fleming, M., Wicks, A. L. and Reinholtz, C.F., "JOUSTER: Joint Unmanned Systems Test, Experimentation and Research Facility," *International Test and Evaluation (ITEA) Journal*, December 2004/January 2005, pp. 9-13.

[2] Weisman, Robert, "Working on next generation of robot warriors," *http://www.boston.com/business/globe/articles/2004/04/13/working_on_next_generation_of_robot_warriors/* (Boston, MA: The Boston Globe, April 13, 2004).

[3] Yamauchi, Brian, "PackBot: A Versatile Platform for Military Robotics," Proceedings of SPIE Vol. 5422: Unmanned Ground Vehicle Technology VI, April 12-16, 2004, Orlando, FL.

[4] Yamauchi, Brian, "The Robot Gallery," *http://robotfrontier.com/gallery.html* (Burlington, MA).

[5] Navy Spawar Website http://robot.spawar.navy.mil/home.asp?item=browse&cat=plat&table=UGV.

[6] Mesa-Robotics, "Performance Specifications and Features – MATILDA Robotic Platform," http://www.mesa-robotics.com/mat_perf.pdf (Madison, AL: Mesa Robotics, 5-20-05).

[7] Grabianowski, Edward, "How Military Robots Work," http://science.howstuffworks.com/military-robot4.htm.

[8] Foster-Miller, Inc., "TALON Robots," http://www.foster-miller.com/literature/documents/TALON_Brochure.pdf (Foster-Miller, Inc.: Waltham, MA, 2005).

[9] Hauesian, Brooke, Mobility Analysis of Small, Lightweight Robotic Vehicles.

[10] http://www.dynamometer.fsnet.co.uk/chassis-dynamometer.htm

[11] http://www.atc.army.mil/fac_guide/facilities/dynamometer.html

[12] http://www.superflow.com/towdyn/td1200.html

[13] http://www.kong.net/VidSat/pix/truck11.jpg

[14] Haueisen, Hudas, Gorsich, Ahlvin, Jones, Priddy, Mason, Hulbert, Case Study of the Evaluation and Verification of a PackBot Model in NRMM. Technical Report 2005-05MV-58

[15] U.S. Army Aberdeen Proving Ground, “Test Operations Procedure “Drawbar Pull,” TOP 2-2-604 (Aberdeen, MD: U.S. Army, July 18, 1980).

[16] Hanamoto, B. *Effects of Variation in Drawbar Hitch Location on Vehicle Performance, Special Report 237*,(Cold Regions Research and Engineering Laboratory, Hanover, NH, September 1975).

[17] Shigley, J.E., Mischke, C.R., *Mechanical Engineering Design*, McGraw Hill, 2001, p. 1185