A PARAMETRIC STUDY EVALUATING VARIOUS FABRICS TO MINIMIZE DRAG EFFECTS ON A SPRINTER

by

Brendan Daly

and

Michael Davis

Submitted in partial fulfillment of the

requirements for Departmental Honors in

the Department of Engineering

Texas Christian University

Fort Worth, Texas

May 2, 2016

A PARAMETRIC STUDY EVALUATING VARIOUS FABRICS

TO MINIMIZE DRAG EFFECTS ON A SPRINTER

Project Approved:

Supervising Professor: Robert R. Bittle, Ph.D.

Department of Engineering

Walt Williamson, Ph.D.

Department of Engineering

Efton Park, Ph.D.

Department of Mathematics

Abstract

This report describes each step of the parametric study conducted on various fabrics to minimize the drag force experienced by a sprinter. This study includes wind tunnel testing to characterize the aerodynamics drag of each tested fabric; the creation of a mathematical model of Usain Bolt sprinting his world record 100-meter time using cylindrical approximations for the parts of the body; and the application of the drag test data to the mathematical model to characterize the performance of each fabric should it be worn by a sprinter. The purpose of this study is to create the methodology that should be used to conduct similar drag optimization studies on a larger scale in the future. The wind tunnel testing provides drag measurement and the body analysis of Usain Bolt sprinting provides geometric, time and velocity data for his running type. The application of the data with the model yields the best and worst performance fabrics over various parts of his body. The result of this parametric study is reasonable, suggesting the tighter, thinner fabrics with slight surface imperfections provide the least drag force and highest performance, whereas the thicker, smoother, more porous fabrics in addition to the smooth test cylinder yield the highest drag force on a runner and the worst performance. The difference between the best and worst fabrics yields a significant difference in performance and consequentially the time it would take a runner to run a race. This study methodology would be applicable to larger scale studies introducing more complex models and more fabrics.

Table of Contents

Abstract **3**

Table of Figures **6**

Table of Tables **8**

Theory and Background **9**

Wind Tunnel Testing Principles 9

Comparison of TCU and Lockheed Martin Wind Tunnel 10

Drag 10

Reynolds Number 11

Effect of Reynolds Number on Drag Coefficient 12

Materials **13**

..... **15**

Procedures **16**

Lockheed Martin Testing- 16

TCU Testing 17

Analysis **18**

Wind Tunnel Data 18

Mathematical Sprinter Body Model 19

Static Model Construction 19

Biomechanical Modeling 22

Application of Test Data to Model 26

Results **28**

Lockheed Martin Testing Results 28

TCU Wind Tunnel Test Results 30

Drag Modeling Results 31

Lockheed Testing 34

Performance Comparison of Best to Worst Fabrics 34

Conclusion **35**

Appendix A **36**

Appendix B **41**

Aspect Ratio Calculator 41

Body Modeling and Analysis 44

Drag Coefficient Extraction Function..... 61
Frame Extraction Function..... 62
Scaling Vector Function 64
Velocity Component of Model Function 65
Excel Column Converter Function (Downloaded from free function site)..... 67

Table of Figures

Figure 1 – Wind tunnel tests are used for a variety of experimental modeling with the goal of optimizing a body's performance traveling through a fluid. 9
Figure 2 – The closed circuit wind tunnel (pictured left) is the used at TCU. The open circuit wind tunnel (pictured right) is the type used at L-M 10
Figure 3 – The arrows represent flow lines within a fluid. The laminar flow has clean, uniform flow lines with constant direction and magnitude. The turbulent flow has tumbling flow lines of various direction and magnitude. 11
Figure 4 – Surface roughness can reduce drag coefficients at lower Reynolds numbers. 12
Figure 5- Fabric 1 – smooth, thin fabric with raised notched (white) on the top and bottom of the cylinder. 13
Figure 6 - Fabric 4 – Tight, thick, neoprene material. Seam places downstream. 14
Figure 7 - Fabric 3 – tight, thin, glossy fabric with raised white bumps covering all surface but seam. Seam placed on downstream side of cylinder. 14
Figure 8 - Fabric 2 – tight, medium thickness material with raised, white pattern over entire surface. Seam on downstream side of cylinder. 14
Figure 9 - Fabric 7 – tight, thin material with dimples in fabric, similar to a golf ball. Loose knit. 15
Figure 10 - Fabric 6 – tight, smooth, medium thickness, semi-glossy material. Similar to fabric 4 but slightly thinner with a glossier finish. 15
Figure 11 - Fabric 5 – Tight, thin material with raised, stitched, frills in concentric rings down the length of the cylinder 15
Figure 12 - Smooth cylinder setup. 16
Figure 13 – The test piece used in the TCU wind tunnel for confirmation of data obtained at L-M. 17
Figure 14- This is the image used to create a scaled static model of Usain Bolt 20
Figure 15 – This figure shows the relative size of each cylinder used in the model. They are color coodinated to correspond to the annotated images found in Figure 17, Figure 18, and Appendix A..... 21
Figure 16 – Frame 555 (pictured left) is the first frame of one stride cycle. Frame 710 (pictured right) is the last frame of the same cycle. Notice the simimlarities in the position of each limb. 22
Figure 17 - Frame 1, Annotated. 23
Figure 18 - Frame 61, Annotated. 23
Figure 19 - Drag Data at 40 feet per second 28
Figure 20 - Drag Data at 60 feet per second 28
Figure 21 - Drag Data at 80 feet per second 29
Figure 22 - Drag Data at 100 feet per second 29

Figure 23 – TCU wind tunnel setup. 30
Figure 24 – This plot contains a variety of drag coefficient data versus Reynolds number for comparison. The experimental drag data with the finite cylinder has consistently lower drag coefficients than the infinite cylinder. The TCU data is taken at smaller Reynolds numbers than the L-M data due to the size of the cylinder but still produce comparable drag coefficients. 31
Figure 25 – Bar graphs illustrating the best and worst fabrics, by net drag force, over one stride cycle for each segment on the limbs. 33
Figure 26 - Annotated Frame 1 36
Figure 27 - Annotated Frame 46 37
Figure 28 - Annotated Frame 31 37
Figure 29 - Annotated Frame 16 37
Figure 30 - Annotated Frame 91 38
Figure 31 - Annotated Frame 76 38
Figure 32 - Annotated Frame 61 38
Figure 33 - Annotated Frame 136 39
Figure 34 - Annotated Frame 121 39
Figure 35 - Annotated Frame 106 39
Figure 36 - Annotated Frame 156 40

Table of Tables

Table 1 – These dimensions are the result of the scaled approximation. These are the dimensions used to create the mathematical model of Usain Bolt sprinting. 20

Table 2 – The data is reduced from numbers between Min Alpha and Max Alpha to the number in Reduced Alhpa. 26

Table 3 - The data is reduced from numbers between Min Vel. and Max Vel. to the number in Reduced Vel. 26

Table 4 - This table shows the fabric numbers for the minimized and maximized drag at each segment. The first segment is the one corresponding to the region closes to the torso of the body and the fourth segment corresponds to the segment closest to the end of the lower limbs (hands on the arms and feet on the legs). 32

Table 5 - Shows the measured angle data off the attached annotated frames (also included in Appendix A). 36

Theory and Background

The performance of each fabric can be assessed by measuring the drag coefficients for each material at varying speeds and angles.

Wind Tunnel Testing Principles

Wind tunnel testing originated at the turn of the twentieth century to test the aerodynamic properties of early aircraft. Presently wind tunnel testing is used to analyze the aerodynamic performance of any object moving through fluids ranging from automobiles to baseballs. The key principle behind this method of testing is that a stationary object with fluid moving around it will experience the same forces as if the object was moving through a stationary fluid. Pressure sensors and force transducers on the experimental model provide the data needed to analyze model performance. Some wind tunnels are large enough to accommodate full size models, but in most cases a scaled down model is used for testing.

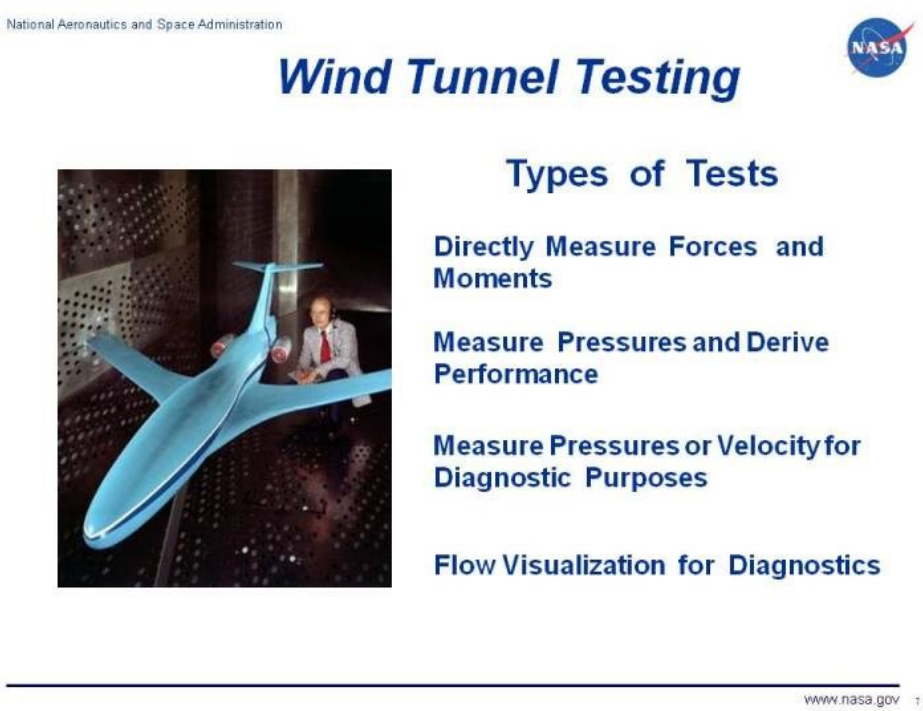

Figure 1 – Wind tunnel tests are used for a variety of experimental modeling with the goal of optimizing a body's performance traveling through a fluid.

Comparison of TCU and Lockheed Martin Wind Tunnel

Testing was performed in both the TCU engineering department wind tunnel and Lockheed Martin low speed wind tunnel. To analyze and compare results from the two wind tunnels one must understand their differences. The TCU wind tunnel is a closed circuit temperature controlled wind tunnel, while the Lockheed wind tunnel uses an open circuit design that does not regulate temperature. The temperature for Lockheed testing was recorded for each test while the TCU wind tunnel was kept at a constant 70°F. Temperature affects aerodynamic testing by changing the density of air which in turn affects the drag experienced by the object. This temperature difference was accounted for appropriately in the comparison TCU and Lockheed Martin data.

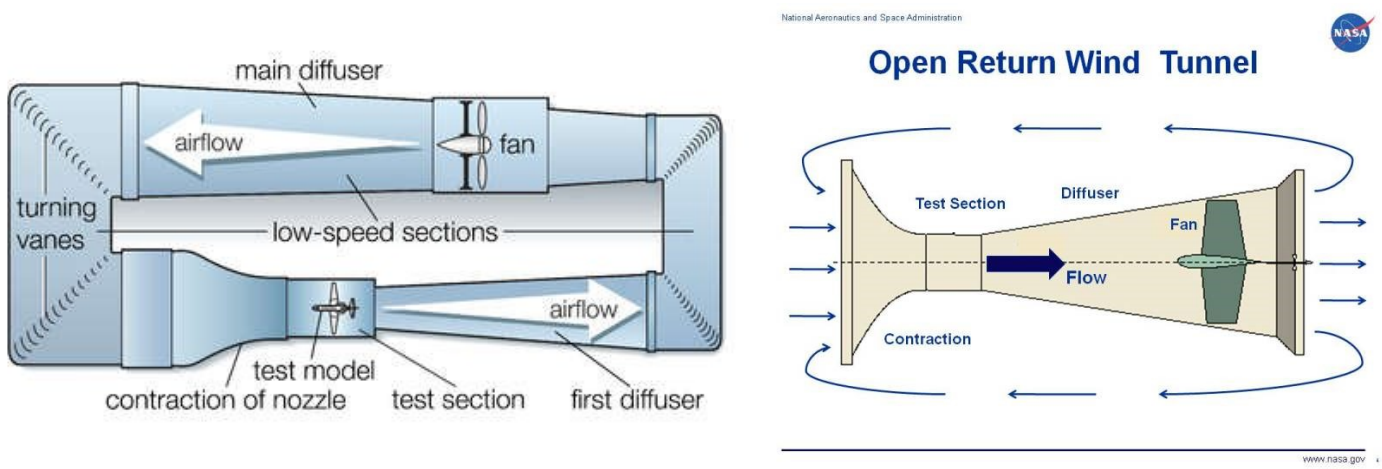

Figure 2 – The closed circuit wind tunnel (pictured left) is the used at TCU. The open circuit wind tunnel (pictured right) is the type used at L-M

Drag

Drag is the force exerted by a fluid on an object as it is moving through the fluid. As the fluid particles collide with the test object they lose momentum, which results in this drag force opposing the motion of the object. Because the drag force opposes motion it will cause the object to slow down; so, minimizing the drag of an object is a common goal for designers and

engineers. The drag force will be dependent on the density of the fluid, cross sectional area, velocity, and the drag coefficient of the object.

$$F_D = \frac{1}{2}\rho C_D A V^2$$

The drag coefficient is determined experimentally through the wind tunnel testing. The drag coefficient for an object will change as speed and orientation change. The optimum track suit would utilize fabrics that experience the lowest drag coefficients for the speed and angle they typically experience in actual use.

Reynolds Number

The Reynolds number is a dimensionless quantity used to compare similar flow situations and to predict fluid behavior. It is a ratio of the inertial forces seen by a fluid to a fluid's internal viscous forces. At low Reynolds numbers the fluid is typically in a laminar flow regime that is dominated by the internal viscous forces. As the Reynolds number increases the inertial forces begin to dominate and the flow becomes more chaotic, and subsequently enters a turbulent flow regime. There is a range of Reynolds numbers where fluids exhibit properties of both laminar

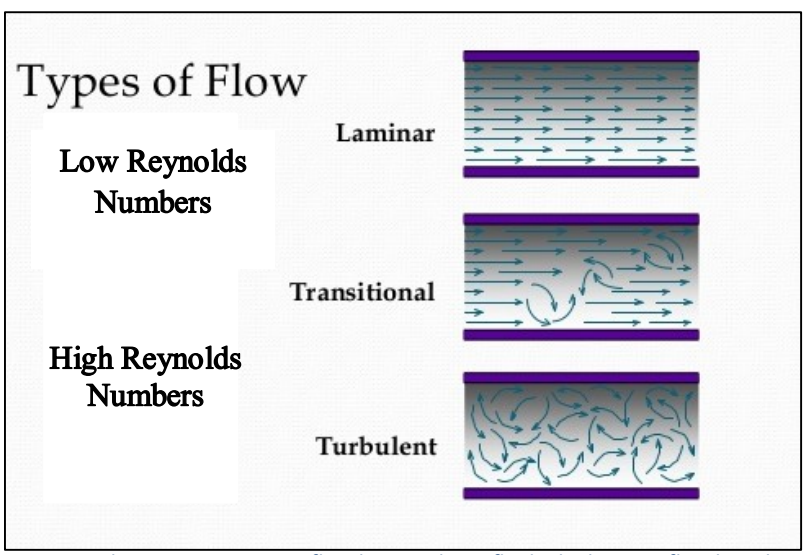

Figure 3 – The arrows represent flow lines within a fluid. The laminar flow has clean, uniform flow lines with constant direction and magnitude. The turbulent flow has tumbling flow lines of various direction and magnitude.

and turbulent flows known as the transitional region. In most practical applications including this study, the flow situations encountered will be turbulent in nature.

Effect of Reynolds Number on Drag Coefficient

The drag coefficient of an object displays a dependence on the Reynolds number and can change significantly as a boundary layer transitions from laminar to turbulent flow. For streamlined objects the transition to turbulent boundary flow causes the drag coefficient to increase, but for blunt objects such as spheres or cylinders there is a decrease in the drag coefficient. The Reynolds number where this transition occurs is a function of the shape of the body (a cylinder in this case) and the surface of the object. By varying the texture and pattern of the fabric, the point where this transition occurs is changed and can significantly affect drag. The graph below shows the effect surface roughness can have in reducing the drag coefficient. A real life example of this principle would be the dimples on a golf ball that cause the transition to turbulent boundary flow to occur much sooner accompanied by a large decrease in the drag coefficient.

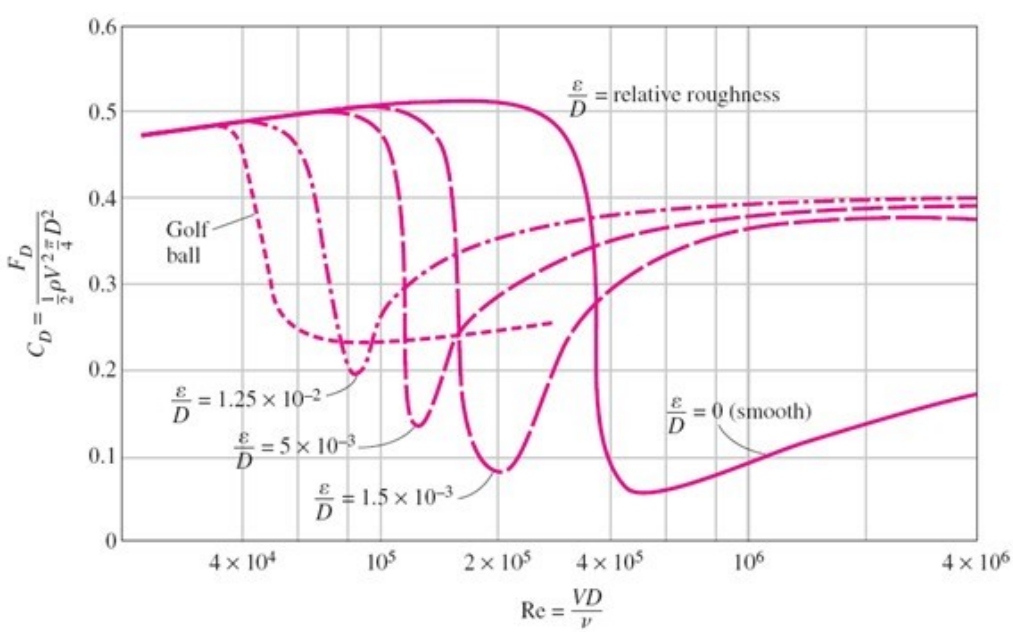

Figure 4 – Surface roughness can reduce drag coefficients at lower Reynolds numbers.

Materials

This test evaluates the drag performance of a variety of fabrics. It does not take into account the friction interactions between various limbs of the body. The fabrics below are a variety of thicknesses, materials, and textures. The primary differences between each fabric are material, which provides various porosities, how tight the fabric is, its base texture, and how the seam is fused or stitched, as well as additive texture, which is created by adding material in various patterns on the outside of the sleeve. See Fabric 1 (Figure 5 below, has notches protruding lined up with the crossflow orientation on the top and bottom. Fabric 2 has a geometric maze pattern of raised soft plastic material. Fabric 3 has raised plastic bumps covering its surface, like the opposite of a surface of a golf ball.

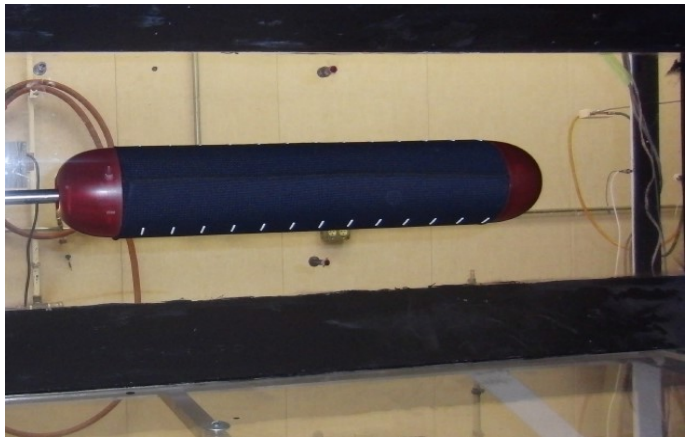

Figure 5- Fabric 1 – smooth, thin fabric with raised notched (white) on the top and bottom of the cylinder.

Figure 8 - Fabric 2 – tight, medium thickness material with raised, white pattern over entire surface. Seam on downstream side of cylinder.

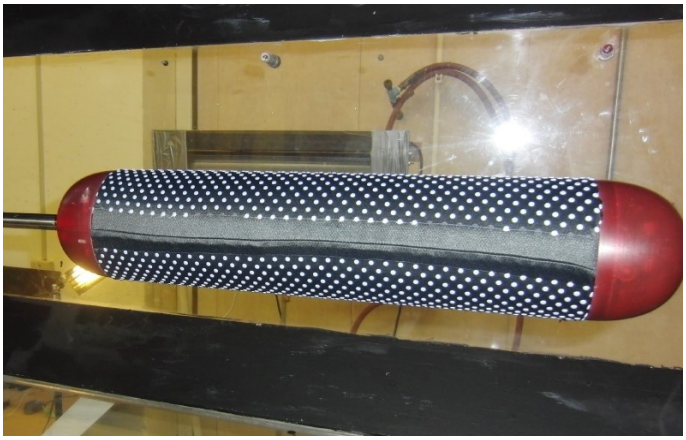

Figure 7 - Fabric 3 – tight, thin, glossy fabric with raised white bumps covering all surface but seam. Seam placed on downstream side of cylinder.

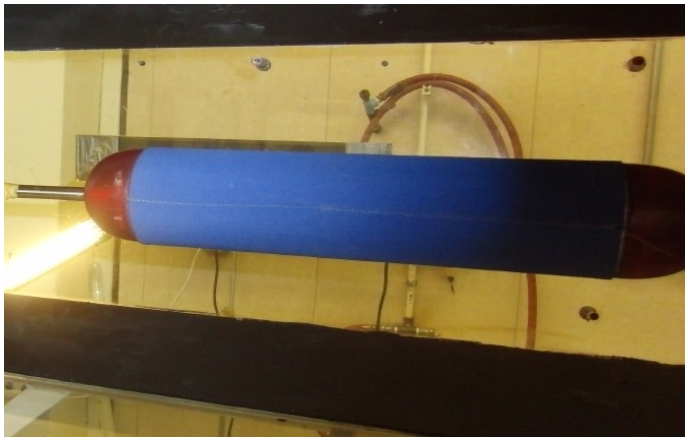

Figure 6 - Fabric 4 – Tight, thick, neoprene material. Seam places downstream.

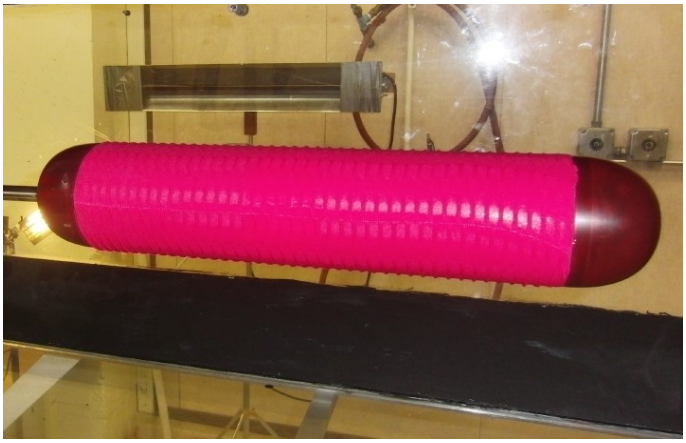

Figure 11 - Fabric 5 – Tight, thin material with raised, stitched, frills in concentric rings down the length of the cylinder

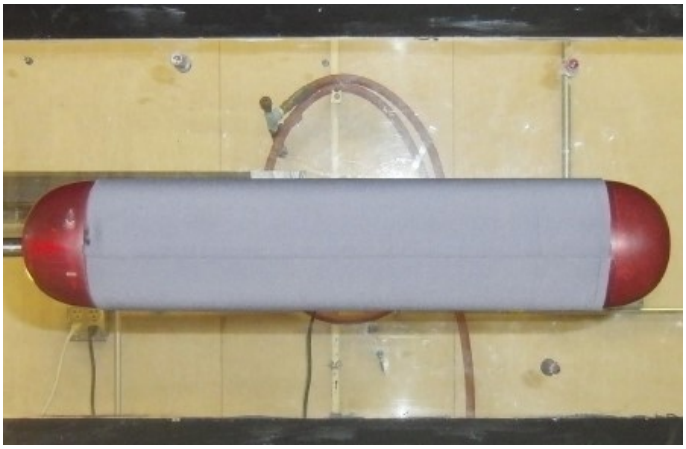

Figure 10 - Fabric 6 – tight, smooth, medium thickness, semi-glossy material. Similar to fabric 4 but slightly thinner with a glossier finish.

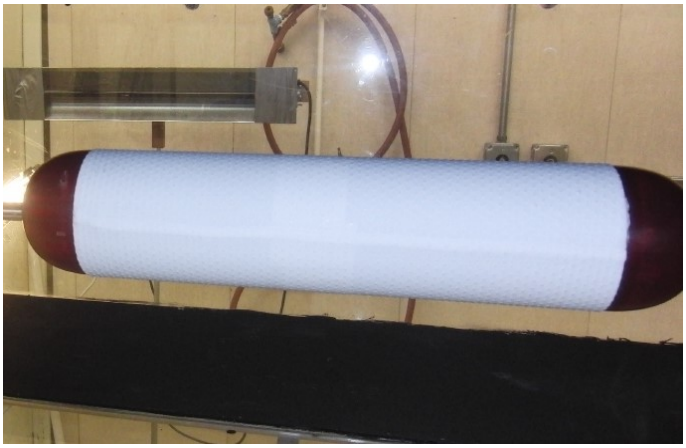

Figure 9 - Fabric 7 – tight, thin material with dimples in fabric, similar to a golf ball. Loose knit.

Procedures

Lockheed Martin Testing-

The goal of Lockheed Martin (L-M) testing was to obtain data for each fabric to compare performance for the body model.

1. L-M wind tunnel was equipped with a cylinder containing a force transducer.
2. Fabric samples were selected and prepped for installation on the cylinder
3. Each test run consisted of four tests where the wind tunnel was at speeds of 35feet per second (fps), 50fps, 70fps, and 90 fps.
4. For each speed the cylinder was rotated from a 90° (cross flow position) to a 30° position in 15° increments while the force transducer recorded data
5. The cylinder was first tested with no fabric to provide a smooth cylinder reference condition
6. Then the first test fabric was installed on the cylinder and data was recorded over the angle sweep for each of the four speeds
7. The process in step 6 was repeated to gather data for the remaining six test fabrics.

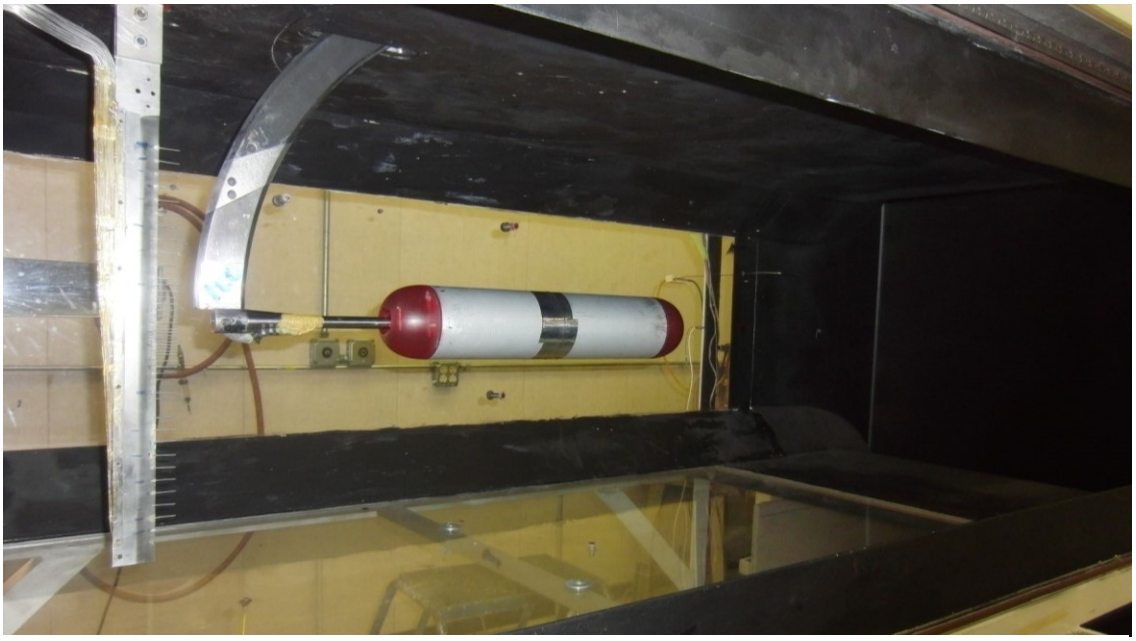

Figure 12 - Smooth cylinder setup.

TCU Testing

The goal of TCU testing was to provide validation of the Lockheed Martin data and better understand the end effects of the cylinder.

1. A cylinder was designed to have the same aspect ratio as the L-M test cylinder and be mounted in the TCU wind tunnel in cross flow (90°) position
2. The cylinder was installed in the TCU wind tunnel by attaching it to a force transducer

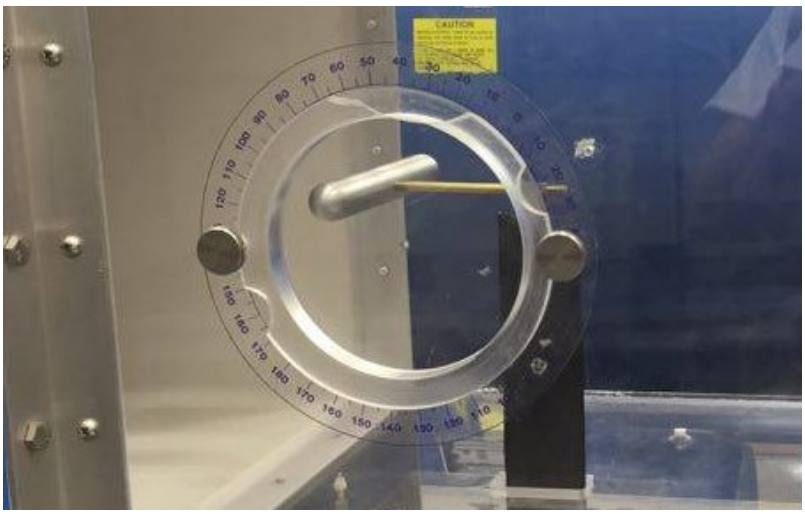

Figure 13 – The test piece used in the TCU wind tunnel for confirmation of data obtained at L-M.

3. The cylinder was tested at the same four speeds as the L-M tests (35, 50, 70, and 90fps)
4. Force transducer data was gathered and analyzed for comparison

Analysis

Wind Tunnel Data

Lockheed's data acquisition system automatically recorded all pertinent data such as Reynolds number and drag coefficients. For the TCU wind tunnel data, further calculations and corrections had to be performed.

To account for the drag seen by the support post in the wind tunnel setup, this drag was recorded separately and defined as a function of the wind tunnel speed. This way the post drag force could be subtracted from the measured drag force. The following post correction was determined experimentally and used in calculations

$$F_{Post} = (5.0856 \times 10^{-5})f^{2}$$

Where F_{Post} was the drag force seen by the post, f was the power frequency of the wind tunnel fan in hertz (Hz), and 5.0856×10^{-5} was an experimentally determined constant. The result was a net drag force on the finite cylinder itself. The Reynolds number for each data point was also calculated using the following equation where, ρ is the density of fluid (air in this case), V is the velocity of the fluid, d is the diameter of the cylinder, and μ is the dynamic viscosity of the fluid.

$$Re = \frac{\rho V d}{\mu}$$

The drag coefficient for each data point was then calculated using the following equation where F_d is the net drag force on the object, and A is the cross sectional area of the cylinder (in the longitudinal direction). The net drag force was calculated using the measured drag force and the drag force from the post.

$$F_d = F_{measured} - F_{Post}$$

$$C_D = \frac{2F_d}{\frac{1}{2}\rho V^2 A}$$

Once calculated, this data could be used further analyzed in the mathematical sprinter body model.

Mathematical Sprinter Body Model

A mathematical model of a sprinter was created using MATLAB based on Usain Bolt sprinting at a steady state and maximum speed. Bolt runs at this pace and gait for 90% of a 100m sprint – this means that our results will apply to the majority of his time in the race creating the potential for very significant results. His body is approximated by a series of cylinders representing various major body parts. The local velocities and flow angles are used to identify the corresponding experimental test coefficients for each condition and fabric. The net drag force is them computed for each discrete part of the body and each fabric type. The output of the program is the best and worst fabrics for each part of the body based upon their drag force. Each part of this process will be further explained in the following subsections.

Static Model Construction

First it was necessary to create static approximations of each one of Usain Bolts limbs. The model is broken down into 10 cylinders, each with a specified diameter and length:

1. Head
2. Torso
3. 2 upper legs
4. 2 lower legs
5. 2 upper arms

6. 2 lower arms

The critical dimensions of each cylinder (length and diameter) were measured by ruler from the printed image, then scaled up to life-sized dimensions. Additionally, the median width of each limb was used as the diameter dimension approximation. The length is measured joint to joint.

Figure 14- This is the image used to create a scaled static model of Usain Bolt

The cylinderical dimensions used to approximate each cylinder is listed in the table below. Additionally, the aspect ratio (ratio of cylinder length to diameter) is shown in the far right column.

Body Part	Length (in)	Diameter (in)	Aspect Ratio
Head	10.3	5.9	1.75
Torso	25.1	12.5	2.00
Upper Leg (x2)	15.8	7.4	2.13
Lower Leg (x2)	25.1	5.0	5
Upper Arm (x2)	13.3	3.8	3.46
Lower Arm (x2)	16.2	3.5	4.58

Table 1 – These dimensions are the result of the scaled approximation. These are the dimensions used to create the mathematical model of Usain Bolt sprinting.

Additionally, below is a visual representation of each cylinder (all axes are equal so each cylinder can be compared).

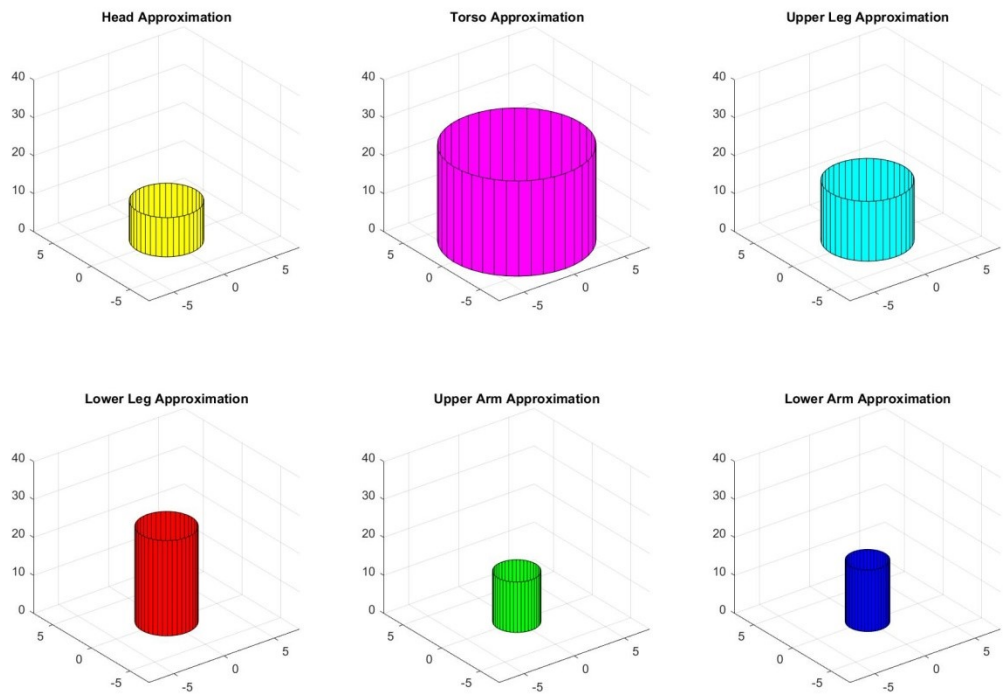

Figure 15 – This figure shows the relative size of each cylinder used in the model. They are color coodinated to correspond to the annotated images found in Figure 17, Figure 18, and Appendix A.

.

The head cylinder and torso cylinder will no longer be considered in the parametric study both because of their aspect ratios and geometry being significantly different than the scope of our experimental data and also because their relative angles remain constant (addressed later in *Biomechanical Modeling* subsection). Notice the relative size of each cylinder and visualize how each cylinder will be connected to form a body.

For simplicity in the model, each cylinder is treated as if it has two free ends, although in reality there is a different interaction at joints, with only the hands and feet being free end approximations.

Biomechanical Modeling

The biomechanical analysis of Usain Bolt utilizes discretized angle and geometry data to approximate local velocities along each limb. The end result is a mathematical model that will allow velocity and time points to be numerically integrated over the time span of the race. The result is net drag force for each fabric for each discrete segment.

The initial video of Bolt sprinting was cropped into one stride cycle (this is one complete cycle for his body to return to its starting positon at steady state, evaluated of the vertical position of his left arm). The frames below show the first and last frames of one stride cycle:

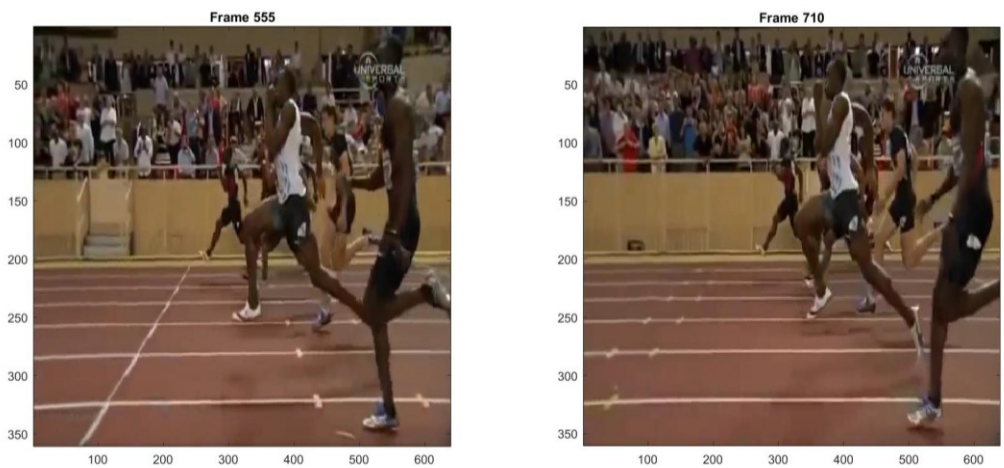

Figure 16 – Frame 555 (pictured left) is the first frame of one stride cycle. Frame 710 (pictured right) is the last frame of the same cycle. Notice the simimlarities in the position of each limb.

The left frame, frame 555 of the original video, is the first frame of the cycle, and the right frame, frame 710 of the original video, is the last frame of one cycle. From here on, the only video analyzed will be frames 555 through 710 (here on referred to as frames 1 through 156).

After isolating the video for just one stride cycle, every fifteenth frame was selected to be annotated and have angles measured; the last frame was selected for analysis as well. The angle

data for each cylindrical approximation was obtained from these images by drawing lines representing the angular positions of each analyzed limb. The colors correspond to the cylindrical approximations seen in Figure 15, previously discussed.

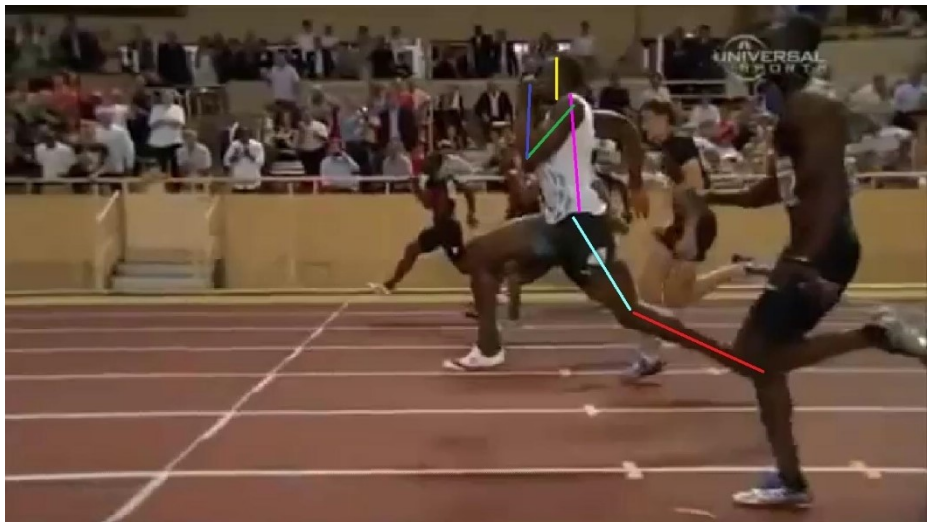

Figure 17 - Frame 1, Annotated.

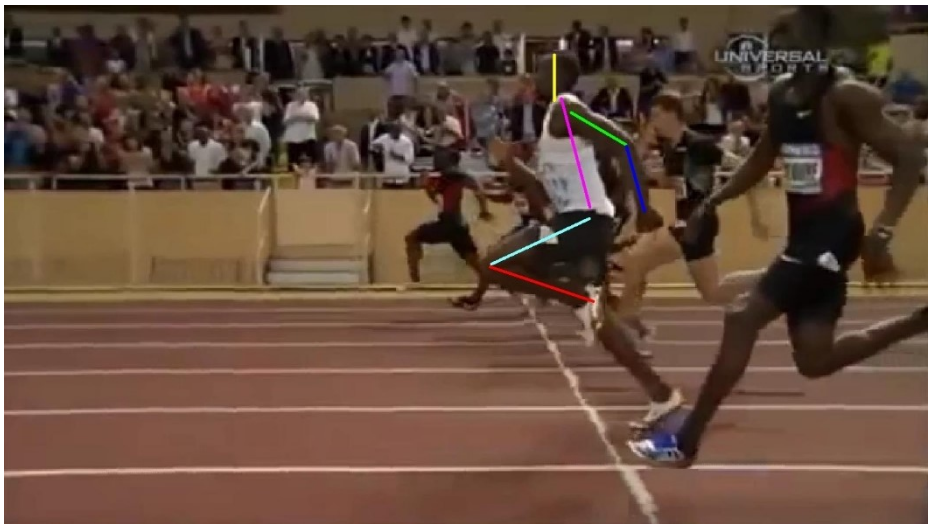

Figure 18 - Frame 61, Annotated.

The absolute angle of each body part was measured from each annotated image, with 0° being directly horizontal and pointed in the direction of net body motion (with the exception of the

head which is measured at a constant 90°), and positive angles being in the direction of the right hand rule (e.g. directly pointing down is 90°, pointing directly back is 180°, and straight up is 270°). See the recorded angle data in Conclusion

This study began with wind tunnel testing at Lockheed Martin's low speed wind tunnel. A cylinder was fitted with several fabrics and the drag force was measured over a range of angles and speeds. A separate test was performed in the TCU engineering wind tunnel to validate the results of this first data set. Once validated, this drag data was incorporated into a body model of a sprinter that could characterize the performance of each fabric for the arms and legs of a sprinter. Finally, the best and worst fabrics were compared to determine the performance difference.

- Lockheed Martin test results were validated by a similar study in the TCU wind tunnel
- There was a significant drag difference between the best and worst fabric that would have a significant effect on performance
- Different fabrics provided the lowest drag behavior depending on the location on the body

.

A kinematic analysis for Bolt's 100m world record race in combination with the number of strides he takes per race reveals the number of seconds per stride. This number can be divided by the number of frames in one stride cycle to arrive at the elapsed time in each frame, which, for the slow motion video used in this analysis, is 3ms. This number will be used to create a delta time that can be applied between each set of angle data that is a function of the number of frames between each annotated image.

The angle data is combined with the time difference between angle positions to calculate angular velocities of each limb using the equation below, where Δθ is the change in angle from one annotated frame to the next in *radians*, Δt is the time elapsed from one annotated frame to the next in *seondsc*, and ω is the angular velocity in *radians/s*:

$$\omega = \frac{\Delta\theta}{\Delta t}$$

Functions for the approximate velocity at any point along each limb were then created based on simple rotational dynamics equations. These functions produce velocity vectors magnitude (ft/s) and direction (rad) for all desired points and annotated positions. All annotated frames' position data, except the last, are included in the analysis, with the angular velocities applied to the first frame (e.g. the angular velocity data calculated between frame 1 and 16 are applied to frame 1, from frame 16 and 31 to 16, and so on). This approximation method is a common one for applying discrete experimental data to modeling.

With functions created for the velocity along each limb at a given point, each limb was segmented further so that fabric data was applied to smaller regions and provided higher granularity outputs. Each of these regions with provide an area over which drag data can be numerically integrated. For improved resolution around the joints the regions are reduced to half the size of those not touching a joint (the joints are the shoulder, elbow, end of the lower arm, the hip, the knee, and end of the lower leg).

Each limb was broken into 4 regions - the end two regions being 1/6 the total length and the middle two being 1/3 the total length. With the dynamic model created and discretized to testable sections the final step was to apply the data to the model.

Application of Test Data to Model

The first step in applying the test data to the model was reducing the data to a discrete form that matched that of the model. Within the drag data for each fabric, the test data points were reduced into the following discrete angles and velocities:

Angle of Attack (Alpha) [deg] Data Reductions		
Min Alpha	**Max Alpha**	**Reduced Alpha**
-	37.5	30
37.5	52.5	45
52.5	67.5	60
67.5	82.5	75
82.5	-	90

*Table 2 – The data is reduced from numbers between **Min Alpha** and **Max Alpha** to the number in **Reduced Alhpa.***

Velocity [ft/s] Data Reductions		
Min Vel.	**Max Vel.**	**Reduced Vel.**
-	50	40
50	70	60
70	90	80
90	-	100

*Table 3 - The data is reduced from numbers between **Min Vel.** and **Max Vel.** to the number in **Reduced Vel.***

With the data reduced into a function of the above discrete angles of attack and free stream velocities, the same process is applied to the resultant velocity vectors of the model – each magnitude corresponds to the free stream velocity point from the testing and each angle, theta, corresponds to a test angle of attack. This data correlation applies the experimental drag coefficients to the created model as a function of free stream velocity and angle of attack.

With the test data and model data now correlated, the next step was to calculate a net drag force for each fabric applied to each segment of the body. This required a simple numerical integration of the drag data for corresponding velocity, angle of attack pairs over the area of the cylindrical approximation. Then one more numerical integration of each segment's drag force over the time between each discrete position point. By integrating each segment's drag data over its area then its time the model approximates the net drag force on one segment over one stride cycle with a

given fabric. This drag force was then compared for each fabric to identify the best and worst performing fabrics with relation to aerodynamic drag force.

Results

Lockheed Martin Testing Results

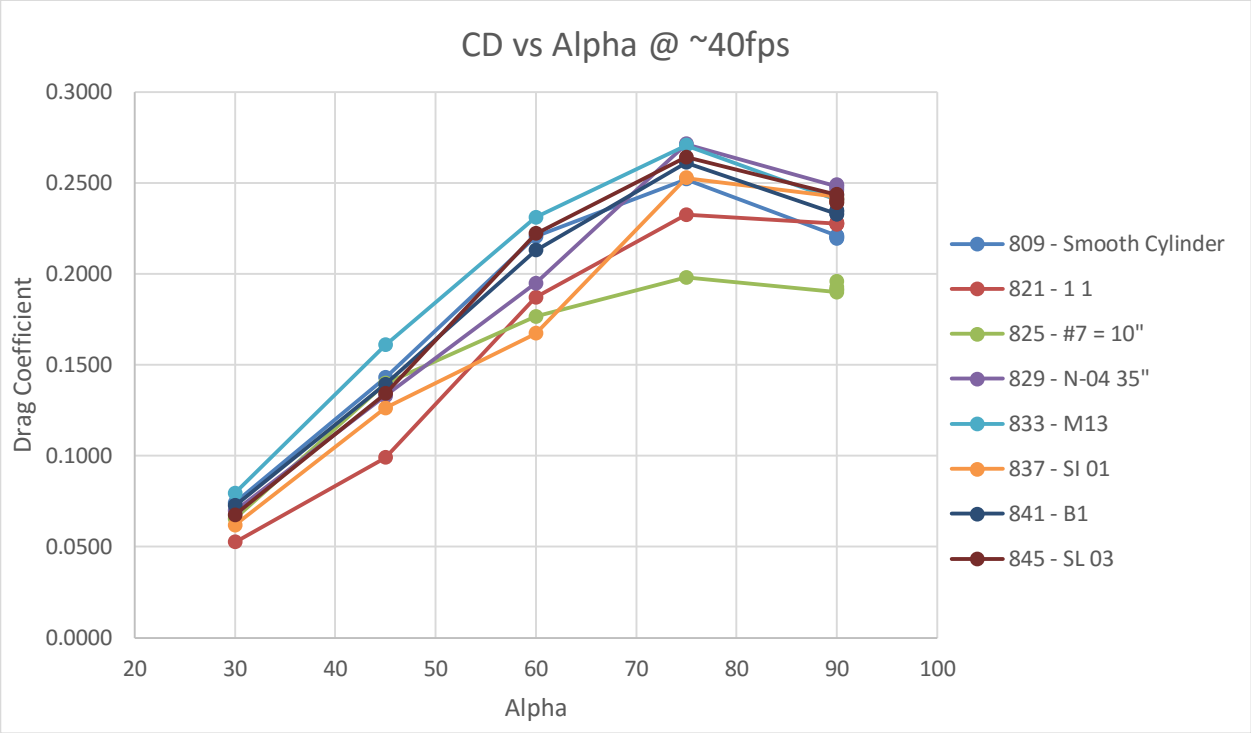

Figure 19 - Drag Data at 40 feet per second

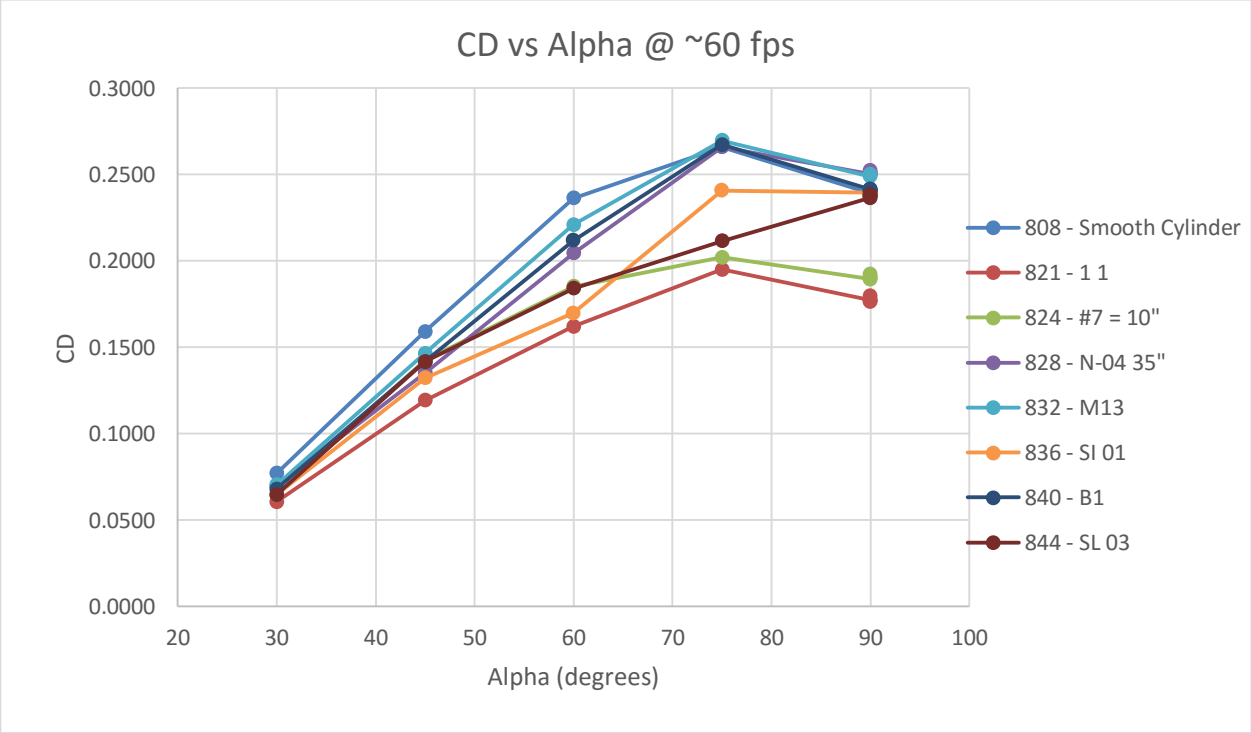

Figure 20 - Drag Data at 60 feet per second

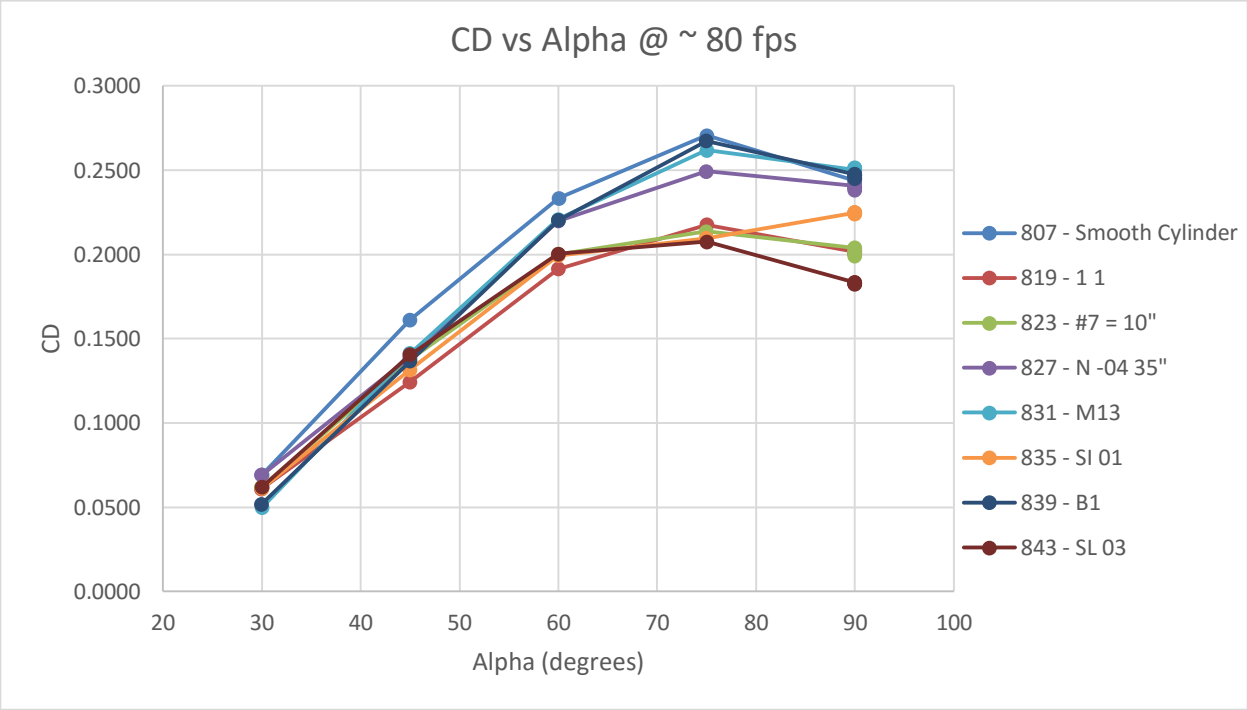

Figure 21 - Drag Data at 80 feet per second

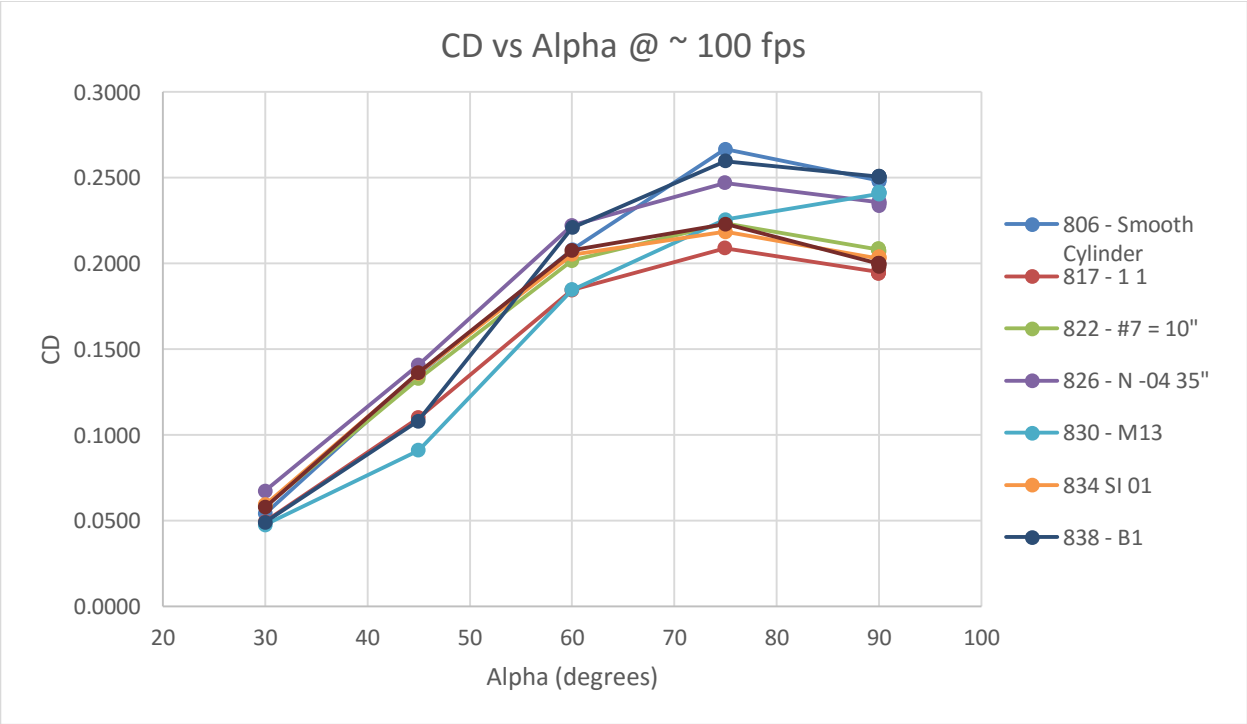

Figure 22 - Drag Data at 100 feet per second

TCU Wind Tunnel Test Results

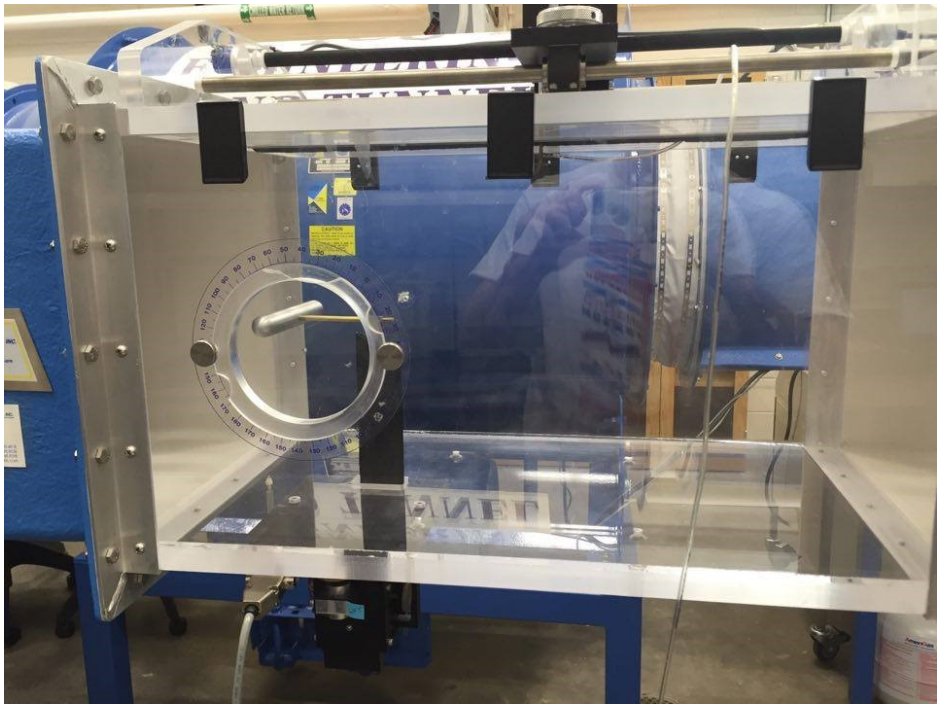

Figure 23 – TCU wind tunnel setup.

The test setup shown in Figure 23 was used for TCU Wind Tunnel testing. The test results from the TCU wind tunnel data are shown in Figure 24. To conduct this test, a smooth surface finite cylinder was used that had the same aspect ratio as the one used during the Lockheed Martin testing. This cylinder was used because the test data from Lockheed did not include data for an infinite cylinder against which comparisons could be made to assess data quality. By gathering our own data on a smooth finite cylinder we could determine if the Lockheed data obtained was

accurate.

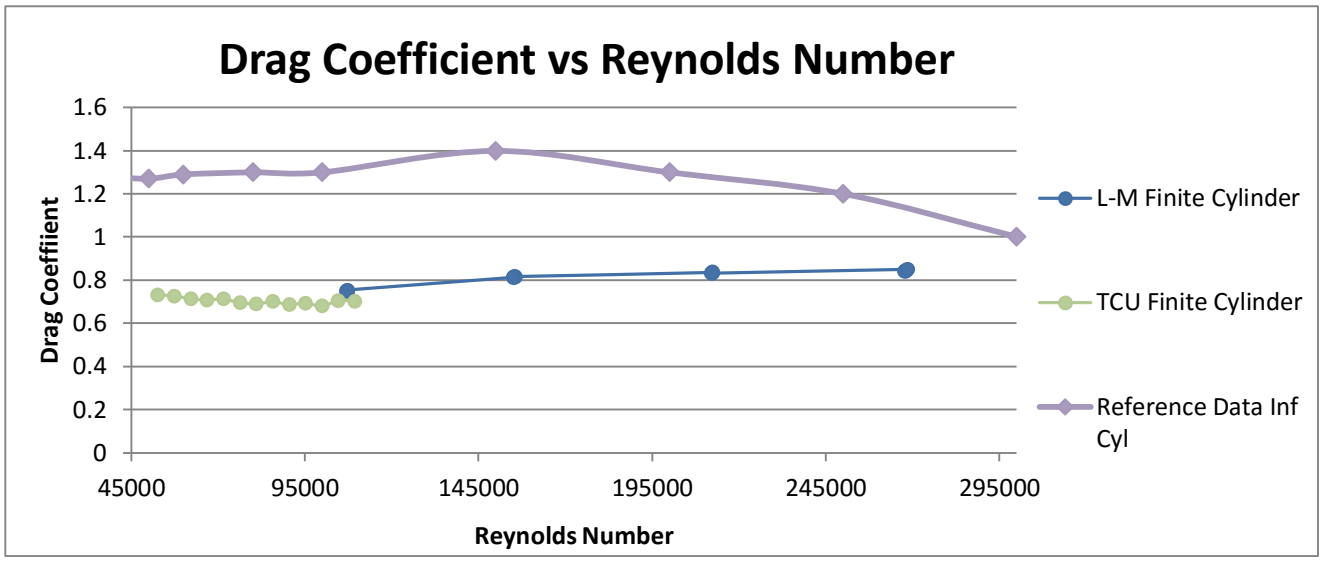

Figure 24 – This plot contains a variety of drag coefficient data versus Reynolds number for comparison. The experimental drag data with the finite cylinder has consistently lower drag coefficients than the infinite cylinder. The TCU data is taken at smaller Reynolds numbers than the L-M data due to the size of the cylinder but still produce comparable drag coefficients.

Figure 24 shows the Reynolds number plotted against the smooth cylinder drag coefficient for the Lockheed Martin data, the TCU data, and infinite cylinder reference data. The TCU data was at a lower Reynolds number than the L-M data because of the smaller cylinder size but still had some slight overlap. The two sets of finite cylinder data matched up which provided confidence in the L-M data set.

Drag Modeling Results

The best fabric for a given segment is the one which produced the lowest drag force over one stride cycle and the worst fabric is the one with the maximum drag force over the same cycle. The best and worst fabrics are displayed in Table 4 below:

Fabric Numbers								
	Minimum Drag				Maximum Drag			
Segment	Upper Leg	Lower Leg	Upper Arm	Lower Arm	Upper Leg	Lower Leg	Upper Arm	Lower Arm
1	2	2	2	2	4	4	4	4
2	2	1	2	2	4	8	4	4
3	2	1	2	2	4	8	4	4
4	2	1	2	2	4	8	4	4

Table 4 - This table shows the fabric numbers for the minimized and maximized drag at each segment. The first segment is the one corresponding to the region closes to the torso of the body and the fourth segment corresponds to the segment closest to the end of the lower limbs (hands on the arms and feet on the legs).

Fabric 2 performs the best over all the limbs with the exception of the lowest three segments on the lower leg, where it is outperformed by fabric 1. Fabric 4 is consistently the worst fabric over the same segments as fabric 2 is best, and fabric 8, which is not actually a fabric but is the smooth cylinder reference data, performs worst on the three lower leg segments where fabric 1 was the best.

The best and worst net drag forces experienced over one stride cycle are shown in the bar graphs below. The yellow bars are the minimum drag force values and the blue are the maximum. See Table 4 above to identify the corresponding fabric for each bar.

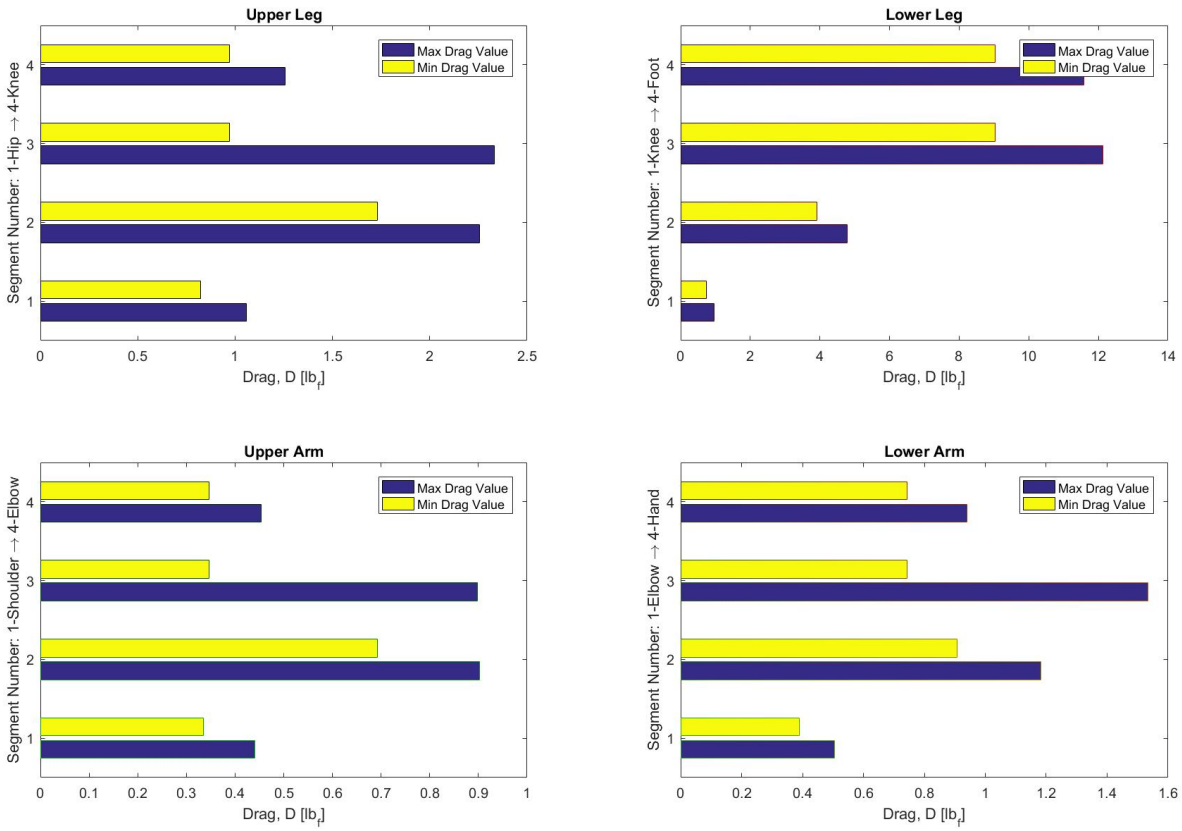

Figure 25 – Bar graphs illustrating the best and worst fabrics, by net drag force, over one stride cycle for each segment on the limbs.

Discussion

Lockheed Testing

Figure 19-Figure 22 show the drag coefficient C_D, plotted against the angle of the cylinder alpha for the four different tested speeds. It can be seen as the angle approaches 90° (cylinder in cross flow) there is a greater separation in coefficients between materials. It also worth noticing that the fabrics that perform well at low angles may not be the best performers at larger angles and similarly fabrics that perform well at low speeds may not do as well at higher speeds. Therefore, to design the optimum low drag suit, different materials need to be used depending on the speed and angle encountered by parts of the body.

Performance Comparison of Best to Worst Fabrics

From the model results a performance comparison can be made between the best and worst fabrics. The average drag force seen on the worst fabric was 2.73 lbf while the best fabric saw an average drag force of 2.007 lbf. This difference in drag force can be multiplied by a velocity to determine the power needed to overcome this additional drag. A velocity of 10.44 m/s is the speed Usain Bolt averages to run his world record time of 9.58 seconds. The additional power needed would then be 0.045 horsepower. This additional power needed instead is a reduction in the velocity and results in a speed 0.715m/s slower for a runner wearing the worst fabric when compared to the best fabric. In a 100-meter race this results in an additional 0.7 seconds added to a runner's time. Assuming Usain Bolt was wearing the optimum fabric during his world record run of 9.58 seconds, his time wearing the worst fabric tested would have been 10.28 seconds. This is a significant decrease in performance that could take the runner from a podium position to last place. This may seem like a drastic result but it is possible when comparing the two fabrics. The last place fabric was a thick neoprene that the air could have permeated through causing it to have more of a parachute effect, drastically slowing down the runner.

Conclusion

This study began with wind tunnel testing at Lockheed Martin's low speed wind tunnel. A cylinder was fitted with several fabrics and the drag force was measured over a range of angles and speeds. A separate test was performed in the TCU engineering wind tunnel to validate the results of this first data set. Once validated, this drag data was incorporated into a body model of a sprinter that could characterize the performance of each fabric for the arms and legs of a sprinter. Finally, the best and worst fabrics were compared to determine the performance difference.

- Lockheed Martin test results were validated by a similar study in the TCU wind tunnel
- There was a significant drag difference between the best and worst fabric that would have a significant effect on performance
- Different fabrics provided the lowest drag behavior depending on the location on the body

Appendix A

This appendix shows the measured angle data for each limb from each annotated frame. Also the annotated frames are shown.

Table 5 - Shows the measured angle data off the attached annotated frames (also included in Appendix A).

Senior Honors Thesis							
Body Model Angles							
Brendan Daly							
3/22/2016							
		head	torso	u.leg	l.leg	u.arm	l.arm
Frame	**Picture**	**h**	**t**	**ul**	**ll**	**ua**	**la**
Color		**Y**	**M**	**C**	**R**	**G**	**B**
num	**num**	**deg**	**deg**	**deg**	**deg**	**deg**	**deg**
1	1	90	105	123	154	47	-92
16	2	90	102	104	-14	75	-20
31	3	90	100	81	-144	94	46
46	4	90	102	51	-161	122	94
61	5	90	105	24	162	150	105
76	6	90	92	5	117	168	101
91	7	90	90	30	83	157	87
106	8	90	96	51	72	147	64
121	9	90	102	67	91	107	28
136	10	90	95	89	126	69	-30
156	11	90	91	118	149	56	-82

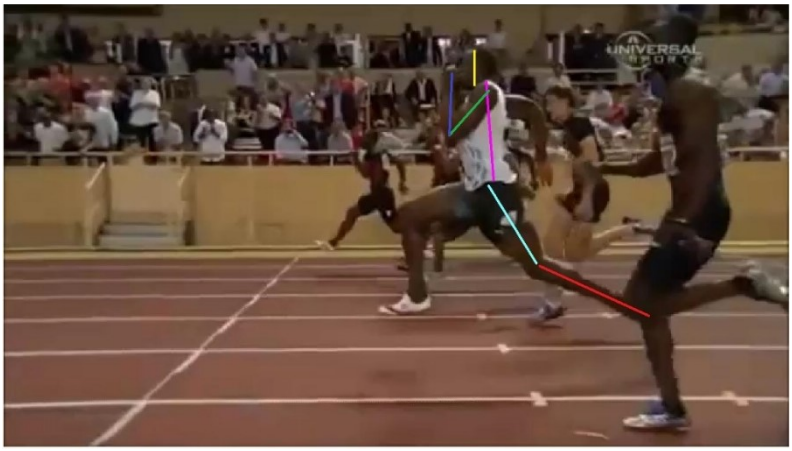

Figure 26 - Annotated Frame 1

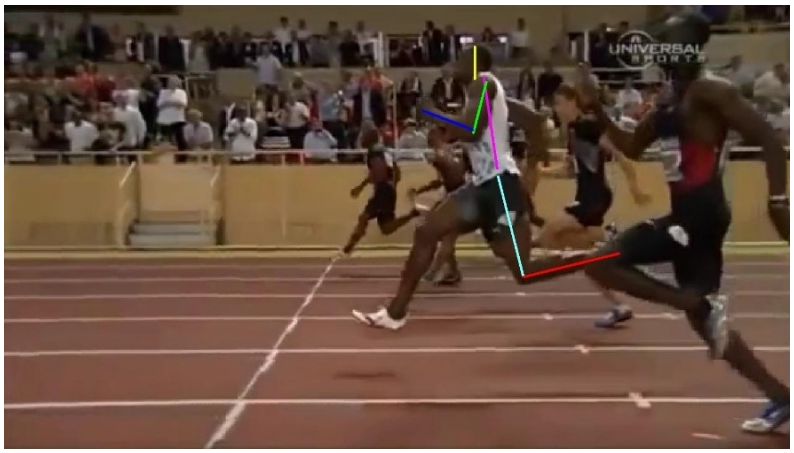

Figure 29 - Annotated Frame 16

Figure 28 - Annotated Frame 31

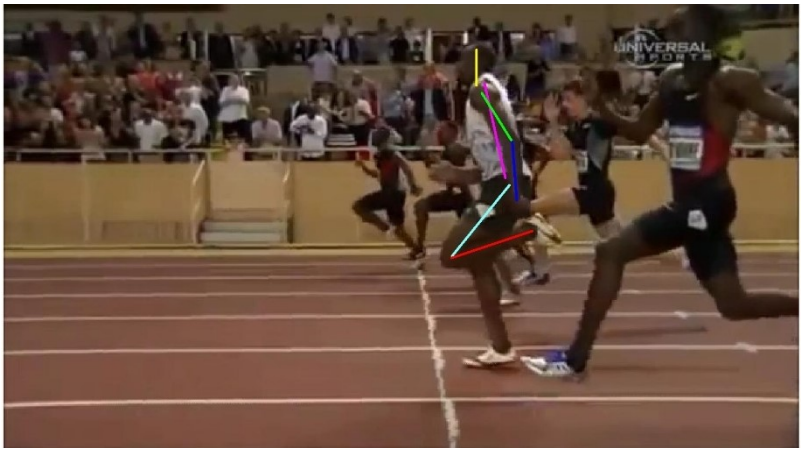

Figure 27 - Annotated Frame 46

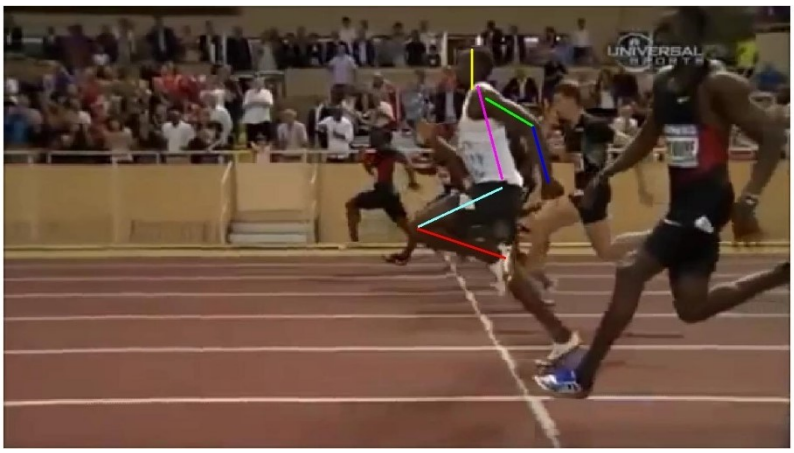

Figure 32 - Annotated Frame 61

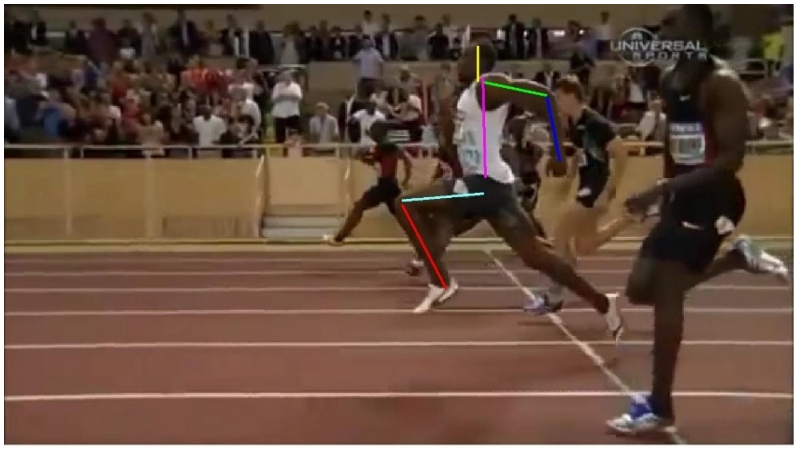

Figure 31 - Annotated Frame 76

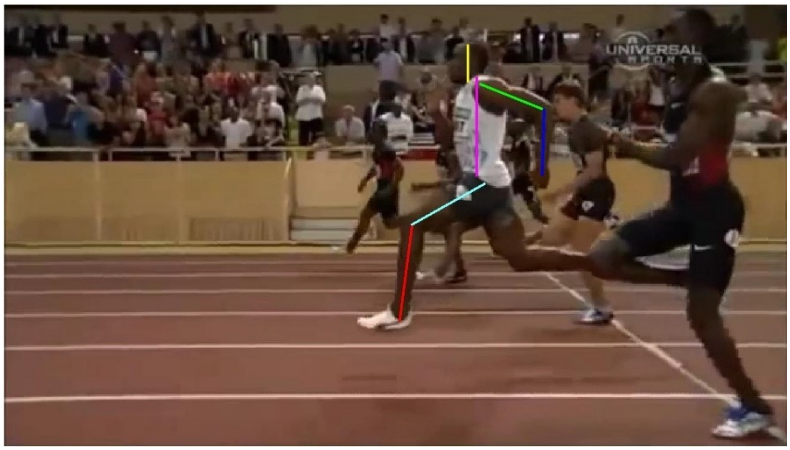

Figure 30 - Annotated Frame 91

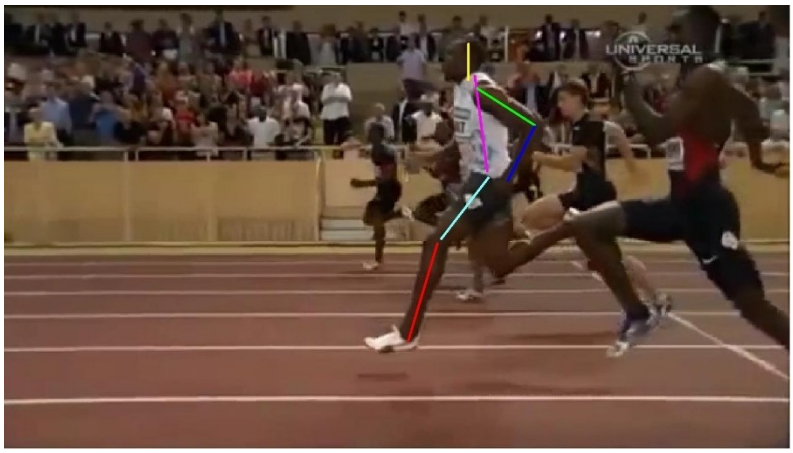

Figure 35 - Annotated Frame 106

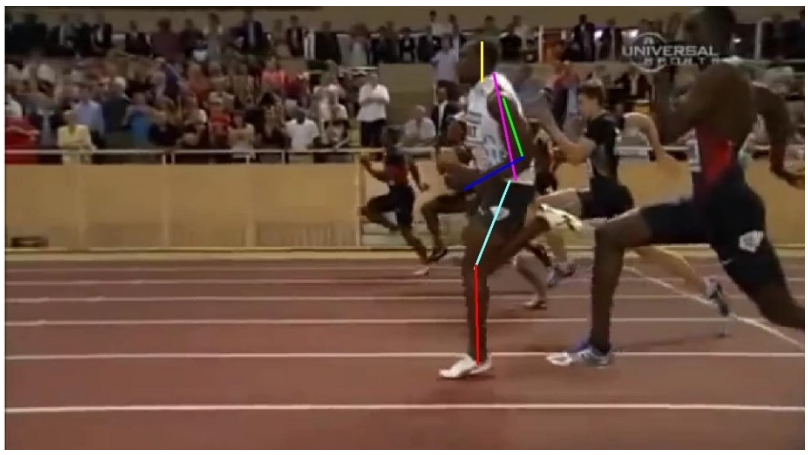

Figure 34 - Annotated Frame 121

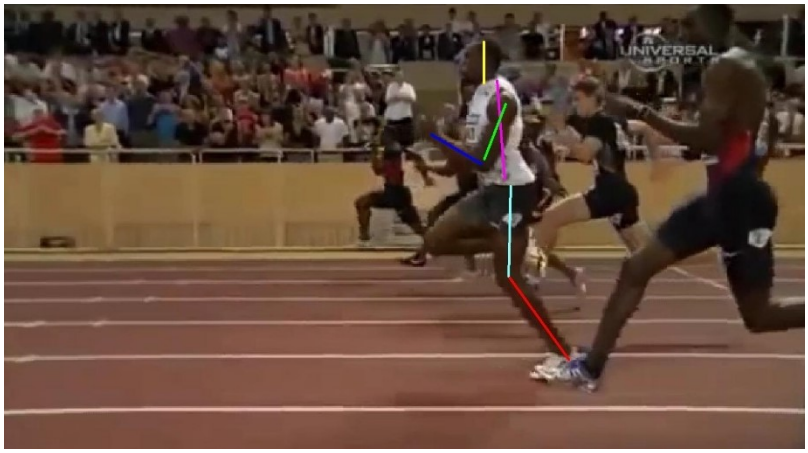

Figure 33 - Annotated Frame 136

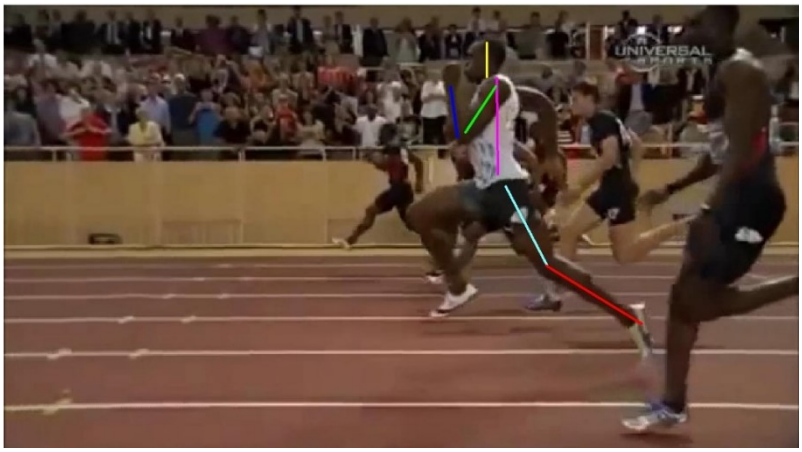

Figure 36 - Annotated Frame 156

Appendix B

This appendix has all the annotated code used to complete the body modeling and analysis.

Aspect Ratio Calculator

```
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          ARCalculator.m
%   Description:    Stores the geometric parameters of the drawing of Usain
%                   Bolt. Calculates scaled up version to be used in model.
%                   This will be the life-sized unites. Also calculates the
%                   aspect ratio (AR) for each cylinder in the model.
%   Inputs:         Lengths and diameters from the printed picture [in]
%   Outputs:        Lengths and diameters of the mathematical model [in]
%                       aspect ratios of each cylinder
%                   Plot comparing each of the model cylinders on a like
%                       scale
%-------------------------------------------------------------------------
% Conversion factor
cf = 7.5;

% Photograph dimmensions (in)
pic = struct;
pic.Length.H = 1.378;
pic.Length.T = 3.3465;
pic.Length.UL = 2.1;
pic.Length.LL = 3.3464567;
pic.Length.UA = 1.771654;
pic.Length.LA = 2.1653543;

pic.Dia.H = 0.7874016;
pic.Dia.T = 1.67322835;
pic.Dia.UL = 0.98425197;
pic.Dia.LL = 0.66929134;
pic.Dia.UA = 0.51181102;
pic.Dia.LA = 0.47244094;

% Conversion to world units
BodyDim.Length = pic.Length;
BodyDim.Length.H  = BodyDim.Length.H  * cf;
BodyDim.Length.T  = BodyDim.Length.T  * cf;
BodyDim.Length.UL = BodyDim.Length.UL * cf;
BodyDim.Length.LL = BodyDim.Length.LL * cf;
BodyDim.Length.UA = BodyDim.Length.UA * cf;
BodyDim.Length.LA = BodyDim.Length.LA * cf;

BodyDim.Dia = pic.Dia;
BodyDim.Dia.H  = BodyDim.Dia.H  * cf;
BodyDim.Dia.T  = BodyDim.Dia.T  * cf;
BodyDim.Dia.UL = BodyDim.Dia.UL * cf;
```

```
BodyDim.Dia.LL = BodyDim.Dia.LL * cf;
BodyDim.Dia.UA = BodyDim.Dia.UA * cf;
BodyDim.Dia.LA = BodyDim.Dia.LA * cf;

% Calculating Aspect Ratios
BodyDim.AR.H  = BodyDim.Length.H  / BodyDim.Dia.H;
BodyDim.AR.T  = BodyDim.Length.T  / BodyDim.Dia.T;
BodyDim.AR.UL = BodyDim.Length.UL / BodyDim.Dia.UL;
BodyDim.AR.LL = BodyDim.Length.LL / BodyDim.Dia.LL;
BodyDim.AR.UA = BodyDim.Length.UA / BodyDim.Dia.UA;
BodyDim.AR.LA = BodyDim.Length.LA / BodyDim.Dia.LA;

%*************************************************************************
% PLOTTING
%*************************************************************************

% PLOT ALL CYLINDRICAL APPROXIMATIONS

% Calculate each radius
BodyDim.Rad.H  = BodyDim.Dia.H  / 2;
BodyDim.Rad.T  = BodyDim.Dia.T  / 2;
BodyDim.Rad.UL = BodyDim.Dia.UL / 2;
BodyDim.Rad.LL = BodyDim.Dia.LL / 2;
BodyDim.Rad.UA = BodyDim.Dia.UA / 2;
BodyDim.Rad.LA = BodyDim.Dia.LA / 2;

% Create cylindrical points
X = struct; Y = struct; Z = struct;
[X.H,  Y.H,  Z.H ] = cylinder(BodyDim.Rad.H,40);
[X.T,  Y.T,  Z.T ] = cylinder(BodyDim.Rad.T,40);
[X.UL, Y.UL, Z.UL] = cylinder(BodyDim.Rad.UL,40);
[X.LL, Y.LL, Z.LL] = cylinder(BodyDim.Rad.LL,40);
[X.UA, Y.UA, Z.UA] = cylinder(BodyDim.Rad.UA,40);
[X.LA, Y.LA, Z.LA] = cylinder(BodyDim.Rad.LA,40);

% Cylinder output is normalized. Multiply by actual cylinder lengths
Z.H  = BodyDim.Length.H  * Z.H;
Z.T  = BodyDim.Length.T  * Z.T;
Z.UL = BodyDim.Length.UL * Z.UL;
Z.LL = BodyDim.Length.LL * Z.LL;
Z.UA = BodyDim.Length.UA * Z.UA;
Z.LA = BodyDim.Length.LA * Z.LA;

% Head is yellow
Col.H = ones(size(Z.H,1),size(Z.H,2),3);
    Col.H(:,:,1) = 1;
    Col.H(:,:,2) = 1;
    Col.H(:,:,3) = 0;
% Torso is magenta
Col.T = ones(size(Z.T,1),size(Z.T,2),3);
    Col.T(:,:,1) = 1;
    Col.T(:,:,2) = 0;
    Col.T(:,:,3) = 1;
% Upper leg is cyan
Col.UL = ones(size(Z.UL,1),size(Z.UL,2),3);
    Col.UL(:,:,1) = 0;
    Col.UL(:,:,2) = 1;
```

```
    Col.UL(:,:,3) = 1;
% Lower leg is red
Col.LL = ones(size(Z.LL,1),size(Z.LL,2),3);
    Col.LL(:,:,1) = 1;
    Col.LL(:,:,2) = 0;
    Col.LL(:,:,3) = 0;
% Upper arm is green
Col.UA = ones(size(Z.UA,1),size(Z.UA,2),3);
    Col.UA(:,:,1) = 0;
    Col.UA(:,:,2) = 1;
    Col.UA(:,:,3) = 0;
% Lower arm is blue
Col.LA = ones(size(Z.LA,1),size(Z.LA,2),3);
    Col.LA(:,:,1) = 0;
    Col.LA(:,:,2) = 0;
    Col.LA(:,:,3) = 1;

% Titles
Tit.H = {'Head Approximation'};
Tit.T = {'Torso Approximation'};
Tit.UL = {'Upper Leg Approximation'};
Tit.LL = {'Lower Leg Approximation'};
Tit.UA = {'Upper Arm Approximation'};
Tit.LA = {'Lower Arm Approximation'};

% Plot
figure
s(1) = subplot(2,3,1);
    surf(X.H,Y.H,Z.H,Col.H)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(1),Tit.H)
s(2) = subplot(2,3,2);
    surf(X.T,Y.T,Z.T,Col.T)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(2),Tit.T)
s(3) = subplot(2,3,3);
    surf(X.UL,Y.UL,Z.UL,Col.UL)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(3),Tit.UL)
s(4) = subplot(2,3,4);
    surf(X.LL,Y.LL,Z.LL,Col.LL)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(4),Tit.LL)
s(5) = subplot(2,3,5);
    surf(X.UA,Y.UA,Z.UA,Col.UA)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(5),Tit.UA)
s(6) = subplot(2,3,6);
    surf(X.LA,Y.LA,Z.LA,Col.LA)
    axis([-7.5 7.5 -7.5 7.5 0 40])
    title(s(6),Tit.LA)
```

Body Modeling and Analysis

```
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          BodyAnalysis.m
%   Description:    Limb by limb analysis of one running cycle to identifty
%                   minimum and maximum drag and corresponding fabrics.
%   Inputs:         Body Model Data
%                   Video of one sprint cycle
%   Outputs:        Results.xlsx ~ excel sheet showing best and worst
%                       fabric indecies based off drag performance
%                   Resultant Power data
%-------------------------------------------------------------------------
% Limb by limb analysis of one running cycle.

%% Video Input
filename = 'OneCycleOfSprint.mp4';

% Construct a VideoReader object associated w/ the file
% 'UsainBolt100mSlowMo'
vidObj = VideoReader(filename);

% Determine the height and width of the frames
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

% Create a MATLAB movie structure array
s = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),...
    'colormap',[]);

% Read one fram at a time using ReadFrame unitl the end of the file is
% reached. APpend data from each video frame to the structure array.
k=1;
while hasFrame(vidObj)
    s(k).cdata = readFrame(vidObj);
    k = k+1;
end

% whos s

%% Processing - Frame Extraction
%
% % Extract every 15th frame for analysis. Additionally extract the final
% % frame.
%
% scrsz = get(groot,'ScreenSize');
%
% last = floor(size(s,2)/15);
% for i = 1:last
%     frameNumber = 15 * i - 14;
%     filepath = strcat('C:\Users\Brendan\Documents\MATLAB\HonorsThesis',...
%         '\FrameByFrameAnalysis\Frame',num2str(frameNumber),'.jpg');
%     Frame = s(frameNumber).cdata;
```

```
%     fig = figure;
%     imshow(Frame)
%     fig.OuterPosition = [1 1 scrsz(3)-50 scrsz(4)];
%     saveas(fig,filepath)
%     close(gcf)
% end
%
% frameNumber = size(s,2);
% filepath = strcat('C:\Users\Brendan\Documents\MATLAB\HonorsThesis',...
%     '\FrameByFrameAnalysis\Frame',num2str(frameNumber),'.jpg');
% Frame = s(frameNumber).cdata;
% fig = figure;
% imshow(Frame)
% fig.OuterPosition = [1 1 scrsz(3)-50 scrsz(4)];
% saveas(fig,filepath)
% close(gcf)
%
% %% Processing - Angle Calculations
% % Calculating angle of each position
% % Lines in annotated images represent angles of body parts.
% %   Head - yellow       [255 255 0  ]
% %   Torso - magenta     [255 0   255]
% %   U. Leg - cyan       [0   255 255]
% %   L. Leg - red        [255 0   0  ]
% %   U. Arm - green      [50  174 88 ]
% %   L. Arm - blue       [63  72  203]
%
% % Create a structure array containing the data from each picture
% filepath = strcat('C:\Users\Brendan\Documents\MATLAB\HonorsThesis\',...
%     'FrameByFrameAnalysis\Annotated\');
%
% angledata(1) =
struct('filepath',{strcat(filepath,'Frame1Annotated.jpg')},...
%                 'origFrame',1);
% angledata(2) =
struct('filepath',{strcat(filepath,'Frame16Annotated.jpg')},...
%                 'origFrame',16);
% angledata(3) =
struct('filepath',{strcat(filepath,'Frame31Annotated.jpg')},...
%                 'origFrame',31);
% angledata(4) =
struct('filepath',{strcat(filepath,'Frame46Annotated.jpg')},...
%                 'origFrame',46);
% angledata(5) =
struct('filepath',{strcat(filepath,'Frame61Annotated.jpg')},...
%                 'origFrame',61);
% angledata(6) =
struct('filepath',{strcat(filepath,'Frame76Annotated.jpg')},...
%                 'origFrame',76);
% angledata(7) =
struct('filepath',{strcat(filepath,'Frame91Annotated.jpg')},...
%                 'origFrame',91);
% angledata(8) =
struct('filepath',{strcat(filepath,'Frame106Annotated.jpg')},...
%                 'origFrame',106);
% angledata(9) =
struct('filepath',{strcat(filepath,'Frame121Annotated.jpg')},...
```

```
%                 'origFrame',121);
% angledata(10) =
struct('filepath',{strcat(filepath,'Frame136Annotated.jpg')},...
%                 'origFrame',136);
% angledata(11) =
struct('filepath',{strcat(filepath,'Frame156Annotated.jpg')},...
%                 'origFrame',156);
%
% % Input the RGB color data for each of the lines to be identified.
% colordata.h  = [255 255 0  ];
% colordata.t  = [255 0   255];
% colordata.ul = [0   255 255];
% colordata.ll = [255 0   0  ];
% colordata.ua = [50  174 88 ];
% colordata.la = [63  72  203];
%
% % Loop through each image and extract the angles
%
% numImages = size(angledata,2);
%
% for k = 1:numImages
%     I = imread(angledata(k).filepath);
%     angles = angle_extractor(colordata,I);
%     angledata(k).meas.h  = angles(1);
%     angledata(k).meas.t  = angles(2);
%     angledata(k).meas.ul = angles(3);
%     angledata(k).meas.ll = angles(4);
%     angledata(k).meas.ua = angles(5);
%     angledata(k).meas.la = angles(6);
% end

%% Processing - Model Creation

% Gather cylinder data
% Conversion factor
cf = 7.5;

% Photograph dimmensions (in)
pic = struct;
pic.Length.H = 1.378;
pic.Length.T = 3.3465;
pic.Length.UL = 2.1;
pic.Length.LL = 3.3464567;
pic.Length.UA = 1.771654;
pic.Length.LA = 2.1653543;

pic.Dia.H = 0.7874016;
pic.Dia.T = 1.67322835;
pic.Dia.UL = 0.98425197;
pic.Dia.LL = 0.66929134;
pic.Dia.UA = 0.51181102;
pic.Dia.LA = 0.47244094;

% Conversion to world units (in)
BodyDim.Length = pic.Length;
BodyDim.Length.H  = BodyDim.Length.H  * cf;
BodyDim.Length.T  = BodyDim.Length.T  * cf;
```

```
BodyDim.Length.UL = BodyDim.Length.UL * cf;
BodyDim.Length.LL = BodyDim.Length.LL * cf;
BodyDim.Length.UA = BodyDim.Length.UA * cf;
BodyDim.Length.LA = BodyDim.Length.LA * cf;

BodyDim.Dia = pic.Dia;
BodyDim.Dia.H  = BodyDim.Dia.H  * cf;
BodyDim.Dia.T  = BodyDim.Dia.T  * cf;
BodyDim.Dia.UL = BodyDim.Dia.UL * cf;
BodyDim.Dia.LL = BodyDim.Dia.LL * cf;
BodyDim.Dia.UA = BodyDim.Dia.UA * cf;
BodyDim.Dia.LA = BodyDim.Dia.LA * cf;

% Calculating Aspect Ratios
BodyDim.AR.H  = BodyDim.Length.H  / BodyDim.Dia.H;
BodyDim.AR.T  = BodyDim.Length.T  / BodyDim.Dia.T;
BodyDim.AR.UL = BodyDim.Length.UL / BodyDim.Dia.UL;
BodyDim.AR.LL = BodyDim.Length.LL / BodyDim.Dia.LL;
BodyDim.AR.UA = BodyDim.Length.UA / BodyDim.Dia.UA;
BodyDim.AR.LA = BodyDim.Length.LA / BodyDim.Dia.LA;

% Read in angle data from excel sheet.
filename = 'BodyModelAngleData.xlsx';
[num,txt,raw] = xlsread(filename);
num_frames = size(num,1);
t_frame = .003; %s/frame

theta = struct([]);
theta_deg = num(:,3:8);
theta_rad = theta_deg * pi/180;

for i = 1:num_frames
    theta(i).frame  = num(i,1);
    theta(i).h      = theta_rad(i,1);
    theta(i).t      = theta_rad(i,2);
    theta(i).ul     = theta_rad(i,3);
    theta(i).ll     = theta_rad(i,4);
    theta(i).ua     = theta_rad(i,5);
    theta(i).la     = theta_rad(i,6);
    % Add a field to angledata with the time elapsed, assuming t0 = frame1
    theta(i).time = (theta(i).frame - 1) * t_frame;
end

% Calculate incremental angular frequencies rad/(delta_frame)
w_rad = zeros(num_frames-1,size(theta_rad,2));
w_deg = zeros(num_frames-1,size(theta_deg,2));
for i = 1:(num_frames-1)
    delta_frame = theta(i+1).frame - theta(i).frame;
    delta_time = delta_frame * t_frame;
    w_rad(i,:) = theta_rad(i+1,:) - theta_rad(i,:); %rad/delta_frame
    w_deg(i,:) = theta_deg(i+1,:) - theta_deg(i,:); %deg/delta_frame
end

% correction factor for number of frames
% -> normalize w to units of rad/s or deg/s
% w [rad/delta_frame] * [delta_frame/delta_time] = w [rad/s] or [deg/s]
w_rad = w_rad / delta_time;
```

```
w_deg = w_deg / delta_time;

% Use rigid multibody rotational dynamics to model arms and legs.
% Assume head and torso act as one fixed rigid body with constant velocity
% forward of V_FreeStream or Vinf.
%   Upper Limb ~ ab
%   Lower Limb ~ bc
%   V_ab(x) = [ Vinf - L_ab(x)*sin(theta_ab_i)*w_ab_i ] * I ...
%           + [ L_ab(x)*cos(theta_ab_i)*w_ab_1 ] * J;
%   V_bc(x) = [ Vinf - L_ab*sin(theta_ab_i)*w_ab_i ...
%             - L_bc(x)*sin(theta_bc_i)*w_bc_i ] * I ...
%           + [ L_ab*cos(theta_ab_i)*w_ab ...
%             + L_bc(x)*cos(theta_bc_i)*w_bc_i ] * J;

% Free stream velocity, or rigid body velocity is 411.137 in/s
Vinf = 411.137; %in/s

% Call the function VCalc to calculate the velocities at n points along
% each limb.
% n ~ the number of data points along each limb. The first is the joint
%       closest to the body and the last is the end of the limb furthest
%       from the body.
v = struct('ul',[],'ll',[],'ua',[],'la',[]);
n = 4;
[v.ul , v.ll] = VCalc(Vinf , BodyDim.Length.UL , BodyDim.Length.LL , ...
                    theta_rad(1:10,3) , w_rad(:,3) , ...
                    theta_rad(1:10,4) , w_rad(:,4) , n );

[v.ua , v.la] = VCalc(Vinf , BodyDim.Length.UA , BodyDim.Length.LA , ...
                    theta_rad(1:10,5) , w_rad(:,5) , ...
                    theta_rad(1:10,6) , w_rad(:,6) , n );

% Convert velocities from i,j components to magnitude, angle form.
%   i ~ left from the rigid body
%   j ~ down from the rigid body
%       this results in...
%   mag = sqrt(mag(i_component)^2 + mag(j_component)^2
%   phi = arctan(j_component/i_component)
%       ~ angle measured from the positive i-axis. CCW~(+)
% Calculate angle of attack of the velocity vector relative to the location
% of the cyinder, alpha.
%   alpha = |theta - phi|, if alpha > 90deg...
%                          -> alpha = 180deg - alpha
%   alpha ~ velocity angle relative to body
% Calculate Reynolds number, ReD, for each data point.
%   ReD = rho*V*d/mu
%       rho ~ density at 70degF
%       rho = 0.07234 lbm/ft^3
%       mu ~ dynamic viscocity
%       mu = 3.82E-7 lbf*s/ft^2
rho = 0.07234; %lbm/ft^3
mu = 3.82E-7; %lbf*s/ft^2

for m = 1:n
    % Mag/Phase
    v.ul(m).mag = sqrt((v.ul(m).i).^2+(v.ul(m).j).^2);
    v.ul(m).phi = atan2((v.ul(m).j),(v.ul(m).i));
```

```
    v.ll(m).mag = sqrt((v.ll(m).i).^2+(v.ll(m).j).^2);
    v.ll(m).phi = atan2((v.ll(m).j),(v.ll(m).i));
    v.ua(m).mag = sqrt((v.ua(m).i).^2+(v.ua(m).j).^2);
    v.ua(m).phi = atan2((v.ua(m).j),(v.ua(m).i));
    v.la(m).mag = sqrt((v.la(m).i).^2+(v.la(m).j).^2);
    v.la(m).phi = atan2((v.la(m).j),(v.la(m).i));
    % alpha
    v.ul(m).alpha = abs(theta_rad(1:10,3) - v.ul(m).phi);
    v.ll(m).alpha = abs(theta_rad(1:10,4) - v.ll(m).phi);
    v.ua(m).alpha = abs(theta_rad(1:10,5) - v.ua(m).phi);
    v.la(m).alpha = abs(theta_rad(1:10,6) - v.la(m).phi);
    % Reducing alpha to 0 <= alpha <= pi/2
    for l = 1:size(v.ul(m).alpha,1)
        while v.ul(m).alpha(l) > (pi/2)
            v.ul(m).alpha(l) = pi - v.ul(m).alpha(l);
        end
        while v.ll(m).alpha(l) > (pi/2)
            v.ll(m).alpha(l) = pi - v.ll(m).alpha(l);
        end
        while v.ua(m).alpha(l) > (pi/2)
            v.ua(m).alpha(l) = pi - v.ua(m).alpha(l);
        end
        while v.la(m).alpha(l) > (pi/2)
            v.la(m).alpha(l) = pi - v.la(m).alpha(l);
        end
    end
%     % Reynolds Number
%     v.ul(m).ReD = (rho/(144*32.2*mu)) * v.ul(m).mag...
%                       .*(BodyDim.Dia.UL./cos(v.ul(m).alpha));

end



%% Processing - Test Data Reduction
% Reduce the data from testing to a form that can be used in a parametric
% study. The parametric study will compare drag forces at each fabric for
% each part of the body in the model.
filename = 'DragData.xlsx';
sheet = 'NewPlotData';
[num,txt,raw] = xlsread(filename,sheet);

% Reducing data to size
fab = struct([]);
cnt = ones(8,1);
for p = 1:size(raw,1)
    if strcmp('smooth',raw(p,1)) % Stored in fab(8)
        ind = cnt(8);
        fab(8).fabric = raw(p,1);
        fab(8).run(ind,1) = raw(p,3);
        fab(8).alpha(ind,1) = raw(p,10);
        fab(8).v(ind,1) = raw(p,14);
        fab(8).Cd(ind,1) = raw(p,98);
        cnt(8) = cnt(8) + 1;
    elseif strcmp('1 1',raw(p,1)) % Fabric 1
        ind = cnt(1);
        fab(1).fabric = raw(p,1);
```

```
        fab(1).run(ind,1) = raw(p,3);
        fab(1).alpha(ind,1) = raw(p,10);
        fab(1).v(ind,1) = raw(p,14);
        fab(1).Cd(ind,1) = raw(p,98);
        cnt(1) = cnt(1) + 1;
    elseif strcmp({'#7 = 10"'},raw(p,1)) % Fabric 2
        ind = cnt(2);
        fab(2).fabric = raw(p,1);
        fab(2).run(ind,1) = raw(p,3);
        fab(2).alpha(ind,1) = raw(p,10);
        fab(2).v(ind,1) = raw(p,14);
        fab(2).Cd(ind,1) = raw(p,98);
        cnt(2) = cnt(2) + 1;
    elseif strcmp({'N -04 35"'},raw(p,1)) % Fabric 3
        ind = cnt(3);
        fab(3).fabric = raw(p,1);
        fab(3).run(ind,1) = raw(p,3);
        fab(3).alpha(ind,1) = raw(p,10);
        fab(3).v(ind,1) = raw(p,14);
        fab(3).Cd(ind,1) = raw(p,98);
        cnt(3) = cnt(3) + 1;
    elseif strcmp({' M 13'},raw(p,1)) % Fabric 4
        ind = cnt(4);
        fab(4).fabric = raw(p,1);
        fab(4).run(ind,1) = raw(p,3);
        fab(4).alpha(ind,1) = raw(p,10);
        fab(4).v(ind,1) = raw(p,14);
        fab(4).Cd(ind,1) = raw(p,98);
        cnt(4) = cnt(4) + 1;
    elseif strcmp({'SI 01'},raw(p,1)) % Fabric 5
        ind = cnt(5);
        fab(5).fabric = raw(p,1);
        fab(5).run(ind,1) = raw(p,3);
        fab(5).alpha(ind,1) = raw(p,10);
        fab(5).v(ind,1) = raw(p,14);
        fab(5).Cd(ind,1) = raw(p,98);
        cnt(5) = cnt(5) + 1;
    elseif strcmp({'B 1'},raw(p,1)) % Fabric 6
        ind = cnt(6);
        fab(6).fabric = raw(p,1);
        fab(6).run(ind,1) = raw(p,3);
        fab(6).alpha(ind,1) = raw(p,10);
        fab(6).v(ind,1) = raw(p,14);
        fab(6).Cd(ind,1) = raw(p,98);
        cnt(6) = cnt(6) + 1;
    elseif strcmp({'SL 03'},raw(p,1)) % Fabric 7
        ind = cnt(7);
        fab(7).fabric = raw(p,1);
        fab(7).run(ind,1) = raw(p,3);
        fab(7).alpha(ind,1) = raw(p,10);
        fab(7).v(ind,1) = raw(p,14);
        fab(7).Cd(ind,1) = raw(p,98);
        cnt(7) = cnt(7) + 1;
    end
end

for p = 1:size(fab,2)
```

```
    fab(p).run = cell2mat(fab(p).run);
    fab(p).alpha = cell2mat(fab(p).alpha);
    fab(p).v = cell2mat(fab(p).v);
    fab(p).Cd = cell2mat(fab(p).Cd);
end

% Further reduce experimetnal data to correlation data.
% ALPHA
%   0    < alpha < 37.5deg  -> 30deg
%   37.5 < alhpa < 52.5deg  -> 45deg
%   52.5 < alpha < 67.5deg  -> 60deg
%   67.5 < alpha < 82.5deg  -> 75deg
%   82.5 < alpha            -> 90deg
% VELOCITY
%        vel < 50fps    -> 40fps
%   50 < vel < 70fps    -> 60fps
%   70 < vel < 90fps    -> 80fps
%   90 < vel            -> 100fps
corrdata = fab;
for p = 1:size(corrdata,2)
    for q = 1:size(corrdata(p).alpha,1)
        if corrdata(p).alpha(q) < 37.5
            corrdata(p).alpha(q) = 30;
        elseif corrdata(p).alpha(q) < 52.5
            corrdata(p).alpha(q) = 45;
        elseif corrdata(p).alpha(q) < 67.5
            corrdata(p).alpha(q) = 60;
        elseif corrdata(p).alpha(q) < 82.5
            corrdata(p).alpha(q) = 75;
        else
            corrdata(p).alpha(q) = 90;
        end
        if corrdata(p).v(q) < 50
            corrdata(p).v(q) = 40;
        elseif corrdata(p).v(q) < 70
            corrdata(p).v(q) = 60;
        elseif corrdata(p).v(q) < 90
            corrdata(p).v(q) = 80;
        else
            corrdata(p).v(q) = 100;
        end
    end
end

% Create a normalized vector that represents the endpoints of each region
% to be integrated down each limb. This value is dependent upon 'n'. THis
% vector will be multiplied by the length of a limb to calculate the
% boundaries of each region.
% Joint Region Spans: 1/(2(n-1))
% Middle Regions Span:1/(n-1)
% End Region Spans: 1/(2(n-1))
unityScale = zeros(n+1,1);
unityScale(2,1) = 1/(2*(n-1));
cum = unityScale(2,1);
for p = 1:(n-2)
    unityScale(p+2,1) = cum + 1/(n-1);
    cum = unityScale(p+2,1);
```

```
end
unityScale(n+1,1) = 1;

% Reorganize data in form:
% fabric().vel().alpha()
% Fabrics 1-8
% Velocities
%   1: 40fps
%   2: 60fps
%   3: 80fps
%   4: 100fps
% Alphas:
%   1: 30deg
%   2: 45deg
%   3: 60deg
%   4: 75deg
%   5: 90deg
fabric = struct([]);

for f = 1:size(corrdata,2)
    fabric(f).name = corrdata(f).fabric;
    fabric(f).vel_ind = [40 60 80 100];
    for p = 1:size(corrdata(f).v,1)
        switch corrdata(f).v(p,1)
            case 40
                switch corrdata(f).alpha(p,1)
                    case 30
                        vel(1).alpha(1) = corrdata(f).Cd(p,1);
                    case 45
                        vel(1).alpha(2) = corrdata(f).Cd(p,1);
                    case 60
                        vel(1).alpha(3) = corrdata(f).Cd(p,1);
                    case 75
                        vel(1).alpha(4) = corrdata(f).Cd(p,1);
                    case 90
                        vel(1).alpha(5) = corrdata(f).Cd(p,1);
                end
            case 60
                switch corrdata(f).alpha(p,1)
                    case 30
                        vel(2).alpha(1) = corrdata(f).Cd(p,1);
                    case 45
                        vel(2).alpha(2) = corrdata(f).Cd(p,1);
                    case 60
                        vel(2).alpha(3) = corrdata(f).Cd(p,1);
                    case 75
                        vel(2).alpha(4) = corrdata(f).Cd(p,1);
                    case 90
                        vel(2).alpha(5) = corrdata(f).Cd(p,1);
                end
            case 80
                switch corrdata(f).alpha(p,1)
                    case 30
                        vel(3).alpha(1) = corrdata(f).Cd(p,1);
                    case 45
                        vel(3).alpha(2) = corrdata(f).Cd(p,1);
                    case 60
```

```
                        vel(3).alpha(3) = corrdata(f).Cd(p,1);
                    case 75
                        vel(3).alpha(4) = corrdata(f).Cd(p,1);
                    case 90
                        vel(3).alpha(5) = corrdata(f).Cd(p,1);
                end
            case 100
                switch corrdata(f).alpha(p,1)
                    case 30
                        vel(4).alpha(1) = corrdata(f).Cd(p,1);
                    case 45
                        vel(4).alpha(2) = corrdata(f).Cd(p,1);
                    case 60
                        vel(4).alpha(3) = corrdata(f).Cd(p,1);
                    case 75
                        vel(4).alpha(4) = corrdata(f).Cd(p,1);
                    case 90
                        vel(4).alpha(5) = corrdata(f).Cd(p,1);
                end
        end
    end
    fabric(f).vel = vel;
end
% Rewrite data back to structure 'corrdata'.
% corrdata(f).vel(v_ind).alpha(alpha_ind)
corrdata = fabric;

%% Processing - Applying Test Data to the Model
% Use  experimental angle of attack, alpha, and experimental velocity, v,
% to correlate a drag coefficients for each fabric to the discretized model
% data.

% Correlate model with correlation data.
% Iterate through each fabric applying its corresponding coefficients to
% the body model. Calculate drag force for each discrete section of each
% limb.
% D = 1/2*rho*V^2*A*Cd
Drag = struct([]);
for f = 1:size(corrdata,2)
    for length_ind = 1:n
        % Upper Leg
        segment_length = (BodyDim.Length.UL ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.UL/12; %ft
        Drag(f).ul(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ul(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ul(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^2 * A * Cd;
            Drag(f).ul(length_ind) = Drag(f).ul(length_ind) + D;
        end
        % Lower Leg
```

```
        segment_length = (BodyDim.Length.LL ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.LL/12; %ft
        Drag(f).ll(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ll(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ll(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^2 * A * Cd;
            Drag(f).ll(length_ind) = Drag(f).ll(length_ind) + D;
        end
        % Upper Arm
        segment_length = (BodyDim.Length.UA ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.UA/12; %ft
        Drag(f).ua(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ua(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ua(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^2 * A * Cd;
            Drag(f).ua(length_ind) = Drag(f).ua(length_ind) + D;
        end
        % Lower Arm
        segment_length = (BodyDim.Length.LA ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.LA/12; %ft
        Drag(f).la(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.la(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.la(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^2 * A * Cd;
            Drag(f).la(length_ind) = Drag(f).la(length_ind) + D;
        end
    end
end

% Correlate model with correlation data.
% Iterate through each fabric applying its corresponding coefficients to
% the body model. Calculate power for each discrete section of each
% limb.
% P = 1/2*rho*V^3*A*Cd
Power = struct([]);
for f = 1:size(corrdata,2)
```

```
    for length_ind = 1:n
        % Upper Leg
        segment_length = (BodyDim.Length.UL ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.UL/12; %ft
        Power(f).ul(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ul(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ul(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^3 * A * Cd;
            Power(f).ul(length_ind) = Power(f).ul(length_ind) + D;
        end
        % Lower Leg
        segment_length = (BodyDim.Length.LL ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.LL/12; %ft
        Power(f).ll(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ll(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ll(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^3 * A * Cd;
            Power(f).ll(length_ind) = Power(f).ll(length_ind) + D;
        end
        % Upper Arm
        segment_length = (BodyDim.Length.UA ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.UA/12; %ft
        Power(f).ua(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.ua(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
            Alpha = v.ua(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^3 * A * Cd;
            Power(f).ua(length_ind) = Power(f).ua(length_ind) + D;
        end
        % Lower Arm
        segment_length = (BodyDim.Length.LA ...
            * (unityScale(length_ind+1) - unityScale(length_ind)))/12; %ft
        segment_dia = BodyDim.Dia.LA/12; %ft
        Power(f).la(length_ind) = 0; %Initialize drag force to 0 for segment
        for fstep = 1:(num_frames-1)
            V = v.la(length_ind).mag(fstep)/12; %[ft/s] Velocity for this
frame step
```

```
            Alpha = v.la(length_ind).alpha(fstep); %[rad/s] alpha for this
frame step
            Cd = dragCoefXtract(corrdata,f,V,Alpha); %extract
Cd~f(fabric,V,Alpha)
            A = segment_dia * segment_length * sin(Alpha);
            D = 1/2 * rho/32.2 * V^3 * A * Cd;
            Power(f).la(length_ind) = Power(f).la(length_ind) + D;
        end
    end
end

%% Analysis - Optimizing Fabric Location
% Identify the minimum drag force for each segment of the body and the
% corresponding fabric.

% Loop throught each segment of each body part. Loop through segments.
% Store min and max in matricies. Fabric index stored in third dimension.
%            UL  LL  UA  LA
%   segment1
%   segment2
%   segment3
%   segment4
% Initialize the min and max matricies.
Min = flintmax * ones(4);
Max = zeros(4);
for length_ind = 1:n
    for fabric_ind = 1:8
        % Upper Leg (length_ind,1,:)
        if Drag(fabric_ind).ul(length_ind) < Min(length_ind,1,1)
            Min(length_ind,1,1) = Drag(fabric_ind).ul(length_ind);
            Min(length_ind,1,2) = fabric_ind;
        end
        if Drag(fabric_ind).ul(length_ind) > Max(length_ind,1,1)
            Max(length_ind,1,1) = Drag(fabric_ind).ul(length_ind);
            Max(length_ind,1,2) = fabric_ind;
        end
        % Lower Leg (length_ind,2,:)
        if Drag(fabric_ind).ll(length_ind) < Min(length_ind,2,1)
            Min(length_ind,2,1) = Drag(fabric_ind).ll(length_ind);
            Min(length_ind,2,2) = fabric_ind;
        end
        if Drag(fabric_ind).ll(length_ind) > Max(length_ind,2,1)
            Max(length_ind,2,1) = Drag(fabric_ind).ll(length_ind);
            Max(length_ind,2,2) = fabric_ind;
        end
        % Upper Arm (length_ind,3,:)
        if Drag(fabric_ind).ua(length_ind) < Min(length_ind,3,1)
            Min(length_ind,3,1) = Drag(fabric_ind).ua(length_ind);
            Min(length_ind,3,2) = fabric_ind;
        end
        if Drag(fabric_ind).ua(length_ind) > Max(length_ind,3,1)
            Max(length_ind,3,1) = Drag(fabric_ind).ua(length_ind);
            Max(length_ind,3,2) = fabric_ind;
        end
        % Lower Arm (length_ind,4,:)
        if Drag(fabric_ind).la(length_ind) < Min(length_ind,4,1)
            Min(length_ind,4,1) = Drag(fabric_ind).la(length_ind);
```

```
            Min(length_ind,4,2) = fabric_ind;
        end
        if Drag(fabric_ind).la(length_ind) > Max(length_ind,4,1)
            Max(length_ind,4,1) = Drag(fabric_ind).la(length_ind);
            Max(length_ind,4,2) = fabric_ind;
        end
    end
end

% Min and Max Power
% Loop throught each segment of each body part. Loop through segments.
% Store min and max in matricies. Fabric index stored in third dimension.
%            UL  LL  UA  LA
%   segment1
%   segment2
%   segment3
%   segment4
% Initialize the min and max matricies.
MinP = flintmax * ones(4);
MaxP = zeros(4);
for length_ind = 1:n
    for fabric_ind = 1:8
        % Upper Leg (length_ind,1,:)
        if Power(fabric_ind).ul(length_ind) < MinP(length_ind,1,1)
            MinP(length_ind,1,1) = Power(fabric_ind).ul(length_ind);
            MinP(length_ind,1,2) = fabric_ind;
        end
        if Power(fabric_ind).ul(length_ind) > MaxP(length_ind,1,1)
            MaxP(length_ind,1,1) = Power(fabric_ind).ul(length_ind);
            MaxP(length_ind,1,2) = fabric_ind;
        end
        % Lower Leg (length_ind,2,:)
        if Power(fabric_ind).ll(length_ind) < MinP(length_ind,2,1)
            MinP(length_ind,2,1) = Power(fabric_ind).ll(length_ind);
            MinP(length_ind,2,2) = fabric_ind;
        end
        if Power(fabric_ind).ll(length_ind) > MaxP(length_ind,2,1)
            MaxP(length_ind,2,1) = Power(fabric_ind).ll(length_ind);
            MaxP(length_ind,2,2) = fabric_ind;
        end
        % Upper Arm (length_ind,3,:)
        if Power(fabric_ind).ua(length_ind) < MinP(length_ind,3,1)
            MinP(length_ind,3,1) = Power(fabric_ind).ua(length_ind);
            MinP(length_ind,3,2) = fabric_ind;
        end
        if Power(fabric_ind).ua(length_ind) > MaxP(length_ind,3,1)
            MaxP(length_ind,3,1) = Power(fabric_ind).ua(length_ind);
            MaxP(length_ind,3,2) = fabric_ind;
        end
        % Lower Arm (length_ind,4,:)
        if Power(fabric_ind).la(length_ind) < MinP(length_ind,4,1)
            MinP(length_ind,4,1) = Power(fabric_ind).la(length_ind);
            MinP(length_ind,4,2) = fabric_ind;
        end
        if Power(fabric_ind).la(length_ind) > MaxP(length_ind,4,1)
            MaxP(length_ind,4,1) = Power(fabric_ind).la(length_ind);
            MaxP(length_ind,4,2) = fabric_ind;
```

```
        end
    end
end

%% Results - Visual
%*************************************************************************
% Horizontal bar graph for each limb comparing best and worst segment.
% Subplot1: UL
% Subplot2: LL
% Subplot3: UA
% Subplot4: LA
%*************************************************************************
scrsz = get(groot,'ScreenSize');
fig = figure;
%UL
subplot(2,2,1)
    x = [1 2 3 4];
    y = [Max(1,1),Min(1,1);...
         Max(2,1),Min(2,1);...
         Max(3,1),Min(4,1);...
         Max(4,1),Min(4,1)];
    barh(x,y,'grouped')
    title('Upper Leg')
    legend('Max Drag Value','Min Drag Value')
    ylabel('Segment Number: 1-Hip \rightarrow 4-Knee')
    xlabel('Drag, D [lb_f]')

%LL
subplot(2,2,2)
    x = [1 2 3 4];
    y = [Max(1,2),Min(1,2);...
         Max(2,2),Min(2,2);...
         Max(3,2),Min(4,2);...
         Max(4,2),Min(4,2)];
    barh(x,y,'grouped')
    title('Lower Leg')
    legend('Max Drag Value','Min Drag Value')
    ylabel('Segment Number: 1-Knee \rightarrow 4-Foot')
    xlabel('Drag, D [lb_f]')

%UA
subplot(2,2,3)
    x = [1 2 3 4];
    y = [Max(1,3),Min(1,3);...
         Max(2,3),Min(2,3);...
         Max(3,3),Min(4,3);...
         Max(4,3),Min(4,3)];
    barh(x,y,'grouped')
    title('Upper Arm')
    legend('Max Drag Value','Min Drag Value')
    ylabel('Segment Number: 1-Shoulder \rightarrow 4-Elbow')
    xlabel('Drag, D [lb_f]')

%LA
subplot(2,2,4)
    x = [1 2 3 4];
    y = [Max(1,4),Min(1,4);...
```

```
        Max(2,4),Min(2,4);...
        Max(3,4),Min(4,4);...
        Max(4,4),Min(4,4)];
    barh(x,y,'grouped')
    title('Lower Arm')
    legend('Max Drag Value','Min Drag Value')
    ylabel('Segment Number: 1-Elbow \rightarrow 4-Hand')
    xlabel('Drag, D [lb_f]')

fig.OuterPosition = [1 1 scrsz(3)-50 scrsz(4)];
%*********************************************************************
% Table of Best and Worst Fabrics
xlswrite('Results.xlsx',Min(:,:,2),'B2:E5')
xlswrite('Results.xlsx',Max(:,:,2),'G2:J5')

%*********************************************************************
% Horizontal bar graph for each limb comparing best and worst segment.
% Power
% Subplot1: UL
% Subplot2: LL
% Subplot3: UA
% Subplot4: LA
%*********************************************************************
scrsz = get(groot,'ScreenSize');
fig = figure;
%UL
subplot(2,2,1)
    x = [1 2 3 4];
    y = [MaxP(1,1),MinP(1,1);...
         MaxP(2,1),MinP(2,1);...
         MaxP(3,1),MinP(4,1);...
         MaxP(4,1),MinP(4,1)];
    barh(x,y,'grouped')
    title('Upper Leg')
    legend('Max Power Value','Min Power Value')
    ylabel('Segment Number: 1-Hip \rightarrow 4-Knee')
    xlabel('Power, P [lb_f*ft/s]')

%LL
subplot(2,2,2)
    x = [1 2 3 4];
    y = [MaxP(1,2),MinP(1,2);...
         MaxP(2,2),MinP(2,2);...
         MaxP(3,2),MinP(4,2);...
         MaxP(4,2),MinP(4,2)];
    barh(x,y,'grouped')
    title('Lower Leg')
    legend('Max Power Value','Min Power Value')
    ylabel('Segment Number: 1-Knee \rightarrow 4-Foot')
    xlabel('Power, P [lb_f*ft/s]')

%UA
subplot(2,2,3)
    x = [1 2 3 4];
    y = [MaxP(1,3),MinP(1,3);...
         MaxP(2,3),MinP(2,3);...
         MaxP(3,3),MinP(4,3);...
```

```
        MaxP(4,3),MinP(4,3)];
    barh(x,y,'grouped')
    title('Upper Arm')
    legend('Max Power Value','Min Power Value')
    ylabel('Segment Number: 1-Shoulder \rightarrow 4-Elbow')
    xlabel('Power, P [lb_f*ft/s]')

%LA
subplot(2,2,4)
    x = [1 2 3 4];
    y = [MaxP(1,4),MinP(1,4);...
         MaxP(2,4),MinP(2,4);...
         MaxP(3,4),MinP(4,4);...
         MaxP(4,4),MinP(4,4)];
    barh(x,y,'grouped')
    title('Lower Arm')
    legend('Max Power Value','Min Power Value')
    ylabel('Segment Number: 1-Elbow \rightarrow 4-Hand')
    xlabel('Power, P [lb_f*ft/s]')

fig.OuterPosition = [1 1 scrsz(3)-50 scrsz(4)];
```

Drag Coefficient Extraction Function

```
function Cd = dragCoefXtract (fabric, f , V , Alpha )
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          dragCoefXtract.m
%   Description:    Identifies the correct drag coeficient to apply to the
%                   model given the input variable.
%   Inputs:         fabric ~ structure containing all fabrics and
%                       corresponding drag data
%                   f ~ fabric index
%                   V ~ velocity magnitude of model data point [ft/s]
%                   Alpha ~ angle of attack of model data points [deg]
%   Outputs:        Cd ~ drag coefficient given parameters
%-------------------------------------------------------------------------
% obtain the velocity index
    if V < 50 %fps
        v_ind = 1;
    elseif V < 70
        v_ind = 2;
    elseif V < 90
        v_ind = 3;
    else
        v_ind = 4;
    end
% use fabric, f, the velocity index, v_ind, and the angle Alpha to identify
% the correct drag coefficiet, Cd. Alhpa in radians.
    if Alpha < (37.5*pi/180) %rad
        Cd = fabric(f).vel(v_ind).alpha(1);
    elseif Alpha < (52.5*pi/180)
        Cd = fabric(f).vel(v_ind).alpha(2);
    elseif Alpha < (67.5*pi/180)
        Cd = fabric(f).vel(v_ind).alpha(3);
    elseif Alpha < (82.5*pi/180)
        Cd = fabric(f).vel(v_ind).alpha(4);
    else
        Cd = fabric(f).vel(v_ind).alpha(5);
    end


end
```

Frame Extraction Function

```
function sOp =  FrameExtraction ()
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          FrameExtraction.m
%   Description:    Extracts the desired video frame for further analysis.
%   Inputs:         filename = 'UsainBolt100mSlowMo.mp4'
%   Outputs:        filename = 'OneCycleOfSprint.mp4'
%                   filename = 'FrameComparison_titles.jpg'
%-------------------------------------------------------------------------
%% Video Input
filename = 'UsainBolt100mSlowMo.mp4';

% Construct a VideoReader object associated w/ the file
% 'UsainBolt100mSlowMo'
vidObj = VideoReader(filename);

% Determine the height and width of the frames
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

% Create a MATLAB movie structure array
s = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),...
    'colormap',[]);

% Read one frame at a time using ReadFrame unitl the end of the file is
% reached. Apend data from each video frame to the structure array.
k=1;
while hasFrame(vidObj)
    s(k).cdata = readFrame(vidObj);
    k = k+1;
end

% whos s

%% Processing

% Extract the video for one sprint cycle to a unique structure, 'sOp'.
% Frame555 to Frame 710
sOp = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),...
    'colormap',[]);

indVec = 555:1:710;
for i = 1:size(indVec,2)
    ind = indVec(i);
    sOp(i) = s(ind);
end

% whos sOp
```

```
%% Video Output

% Display the 555th fram stored in 's'. This is the start of one full cycle
% in Usain Bolt's sprint - based upon the position of his left lower-arm.
% Additionally, display frame 710 - the last frame in one full sprint cycle,
% once again defined by the vertical postion of his lower left arm.
F = figure;
hold on
subplot(1,2,1)
    image(s(555).cdata)
    axis square
    ax = gca;
%     ax.Visible = 'off';
    ax.Title.String = 'Frame 555';
subplot(1,2,2)
    image(s(710).cdata)
    title('Frame 710')
    axis square
    ax = gca;
%     ax.Visible = 'off';
    ax.Title.String = 'Frame 710';
scrsz = get(groot,'ScreenSize');
F.OuterPosition = [1 1 scrsz(3)-50 scrsz(4)];
hold off

%% Export one cycle video
savetit = 'OneCycleOfSprint.mp4';
profile = 'MPEG-4';

% Setting save parameters.
% Save to:
%   OneCycleOfSprint.mp4
writer = VideoWriter(savetit,profile);
writer.Quality = 100;

% Save in shortened video.
open(writer);
    writeVideo(writer,sOp);
close(writer);

end
```

Scaling Vector Function

```
function vec = scalingVec (n)
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          scalingVec.m
%   Description:    Creates a scaled vector to create n evenly spaced
%                   points along a cylinder, including the start and end
%                   points along it.
%   Inputs:         n ~ the number of points (n-1 regions)
%   Outputs:        vec ~ the unit vector for the cylinder. Multiply by a
%                       scalar length to apply to a cylinder.
%-------------------------------------------------------------------------

vec = zeros(n,1);
for i = 1:n
    ind = n - i + 1;
    vec(ind) = (n-i)/(n-1);
end

end
```

Velocity Component of Model Function

```
function [v_ab, v_bc] = VCalc(Vinf, L_ab, L_bc, th_ab, w_ab, th_bc, w_bc,
n_x)
%-------------------------------------------------------------------------
%
%   Institution:    Texas Christian University
%   Project:        Senior Honors Thesis
%                   Minimizing Drag Effects on a Sprinter
%   Author:         Brendan Daly
%   Title:          Velocity Component Calculator for Sprinter Body Model
%   Description:    Calculate the net velocities of various points along
%                   the arms and legs of a sprinter model. The velocity
%                   data will then be used for a parametric study to
%                   minimize the drag on the sprinter.
%                   Upper Limb ~ ab
%                   Lower Limb ~ bc
%                   V_ab(x) = [ Vinf - L_ab(x)*sin(theta_ab_i)*w_ab_i ] * I
...
%                           + [ L_ab(x)*cos(theta_ab_i)*w_ab_1 ] * J;
%                   V_bc(x) = [ Vinf - L_ab*sin(theta_ab_i)*w_ab_i ...
%                             - L_bc(x)*sin(theta_bc_i)*w_bc_i ] * I ...
%                           + [ L_ab*cos(theta_ab_i)*w_ab ...
%                             + L_bc(x)*cos(theta_bc_i)*w_bc_i ] * J;
%   Inputs:         Vinf    ~ rigid body velocity
%                           [in/s]
%                   L_ab    ~ model length of the upper limb (hinged to the
%                               rigid body)
%                           [in]
%                   L_bc    ~ model length up the lower limb (hinged to Lab)
%                           [in]
%                   th_ab   ~ vector containing absolute angle data for each
%                               of the frames for length ab
%                           [rad]
%                   w_ab    ~ vector containing absolute angular velocity
%                               data for each of the frames for length ab
%                           [rad/s]
%                   th_bc   ~ vector containing absolute angle data for each
%                               of the frames for length ab
%                           [rad]
%                   w_bc    ~ vector containing absolute angular velocity
%                               data for each of the frames for length ab
%                           [rad/s]
%   Outputs:        v_ab    ~ structure containing fields i & j. Each field
%                               is a vector containing the various velocity
%                               points along body ab.
%                   v_bc    ~ structure containing fields i & j. Each field
%                               is a vector containing the various velocity
%                               points along body ab.
%
%                   v_ab or v_bc can be accessed using the notation:
%                       v_ab(x).i  and  v_ab(x).j
%                       x ~ index from 1:n_x of velocity calculations down
%                           a rigid body ab or bc.
%                       Example:
%                         + v_ab(1).i is the i component of velocity at the
%                           joint a.
```

```
%                          + v_bc(2).j if n_x=3 is the j component of
%                            velocity at the second of 3 points along the
%                            body bc. This would be the midway point.
%
%-------------------------------------------------------------------------

% Discretize the length data in order to obtain n_x velocity data points.
% n_x = 3;
scale_vec = scalingVec(n_x);
l_ab = L_ab*scale_vec;
l_bc = L_bc*scale_vec;

% Segment the equations
termI_1 = Vinf;
termI_2 = -sin(th_ab).*w_ab;
termI_3 = -sin(th_bc).*w_bc;

termJ_1 = cos(th_ab).*w_ab;
termJ_2 = cos(th_bc).*w_bc;

% Loop through the lengths for l_ab and store in struct v_ab
v_ab = struct('i',[],'j',[]);
for k = 1:n_x
    v_ab(k).i = termI_1 + l_ab(k)*termI_2;
    v_ab(k).j = l_ab(k)*termJ_1;
end

% Loop through the lengths for l_bc and store in struct v_bc
v_bc = struct('i',[],'j',[]);
for k = 1:n_x
    v_bc(k).i = termI_1 + l_ab(n_x)*termI_2 + l_bc(k)*termI_3;
    v_bc(k).j = l_ab(n_x)*termJ_1 + l_bc(k)*termJ_2;
end


end
```

Excel Column Converter Function (Downloaded from free function site)

```
function b = xlscol(a)

%XLSCOL Convert Excel column letters to numbers or vice versa.
%   B = XLSCOL(A) takes input A, and converts to corresponding output B.
%   The input may be a number, a string, an array or matrix, an Excel
%   range, a cell, or a combination of each within a cell, including nested
%   cells and arrays. The output maintains the shape of the input and
%   attempts to "flatten" the cell to remove nesting.  Numbers and symbols
%   within strings or Excel ranges are ignored.
%
%   Examples
%   --------
%       xlscol(256)   % returns 'IV'
%
%       xlscol('IV')  % returns 256
%
%       xlscol([405 892])  % returns {'OO' 'AHH'}
%
%       xlscol('A1:IV65536')  % returns [1 256]
%
%       xlscol({8838 2430; 253 'XFD'}) % returns {'MAX' 'COL'; 'IS' 16384}
%
%       xlscol(xlscol({8838 2430; 253 'XFD'})) % returns same as input
%
%       b = xlscol({'A10' {'IV' 'ALL34:XFC66'} {'!@#$%^&*()'} '@#$' ...
%         {[2 3]} [5 7] 11})
%       % returns {1 [1x3 double] 'B' 'C' 'E' 'G' 'K'}
%       %   with b{2} = [256 1000 16383]
%
%   Notes
%   -----
%       CELLFUN and ARRAYFUN allow the program to recursively handle
%       multiple inputs.  An interesting side effect is that mixed input,
%       nested cells, and matrix shapes can be processed.
%
%   See also XLSREAD, XLSWRITE.
%
%   Version 1.1 - Kevin Crosby

% DATE      VER  NAME          DESCRIPTION
% 07-30-10  1.0  K. Crosby     First Release
% 08-02-10  1.1  K. Crosby     Vectorized loop for numerics.

% Contact: Kevin.L.Crosby@gmail.com

base = 26;
if iscell(a)
  b = cellfun(@xlscol, a, 'UniformOutput', false); % handles mixed case too
elseif ischar(a)
  if ~isempty(strfind(a, ':')) % i.e. if is a range
    b = cellfun(@xlscol, regexp(a, ':', 'split'));
  else % if isempty(strfind(a, ':')) % i.e. if not a range
    b = a(isletter(a));        % get rid of numbers and symbols
    if isempty(b)
      b = {[]};
```

```
    else % if ~isempty(a);
      b = double(upper(b)) - 64; % convert ASCII to number from 1 to 26
      n = length(b);             % number of characters
      b = b * base.^((n-1):-1:0)';
    end % if isempty(a)
  end % if ~isempty(strfind(a, ':')) % i.e. if is a range
elseif isnumeric(a) && numel(a) ~= 1
  b = arrayfun(@xlscol, a, 'UniformOutput', false);
else % if isnumeric(a) && numel(a) == 1
  n = ceil(log(a)/log(base));  % estimate number of digits
  d = cumsum(base.^(0:n+1));   % offset
  n = find(a >= d, 1, 'last'); % actual number of digits
  d = d(n:-1:1);               % reverse and shorten
  r = mod(floor((a-d)./base.^(n-1:-1:0)), base) + 1;  % modulus
  b = char(r+64);  % convert number to ASCII
end % if iscell(a)

% attempt to "flatten" cell, by removing nesting
if iscell(b) && (iscell([b{:}]) || isnumeric([b{:}]))
  b = [b{:}];
end % if iscell(b) && (iscell([b{:}]) || isnumeric([ba{:}]))
```