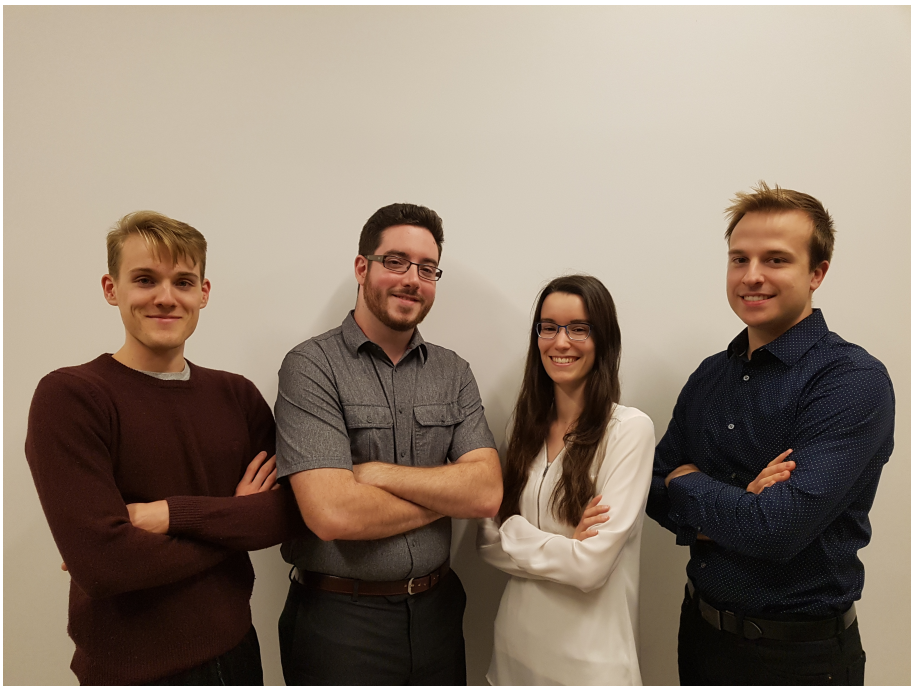
Waterfront 2A - Capstone Report
MCG4322



Waterfront Robot 2A

Marc-André Arsenault (8172498)
Mathieu Carroll (8089784)
Alexane Lahaie (8204533)
Joshua O'Reilly (8359885)

University of Ottawa
Department of Mechanical Engineering
December 06, 2019

Abstract

The O-Crab is an autonomous biologically inspired robot designed to navigate the harsh conditions and terrain of waterfronts and collect the ever increasing amounts of litter found near waterways. To do so, it is equipped with five crab-like legs, each providing two degrees of freedom. These legs are powered by a set of motors and harmonic drive reducers. The O-Crab's fully waterproof construction and resistant materials enable it to operate in moisture and potentially salt intensive environments. Thanks to lid-mounted solar panels, it operates independently from centralized electricity sources and for long periods of time by alternating between solar power and the on-board batteries.

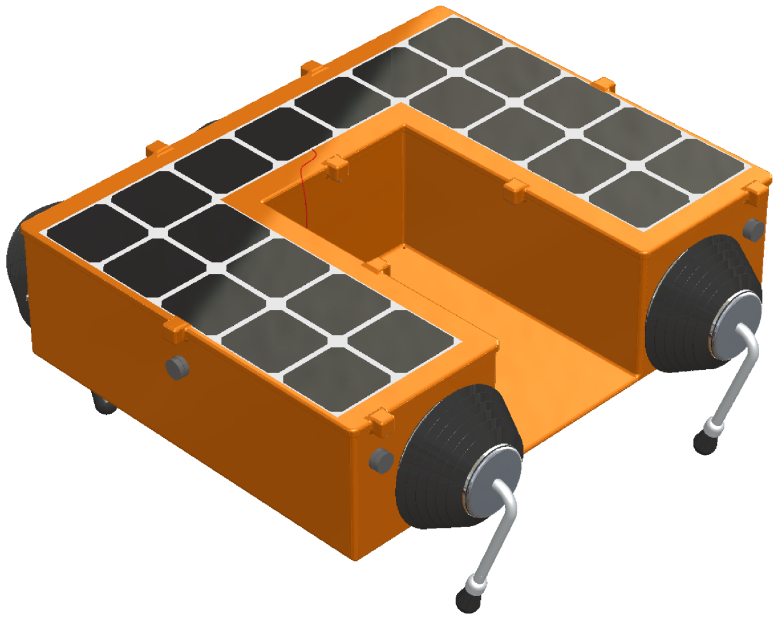


Figure 1: The O-Crab waterfront robot

Contents

List of Figures

List of Tables

1 Project Charter

1.1 Mandate

Group WR2A is mandated to develop a rugged device which uses a biomimetic inspired locomotion system to remove waste from waterfronts. It must be self-reliant and resistant to exterior environments such as areas with minimal accessibility, rough weather and arduous terrain.

1.2 Requirements

The design is to be solar powered and have a biomimetic locomotion system. It must operate in rain, water, high heat, and navigate terrain such as sand, mud, plants and bramble. Human intervention should only be necessary to empty the litter container. It should resist vandalism. The litter collector size will vary from 1 to 5000 cm^3 and 1 to 5 kg.

1.3 Constraints

The device should not have any continuously rotating joints. Additionally, bellows must be used to waterproof joints instead of o-rings to avoid corrosion damage from salt water.

1.4 Criteria

Power consumption per kilogram (device and litter) should be minimized. The operating time should be maximized for better litter collection efficiency. The robot's capacity to navigate a variety of environments, including sand and pebble beaches, shallow water, mud and around small plants, should be maximized. The mechanical stability of the robot should be optimized as well to allow navigation of rough terrain. Finally, aesthetics should be considered as it will operate in public spaces.

1.5 Design Optimization

The user may input the desired litter weight, litter box size, and horizontal and vertical leg reach. The program outputs component dimensions based on structural and geometrical analyses, as well as specifications such as the number of solar cells, operating time per day, walking speed and total mass.

2 Proposed Design

2.1 The O-Crab Design

The general assembly of the O-Crab locomotion and chassis is shown in Figure 2. The robot recharges using Maxeon SunPower Solar cells on the top of the chassis [1].

Due to the nature of the robot's application, the chassis is constructed of acrylic as it is UV stable, resistance to cold weather and salt, and relatively strong [2] [3]. Safety orange was selected for high visibility. It was designed to be fully sealed from the elements (sand, moisture, salt, dust, etc). This was achieved by the use of molded silicone bellows at the leg joints, which are flexible enough to allow the repeating angular motion of the legs [4]. Another aspect of sealing is the use of a compressed gasket between the chassis and its lid [5]. This feature is shown in Figure 3, where the gasket is positioned inside a groove which protrudes upward, creating a further barrier for water and particles. To prevent vandalism, tamper resistant bolts were used for the lid of the robot, making it difficult to dismantle [6]. Furthermore, any fasteners penetrating the chassis (other than those holding the lid due to the gasket) have a compressed o-Ring to prevent water and contaminant ingress through the hole [7].

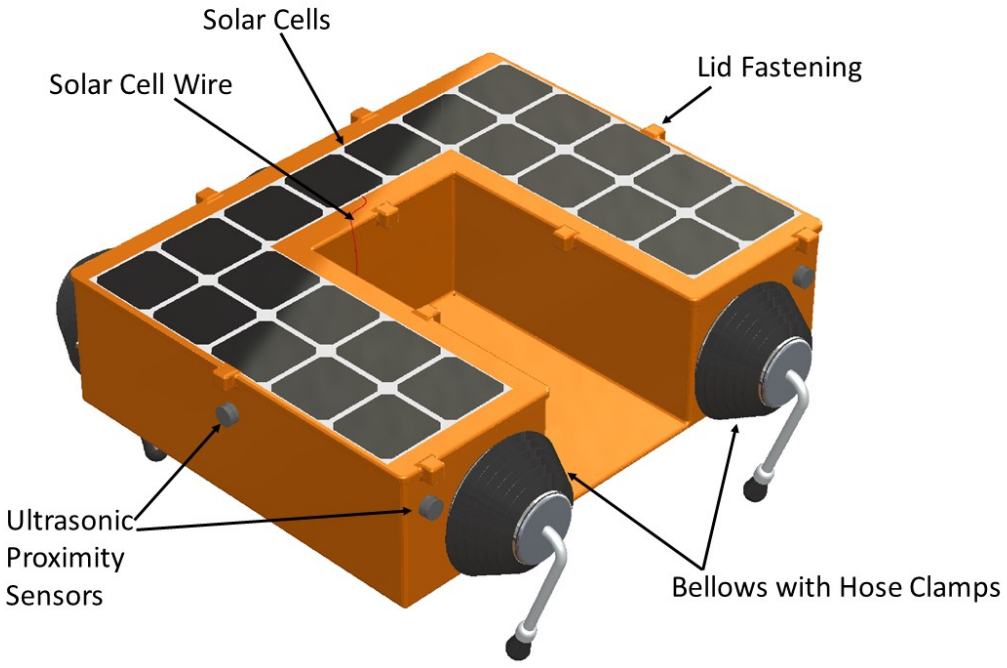


Figure 2: O-Crab general assembly without litter system

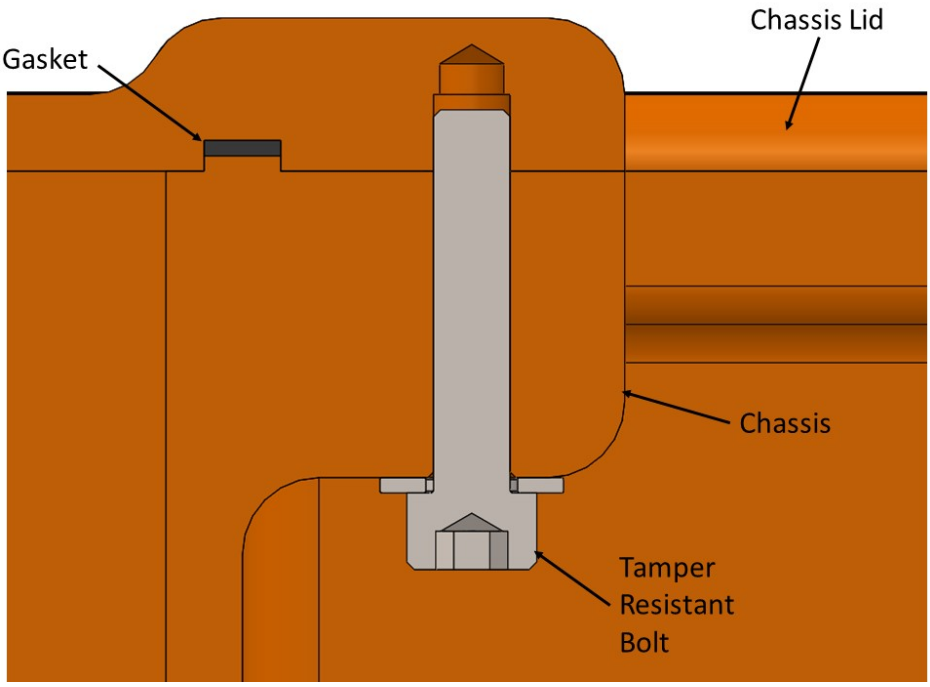


Figure 3: Chassis gasket

2.1.1 Electronics

The robot is also equipped with sensor equipment, enabling independent operation and navigation of its environment. Ultrasonic proximity sensors are mounted through all four sides of the chassis to detect nearby objects. [8]. This will allow it to avoid obstacles and people [9]. The motors each have a built in hall sensor for leg position and velocity control [10].

Other sensor equipment is found inside the chassis. A GPS/IMU unit from Inertial Sense allows for programming way-points, path-planning and retrieval assistance in case the robot becomes damaged or stolen [11]. The IMU, or inertial measurement unit, detects orientation and acceleration of the chassis and will be used in the control system [12].

These elements are shown in Figure 4 which shows a top view of the robot, with the chassis lid removed. Other elements shown in the top view are the controllers. An NVIDIA Jetson TX2 Module leverages its GPU to perform computer vision tasks and GPU based SLAM [13] [14]. A Raspberry Pi 4 B is used for all other functions of the robot such as the overall control loop and communication [15] [16]. oDrives power the leg and arm motors. Each one can control two motors simultaneously and allows for over 100A draw at 48V, well

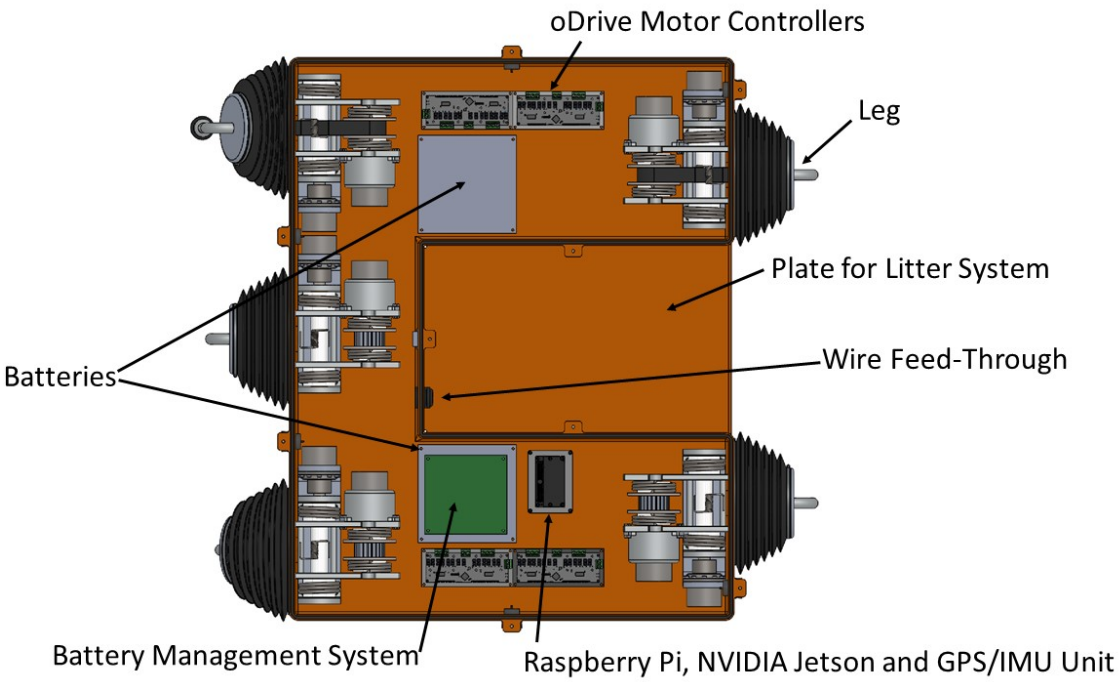


Figure 4: Top view of component layout inside the chassis

within operating conditions [17].

Also visible in Figure 4 are the batteries and battery management system. The batteries stock energy from the solar cells for when the robot is operating, and are composed of Panasonic NRC18650B cells [18]. The battery management system ensures acceptable charge and discharge rates between cells [19]. As shown in the top view, the cells are split into two batteries, one on either side of the robot to distribute the weight more evenly.

2.1.2 Leg Configuration and Movement

Figure 4 also shows the leg configuration (note that some of the timing belts were removed to show other leg details). Three legs are placed at the back and two at the front, with the litter and robot arm between them (see Section 2.2). The legs have two degrees of freedom, allowing them to extend in and out. A third degree of freedom for rotation of the leg around the vertical axis of the robot was not added to reduce complexity. The robot turns by moving legs on one side (right or left) faster than the other side. The two front legs are positioned as close to each other as possible. This makes the robot more stable, as shown in the stability triangle analysis of the Modelling Report.

2.1.3 Major Leg Components

Figure 5 shows a section view of the foot assembly. The semi-spherical elongated shape was chosen to ensure the foot would not dig in deep or get caught in mud or sand. A flexible silicone "sock" was chosen as the outer material to provide some damping to the legs. Its flexibility also allows it to be slipped on and secured in the ridge on the foot rod by use of its elasticity. Once installed, it is further secured by a cap which is screwed down onto the outer portion of the silicone. This cap would have been installed on the foot rod, which is manufactured with a threaded portion, prior to installing the silicone sock. The foot assembly is then press fit into the tibia tube.

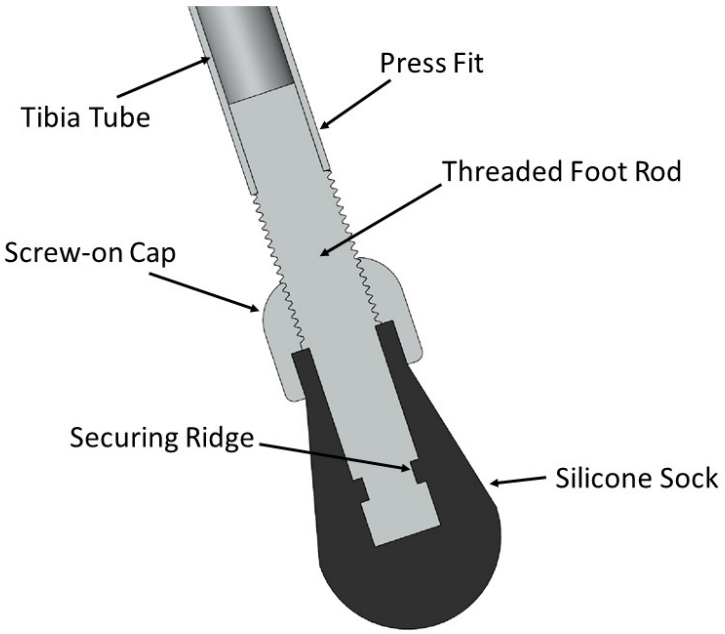


Figure 5: Section view of foot assembly

As shown in the leg section side view of Figure 6, the robot uses a belt and pulley system to rotate the knee, reducing the inertia of the leg and the required bellow size by keeping the Harmonic Drives and motors within the chassis. A steel reinforced HTD8 urethane timing belt was used for synchronisation and to prevent slipping [20]. To facilitate assembly of the belt and ensure that belt stretching with temperature and usage is compensated, a small torsion spring tensioner is added directly onto the belt shown in Figure 7. As the movement of the leg is oscillatory instead of performing full rotations, this spring is positioned as to

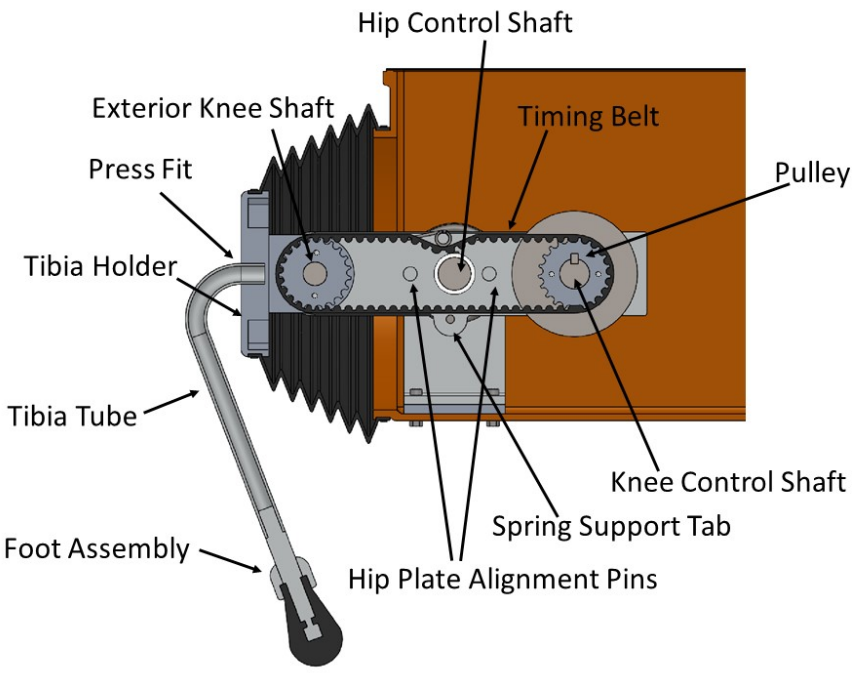


Figure 6: Side section view of leg showing belt assembly

not interfere with the pulleys.

The bellow is attached to the chassis using a hose clamp around a protruding circular flange on the chassis. The other end of the bellow is attached to the tibia holder, which simultaneously connects the bellow, the exterior knee pulley and the tibia tube. The tibia holder's large diameter is required for the bellow, as it needs to be larger than the hypotenuse of the shaft length and thigh plate height. It has two "arms" extending out which are bolted to both sides of the pulley, shown in Figure 7. The tibia holder must be relatively thick as the tibia tube is press fit into it. Thus, some material was removed between the various features, as the outer portion holding the bellow does not take much force. The tibia tube was chosen as an aluminum circular tube to remain lightweight.

Figure 7 shows a top view of the thigh portion of the leg without the chassis or bellow. Contrary to the tibia hollow tube, it consists only of two two hip plates. This design was chosen as it takes less space than a circular tube for the same size of belt system. It also makes the leg easier to assemble and reduces the complexity of the leg. The plates are made relatively thick to support the exterior knee shaft, knee control shafts and to be able to fix them onto the hip shaft with keys.

Alignment pins for the hip plates are found on either side of the hip control shaft. They

extend from one hip plate and are inserted into holes inside the other hip plate. They allow the holes for the shafts to be machined simultaneously in the correct final plate position, ensuring proper shaft alignment. Their second purpose is also to provide some additional structure to the hip plates, and to secure the plates together by using a bolt at the end of each pin.

Both the hip control shaft and knee control shaft are connected to a Harmonic Drive reducer and electric DC motor [21] [10]. The motor and Harmonic Drive connected to the knee control shaft are mounted directly to the hip plate using the knee adaptor, with the Harmonic Drive hidden inside the adaptor. This entire assembly moves with the hip plates.

Torsion springs are positioned on the hip control shaft and knee control shaft. While the legs are not moving, the joint torques are brought below the Harmonic Drive non-backdriving torque, allowing them to passively hold up the robot. This solution provided significant power savings, as the robot generally moves only one leg at a time off the ground. The only drawback with this solution is that the moving leg will have to use more energy to fight the springs, however, this value is much less than constantly having all legs powered. Two identical but mirrored springs were used for each shaft, as this provided springs of smaller diameter, making the hip plates more compact and the shaft assemblies more symmetrical.

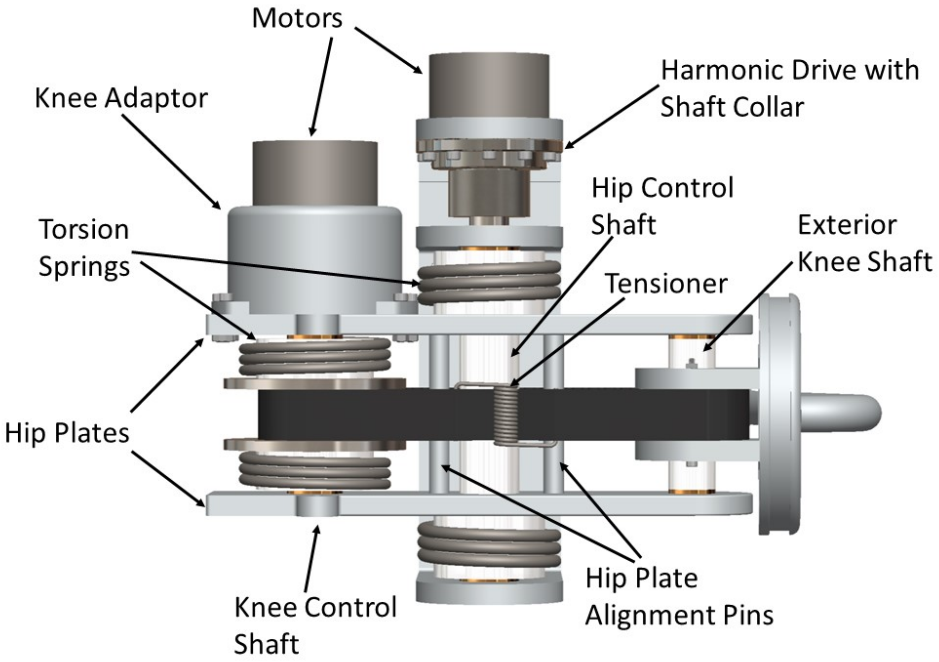


Figure 7: Top view of leg thigh without chassis or bellow

2.1.4 Leg Shafts Assemblies

Figure 8 shows a section view of the exterior knee shaft and its components. Spacers are used to position the components axially. The pulley has no key as it is directly connected to the tibia holder using bolts. The exterior knee shaft is only structural and does not transfer any torque. The shaft is supported by sintered bronze sleeve bearings with flanges [22]. The other shafts are also supported by sleeve bearings as shown in Figures 9 and 11. These were selected over ball or needle bearings as the legs are often static, and radial forces on the shafts were quite high due to the belt tensions and forces from the legs. The flanges were added to prevent the shaft step and spacer from rubbing on the plates and to position the sleeve bearings in the plates.

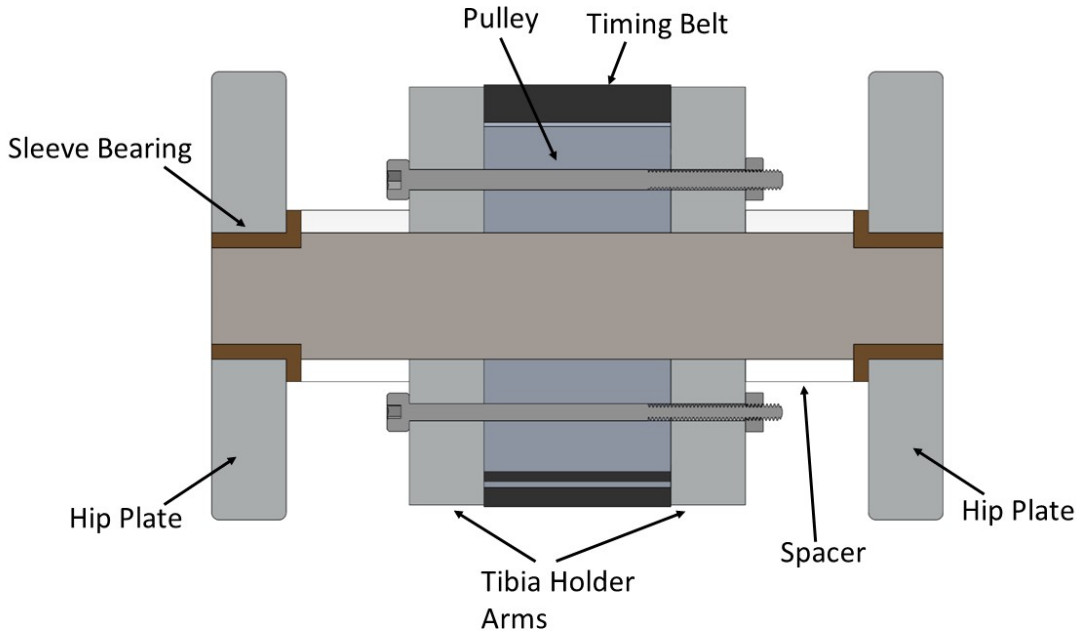


Figure 8: Section view of exterior knee shaft

Figure 9 shows a section view of the hip control shaft, its components and the brackets and adaptor used to support it. The hip plates are held on the shaft by keys and are axially positioned by spacers. The spacers at the springs are larger to support the springs as they move.

The shaft is supported by two L-brackets which are attached to a base plate, which is then attached to the chassis. One end of the springs connect to the hip plates and the other

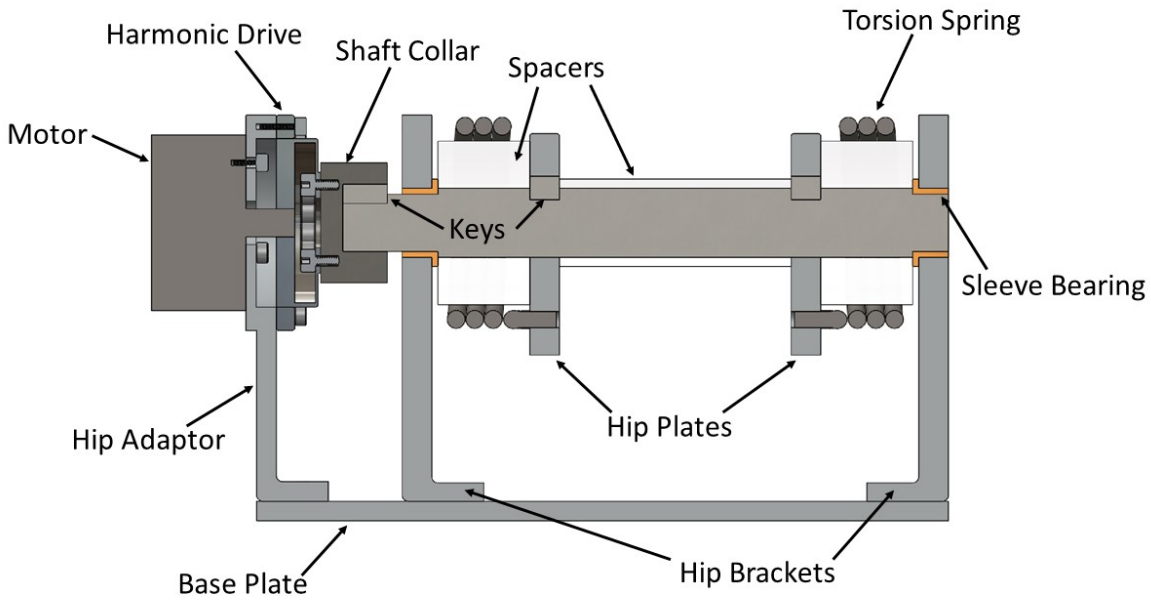


Figure 9: Section view of hip control shaft and its supporting brackets and adaptor

connects to its corresponding L-bracket. If the spring is larger than the hip plate height, a "tab" of additional material is added on the hip plate instead of increasing its total height, which would increase the size of other parts. This detail is shown in Figure 6. To allow for the proper alignment of the hip control shaft, the L-brackets each have two alignment pins at their base which are positioned into holes on the base plate. This assembly is shown in Figure 10. To support the motor and Harmonic Drive, an adaptor piece was created, and extends down to the base plate. It has a similar shape to the L-brackets and uses alignment pins as well. The motor is mounted first onto the adaptor, then the Harmonic Drive.

The shaft collar used to connect the output of the Harmonic Drive to the hip control shaft is also of interest. As the stainless steel shafts are larger in diameter than typical steel shafts would have been, a regular flange collar did not work for the connection, as the Harmonic Drive flexspline bolt holes are quite close together. A flange-less and slightly thicker custom collar was thus created for the purpose of this application.

Figure 11 shows a section view of the knee control shaft, its components and the adaptor used to support its motor and Harmonic Drive. The knee adaptor uses a similar configuration to the adaptor at the hip control shaft, but attaches to the hip plate instead of the base plate and chassis. The pulley uses a keyway as this shaft transmits torque, unlike the exterior

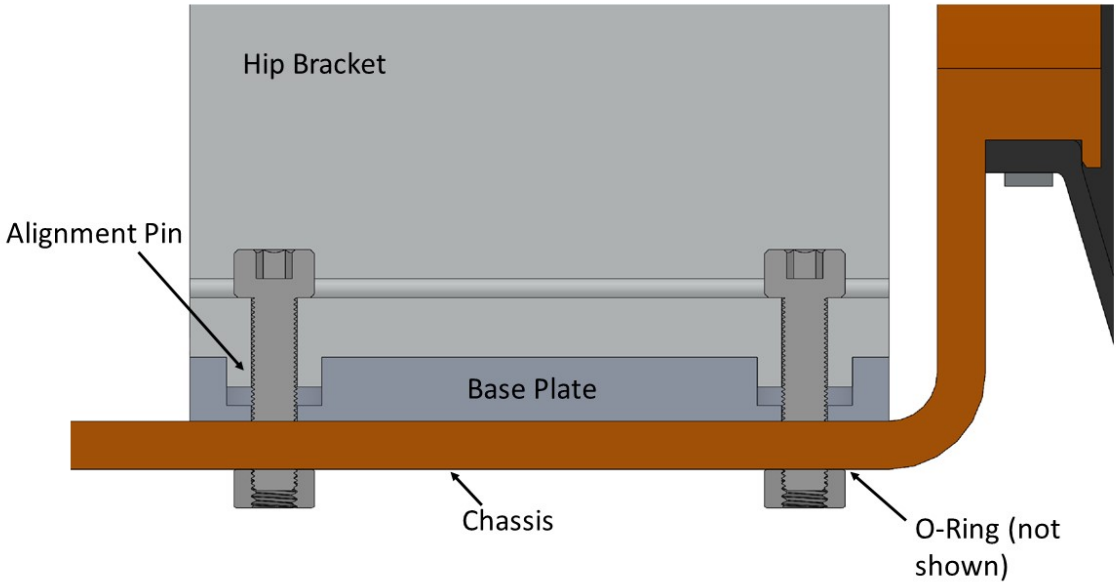


Figure 10: Section view of hip bracket alignment pins and fastening to chassis

knee shaft. A pulley plate is attached on either side of the pulley (fasteners are not visible in this view as they were made to be offset from the keyway). The purpose of these parts is to provide a surface on which to attach the torsion springs. The other side of the springs is attached on the hip plates (with an added "tab" if required, as with the hip springs). Once again, a larger spacer is used to support the spring. Pins are used to fasten the pulley plates to the pulley instead of bolts to avoid interference. As with the hip control shaft, a custom shaft collar is used to attach the Harmonic Drive output to the shaft.

2.2 Interconnectivity

Figure 12 shows the robot combined with the litter collection system. As the litter collection system is fully waterproof, the chassis has a dedicated plate for mounting on its exterior, positioned at the front center of the robot. This arrangement allows the litter collection system to have a better view of potential litter when the robot is moving around the beach. The plate is also positioned at the bottom of the chassis height, reducing the necessary length of the litter collector to reach the ground and lowering the center of gravity. The area of the plate can easily be modified to accommodate various sizes of litter collection systems,

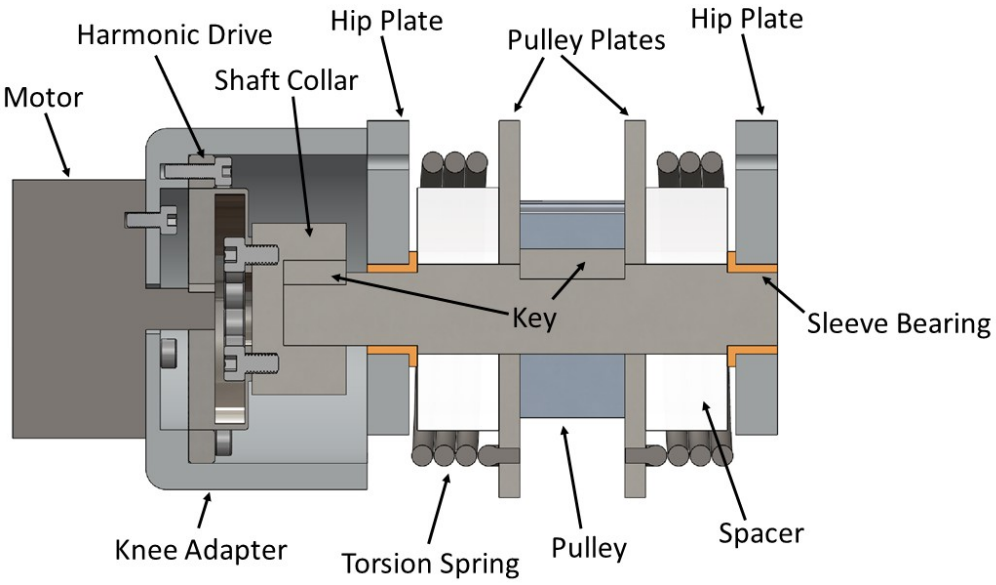


Figure 11: Section view of knee control shaft and its adaptor

depending on litter box dimensions. The spacing of the front legs and area on the lid for solar panels change accordingly to accommodate various litter boxes.

Another advantage of the plate design is that the fasteners used to hold the litter system do not go through the waterproofed chassis. The only connection to the inside of the chassis is for the electrical wires. A waterproof wire feed through seal is used for this purpose [23], as shown in Figure 4. Small holes were added in the corner of the mounting plate to evacuate any accumulated water.

The oDrive controllers for the litter collector are placed inside the chassis.

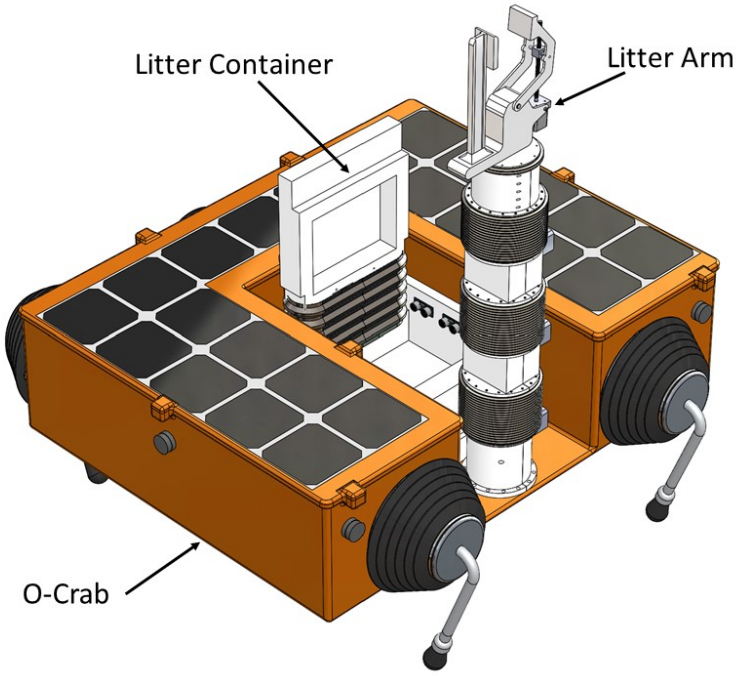


Figure 12: Robot with litter collector system

3 Parametrization

3.1 Outline

Parameterization inputs include the litter box volume, maximum litter mass, and desired horizontal (x) and vertical (y) reach of the legs. The litter mass is limited between 0kg and 5kg, as per the project mandate. The litter box volume is limited between 5000cm^3 and 15000cm^3. These values were determined based on the project mandate and dimensions provided by group WR2B. The maximum reach in x and y determine the obstacles the robot can navigate (for example: stairs, size of rocks, crevices). They are limited between 170mm and 405mm, and 40mm and 95mm respectively. A different design solution would be required to operate outside these intervals, as detailed in the discussion.

Table 1 only lists the significant key components being parameterized based on the user inputs, as well as the range of their key parameterized dimensions. All components have parameterized dimensions which may not be shown in the table.

Table 1: Summary of parameterized parts (Str. : Structural, Geo. : Geometric)

Assembly	Component Optimized Parameters	Type	Optimized Dimensions	Units
Exterior Knee Shaft	Large diameter	Str.	$15.0 < D_{S\_K} < 24.5$	mm
	Small diameter	Str.	$11.0 < d_{S\_K} < 20.5$	mm
	Length of shaft	Str.	$90.9 < L_{S\_K} < 126.2$	mm
Hip Control Shaft	Large diameter	Str.	$21.5 < D_{S\_H} < 39.5$	mm
	Small diameter	Str.	$17.5 < d_{S\_H} < 35.5$	mm
	Length of shaft	Str.	$153.5 < L_{S\_H} < 232.1$	mm
Knee Control Shaft	Large diameter	Str.	$19.5 < D_{S\_HK} < 35.0$	mm
	Small diameter	Str.	$15.5 < d_{S\_HK} < 31.0$	mm
	Length of shaft	Str.	$90.9 < L_{S\_HK} < 126.2$	mm
Leg	Length of lower tibia	Geo.	$228.4 < r_3 < 542.5$	mm
	Length of upper tibia	Geo.	$91.4 < r_2 < 217.0$	mm
	Length of thigh	Geo.	$76.1 < r_1 < 180.8$	mm
	Radius of bend	Geo.	$27.1 < r_{bend} < 66.2$	mm
	Outer tube diameter	Str.	$16.0 < D_{tube} < 44.0$	mm
	Inner tube diameter	Str.	$12.8 < d_{tube} < 40.8$	mm
Pulley & Belt	Pitch diameter	Str.	$50.9 < D_{pulley} < 188.4$	mm
Springs Knee	Coil Diameter	Str.	$42.0 < D_{sp\_K} < 89.0$	mm
	Wire Diameter	Str.	$4.0 < d_{w\_K} < 8.4$	mm
	Number of Turns	Str.	$2 < n_{sp\_K} < 2$	quantity
Spring Hip	Coil Diameter	Str.	$35.0 < D_{sp\_H} < 80.0$	mm
	Wire Diameter	Str.	$4.8 < d_{w\_H} < 10.2$	mm
	Number of Turns	Str.	$2 < n_{sp\_H} < 2$	quantity
Motor Hip	Outer Diameter	Geo.	$52.9 < D_{m\_H} < 145.1$	mm
	Thickness	Geo.	$30.6 < t_{m\_H} < 55.3$	mm
Motor Hip Knee	Outer Diameter	Geo.	$45.7 < D_{m\_HK} < 110.7$	mm
	Thickness	Geo.	$29.2 < t_{m\_HK} < 43.5$	mm
Harmonic Drive Hip	Outer Diameter	Geo.	$73.0 < D_{HD\_H} < 145.1$	mm
	Thickness	Geo.	$15.0 < t_{HD\_H} < 28.5$	mm
Harmonic Drive Hip Knee	Outer Diameter	Geo.	$68.8 < D_{HD\_HK} < 110.7$	mm
	Thickness	Geo.	$14.2 < t_{HD\_HK} < 22.0$	mm
Battery	Width	Geo.	$93.2 < w_B < 149.0$	mm
	Length	Geo.	$93.2 < l_B < 149.0$	mm
Hip Bracket	Height	Geo.	$93.2 < h_{br} < 199.9$	mm

3.2 High Level Parameterization

The main program loop calculates the robot weight. This incorporates all calculations involving forces and component geometry, as both will vary with and influence the weight. The first components calculated are the Leg and Force object, which contain important properties such as the torque at joints and lengths of linkages. The springs are calculated before the shafts, as the shafts' lengths are dependent on the springs' length. The order of the remaining components has no impact on their calculations. The order was chosen based on which components are geometrically dependent and the order which enabled the most components to be fully defined at their initialization.

An initial weight prediction is given, and is either reduced or increased at each iteration as the estimate becomes more precise. Once the total weight of the robot varies by less than 1 kg, the program exits the main iteration cycle and prints the required properties to all text files. A high level flow chart for the parameterization of the O-Crab is shown in Figure 13. For simplification, the exterior knee shaft is referred as knee shaft, the hip control shaft as hip shaft and knee control shaft as knee hip shaft.

3.3 Parameterization Discussion

3.3.1 Simplifications

Certain aspects of the parameterization were simplified. First, some parts were simplified to single variable parameterization problems, whereas in reality they may have multiple changing variables. For example, the belt system calculation was simplified by setting a constant belt pitch, belt width and belt thickness to reduce from volumetric parameterization to single variable parameterization. The only dimension that is calculated is the pulley diameter. The length of the belt is geometrically set by other parameters such as the length of the thigh.

When calculating the load carried by each leg, the position of each foot must be approximated to use the matrix approach as done in Section 4.3 of the Modelling report. This approach calculates higher normal forces for one foot as the centre of mass is not evenly distributed, compared to dividing the mass by 3 (loosely approximating 3 legs making contact with the ground). However, the position of the feet are not re-calculated in the program, this was due to time constraints, resulting in a simplification of the method to a ratio approach.

The ratio between the leg linkage lengths, as defined in Section 3.4 of the Analysis Report, are constant and were determined experimentally as shown in Appendix B.2. When

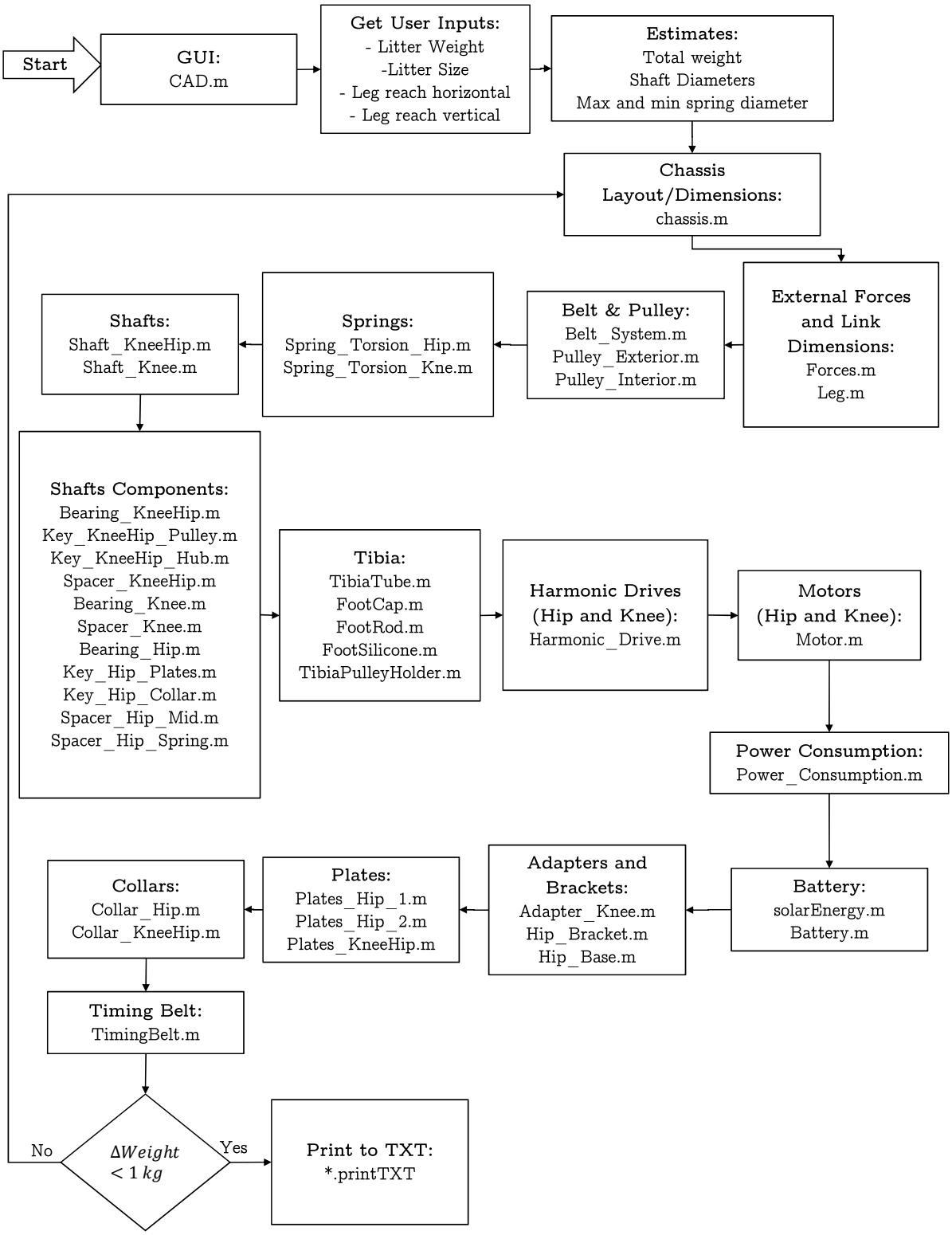


Figure 13: Master parameterization flowchart

leg reaches in x and y are passed by the GUI, they are compared to reference leg reaches and scaled accordingly. If the desired x reach is half the reference one, then the linkage lengths will also be halved. Since the leg linkages are related by ratios, this causes the y reach to scale with the x reach, in that it may be larger than the required value. A more ideal approach would have been to independently vary the lengths of the linkages to produce a configuration that more closely matches the given x and y reach.

The bellow was not parameterized as per the equations presented in Section 4.9 of the Analysis Report. The construction of the part in SolidWorks was limited in that it was not possible to change the number and size of the folds to match the equations. Thus, only the diameters of the bellow scale with the robot. The change in those values also modifies the length of the folds consequently. The thigh angle varies from $0°$ to $28°$ from horizontal, and the tibia angle varies from $-22.5°$ to $22.5°$ from the thigh. These constraints are to ensure the bellow does not stretch too far in either direction.

The solar cells were not added in the parameterized assembly due to time constraints and the complexity of positioning parts on a non-rectangular area. The number of solar cells is calculated and is output into the Matlab Command Window.

The sleeve bearings were not parameterized, as the results of the calculations gave small and sometimes unrealistic lengths. These lengths did not ensure enough penetration into the hip plates or hip brackets for a stable assembly. As a simplification, the length was set to a larger constant value. As the width of the hip plates and hip brackets is constant, the bearings were made to match that width. Their diameters scale with that of the shafts.

To ensure that the steps on the shafts were sufficiently large to properly position components axially, the difference between the small and large diameter of the shafts was always set to 4 mm.

The foot assembly was simplified by only using the tibia tube diameter as a parameterization input, and maintaining a total assembly height of 100mm. This results in a constant radius for the spherical portion of the silicone sock. However, when the size of the tibia tube changes, the shape of the silicone sock changes. When the tube is larger, the silicone becomes thin in some areas and can cause the parameterization to fail once the tube surpasses the set radius. More parameterized dimensions should be added to ensure the silicone sock keeps its required shape, regardless of how large the tibia tube becomes.

Traditional `for` loops were used for most of the components; a single variable is manipulated in the loop to find an acceptable solution, while all other variables are set as a constant or as ratios of others. The usage of vectorized loops (as shown in Appendix B.4.2)

for the springs simplified parameterizing multiple variables simultaneously. This approach could have also been applied to other parts whose geometry suffered due to variables that were set as constants or ratios.

The stability analysis summarized in Section 3.3 of the Analysis Report was not performed due to time constraints. The centre of mass of the robot would have needed to be computed, which would have been a long process due to the large amount of parts in the robot. Components were positioned in a manner to provide as much stability as possible, as per observations made in the previous stability analysis. However, the stability of the robot on the required slope was not verified.

As mentioned in the Analysis Report, the tensioner torsion spring was not parameterized as the equations were never fully developed.

The electronics were not positioned in the parameterized assembly, as the internal dimensions vary significantly between different leg reaches and litter box dimensions. There is no simple way of placing them that would satisfy all possible robot configurations. A comparison of a small and large robot with the same litter size is shown in Figure 14. The small robot has quite a bit of available space for the electronics in between the front and back legs, however the large robot would require more thought and analysis to ensure that the electronics do not interfere with the legs.

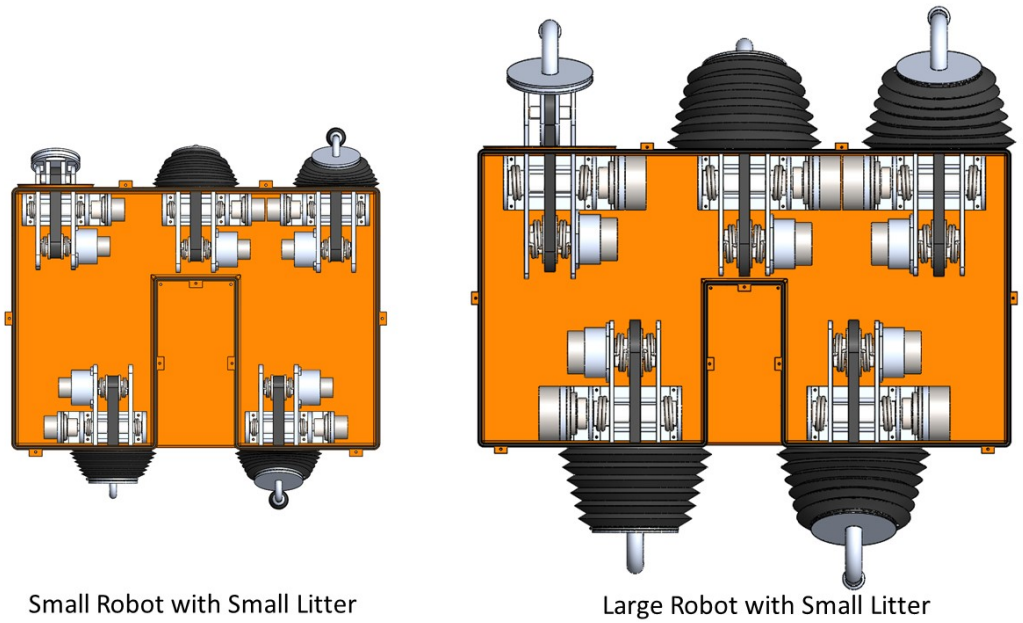


Figure 14: Comparison of available areas for electronics in two robot configurations

The torsion spring programming was also simplified slightly. The selection method should

verify the shaft and pulley diameters to ensure the spring does not interfere with other components. However this method could not be properly integrated into the programming.

3.3.2 Constrained Parts

Some parts deviate from their parameterization loop if certain conditions are met. If the motors are larger than the Harmonic Drives, their mounting bolts start interfering with the Harmonic Drive mounting bolts. Since the motors scale in size faster than the Harmonic Drives, this occurs at the upper end of the leg reach. To counteract this, the motor outer diameter and mounting diameter are limited to that of the Harmonic Drive. In reality, some design changes would need to be made. A different motor or additional gearbox such as a single stage reducer could be used. Another option would be to modify the design of the adaptor onto which the motor and harmonic drive are attached.

All of the keyed components such as the pulley, hip plates and shaft collars were set to a fixed width equal to the width of the component they are in. However, as the torques in the legs increase, some of the keys require a larger length than is available in order to meet their safety factor (this is printed in the command window if it is the case). This would realistically be resolved by increasing the width of the component by adding a hub, or using a higher-strength material for the key.

When the desired x and y reach of the leg become too small and litter mass too large, the tibia tube interferes with the tibia holder. This is due to the tibia tube diameter increase with the robot mass. The mass also makes the torsion springs wider, increasing the shaft diameter and consequently the bellow and tibia holder diameters. The parameterization inputs were limited to ranges that avoid collisions between these parts.

3.3.3 Additional Parameters and Analysis

Many geometric aspects of the assembly were not considered in the Modelling and Analysis Reports. The hip brackets connect the hip plates to the base plate and chassis. Their height had to be parameterized to ensure the leg would not hit the chassis bottom during movement. It was determined as a function of the maximum hip angle, diameters of the hip and knee adaptors, and dimensions of the hip plates. The distance between the knee control shaft and hip control shaft on the hip plates was also calculated to avoid interference between the knee adapter and the hip brackets. Another new geometrical relation was added for the radius of the bent portion of the tibia tube. As mentioned in Section 4.2.6 of the Analysis Report, no precise method could be developed to properly calculate the safety factor of the

tube bend. The parameterization was thus set to always be the maximum possible radius, so as to increase manufacturability.

In some cases, the performed parameterization was unnecessary. The spring parameterization minimizes the number of turns and, within the chosen operating range, always gives 2; this variable could have been set as constant. The fasteners were found to also always give an acceptable safety factor for diameters as low as M2s, even at the upper end of the parameterization input range; the parameterization for them could have been removed entirely.

4 Discussion and Critical Review

Harmonic Drives provide excellent torque in a small form-factor. Their back-driving torque, however, is relatively low when compared to their output torque [21]. Springs were added to bring the torque while resting below this limit, at the cost of increasing the width of the entire leg assembly. Worm gears for transmission may provide a more compact and simple solution, with better resistance to back-driving [24]. It would be beneficial to explore this solution in more depth and compare.

The shafts were made of marine grade stainless steel 316, as it is a common material used in marine applications and is often used for pump shafts [25]. Due to the lower yield strength when compared to regular steel, the shaft diameters were too large for the output of the Harmonic Drives. A custom and relatively bulky adapter was required to interface the output bolts of the drive with the shaft. A stronger grade of steel such as AISI 4140 could be used to reduce the shaft dimension and switch the adapter for a more typical flange collar [26]. Since all shafts are contained within the bellows and chassis, the use of stainless steel may not be necessary.

As shown in Appendix B.2, the method for determining the relative lengths of the leg linkages was fairly experimental and sub-optimal. With more time, the leg linkage ratio optimization could be refined. This might lead to linkage combinations which reduce torques in the legs and match the desired x and y reach better, also likely resulting in a smaller, more compact robot.

The parameterization could also be further developed by including the exact calculations of the normal forces, as well as computing the center of mass and verifying the robot stability at a given slope angle. In this event, maximum terrain slope may have been added as a parameterization input.

As discussed in Section 3.3.2, different motors with higher torques or different mounting dimensions could be explored to avoid interference between the Harmonic Drive and motor when the robot reaches a certain size.

Although the diameter of fasteners was parameterized, they were not included as parts in the final assembly due to time constraints.

Finally, the robot is not very aesthetically pleasing. Considering it will operate in public spaces, a more attractive design could be developed.

5 References

[1] SunPower, "Solar Panels," Jan. 2018. [Online]. Available: https://us.sunpower.com/products/solar-panels

[2] A. Marla_Acme, "The Best Plastics for Outdoor Use," May 2019. [Online]. Available: https://www.acmeplastics.com/content/the-best-plastics-for-outdoor-use/

[3] Curbell Plastics, "UV Resistant Plastics for Outdoor Applications | Curbell Plastics," 2019. [Online]. Available: https://www.curbellplastics.com/Research-Solutions/Applications/Good-Weathering

[4] Custom Rubber Corp, "Rubber Molded Bellows," 2019. [Online]. Available: https://www.customrubbercorp.com/Applications/convoluted-boots-and-tubes

[5] Protocase, "Custom Cut Gaskets made to your specifications, priced right and shipped in days." 2019. [Online]. Available: https://www.protocase.com/products/materials-components-finishes/components/mcf-gaskets.php

[6] McMaster-Carr, "Tamper Resistant Bolts," 2019. [Online]. Available: https://www.mcmaster.com/

[7] Fastenright, "Sealing Fasteners." [Online]. Available: https://www.fastenright.com/products/general-fixings/sealing-fasteners

[8] MaxBotixs, "MB7395 HRXL-MaxSonar-WRBT," 2017. [Online]. Available: https://www.maxbotix.com/Ultrasonic_Sensors/MB7395.htm

[9] F. Corrigan, "_US12 Top Collision Avoidance Drones And Obstacle Detection Explained," Oct. 2019. [Online]. Available: https://www.dronezon.com/learn-about-drones-quadcopters/top-drones-with-obstacle-detection-collision-avoidance-sensors-explained/

[10] maxon motor, "maxon DC motor and maxon EC motor Key information," Nov. 2014. [Online]. Available: https://www.maxongroup.com/medias/sys_master/8815460712478.pdf?attachment=true

[11] Inertial Sense, "µINS+RTK Module," 2019. [Online]. Available: https://inertialsense.com/product/ins/

[12] Unmanned Systems Technology, "Selecting an Inertial Measurement Unit (IMU) for UAV Applications," Nov. 2018. [Online]. Available: https://www.unmannedsystemstechnology.com/2018/11/selecting-an-inertial-measurement-unit-imu-for-uav-applications/

[13] NVIDIA, "Jetson Download Center - Jetson TX2 Module," Nov. 2015. [Online]. Available: https://developer.nvidia.com/embedded/downloads

[14] M. Larabel, *Benchmarks Of Many ARM Boards From The Raspberry Pi To NVIDIA Jetson TX2 - Phoronix*, Mar. 2017. [Online]. Available: https://www.phoronix.com/scan.php?page=article&item=march-2017-arm&num=3

[15] Raspberry Pi, *Raspberry Pi 4 Model B specifications – Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org

[16] RasPi.TV, *How much power does the Pi4B use? Power Measurements*, Jun. 2019. [Online]. Available: https://raspi.tv/2019/how-much-power-does-the-pi4b-use-power-measurements

[17] O. Weigl, "madcowswe/ODriveHardware," Dec. 2019, original-date: 2016-05-31T21:35:29Z. [Online]. Available: https://github.com/madcowswe/ODriveHardware

[18] 18650batterystore, "Panasonic NCR18650b 3400mah 4.9a Battery," 2019. [Online]. Available: https://www.18650batterystore.com/Panasonic-18650-p/panasonic-ncr18650b.htm

[19] Voltaplex, "6s–16s (22.2v–74v, Adjustable) 100a max. BMS Battery Management System for Lithium-ion Battery Pack with Balancing and Communication." [Online]. Available: https://voltaplex.com/22-2v-59-2v-pcb-bms-battery-management-system-for-lithium-ion-battery-pack

[20] Gates Mectrol, "Urethane Belt Program," 2018. [Online]. Available: http://www.gatesmectrol.com/common/downloads/files/mectrol/brochure/GatesMectrol_Belt_Pulley_Catalog.pdf

[21] Harmonic Drive, "CSD-2a Component Set | Harmonic Drive," 2019. [Online]. Available: https://www.harmonicdrive.net/products/component-sets/cup-type/csd-2a

[22] QBC Bearings, "FLANGED SLEEVE BEARINGS | Standard Series - Inch," 2019. [Online]. Available: https://www.qbcbearings.com/BuyRFQ/SleeveB_StandardS_F_SB_I.php

[23] McMaster-Carr, "Plastic Submersible Cord Grips," 2019. [Online]. Available: https://www.mcmaster.com/7310K35

[24] spiroidgearing, "Worm Gears | Right Angle Gearbox | FAQ," 2019. [Online]. Available: https://www.spiroidgearing.com/faq/performance-characteristics/

[25] Metal Supermarkets, "Marine Grade Metals," Apr. 2017. [Online]. Available: https://www.metalsupermarkets.com/marine-grade-metals/

[26] McMaster-Carr, "Rotary Shafts," 2019. [Online]. Available: https://www.mcmaster.com/

[27] MathWorks, "Vectorization - MATLAB & Simulink," 2019. [Online]. Available: https://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

[28] A. Kathuria, "Intro to optimization in deep learning: Gradient Descent," Jun. 2018. [Online]. Available: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

A Instructions for GUI

A.1 Running the software

1. Open the shortcut CAD.m with MATLAB. Ensure your workspace root folder contains CAD.m and the functions folder
2. Run CAD.m from MATLAB. This should open the O-Crab user interface
3. Select your desired configuration either by sliding the sliders, or by entering a value in the text boxes. The values are restricted by the minimum and maximum determined for each option. See Figure 15 for more information.
4. Click the "Compute!" button. Note: If no value is entered, the minimum for all options will be selected.
5. Wait for the "Calculations Complete, Solid Work can now be updated" message to appear in the MatLab command prompt before rebuilding the SolidWorks Assembly.

    - Note: The information above in the MatLab command prompt are properties of the robot which cannot be shown in the SolidWorks assembly.

    - Note: Do not change the input parameters while the program is running

6. Open the shortcut to the O-Crab.SLDASM file from the root folder to open the assembly in SolidWorks.
7. Rebuild the SolidWorks assembly.

    - Note: It is normal for errors to occur on the first rebuild. Without running the MatLab code again, rebuild the SolidWorks assembly a second time if needed. Occasionally, one of the bellow runs through a rebuild issue related to its geometry although every bellow uses the same geometry. That specific bellow can be suppressed if it prevents the rest of the assembly from rebuilding properly.

8. The SolidWorks assembly is now ready. The one leg without the bellow is movable and restricted between the normal walking angle ranges.
9. To continue testing different combinations, change the input parameters and select "Compute!" again.

    - Note: Do not modify the CAD model while the program is running

A.2 Debugging

- If either the CAD.m or O-Crab.SLDASM shortcuts do not work, you can open them directly from `WR2A/MATLAB/CAD.m` or `WR2A/SolidWorks/Parts/Parametrized/O-Crab.SLDASM`.

- If any errors are thrown while running CAD.m, the functions folder (and sub-folders) are likely not added to the path. Ensure your MATLAB root folder is WR2A/MATLAB and add all folders and subfolders.

- If the SolidWorks assembly does not build properly after parameterizing, rebuild.

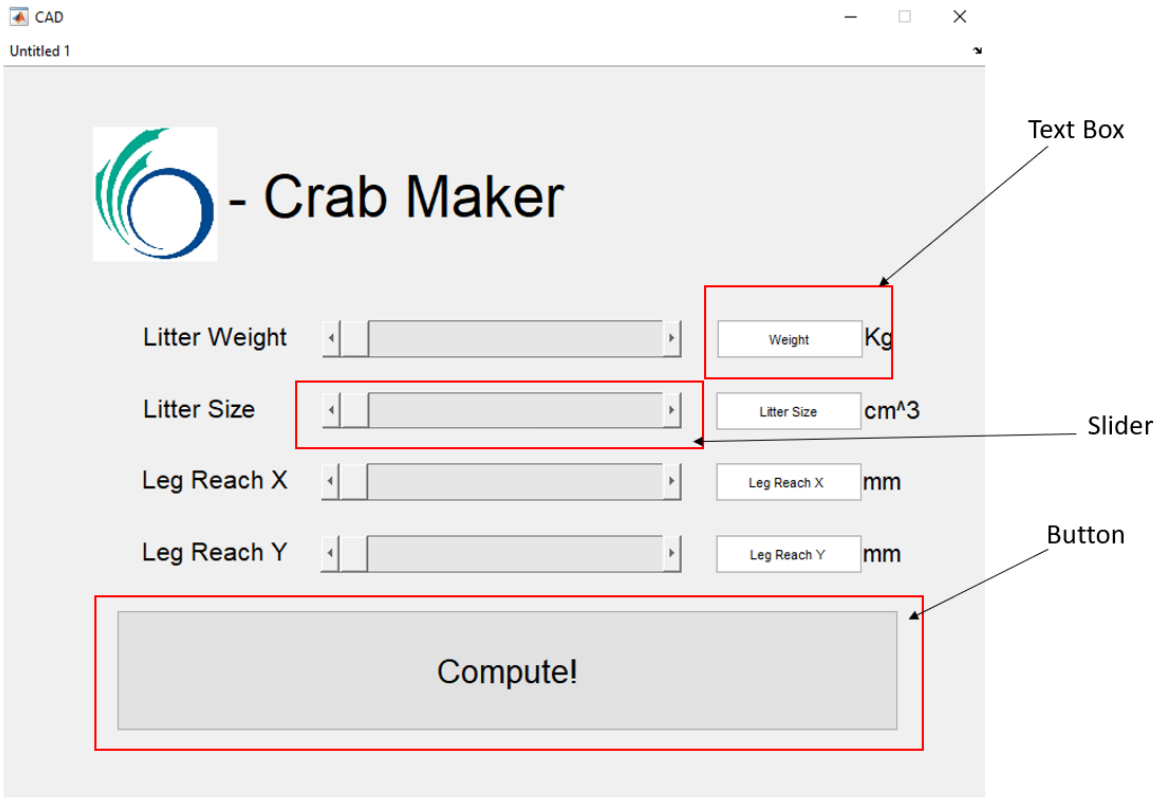


Figure 15: User Interface

B Flowcharts

B.1 Iterative Parameterization

B.1.1 Shafts

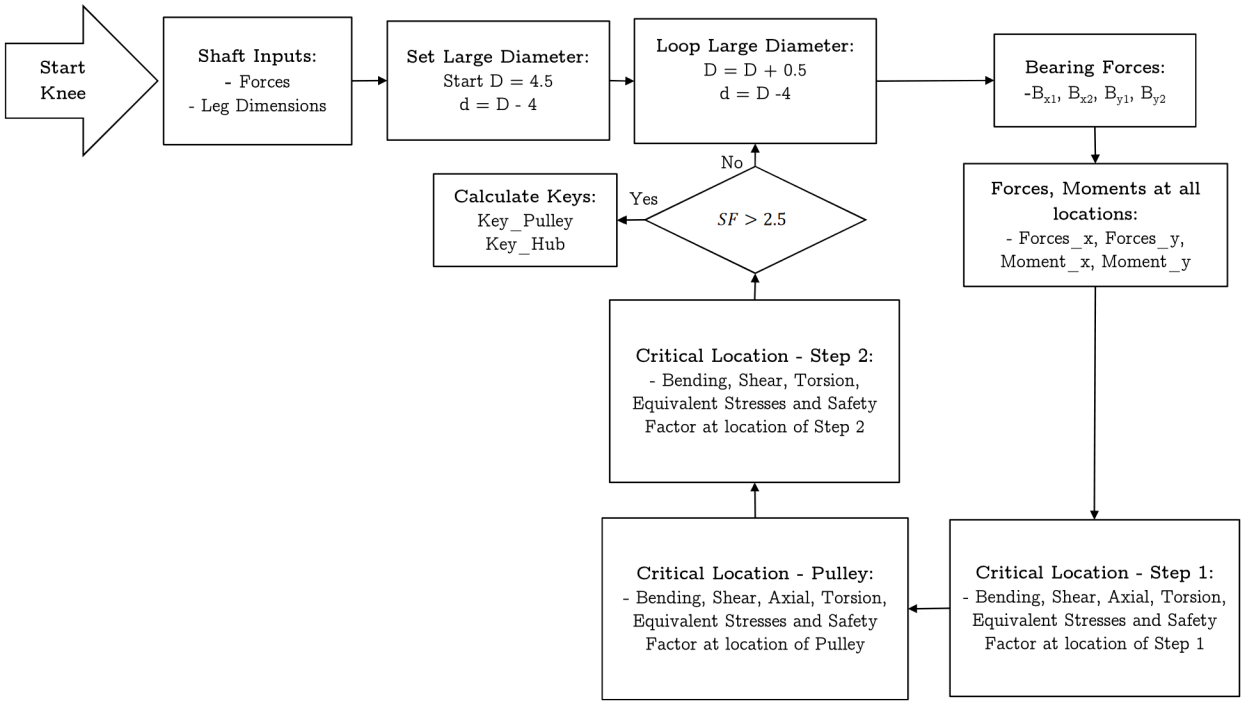


Figure 16: Shaft Flow Chart (Knee)

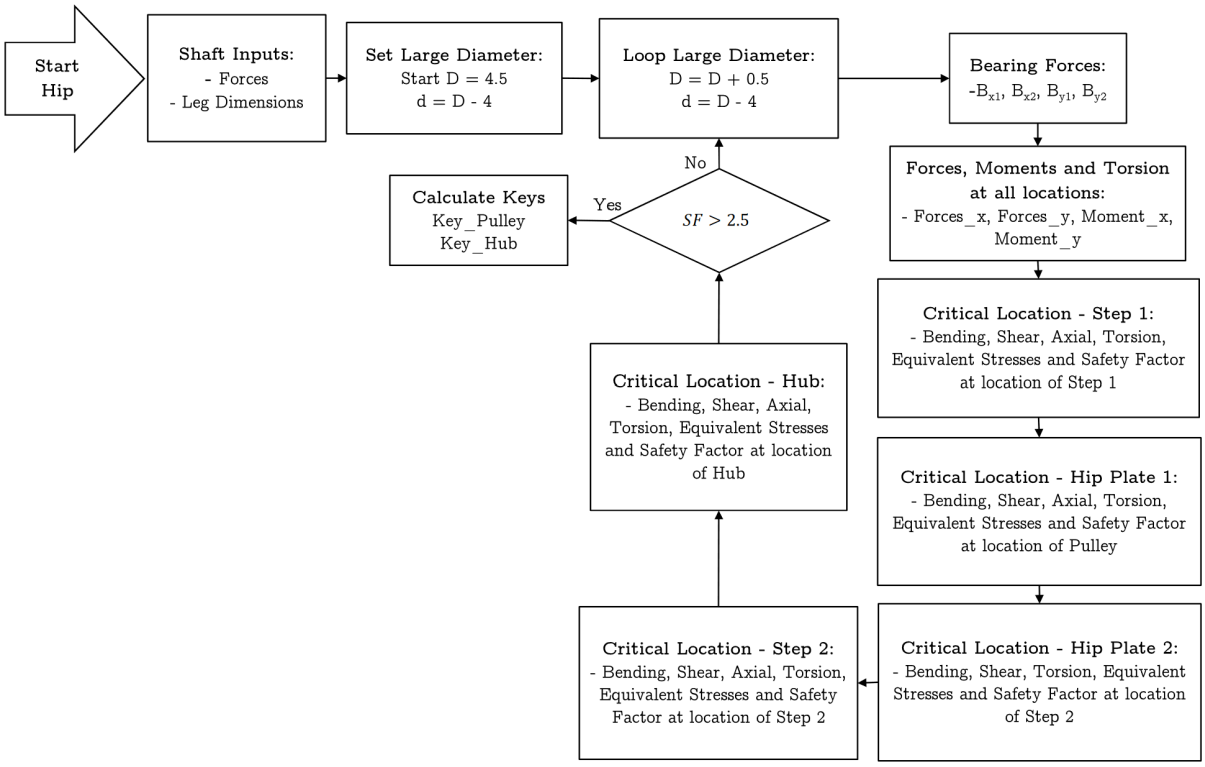


Figure 17: Shaft Flow Chart (Hip)

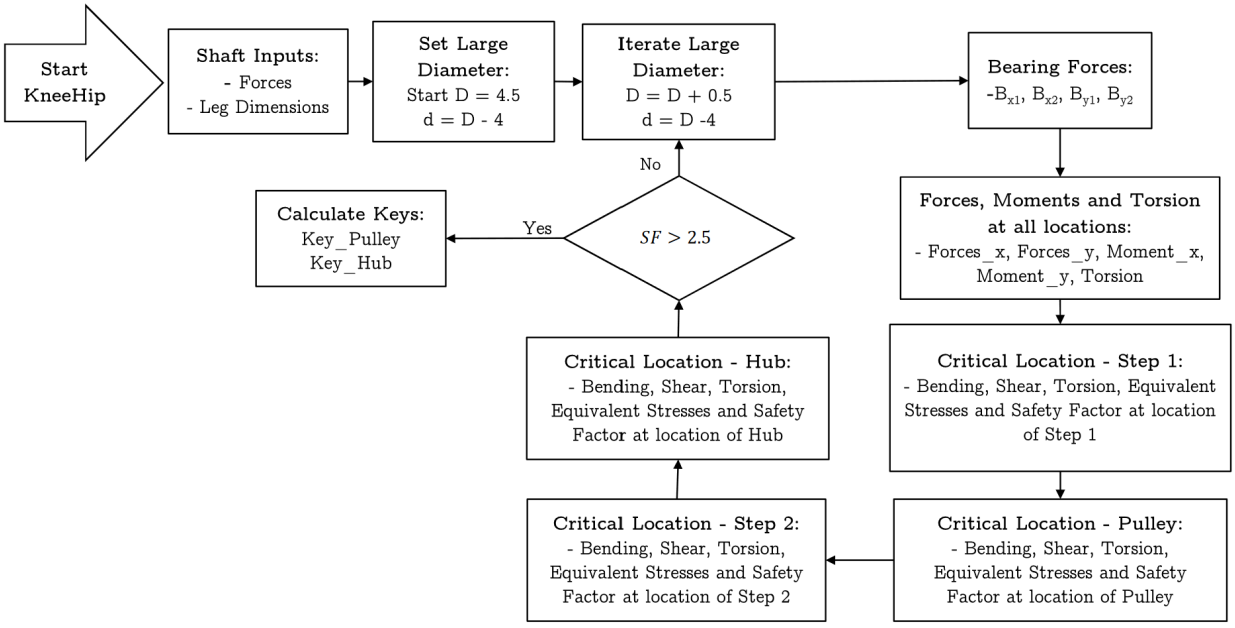


Figure 18: Shaft Flow Chart (Knee Hip)

B.1.2 Belt System

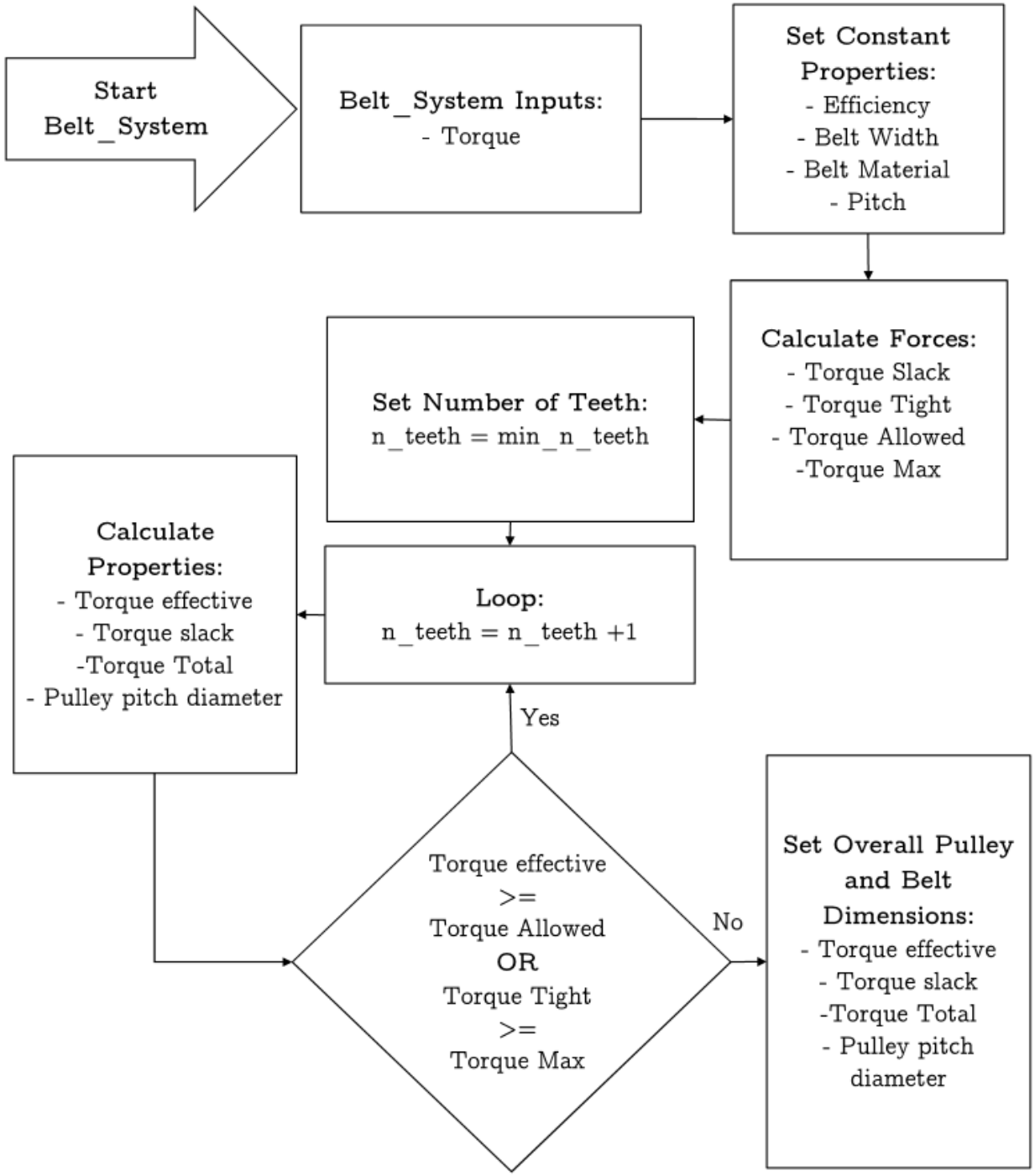


Figure 19: Belt System Flow Chart

B.1.3 Tibia Tube

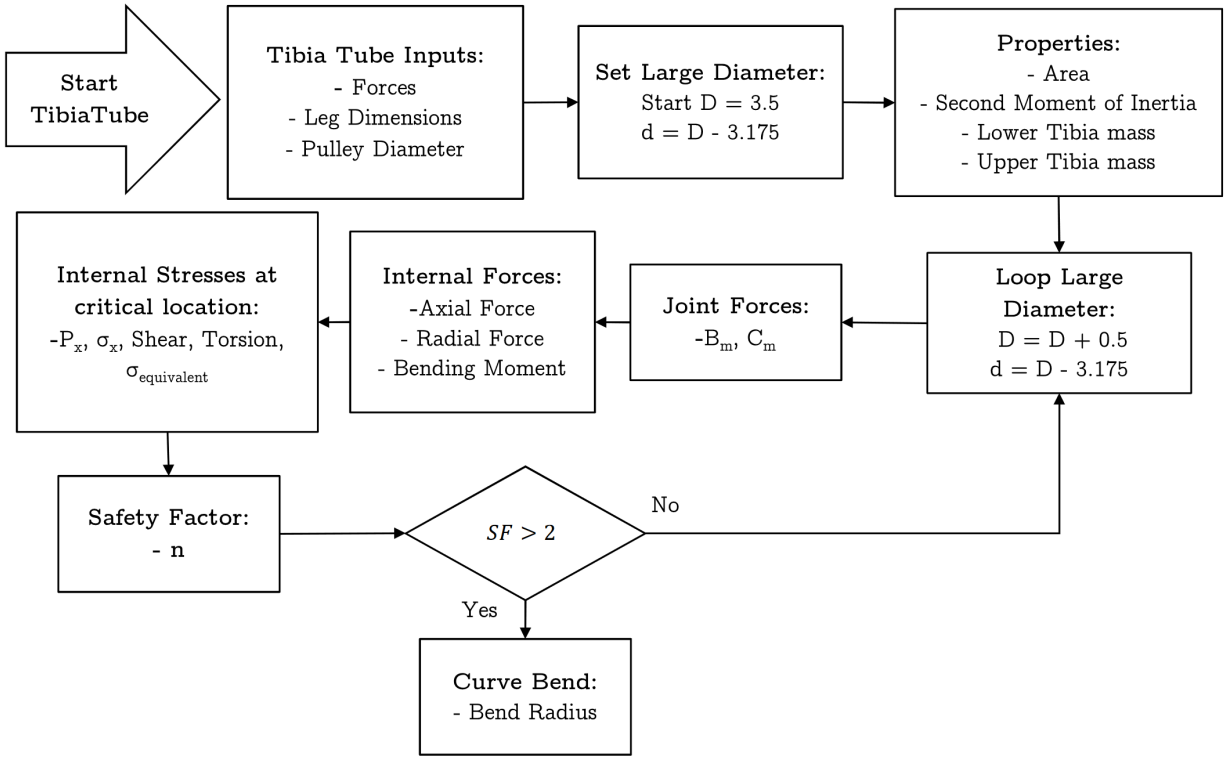


Figure 20: Tibia Tube Flow Chart

B.2 Linkage Optimization

The relative lengths (ratios) of the linkages r_1, r_2 and r_3, as well as the tibia angle α were set as constants in the parameterization. They were determined in Section 3.4 of the Analysis Report by generating a 3D chart of the x and y reach, and height from the ground d. The lengths $r_1 = 100$mm, $r_2 = 50$mm, $r_3 = 300$mm and $\alpha = 69°$ were originally selected to have approximately the same x and y reach. However, r_2 was increased to 120mm after determining that the upper tibia was too short to mount the bellow. This modified the linkage ratios slightly. The overall workspace of the leg and the x and y reach were visualized as shown in Figure 21. When a user inputs the x and y reach, the program simply scales the linkage lengths, walking height and walking range in x to match the highest value of the two inputs using the reference reaches found in `workspace.m`. This results in the desired reach in only x or y being respected, with the other higher than necessary. The relative size of the linkages has been determined experimentally and is likely far from perfect. An ideal

approach would have been to modify the lengths r_1, r_2, r_3, and α to generate a workspace that better matches both input parameters.

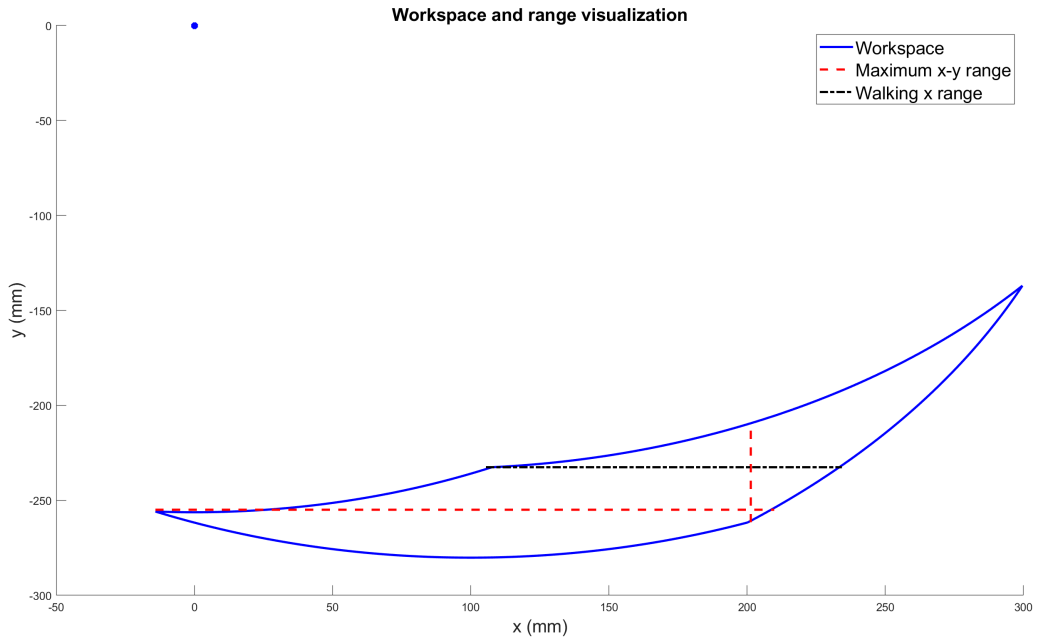


Figure 21: Workspace visualization for $r_1 = 100$mm, $r_2 = 50$mm, $r_3 = 300$mm and $\alpha = 69°$

B.3 Motor and Harmonic Drive Flowcharts

The motor and Harmonic Drive specs were determined by first collecting specifications for multiple sizes of motors and drives. This data was polyfit relative to their rated torques in MATLAB and the corresponding first order polynomial coefficients T_1 and T_2 were inserted into the MATLAB files for both parts. The maximum required torque is input to the functions and all dimensions, efficiency, and other relevant specifications are output using the fit. This procedure is outlined in Figure 22.

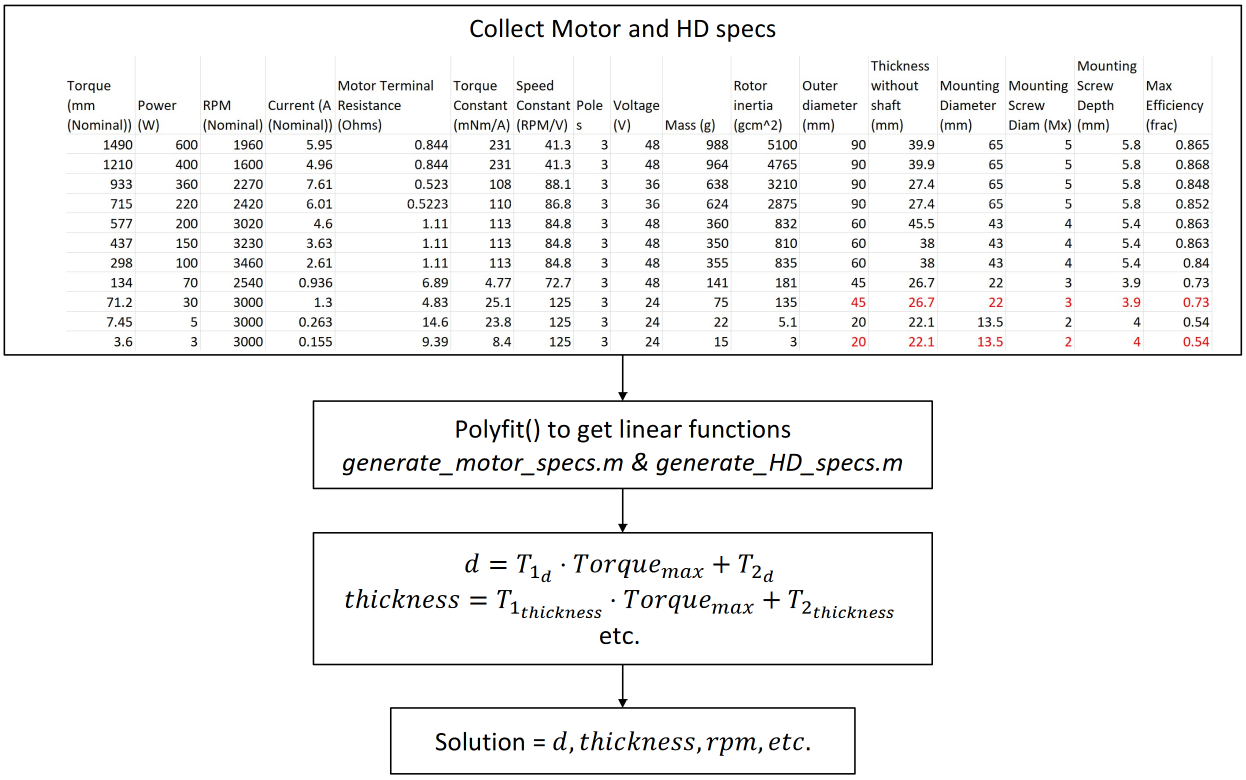


Figure 22: Parameterization of Harmonic Drives and Motors

B.4 Volumetric Parameterization and Vectorization

B.4.1 Bolts

All components but the springs and bolts were determined using the classic `for` loops found in Appendix B; all geometric variables are set as constants or ratios of a single parameterized variable. The bolts employ a simple vectorized approach, where a range of bolt diameters is generated using `d = linspace(startval,endval,numberofsteps);`. The regular stress and safety factor equations are vectorized, giving an array of safety factors for each diameter. These can be filtered using `find(SF > 1.5, etc.)` to find all diameters matching certain conditions, and the final diameter is minimized by selecting the smallest amongst the matching diameters. This procedure is demonstrated in Figure 23.

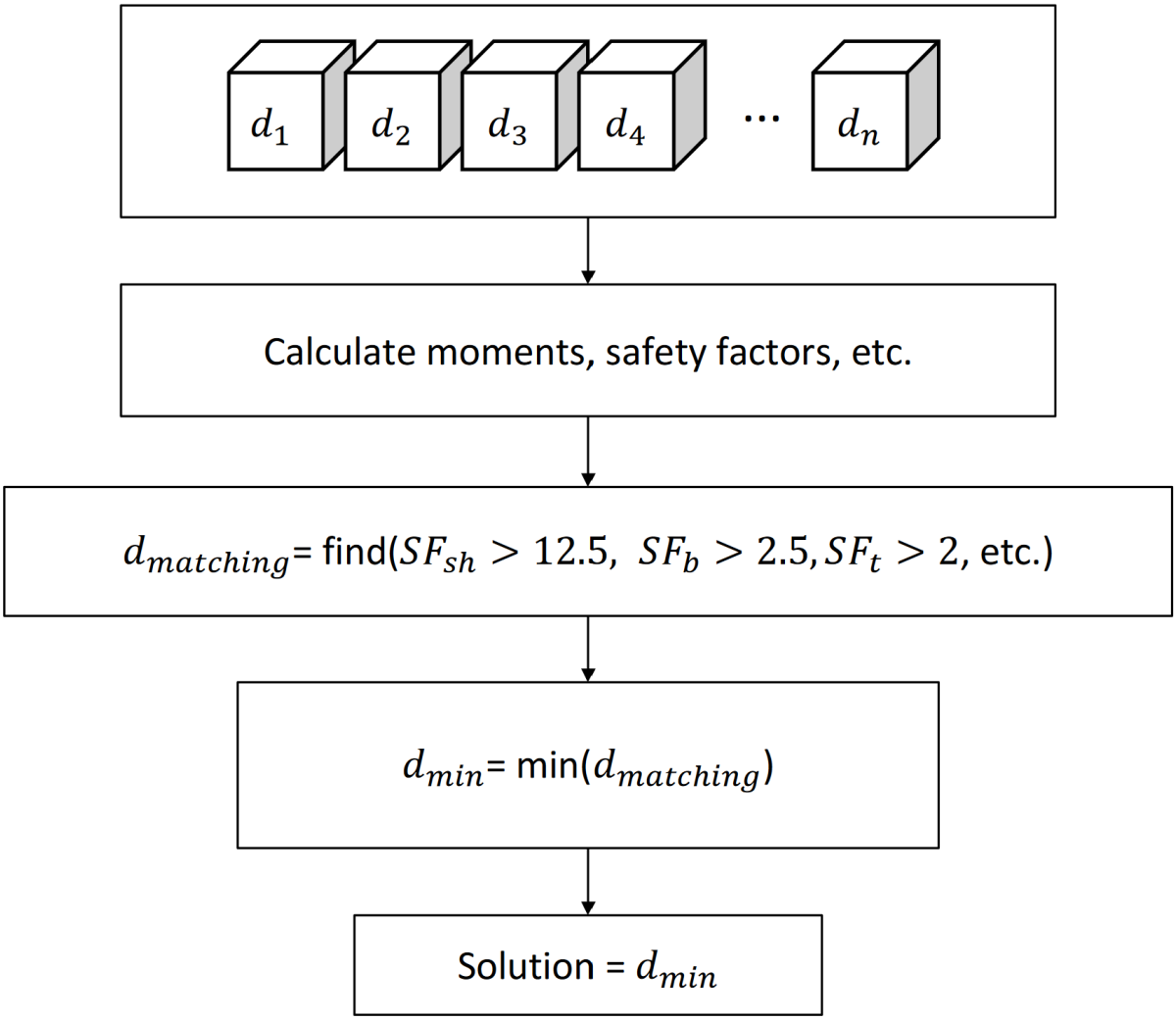


Figure 23: Flowchart of vectorized bolts. Note how looping is not necessary and the time to find a solution is constant

As well as computing the ideal solution in constant time and taking advantage of MATLAB's superior vectorization performance, this technique can easily be extended to multiple dimensions [27].

B.4.2 Springs

The spring has three primary parameters: the number of twists N_f, wire diameter d and total diameter D. All three can be expressed as row, column and depth vectors using `linspace()`, then duplicated into 3D matrices. These are then passed through the stress analysis equations, and 3D matrices of safety factors and spring constants are found. The

same `find()` function can be used to find combinations meeting minimum safety factor and maximum diameter requirements, amongst others. The ideal combinations is then selected by minimizing the total spring length $L = N_f \cdot d$. The 3D procedure is demonstrated in Figure 24.

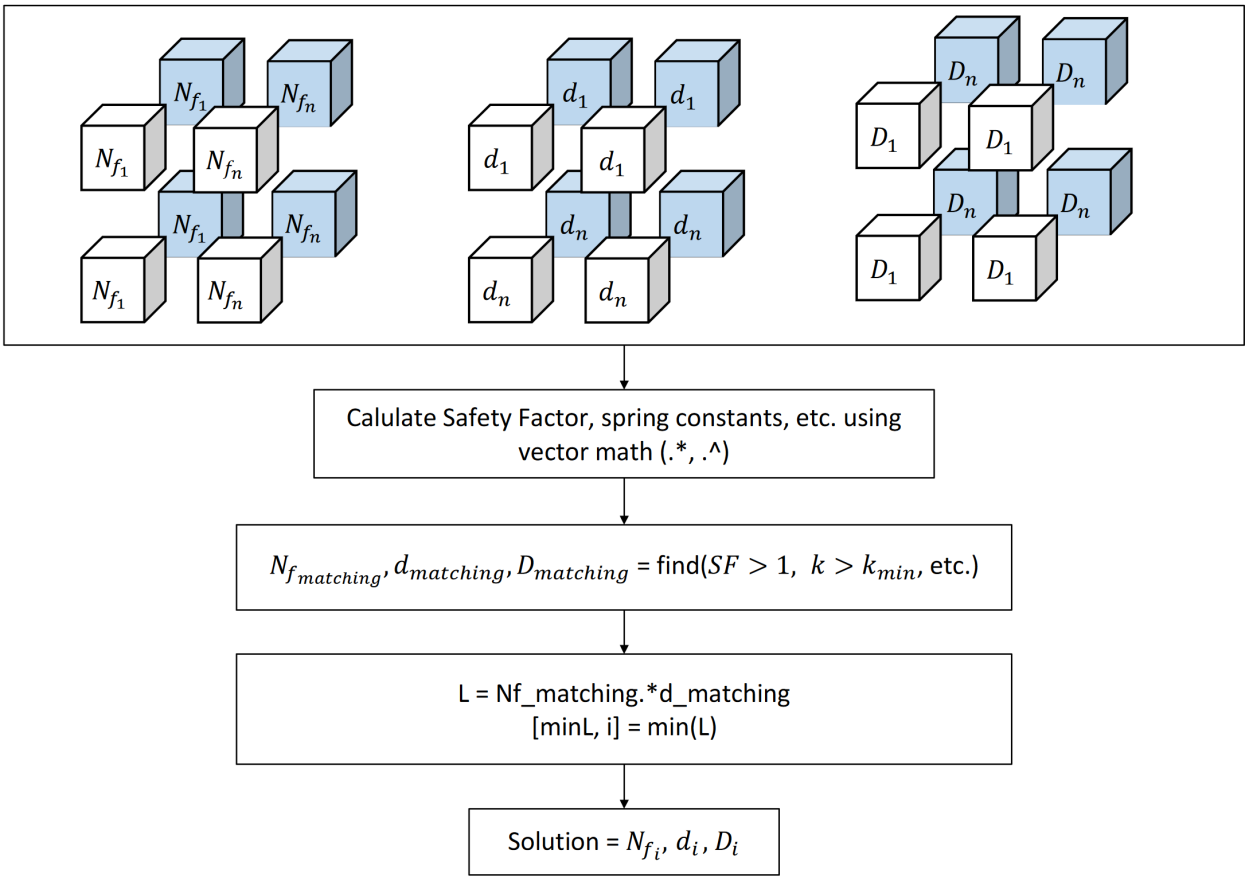


Figure 24: Flowchart of vectorized springs. As with Figure 23, there is no looping required

The consequence of this method is a larger memory footprint, as large matrices must be stored. It greatly simplifies the optimization procedure, is much more simple to implement than a gradient descent algorithm, and in hindsight could have been employed for parts where simplifications had to be made to reduce the optimization problem to one variable [28].

C Design Code

C.1 Main Program

```
function[] = main(litterWeight, BoxArea, x_reach, y_reach)

%% Assumed Constant Forever
runTimeDays =1; % [Days]
armWeight = 6; % [kg]
electronicsWeight = 0.5; %[kg]
t_bearing = 12; %[mm]
L3 = 7.5+5; % [mm]
t_plate = 10; % [mm]
Spring_free_angle_h = 180; % [deg]
Spring_free_angle_k = 180; % [deg]

%% Estimated first
BoxArea = BoxArea*(10^3)/150; % Convert to mm^3
TopArea = 0.5*1000^2; % Area of solar cells
D_knee_shaft = 21.5; % Outer diameter Knee Shaft
D_kneeHip_shaft = 21.5; % Outer diameter Hip Knee Shaft
Dmax_h = 500; % Max hip diameter
Dmin_h = 25; % Min hip diameter
Dmax_k = 500; % Diametre Hd
Dmin_k = 1; % Spacer
M =20 + litterWeight; % Total Robot Weight
M_old = 0; % Previous estimated robot weight

% Main Iteration Cycle
while(abs(M-M_old)>1) % Loops until the change in weight it less than 1 kg

    %% Length and Angles
    Leg_Object = Leg(x_reach,y_reach);
    %% Normal Force
    Force_Object = Forces(M, Leg_Object);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%% Belt %%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
belt_System = Belt_System(Force_Object.Torque_hipKnee*1000);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Spring %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Set Max Torque at Joints
Mmax_h = Force_Object.Torque_hip*1000;
Mmin_h = Force_Object.Torque_hip_min*1000;
Mmax_k = Force_Object.Torque_hipKnee*1000;
Mmin_k = Force_Object.Torque_hipKnee_min*1000;

%Set Max and Min angle
MaxTheta_h = Leg_Object.Theta;
MinTheta_h = Leg_Object.Theta_min;
MaxTheta_k = Leg_Object.Psy_relative;
MinTheta_k = Leg_Object.Psy_relative_min;

%Calculate the spring
spring_Torsion_Hip = Spring_Torsion_Hip(Leg_Object,Mmax_h,Mmin_h,Dmin_h,
    Dmax_h,MaxTheta_h,MinTheta_h,Spring_free_angle_h);
spring_Torsion_Knee = Spring_Torsion_Knee(Leg_Object,Mmax_k,Mmin_k,Dmin_k,
    Dmax_k,MaxTheta_k,MinTheta_k,Spring_free_angle_k);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%   Shafts  %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KneeHip
shaft_KneeHip = Shaft_KneeHip(belt_System.T_tight, belt_System.T_slack,
    Force_Object.Torque_hipKnee*1000, t_bearing, spring_Torsion_Knee.L, L3
    , belt_System.b_width);

% Knee
shaft_Knee = Shaft_Knee(Force_Object,Leg_Object, belt_System.T_tight,
    belt_System.T_slack,spring_Torsion_Knee.L,belt_System.b_width,
    shaft_KneeHip.L_mid,t_bearing);

% Hip
shaft_Hip = Shaft_Hip(Force_Object,Leg_Object, spring_Torsion_Hip.L, L3,
    t_plate,t_bearing,shaft_KneeHip.L_mid,spring_Torsion_Hip.L);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%   Tibia  %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tibia and components
tibiaTube = TibiaTube(Force_Object, Leg_Object,
    belt_System.Pulley_Belt_Total_Dia);
footCap = FootCap(tibiaTube.d,tibiaTube.D);
footRod = FootRod(tibiaTube.d,tibiaTube.D);
footSilicone = FootSilicone(tibiaTube.d);
tibiaPulleyHolder = TibiaPulleyHolder(shaft_Knee, tibiaTube.D,
    belt_System.Pulley_Belt_Total_Dia);
tibiaPulleyHolder = tibiaPulleyHolder.calculateBolts(Leg_Object.R,
    belt_System.Pulley_Inner_Dia, belt_System.b_width, 10, Force_Object.FF
    , Force_Object.FN);


%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% Harmonic Drive %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get torques and angles, anglesd, anglesdd over walking cycle
[max_torque_theta,max_torque_phi,max_thetad,max_phid,theta, phi,thetad,
    phid,thetadd, phidd,torque_theta,torque_phi] = Torques_Angles(
    Leg_Object, Force_Object, spring_Torsion_Hip.k, spring_Torsion_Knee.k,
     spring_Torsion_Hip.correction_angle,
    spring_Torsion_Knee.correction_angle);
% Get harmonic drive specs
HD_hip = Harmonic_Drive(max_torque_theta,'Hip');
HD_knee = Harmonic_Drive(max_torque_phi,'Knee');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%   Pulley  %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pulley_Exterior = Pulley_Exterior(belt_System,shaft_Knee.D_big);
pulley_Interior = Pulley_Interior(belt_System,shaft_KneeHip.D_big);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%   Shaft Components  %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% KneeHip Shaft Components
pulley_Exterior = pulley_Exterior.setBolts(tibiaPulleyHolder);
bearing_KneeHip = Bearing_KneeHip(shaft_KneeHip.d_small);
key_KneeHip_Pulley = Key_KneeHip_Pulley(shaft_KneeHip);
pulley_Interior = pulley_Interior.setKeys(key_KneeHip_Pulley);
key_KneeHip_Hub = Key_KneeHip_Hub(shaft_KneeHip);
spacer_KneeHip = Spacer_KneeHip(shaft_KneeHip,spring_Torsion_Knee.L,
    spring_Torsion_Knee.Di,fasteners_general(pulley_Interior.d_bolts));

% Knee Shaft Components
bearing_Knee = Bearing_Knee(shaft_Knee.d_small);
spacer_Knee = Spacer_Knee(shaft_Knee, belt_System.b_width);

% Hip Shaft Components
bearing_Hip = Bearing_Hip(shaft_Hip.d_small);
key_Hip_Plates = Key_Hip_Plates(shaft_Hip);
key_Hip_Collar = Key_Hip_Collar(shaft_Hip);
spacer_Hip_Mid = Spacer_Hip_Mid(shaft_Hip.D_big, shaft_KneeHip.L_mid);
spacer_Hip_Spring = Spacer_Hip_Spring(shaft_Hip.D_big,
    spring_Torsion_Hip.L,spring_Torsion_Hip.Di);


%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Loop Variables
D_knee_shaft = shaft_Knee.D_big;
D_kneeHip_shaft = shaft_KneeHip.D_big;

Dmin_h = spacer_Hip_Spring.D_shafthip +9;
Dmin_k = spacer_KneeHip.D_shafthipknee +9; %Spacer
Dmax_h = max([HD_hip.spline_outer_diameter,
    belt_System.Pulley_Belt_Total_Dia]);
Dmax_k = max([HD_knee.spline_outer_diameter,
    belt_System.Pulley_Belt_Total_Dia]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%   Motor  %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get maximum torque and angular velocity experienced by motor
```

```
[max_torque_theta_motor,max_thetad_motor] = HD_hip.getInputs(
    max_torque_theta,max_thetad);
[max_torque_phi_motor,max_phid_motor] = HD_knee.getInputs(max_torque_phi,
    max_phid);

% Get motor specs
motor_hip = Motor(max_torque_theta_motor,HD_hip,'Hip');
motor_knee = Motor(max_torque_phi_motor,HD_knee,'Knee');

% Get average current draw of robot
[current_consumed,walking_speed] = Power_Consumption(Leg_Object,HD_hip,
    HD_knee,motor_hip,motor_knee,theta,phi,thetad,phid,torque_theta,
    torque_phi);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% Battery %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get battery specs
battery = Battery(runTimeDays,10,solarEnergy(TopArea),current_consumed);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% Adpaters %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get knee adapter
adapter_knee = Adapter_Knee(HD_knee,motor_knee,27.5,
    belt_System.Pulley_Belt_Total_Dia);

% Get hip brackets
hip_bracket = Hip_Bracket(HD_hip,adapter_knee,Leg_Object,(
    shaft_KneeHip.L_mid + 2*2 + 2*10 +2*spring_Torsion_Hip.L+2*2),(
    shaft_Hip.d_small +2*2),belt_System.Pulley_Belt_Total_Dia,
    Force_Object.FF,Force_Object.ff,Force_Object.FN); % 2 mm bushing
    diameter

% Get hip adapter
adapter_hip = Adapter_Hip(HD_hip,motor_hip,hip_bracket);

% Get hip base
hip_base = Hip_Base(HD_hip,hip_bracket,(shaft_KneeHip.L_mid + 2*2 + 2*10
```

```
    +2*spring_Torsion_Hip.L+2*2),27.5);


%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Plates %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
plates_Hip_1 = Plates_Hip_1(shaft_Knee.d_small,shaft_Hip.D_big,
    shaft_KneeHip,belt_System.Pulley_Belt_Total_Dia,Leg_Object.r1,
    hip_bracket.distance_between_shafts,adapter_knee,key_Hip_Plates,
    spring_Torsion_Hip,spring_Torsion_Knee);
plates_Hip_2 = Plates_Hip_2(shaft_Knee.d_small,shaft_Hip.D_big,
    shaft_KneeHip,belt_System.Pulley_Belt_Total_Dia,Leg_Object.r1,
    hip_bracket.distance_between_shafts,adapter_knee,key_Hip_Plates,
    spring_Torsion_Hip,spring_Torsion_Knee);
plates_KneeHip = Plates_KneeHip(shaft_KneeHip.D_big,
    pulley_Interior.d_bolts,pulley_Interior.L_bolts,
    spring_Torsion_Knee.bigD,spring_Torsion_Knee.smalld);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% Other %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Collar
collar_Hip = Collar_Hip(shaft_Hip.d_small, key_Hip_Collar,
    HD_hip.flex_bolt_diameter,HD_hip.flex_mounting_diameter,
    HD_hip.flex_num_bolts);
collar_KneeHip = Collar_KneeHip(shaft_KneeHip.d_small, key_KneeHip_Hub,
    HD_knee.flex_bolt_diameter,HD_knee.flex_mounting_diameter,
    HD_knee.flex_num_bolts);

% Re calculate object dimensions
belt_System = belt_System.setBeltDimensions(Leg_Object.r1,
    hip_bracket.distance_between_shafts);
timingBelt = TimingBelt(belt_System);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% Chassis %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
chassis = Chassis(shaft_KneeHip,spring_Torsion_Hip,plates_Hip_1,BoxArea,
    hip_base,hip_bracket,motor_hip,HD_hip,belt_System);
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%% Bellow %%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    bellow = Bellow(plates_Hip_1,chassis,tibiaPulleyHolder);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%% Weight %%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    M_old = M; % Set approximated as old weight calculated
    M = 0; % Reset approximated weight

    % Tibia
    M = M + tibiaTube.m2 + tibiaTube.m3 + tibiaPulleyHolder.mass;

    % Shaft
    M = M + shaft_KneeHip.mass + shaft_Knee.mass + shaft_Hip.mass;

    % Brackets
    M = M + hip_bracket.mass*2 + hip_base.mass + adapter_hip.mass;

    % Motor
    M = M + motor_hip.mass + motor_knee.mass;

    % HD
    M = M + HD_hip.mass + HD_knee.mass;

    % Battery
    M = M*5 + battery.mass;

    % Total
    M = M + litterWeight + armWeight + electronicsWeight + chassis.mass;
    TopArea = chassis.area;
end


%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Print %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% KneeHip %%%%%%%%%%
shaft_KneeHip.printTXT();
bearing_KneeHip.printTXT();
key_KneeHip_Pulley.printTXT();
key_KneeHip_Hub.printTXT();
spacer_KneeHip.printTXT();
collar_KneeHip.printTXT();
plates_KneeHip.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Knee %%%%%%%%%%
shaft_Knee.printTXT();
bearing_Knee.printTXT();
spacer_Knee.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Hip %%%%%%%%%%
shaft_Hip.printTXT();
bearing_Hip.printTXT();
key_Hip_Plates.printTXT();
key_Hip_Collar.printTXT();
spacer_Hip_Mid.printTXT();
spacer_Hip_Spring.printTXT();
collar_Hip.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Limb %%%%%%%%%%
tibiaTube.printTXT();
tibiaPulleyHolder.printTXT();
footCap.printTXT();
footRod.printTXT();
footSilicone.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Springs %%%%%%%%%%
spring_Torsion_Hip.printTXT();
spring_Torsion_Knee.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%% Pulley %%%%%%%%%%
pulley_Exterior.printTXT();
pulley_Interior.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Plates %%%%%%%%%%
plates_Hip_1.printTXT();
plates_Hip_2.printTXT();

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Josh %%%%%%%%%%
HD_hip.printTXT();
HD_knee.printTXT();
motor_hip.printTXT();
motor_knee.printTXT();
battery.printTXT();
adapter_knee.printTXT(HD_knee,motor_knee);
hip_bracket.printTXT();
adapter_hip.printTXT(HD_hip,motor_hip);
hip_base.printTXT(HD_hip);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Other %%%%%%%%%%%%%%%
timingBelt.printTXT();
chassis.printTXT();
bellow.printTXT();




%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Output %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if(not(shaft_Hip.worked_keyplates))
    disp ("Hip shaft plate keys insufficiently long")
end
if(not(shaft_Hip.worked_keyhub))
    disp ("Hip shaft hub keys insufficiently long")
end
```

```
if(not(shaft_KneeHip.worked_keypulley))
    disp ("Knee Hip shaft pulley keys insufficiently long")
end
if(not(shaft_KneeHip.worked_keyhub))
    disp ("Knee Hip shaft hub keys insufficiently long")
end

disp(['Number of Solar Cells: ', num2str(floor(chassis.area/(125^2)))])

disp(['Run Hour per Day: ', num2str(battery.running_hours_per_day), ' h'])

disp(['Walking Speed: ', num2str(Leg_Object.ed), ' mm/s'])

disp(['Robot Total Weight: ', num2str(M) , ' kg'])

disp("Calculations Complete, Solid Work can now be updated")
end
```

C.2 Leg

```
classdef Leg
    properties
        r1                  % [mm] length of thigh
        r2                  % [mm] length of upper tibia
        r3                  % [mm] length of lower tibia
        R                   % [mm] length of simplified tibia
        phi_min             % [deg] Angle between thigh and simplified tibia
            at x_walking_max
        phi_max             % [deg] Angle between thigh and simplified tibia
            at x_walking_min
        d                   % [mm] Height the thigh motor
        dd                  % [m/s] Vertical velocity of the motor
        ed                  % [m/s] Horizontal velocity of the robot
        curveAngle          % [deg] Angle of the bend in the tibia
        x_walking_min       % [mm] Minimum distance from robot in x
        x_walking_max       % [mm] Maximum distance from robot in x
        Theta               % [deg] Angle between horizontal and thigh at
            x_walking_max
        Phi                 % [deg] Angle between horizontal and simplified
            tibia at x_walking_max
```

```
    Psy                 % [deg] Angle between horizontal and upper tibia
       at x_walking_max
    Alpha               % [deg] Angle between horizontal and lower tibia
       at x_walking_max
    Theta_min           % [deg] Angle between horizontal and thigh at
       x_walking_min
    Phi_min             % [deg] Angle between horizontal and simplifed
       tibia at x_walking_min
    Psy_min             % [deg] Angle between horizontal and upper tibia
       at x_walking_min
    Alpha_min           % [deg] Angle between horizontal and lower tibia
       at x_walking_min
    Thetad              % [rad/s] Angular velocity of Theta
    Phid                % [rad/s] Angular velocity of Phi
    Thetadd             % [rad/s] Angular acceleration of Theta
    Phidd               % [rad/s] Angular acceleration of Phi
    h_tibia             % [mm] Vertical distance of tibia
    l_tibia             % [mm] Horizontal distance of tibia
    h_leg               % [mm] Vertical distance of the leg
    l_leg               % [mm] Horizontal distance of the leg
    Phi_relative        % [deg] Phi relative to thigh at x_walking_max
    Psy_relative        % [deg] Psy relative to upper tibia at
       x_walking_max
    Phi_relative_min    % [deg] Phi relative to thigh at x_walking_min
    Psy_relative_min    % [deg] Psy relative to upper tibia at
       x_walking_min
    small_psy           % [deg] Angle between Phi and Psy
end
methods

    % Leg Constructor
    function obj = Leg(x_reach,y_reach)
        obj = obj.setParameters(x_reach,y_reach);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,x_reach,y_reach)
        obj.dd = 0;
        obj.ed = 0.5*x_reach;
        obj.curveAngle = 69;
```

```
            % Linkage optimization (Section  3.4 of Analysis Report)
            [obj.r1,obj.r2,obj.r3,obj.R,obj.phi_min,obj.phi_max,obj.d,
                obj.x_walking_min,obj.x_walking_max] = linkages(x_reach,
                y_reach);

            % Geometric Calculation (Section 2.1 of Analysis Dossier)
            [obj.Theta, obj.Phi, obj.Psy, obj.Alpha, obj.small_psy] = Dynamics
                (obj.r1,obj.r2,obj.r3,obj.x_walking_max,obj.d,obj.curveAngle,
                obj.R);
            [obj.Theta_min, obj.Phi_min, obj.Psy_min, obj.Alpha_min] =
                Dynamics(obj.r1,obj.r2,obj.r3,obj.x_walking_min,obj.d,
                obj.curveAngle,obj.R);
            [obj.Thetad, obj.Phid] = leg_angles_dot(obj.r1,obj.R,obj.Theta,
                obj.Phi,obj.dd,obj.ed);
            [obj.Thetadd, obj.Phidd] = leg_angles_ddot(obj.r1,obj.R,obj.Theta,
                obj.Thetad,obj.Phi,obj.Phid,0,0);
            obj.h_tibia= obj.d + obj.r1*sind(obj.Theta);
            obj.l_tibia = obj.x_walking_max-obj.r1*cosd(obj.Theta);
            obj.h_leg = obj.d;
            obj.l_leg = obj.x_walking_max;
            obj.Phi_relative = obj.Phi-obj.Theta;
            obj.Psy_relative = obj.Phi_relative + obj.small_psy;
            obj.Phi_relative_min = obj.Phi_min-obj.Theta_min;
            obj.Psy_relative_min = obj.Phi_relative_min + obj.small_psy;
        end

    end
end
```

C.3 Forces

```
classdef Forces
    properties
        FN                  % [N] Normal Force
        ff                  % [N] Friction force in x direction
        FF                  % [N] Friction force in y direction
        Torque_hip          % [Nm] Torque at hip maximum
        Torque_hipKnee      % [Nm] Torque at Hip Knee maximum
        Torque_hip_min      % [Nm] Torque at Hip minimum
        Torque_hipKnee_min  % [Nm] Torque at Hip Knee minimum
```

```
    F_tibia             % [N] Forces at the Tibia
    F_thigh             % [N] Forces at the thigh
    m_foot = 0.25;      % [Kg] Weight of the foot piece
    m_knee = 0.5;       % [Kg] Weight of the knee piece
end
methods

    % Forces Constructor
    function obj = Forces(W, leg)
       obj = obj.setParameters(W,leg);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,W,leg)
        %% Normal Force (Section 3.1.1 of Analysis Report)
        CG = [-559+1062 , -400.02+800, 209.98];
        P1 = [-1062+1062,200,0];
        P2 = [0+1062,800,0];
        P3 = [0+1062,400,0];
        [N1,N2,N3] = forces(P2,P3,P1,CG,W,20);
        obj.FN = max(abs([N1,N2,N3]));

        %% Friction Force (Section 3.1.2 of Analysis Report)
        CG = [596 , 400.04, 209.13];
        P1 = [1136,200,0];
        P2 = [0,800,0];
        P3 = [0,400,0];
        [N1,N2] = friction_Forces(P1,P2,P3,CG,W,20);
        obj.ff = max(abs([N1,N2]));
        obj.FF = 10.6;

        %% Torques (Section 3.2 of Analysis Report)
        [obj.Torque_hip,obj.Torque_hipKnee] = dynamic_equation(obj.m_foot,
           obj.m_knee,leg.R/1000,leg.r1/1000,leg.Theta,leg.Phi,
           leg.Thetadd,leg.Phidd,obj.FN,obj.FF,0,0,0,0);
        [obj.Torque_hip_min,obj.Torque_hipKnee_min] = dynamic_equation(
           obj.m_foot,obj.m_knee,leg.R/1000,leg.r1/1000,leg.Theta_min,
           leg.Phi_min,leg.Thetadd,leg.Phidd,obj.FN,obj.FF,0,0,0,0);

        %% Shaft Forces
        % Section 4.5.5 of Analysis Report
```

```
            obj.F_tibia = obj.FN-9.81*obj.m_foot; % Eq. 86
            obj.F_thigh = obj.F_tibia-9.81*obj.m_knee; % Eq. 97
        end

        function printTXT(obj)

        end

    end
end
```

C.4 Belt System

```
classdef Belt_System
    properties
        Pulley_Belt_Total_Dia   % [mm] Pulley Belt Total Diameter
        Pulley_Inner_Dia        % [mm] Pulley Inner Diameter
        b_width                 % [mm] Belt Width
        T_tight                 % [N] Tight Tension
        T_slack                 % [N] Slack Tension
        pulley_pitch_dia        % [mm] Pulley Pitch Diameter
        Belt_Length             % [mm] Belt Length
        M_k                     % [Nmm] Torque at knee
        b_pitch                 % [mm] Belt Pitch
        C                       % [mm] Distance center to center of pulleys

    end
    methods

        % Belt System Constructor
        function obj = Belt_System(M_k)
            obj = obj.setParameters(M_k);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,M_k)
            obj = obj.calculate(M_k);
        end

        % Parametric Calculation function (Section 4.3 of Analysis Report)
```

```
function obj = calculate(obj,M_k)
    eta_belt = 0.95;        %Belt drive efficiency
    M_h = M_k/eta_belt;

    %% Belt Specifications
    b_width = 25; % mm
    b_pitch = 8;  % 5mm or 8mm
    b_material = 1; % 1:Steel ; 2:Kevlar
    b_pitch_diff = 0.7;

    % Belt Specification based on pitch and material of belt
    if b_pitch == 5
        Te_allow = 1020/25*b_width;
        min_n_teeth = 14;
        b_thick = 3.6;
        b_tooth_height = 2.1;
        if b_material == 1
            T_max = 2340/25*b_width;
        elseif b_material == 2
            T_max = 1170/25*b_width;
        end
    elseif b_pitch == 8
        Te_allow = 1870/25*b_width;
        min_n_teeth = 20;
        b_thick = 5.6;
        b_tooth_height = 3.4;
        if b_material == 1
            T_max = 3741/25*b_width;
        elseif b_material == 2
            T_max = 1870/25*b_width;
        end
    end

    %% Starting values
    pulley_pitch_dia = b_pitch*min_n_teeth/pi;
    T_effective = 2*M_h/pulley_pitch_dia;
    T_slack = 0.3*T_effective;
    T_tight = T_effective+T_slack;
    T_total = T_tight + T_slack;

    %% Loop until maximum belt tension and allowable effective tension
```

```
            conditions are met
        n_teeth = min_n_teeth;

        while (T_effective >= Te_allow || T_tight >= T_max)
            n_teeth = n_teeth + 1;
            pulley_pitch_dia = b_pitch*n_teeth/pi;
            T_effective = 2*M_h/pulley_pitch_dia;
            T_slack = 0.3*T_effective;
            T_tight = T_effective+T_slack;
            T_total = T_tight + T_slack;
        end

        %% Overall pulley and belt dimensions
        Pulley_Belt_Total_Dia = pulley_pitch_dia + 2*(-b_pitch_diff +
           b_thick - b_tooth_height);
        Pulley_Inner_Dia = pulley_pitch_dia - 2*b_pitch_diff - 2*
           b_tooth_height;
        Max_linear_displ = pulley_pitch_dia/2*deg2rad(45);
        obj.Pulley_Belt_Total_Dia = Pulley_Belt_Total_Dia;
        obj.Pulley_Inner_Dia = Pulley_Inner_Dia;
        obj.b_width = b_width;
        obj.T_tight = T_tight;
        obj.T_slack = T_slack;
        obj.pulley_pitch_dia = pulley_pitch_dia;
        obj.M_k = M_k;
        obj.b_pitch = b_pitch;
    end

    % Set Belt Dimensions Function
    % Parameters:
    % L_thigh   [mm] Length of thigh
    % L_motor   [mm] Distance between both motors shaft
    function obj = setBeltDimensions(obj, L_thigh,L_motor)
        obj.C = L_thigh + L_motor;  %Center-to-center dimension
        obj.Belt_Length = pi*obj.pulley_pitch_dia + 2*obj.C;
        b_n_teeth = obj.Belt_Length/obj.b_pitch;
    end

end
end
```

C.5 Pulley Exterior

```
classdef Pulley_Exterior
    properties
        D_p          % [mm] Pitch Diameter
        d_i          % [mm] Hub Diameter
        W_p          % [mm] Width of the pulley
        d_bolts      % [mm] Diameter Bolt
        L_bolts      % [mm] Bolt Radius Location
    end
    methods

        % Pulley Exterior Constructor
        function obj = Pulley_Exterior( belt_System,D_knee_shaft)
            obj = obj.setParameters( belt_System,D_knee_shaft);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, belt_System,D_knee_shaft)
            obj.D_p =  belt_System.pulley_pitch_dia;
            obj.d_i = D_knee_shaft;
            obj.W_p = belt_System.b_width;
        end

        % Set bolt method
        % Parameters
        % TibiPulleyHolder Object
        function obj = setBolts(obj, tibiaPullerHolder)
            obj.d_bolts = tibiaPullerHolder.d_bolts;
            obj.L_bolts = tibiaPullerHolder.L_bolts;
        end

        % Set Pulley Diameter method
        % Parameters
        % [mm] Diameter of knee
        function obj = setPulleyDiameter(obj, innerDiameter)
            obj.d_i = innerDiameter;
        end

        % Print function to TXT file
```

```
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\ExteriorTimingPulley.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_p"= ',num2str(obj.D_p),'\n'));
            fprintf(fid,strcat('"d_i"= ',num2str(obj.d_i),'\n'));
            fprintf(fid,strcat('"W_p"= ',num2str(obj.W_p),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fclose(fid);
        end

    end
end
```

C.6 Pulley Interior

```
classdef Pulley_Interior
    properties
        D_p          % [mm] Pitch Diameter
        d_i          % [mm] Hub Diameter
        w_k          % [mm] Width Key
        h_k          % [mm] Height Key
        W_p          % [mm] Width of the pulley
        d_bolts      % [mm] Diameter Bolt
        L_bolts      % [mm] Bolt Radius Location
    end
    methods

        % Pulley Interior Constructor
        function obj = Pulley_Interior(belt_System,D_kneeHip_shaft)
            obj = obj.setParameters(belt_System,D_kneeHip_shaft);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, belt_System,D_kneeHip_shaft)
            obj.D_p = belt_System.pulley_pitch_dia;
            obj.d_i = D_kneeHip_shaft;
            obj.W_p = belt_System.b_width;
            obj = obj.calculate(belt_System);
        end
```

```
        % Set Key Method
        % Parameters:
        % Key_HipKnee Object
        function obj = setKeys(obj,key_HipKnee)
            obj.w_k = key_HipKnee.w;
            obj.h_k = key_HipKnee.h/2;
        end

        % Calculate Bolt Method
        % Parameters:
        % Belt_System Object
        function obj = calculate(obj,belt_System)
            obj.d_bolts  = fasteners_kneeHip(belt_System.Pulley_Inner_Dia,
               obj.d_i, obj.W_p, 5, belt_System.M_k); %
            obj.L_bolts = (belt_System.Pulley_Inner_Dia - obj.d_i)/4+0.5*
               obj.d_i;
        end

        % Set Pulley Diameter Method
        function obj = setPulleyDiameter(obj, innerDiameter)
            obj.d_i = innerDiameter;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\InteriorTimingPulley.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_p"= ',num2str(obj.D_p),'\n'));
            fprintf(fid,strcat('"d_i"= ',num2str(obj.d_i),'\n'));
            fprintf(fid,strcat('"w_k"= ',num2str(obj.w_k),'\n'));
            fprintf(fid,strcat('"h_k"= ',num2str(obj.h_k),'\n'));
            fprintf(fid,strcat('"W_p"= ',num2str(obj.W_p),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fclose(fid);
        end

    end
end
```

C.7 Spring Hip

```
classdef Spring_Torsion_Hip
    properties
        bigR                % [mm] Coil Radius
        bigD                % [mm] Coil Diameter
        smalld              % [mm] Wire Diameter
        N                   % [quantity] Number of turns
        r                   % [mm] Bend radius
        l =10;              % [mm] Offset length of both ends of spring
        Lmid                % [mm] Distance for mid plane
        k                   % [N/mm] Spring Constants
        SF                  % Safety Factor
        p                   % [mm] Spring coil pitch
        Di                  % [mm] Internal coil diameter
        L                   % [mm] Length of spring
        correction_angle    % [deg] Angle to switch from spring deflection to
            prosition of leg
    end
    methods

        % Spring Torsion Hip Constructor
        function obj = Spring_Torsion_Hip(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,
            MinAngle,free_angle)
             obj = obj.setParameters(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle,
                 free_angle);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,
            MinAngle,free_angle)

             obj = obj.calculate(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle,
                 free_angle);
        end

        % Parametric Calculation function (Section 4.4 of Analysis Report)
        function obj = calculate(obj,leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle
            ,free_angle)
```

```
            [k,d,D,l,SF,N,p,Di,theta_max] = torsionSpring(Mmax,Mmin,Dmin,Dmax,
                MaxAngle,MinAngle,free_angle);
            obj.bigR = D/2;
            obj.bigD =D;
            obj.smalld = d;
            obj.k = k;
            obj.N = N;
            obj.r = d/2;
            obj.Lmid = N*p/2;
            obj.SF=SF;
            obj.p = p;
            obj.Di = Di;
            obj.L = l;
            obj.correction_angle = leg.Theta + theta_max*180/pi;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipTorsionSpring.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"bigR"= ',num2str(obj.bigR),'\n'));
            fprintf(fid,strcat('"smalld"= ',num2str(obj.smalld),'\n'));
            fprintf(fid,strcat('"N"= ',num2str(obj.N),'\n'));
            fprintf(fid,strcat('"r"= ',num2str(obj.r),'\n'));
            fprintf(fid,strcat('"l"= ',num2str(obj.l),'\n'));
            fprintf(fid,strcat('"Lmid"= ',num2str(obj.Lmid),'\n'));
            fprintf(fid,strcat('"p"= ',num2str(obj.p),'\n'));
            fclose(fid);
        end

    end
end
```

C.8 Spring Knee

```
classdef Spring_Torsion_Knee
    properties
        bigR                % [mm] Coil Radius
        bigD                % [mm] Coil Diameter
        smalld              % [mm] Wire Diameter
```

```
    N                   % [quantity] number of turns
    r                   % [mm] Bend radius
    l =10;              % [mm] Offset length of both ends of spring
    Lmid                % [mm] Distance for mid plane
    k                   % [N/mm] Spring Constant
    SF                  % Safety Factor
    p                   % [mm] Spring coil pitch
    Di                  % [mm] Internal coil diameter
    L                   % [mm] Length of spring
    correction_angle    % [deg] Angle to switch from spring deflection to
       prosition of leg

end
methods

    function obj = Spring_Torsion_Knee(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,
       MinAngle,free_angle)
        obj = obj.setParameters(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle,
           free_angle);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,
       MinAngle,free_angle)

        obj = obj.calculate(leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle,
           free_angle);
    end

    % Parametric Calculation function (Section 4.4 of Analysis Report)
    function obj = calculate(obj,leg,Mmax,Mmin,Dmin,Dmax,MaxAngle,MinAngle
       ,free_angle)

        [k,d,D,l,SF,N,p,Di,theta_max] = torsionSpring(Mmax,Mmin,Dmin,Dmax,
           MaxAngle,MinAngle,free_angle);
        obj.bigR = D/2;
        obj.bigD =D;
        obj.smalld = d;
        obj.k = k;
        obj.N = N;
        obj.r = d/2;
```

```
            obj.Lmid = N*p/2;
            obj.SF=SF;
            obj.p = p;
            obj.Di = Di;
            obj.L = l;

            obj.correction_angle = theta_max*180/pi - leg.Phi + leg.Theta-360;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\KneeTorsionSpring.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"bigR"= ',num2str(obj.bigR),'\n'));
            fprintf(fid,strcat('"smalld"= ',num2str(obj.smalld),'\n'));
            fprintf(fid,strcat('"N"= ',num2str(obj.N),'\n'));
            fprintf(fid,strcat('"r"= ',num2str(obj.r),'\n'));
            fprintf(fid,strcat('"l"= ',num2str(obj.l),'\n'));
            fprintf(fid,strcat('"Lmid"= ',num2str(obj.Lmid),'\n'));
            fprintf(fid,strcat('"p"= ',num2str(obj.p),'\n'));
            fclose(fid);
        end

    end
end
```

C.9 Torsion Spring Function

```
% Parameters
% Mmax          [Nmm] Maximum torque
% Mmin          [Nmm] Minimum torque
% Dmin          [mm] Minimum possible diameter
% Dmax          [mm] Maximum possible diameter
% MaxAngle      [deg] Maximum rotation angle
% MinAngle      [deg] Minimum rotation angle
% freeAngle     [deg] Spring free angle

%Outputs
% k             [N/mm] Spring Constant
% d             [mm] Wire Diameter
```

```
% D             [mm] Coil diameter
% l             [mm] Length of spring
% SF            [] Safety Factor
% N             [quantity] Number of turns
% p             [mm] Pitch
% Di            [mm] Internal coil diameter
% theta_max     [deg] Max deflection

% Parametric Calculation function (Section 4.4 of Analysis Report)
function [k,d,D,l,SF,N,p,Di,theta_max] = torsionSpring(Mmax,Mmin,Dmin,Dmax,
   MaxAngle,MinAngle,free_angle)
%% Spring material: music wire
E = 207E3;  % MPa    (E = 207 GPa)
A = 2211;   % MPa mm (A = 2211 MPa mm)
m = 0.145;  %

%% Finding backdriving torque (Section 4.4.3 of Analysis Report)
Back_driving_torque = Mmax/5;
Max_M_spring = 1/2*(Mmax - Back_driving_torque);
Min_M_spring = 1/2*(Mmin - Back_driving_torque);

%% Finding the minimal spring constant k (Section 4.4.6 of Analysis Report)
% Equation 70
k_min = (Max_M_spring - Min_M_spring)/(abs(deg2rad(MaxAngle - MinAngle)));
% Equation 71 and 72
theta_max = Max_M_spring/k_min;
theta_min = Min_M_spring/k_min;

%% Geometric Calculation
if free_angle <= 180
    angle = free_angle + 180;
else
    angle = free_angle - 180;
end
%% Starting Values
L_arm = 0;
N_f = 1;
N_p = -1/360*angle+1; %Eq. 73
N_b = N_f + N_p; %Eq. 74

%% Vectorized parametrization method, see Capstone Report Appendix
```

```
N_f_nomatrix = linspace(1,10,10);
N_f          = repmat(N_f_nomatrix,100,1,100);
d_nomatrix   = linspace(1,15,100);
d            = repmat(d_nomatrix',1,10,100);
D_nomatrix   = linspace(1,100,100);
D_reshaped   = reshape(D_nomatrix,[1,1,100]);
D            = repmat(D_reshaped,100,10,1);

% Section 4.4.6 of Analyis Report
N_b = N_f + N_p; %Eq. 74
N_a = N_b + (2.*L_arm./(3.*pi.*D)); %Eq. 75

k = (d.^4.*E)./(64.*D.*N_a); %Eq. 77

M = k.*theta_max; %Eq. 79

theta_prime = (10.8.*M.*D.*N_b)./(d.^4.*E); %Eq. 80
D_prime     = (N_b.*D)./(N_b + theta_prime); %Eq. 81
D_i_prime   = D_prime - d; %Eq. 82

C     = D./d;
K_i   = (4.*C.^2 - C - 1)./(4.*C.*(C - 1)); %Eq. 83
sigma = K_i.*32.*M./(pi.*d.^3); %Eq. 84
S_ut  = A./d.^m;
S_y   = 0.78.*S_ut;
SF    = S_y./sigma; %Eq. 85

%% Find vectorized solution
%[rows,cols,depth] = ind2sub(size(N_f),find((SF>=1) & (k>=k_min) & (D_i_prime>
   Dmin) & (C<=12) & (C>=4) & ((D+4*d)<Dmax)));
[rows,cols,depth] = ind2sub(size(N_f),find((SF>=1) & (k>=k_min)  & (C<=12) & (
   C>=4)));

Nf_matching = zeros(size(rows,1),1);
d_matching  = zeros(size(rows,1),1);
% D_matching  = zeros(size(rows,1),1);
% SF_matching  = zeros(size(rows,1),1);
% k_matching  = zeros(size(rows,1),1);
% Di_matching  = zeros(size(rows,1),1);
for i=1:1:size(rows,1)
```

```
    Nf_matching(i) = N_f(rows(i),cols(i),depth(i));
    d_matching(i)  = d(rows(i),cols(i),depth(i));
    D_matching(i)  = D(rows(i),cols(i),depth(i));
    SF_matching(i) = SF(rows(i),cols(i),depth(i));
    k_matching(i)  = k(rows(i),cols(i),depth(i));
    Di_matching(i)  = D_i_prime(rows(i),cols(i),depth(i));
end

L     = Nf_matching.*d_matching;
[l,i] = min(L);

N  = Nf_matching(i);
d  = d_matching(i);
D  = D_matching(i);
SF = SF_matching(i);
k  = k_matching(i);
Di = Di_matching(i);

% Small modification to pitch for solid works
p = d*1.01;
l = l/d*p+2*d;

end
```

C.10 Shaft Hip Knee

```
classdef Shaft_KneeHip
    properties
        D_big                       % [mm] Big diameter of shaft
        d_small                     % [mm] Small diameter of shaft
        L_bearing                   % [mm] Length bearing
        L_mid                       % [mm] Leangth of bid diameter
        w_keypulley                 % [mm] Width key pulley
        h_keypulley                 % [mm] Height key pulley
        w_keyhub                    % [mm] Width key hub
        h_keyhub                    % [mm] Height key hub
        Plane_keyhub                % [mm] Plane for collar key
        mass                        % [Kg] Mass
        worked_keypulley            % Boolean if key length is sufficient for
            pulley
```

```
    worked_keyhub               % Boolean if key length is sufficient for
        hub

    % Constants
    M_sy = 240;                 % [MPa] Yield Strength
    density = 8 * (10^(-6));    % [kg/mm^3] Density

    %Concentration Factors
    CF_B = [2,1.8,2,1.8];       % Bending
    CF_S = [1.7,1.8,1.7,1.8];   % Shear
    CF_T = [1.7,1.8,1.7,1.8];   % Torsion
end
methods

    % Shaft Knee Hip Constructor
    function obj = Shaft_KneeHip(F_T1, F_T2, Torque, t_bearing, L1, L3,
        belt_width)
        obj = obj.setParameters(F_T1, F_T2, Torque, t_bearing, L1, L3,
            belt_width);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj, F_T1, F_T2, Torque, t_bearing, L1,
        L3, belt_width)
        obj.L_bearing = t_bearing;
        obj.L_mid = belt_width + 10 + 2*L1;
        obj.Plane_keyhub = obj.L_mid/2+obj.L_bearing+20;
        obj = obj.calculate(F_T1, F_T2, Torque, L1, L3, belt_width);
        obj = obj.keys(Torque,belt_width);
        obj.mass = pi*((obj.D_big/2)^2)*(obj.L_mid)*obj.density;
    end

    % Parametric Calculation function (Section 4.5.5.3 of Analysis Report)
    function obj = calculate(obj, F_T1, F_T2, Torque, L1, L3, belt_width)
        t_pulley = belt_width+10;
        t_bearing = obj.L_bearing;
        T = Torque/0.95;
        delta_1 =180;
        delta_2 =180;

        %% Parametrization Loop
```

```
D =5;
SafetyFactor = [0,0,0,0];
while(SafetyFactor(1) < 2.5 || SafetyFactor(2) < 2.5 ||
    SafetyFactor(3)< 2.5 || SafetyFactor(4)< 2.5)
    D = D + 0.5;
    d = D - 4;

    %% Joint Forces (Equation 104, 105, 106, 107 in Analysis
       Report)
    B_y_2 = ((0.5*t_bearing + L1 + 0.5*t_pulley)*(F_T1*sind(
       delta_1) + F_T2 * sind(delta_2)))/(t_bearing + 2*L1 +
       t_pulley);
    B_x_2 = -((0.5*t_bearing + L1 + 0.5*t_pulley)*(F_T1*cosd(
       delta_1) + F_T2 * cosd(delta_2)))/(t_bearing + 2*L1 +
       t_pulley);

    B_x_1 = -F_T1*cosd(delta_1) - F_T2* cosd(delta_2) - B_x_2;
    B_y_1 = F_T1*sind(delta_1) + F_T2* sind(delta_2) - B_y_2;

    %% Shear and Moment Graph

    %(1) 0, (2) Step1, (3) Pulley, (4) Step 2, (5) Bearing 2, (6)
       Hub
    Location =  [0, t_bearing/2,                0.5*t_bearing+L1+0
       .5*t_pulley,                      0.5*t_bearing+L1+0.5*
       t_pulley+0.5*t_pulley+L1,                      0.5*
       t_bearing+L1+0.5*t_pulley+0.5*t_pulley+L1+0.5*t_bearing, 0
       .5*t_bearing+L1+0.5*t_pulley+0.5*t_pulley+L1+0.5*t_bearing
        + 0.5*t_bearing+L3 ];
    Forces_x =  [0, B_x_1,  B_x_1,             B_x_1+F_T1*cosd(
       delta_2)+F_T2*cosd(delta_2),       B_x_1+F_T1*cosd(
       delta_2)+F_T2*cosd(delta_2),                      B_x_1+
       F_T1*cosd(delta_2)+F_T2*cosd(delta_2)+ B_x_2];
    Forces_y =  [0, -B_y_1,  -B_y_1,            -B_y_1 + F_T1*sind
       (delta_1) + F_T2*sind(delta_2) ,  -B_y_1 + F_T1*sind(
       delta_1) + F_T2*sind(delta_2),                 -B_y_1 +
       F_T1*sind(delta_1) + F_T2*sind(delta_2) - B_y_2];
    Moment_x =  [0, Forces_y(2)*Location(2),    Forces_y(3)*
       Location(3),                           Forces_y(3)*
       Location(3)+Forces_y(4)*(Location(4)-Location(3)),
       Forces_y(3)*Location(3)+Forces_y(4)*(Location(4)-Location
```

```
    (3))+Forces_y(5)*(Location(5)-Location(4))];
Moment_y =  [0, Forces_x(2)*Location(2),    Forces_x(3)*
    Location(3),                            Forces_x(3)*
    Location(3)+Forces_x(4)*(Location(4)-Location(3)),
    Forces_x(3)*Location(3)+Forces_x(4)*(Location(4)-Location
    (3))+Forces_x(5)*(Location(5)-Location(4))];
Torsion  =  [0, 0,                          T,
                                                    T,

    T];


%% Critical Location - Step 1 (Equation 91 to 96 in Analysis
    Report)
Total_Shear = sqrt(Forces_x(2)^2+Forces_y(2)^2);
Total_Moment = sqrt(Moment_x(2)^2+Moment_y(2)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(1))/(pi*d^3);
Shear_Stress = (4/3)*obj.CF_S(1)*Total_Shear/((pi*d^2)/4);
Torsion_Stress = (16*obj.CF_T(1)*Torsion(2)/(pi*d^3));
equivalent_Stress = sqrt(Bending_Stress^2+3*Torsion_Stress^2);
SafetyFactor(1) = obj.M_sy/equivalent_Stress;


%% Critical Location - Pulley (Equation 91 to 96 in Analysis
    Report)
Total_Shear = sqrt(Forces_x(3)^2+Forces_y(3)^2);
Total_Moment = sqrt(Moment_x(3)^2+Moment_y(3)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(2))/(pi*D^3);
Shear_Stress = (4/3)*obj.CF_S(2)*Total_Shear/((pi*D^2)/4);
Torsion_Stress = (16*obj.CF_T(2)*Torsion(3)/(pi*D^3));
equivalent_Stress = sqrt(Bending_Stress^2+3*Torsion_Stress^2);
SafetyFactor(2) = obj.M_sy/equivalent_Stress;


%% Critical Location - Step 2 (Equation 91 to 96 in Analysis
    Report)
Total_Shear = sqrt(Forces_x(4)^2+Forces_y(4)^2);
Total_Moment = sqrt(Moment_x(4)^2+Moment_y(4)^2);
```

```
            Bending_Stress = (32*Total_Moment*obj.CF_B(3))/(pi*d^3);
            Shear_Stress = (4/3)*obj.CF_S(3)*Total_Shear/((pi*d^2)/4);
            Torsion_Stress = (16*obj.CF_T(3)*Torsion(4)/(pi*d^3));
            equivalent_Stress = sqrt(Bending_Stress^2+3*Torsion_Stress^2);
            SafetyFactor(3) = obj.M_sy/equivalent_Stress;


            %% Critical Location - Hub (Equation 91 to 96 in Analysis
               Report)
            Total_Shear = sqrt(Forces_x(5)^2+Forces_y(5)^2);
            Total_Moment = sqrt(Moment_x(5)^2+Moment_y(5)^2);

            Bending_Stress = (32*Total_Moment*obj.CF_B(4))/(pi*d^3);
            Shear_Stress = (4/3)*obj.CF_S(4)*Total_Shear/((pi*d^2)/4);
            Torsion_Stress = (16*obj.CF_T(4)*Torsion(4)/(pi*d^3));
            equivalent_Stress = sqrt(Bending_Stress^2+3*Torsion_Stress^2);
            SafetyFactor(4) = obj.M_sy/equivalent_Stress;

        end
        obj.D_big=D;
        obj.d_small=d;

    end

    % Key calculation function
    function obj = keys(obj,Torque,t_pulley)
        [obj.w_keypulley, h_keypulley,obj.worked_keypulley] = keys(
           obj.D_big,Torque,t_pulley);
        [obj.w_keyhub, h_keyhub, obj.worked_keyhub] = keys(obj.d_small,
           Torque, 15);
        obj.h_keypulley = h_keypulley/2;
        obj.h_keyhub = h_keyhub/2;
    end

    % Print function to TXT file
    function printTXT(obj)
        filePath = '..\Solidworks\Equations\HipKneeShaft.txt';
        fid = fopen(filePath,'wt');
        fprintf(fid,strcat('"D_big"= ',num2str(obj.D_big),'\n'));
        fprintf(fid,strcat('"d_small"= ',num2str(obj.d_small),'\n'));
        fprintf(fid,strcat('"L_bearing"= ',num2str(obj.L_bearing),'\n'));
```

```
            fprintf(fid,strcat('"L_mid"= ',num2str(obj.L_mid),'\n'));
            fprintf(fid,strcat('"w_keypulley"= ',num2str(obj.w_keypulley),'\n'
                ));
            fprintf(fid,strcat('"h_keypulley"= ',num2str(obj.h_keypulley),'\n'
                ));
            fprintf(fid,strcat('"w_keyhub"= ',num2str(obj.w_keyhub),'\n'));
            fprintf(fid,strcat('"h_keyhub"= ',num2str(obj.h_keyhub),'\n'));
            fprintf(fid,strcat('"Plane_keyhub"= ',num2str(obj.Plane_keyhub),'\
                n'));
            fclose(fid);
        end

    end
end
```

C.11 Shaft Knee

```
classdef Shaft_Knee
    properties
        D_big                       % [mm] Big diameter of shaft
        d_small                     % [mm] Small diameter of shaft
        L_mid                       % [mm] Length of big diameter portion
        L_bearing                   % [mm] Length of bearing
        L_total_knee_shaft          % [mm] Length of knee shaft
        L                           % [mm] Distance of torsion spring and
            spring plate
        mass                        % [kg] Mass

        %Constants
        M_sy = 240;                 % [MPa] Yield strength
        density = 8 * (10^(-6));    % [kg/mm^3] Density

        %Concentration Factors
        CF_B = [1.9,1,1.9];         % Bending
        CF_S = [1.6,1,1.6];         % Shear
        CF_A =[2,1,2];              % Axial
    end
    methods

        %Shaft Knee Constructor
```

```
function obj = Shaft_Knee(forces,leg, F_t1, F_t2,torsion_spring_length
    ,t_pulley, l_mid_HipKneeShaft,t_bearing)
   obj = obj.setParameters(forces,leg, F_t1, F_t2,
      torsion_spring_length,t_pulley, l_mid_HipKneeShaft,t_bearing);
end

% Properties and parameters calculation function
function obj = setParameters(obj,forces,leg, F_t1, F_t2,
   torsion_spring_length,t_pulley, l_mid_HipKneeShaft,t_bearing )
    obj.L_mid = l_mid_HipKneeShaft;
    obj.L_total_knee_shaft = obj.L_mid + 2*t_bearing;
    obj.L = torsion_spring_length +5;
    obj.L_bearing = t_bearing;
    obj = obj.calculate(forces,leg,F_t1, F_t2, t_pulley,t_bearing);
    obj.mass = pi*((obj.D_big/2)^2)*(obj.L_mid)*obj.density;

end


% Parametric Calculation function (Section 4.5.5.1 of Analysis Report)
function obj = calculate(obj,forces,leg, F_t1, F_t2,t_pulley,
   t_bearing)

    F_tibia = forces.F_tibia;
    f_friction = forces.ff;
    theta = leg.Theta +90;
    L=obj.L;
    D =5;
    delta_1 =0;
    delta_2 =0;
    SafetyFactor = [0,0,0];
    h_tibia=leg.h_tibia;
    l_tibia =leg.l_tibia;

    while(SafetyFactor(1) < 2.5 || SafetyFactor(2) < 2.5 ||
       SafetyFactor(3)< 2.5)
        D = D + 0.5;
        d = D - 4;

        %% Joint Forces (Equation 87, 88, 89, 90 in Analysis Report)
        B_y_2 = ((0.5*t_bearing + L + 0.5*t_pulley)*(F_tibia*sind(
```

```
    theta) + F_t1 * sind(delta_1) + F_t2 * sind(delta_2))-(
    f_friction*h_tibia*cosd(theta-90))+(f_friction*l_tibia *
    cosd(theta)))/(t_bearing + 2*L+t_pulley);
B_x_2 = ((0.5*t_bearing + L + 0.5*t_pulley)*(F_tibia*cosd(
    theta) + F_t1 * cosd(delta_1) + F_t2 * cosd(delta_2))-(
    f_friction*h_tibia*sind(theta-90))-(f_friction*l_tibia *
    sind(theta)))/(t_bearing + 2*L+t_pulley);

B_x_1 = F_tibia*cosd(theta) + F_t1* cosd(delta_1) + F_t2*cosd(
    delta_2) - B_x_2;
B_y_1 = F_tibia*sind(theta) + F_t1* sind(delta_1) + F_t2*sind(
    delta_2) - B_y_2;

%% Shear and Moment Graph

%(1) 0, (2) Step1, (3) Pulley, (4) Step 2, (5) Bearing 2
Location =[0,   t_bearing/2,    0.5*t_bearing+L+0.5*t_pulley,
     t_bearing+2*L+t_pulley-0.5*t_bearing,   t_bearing+2*L+
    t_pulley];
Forces_x = [0,    B_x_1,  B_x_1,  B_x_1-(F_t1*cosd(delta_1)+
    F_t2*cosd(delta_2)+F_tibia*cosd(theta)),          B_x_1-(
    F_t1*cosd(delta_1)+F_t2*cosd(delta_2)+F_tibia*cosd(theta))
    ,          B_x_1-(F_t1*cosd(delta_1)+F_t2*cosd(delta_2)+
    F_tibia*cosd(theta))+ B_x_2];
Forces_y = [0,    B_y_1,  B_y_1,  B_y_1 - (F_t1*sind(delta_1)
    + F_t2*sind(delta_2) + F_tibia*sind(theta)),    B_y_1 - (
    F_t1*sind(delta_1) + F_t2*sind(delta_2) + F_tibia*sind(
    theta)),    B_y_1 - (F_t1*sind(delta_1) + F_t2*sind(
    delta_2) + F_tibia*sind(theta)) + B_y_2];
Moment_x =[0,    Forces_y(2)*Location(2),  Forces_y(3)*
    Location(3),  Forces_y(3)*Location(3)-f_friction*h_tibia*
    cosd(theta-90) + f_friction*l_tibia*cosd(theta),   (
    Forces_y(3)*Location(3)-f_friction*h_tibia*cosd(theta-90)
    + f_friction*l_tibia*cosd(theta)) + (Location(4) -
    Location(3))*Forces_y(5)];
Moment_y =[0,    Forces_x(2)*Location(2),  Forces_x(3)*
    Location(3),  Forces_x(3)*Location(3)-f_friction*h_tibia*
    sind(theta-90) - f_friction*l_tibia*sind(theta),   (
    Forces_x(3)*Location(3)-f_friction*h_tibia*sind(theta-90)
    - f_friction*l_tibia*sind(theta)) + (Location(4) -
    Location(3))*Forces_x(5)];
```

```
%% Critical Location - Step 1 (Equation 91 to 96 in Analysis
   Report)
Total_Shear = sqrt(Forces_x(2)^2+Forces_y(2)^2);
Total_Moment = sqrt(Moment_x(2)^2+Moment_y(2)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(1))/(pi*d^3);
Shear_Stress = (4/3)*obj.CF_S(1)*Total_Shear/((pi*d^2)/4);
Axial_Stress = (4*f_friction*obj.CF_A(1))/(pi*d^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
   Shear_Stress^2);
SafetyFactor(1) = obj.M_sy/equivalent_Stress;


%% Critical Location - Pulley (Equation 91 to 96 in Analysis
   Report)
Total_Shear = sqrt(Forces_x(3)^2+Forces_y(3)^2);
Total_Moment = sqrt(Moment_x(3)^2+Moment_y(3)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(2))/(pi*D^3);
Shear_Stress = (4/3)*obj.CF_S(2)*Total_Shear/((pi*D^2)/4);
Axial_Stress = (4*f_friction*obj.CF_A(2))/(pi*D^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
   Shear_Stress^2);
SafetyFactor(2) = obj.M_sy/equivalent_Stress;


%% Critical Location - Step 2 (Equation 91 to 96 in Analysis
   Report)
Total_Shear = sqrt(Forces_x(5)^2+Forces_y(5)^2);
Total_Moment = sqrt(Moment_x(5)^2+Moment_y(5)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(3))/(pi*d^3);
Shear_Stress = (4/3)*obj.CF_S(3)*Total_Shear/((pi*d^2)/4);
Axial_Stress = (4*f_friction*obj.CF_A(3))/(pi*d^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
   Shear_Stress^2);
SafetyFactor(3) = obj.M_sy/equivalent_Stress;
```

```
            end
            obj.D_big =D;
            obj.d_small=d;

        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\KneeShaft.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_big"= ',num2str(obj.D_big),'\n'));
            fprintf(fid,strcat('"d_small"= ',num2str(obj.d_small),'\n'));
            fprintf(fid,strcat('"L_mid"= ',num2str(obj.L_mid),'\n'));
            fprintf(fid,strcat('"L_bearing"= ',num2str(obj.L_bearing),'\n'));
            fclose(fid);
        end

    end
end
```

C.12 Shaft Hip

```
classdef Shaft_Hip
    properties
        D_big                       % [mm] Big diameter of shaft
        d_small                     % [mm] Small diameter of shaft
        L_mid                       % [mm] Length of big diameter portion
        L_bearing                   % [mm] Length of bearing
        Plane_keyplates             % [mm] Mid plane of keys for hip
            plates
        w_keyhub                    % [mm] Width key hub
        h_keyhub                    % [mm] Height key hub
        w_keyplates                 % [mm] Width key plates
        h_keyplates                 % [mm] Height key plates
        Plane_keyhub                % [mm] Plane for collar key
        mass                        % [kg] Mass
        worked_keyplates            % Boolean if key length is sufficient
            for plates
        worked_keyhub               % Boolean if key length is sufficient
            for hub
```

```
    %Constants
    M_sy = 240;                     % [MPa] Yield Strength
    density = 8 * (10^(-6));        % [kg/mm^3] Density

    %Concentration Factors
    CF_B = [2,1.8,1.8,2,1.8];       % Bending
    CF_S = [1.7,1.8,1.8,1.7,1.8];   % Shear
    CF_T =[1.7,1.8,1.8,1.7,1.8];    % Torsion
    CF_A =[2.1,1.8,1.8,2.1,1.8];    % Axial
end

methods

    % Shaft Hip Constructor
    function obj = Shaft_Hip(forces,leg, L1, L3, t_plate,t_bearing,
       l_mid_knee_hip_shaft,w_spring_hip)
        obj = obj.setParameters(forces,leg, L1, L3, t_plate,t_bearing,
           l_mid_knee_hip_shaft,w_spring_hip);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,forces,leg, L1, L3, t_plate,t_bearing
       ,l_mid_knee_hip_shaft,w_spring_hip)
        obj.L_mid = 2*w_spring_hip + 2*10 + l_mid_knee_hip_shaft + 2*2;
        obj.L_bearing = t_bearing;
        obj.Plane_keyplates = (l_mid_knee_hip_shaft+14)/2;
        obj.Plane_keyhub = obj.L_mid/2+t_bearing+20;
        obj = obj.calculate(forces,leg, L1, l_mid_knee_hip_shaft + 2*2, L3
           , t_plate,t_bearing);
        obj = obj.keys(forces.Torque_hip*1000);
        obj.mass = pi*((obj.D_big/2)^2)*(obj.L_mid)*obj.density;
    end

    % Keys Calculation
    function obj = keys(obj,Torque)
        [obj.w_keyplates, h_keyplates, obj.worked_keyplates] = keys(
           obj.D_big,Torque,10);
        [obj.w_keyhub, h_keyhub, obj.worked_keyhub] = keys(obj.d_small,
           Torque, 15);
        obj.Plane_keyhub = obj.L_mid/2+obj.L_bearing+20; %5 clearance + 15
```

```
        hub [mm]
    obj.h_keyplates = h_keyplates/2;
    obj.h_keyhub = h_keyhub/2;
end

% Parametric Calculation function (Section 4.5.5.2 of Analysis Report)
function obj = calculate(obj,forces,leg, L1, L2, L3, t_plate,t_bearing
    )
    h_leg = leg.h_leg;
    l_leg = leg.l_leg;
    F_thigh = forces.F_thigh;
    f_friction = forces.ff;
    T = forces.Torque_hip*1000;

    %% Joint Forces (Equation 98, 99, 100, 101 in Analysis Report)
    B_y_2 = ((t_bearing+2*L1+2*t_plate+L2)*0.5*F_thigh-(f_friction*
        h_leg))/(t_bearing+2*L1+2*t_plate+L2);
    B_x_2 = -(f_friction*l_leg)/(t_bearing+2*L1+2*t_plate+L2);

    B_x_1 = -1*B_x_2;
    B_y_1 = F_thigh-B_y_2;

    %% Shear and Moment Graph

    %(1) 0, (2) Step1, (3) Hip Plate 1, (4) Hip Plate 2, (5) Step 2,
        (6)Bearing
    %2 (7) Hub
    Location = [0,  t_bearing/2,    0.5*t_bearing+L1+0.5*t_plate,   0
        .5*t_bearing+L1+0.5*t_plate + t_plate+L2,   0.5*t_bearing+L1+0
        .5*t_plate + t_plate+L2+0.5*t_plate+L1,0.5*t_bearing+L1+0.5*
        t_plate + t_plate+L2+0.5*t_plate+L1+0.5*t_bearing, 0.5*
        t_bearing+L1+0.5*t_plate + t_plate+L2+0.5*t_plate+L1+0.5*
        t_bearing+0.5*t_bearing+L3];
    Forces_x = [0,  B_x_1,  B_x_1,  B_x_1,          B_x_1,
        B_x_1 + B_x_2];
    Forces_y = [0,  B_y_1, B_y_1 - F_thigh/2,    B_y_1 - F_thigh,
        B_y_1 - F_thigh,   B_y_1 - F_thigh+ B_y_2];
    Moment_x = [0,  Forces_y(2)*Location(2),  Forces_y(2)*Location(3),
         Forces_y(2)*Location(3)-0.5*f_friction*h_leg,      Forces_y
        (2)*Location(3)-0.5*f_friction*h_leg + Forces_y(3)*(Location
        (4)-Location(3)),   Forces_y(2)*Location(3)-0.5*f_friction*
```

```
    h_leg+Forces_y(3)*(Location(4)-Location(3))-0.5*f_friction*
    h_leg,      Forces_y(2)*Location(3)-0.5*f_friction*h_leg+
    Forces_y(3)*(Location(4)-Location(3))-0.5*f_friction*h_leg + (
    Location(5)-Location(4))*Forces_y(4), Forces_y(2)*Location(3)
    -0.5*f_friction*h_leg+Forces_y(3)*(Location(4)-Location(3))-0
    .5*f_friction*h_leg + (Location(5)-Location(4))*Forces_y(4)+(
    Location(6)-Location(5))*Forces_y(5)];
Moment_y = [0,  Forces_x(2)*Location(2),  Forces_x(2)*Location(3),
     Forces_x(2)*Location(3)-0.5*f_friction*l_leg,      Forces_x
    (2)*Location(3)-0.5*f_friction*l_leg + Forces_x(3)*(Location
    (4)-Location(3)),   Forces_x(2)*Location(3)-0.5*f_friction*
    l_leg + Forces_x(3)*(Location(4)-Location(3))-0.5*f_friction*
    l_leg,    Forces_x(2)*Location(3)-0.5*f_friction*l_leg +
    Forces_x(3)*(Location(4)-Location(3))-0.5*f_friction*l_leg+ (
    Location(5)-Location(4))*Forces_x(4), Forces_x(2)*Location(3)
    -0.5*f_friction*l_leg + Forces_x(3)*(Location(4)-Location(3))
    -0.5*f_friction*l_leg+ (Location(5)-Location(4))*Forces_x(4) +
     (Location(6)-Location(5))*Forces_x(5)];
Torsion  = [0,  0,  T*0.5,  T,  T,  T,  0];

%% Parametrization Loop
SafetyFactor = [0,0,0,0,0];
D=5;
while(SafetyFactor(1) < 2.5 || SafetyFactor(2) < 2.5 ||
    SafetyFactor(3)< 2.5 || SafetyFactor(4)< 2.5 || SafetyFactor
    (5)< 2.5)
    D = D + 0.5;
    d = D - 4;


    %% Critical Location - Step 1 (Equation 91 to 96, and 102, 103
        in Analysis Report)
    Total_Shear = sqrt(Forces_x(2)^2+Forces_y(2)^2);
    Total_Moment = sqrt(Moment_x(2)^2+Moment_y(2)^2);

    Bending_Stress = (32*Total_Moment*obj.CF_B(1))/(pi*d^3);
    Shear_Stress = (4/3)*obj.CF_S(1)*Total_Shear/((pi*d^2)/4);
    Torsion_Stress = (16*obj.CF_T(1)*Torsion(2)/(pi*d^3));
    Axial_Stress = (4*f_friction*obj.CF_A(1))/(pi*d^2);
    equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
       Torsion_Stress^2);
```

```
SafetyFactor(1) = obj.M_sy/equivalent_Stress;


%% Critical Location - Hip Plate 1 (Equation 91 to 96, and
   102, 103 in Analysis Report)
Total_Shear = sqrt(Forces_x(2)^2+Forces_y(2)^2);
Total_Moment = sqrt(Moment_x(3)^2+Moment_y(4)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(2))/(pi*D^3);
Shear_Stress = (4/3)*obj.CF_S(2)*Total_Shear/((pi*D^2)/4);
Torsion_Stress = (16*obj.CF_T(2)*Torsion(3)/(pi*D^3));
Axial_Stress = (4*f_friction*obj.CF_A(2))/(pi*D^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
   Torsion_Stress^2);
SafetyFactor(2) = obj.M_sy/equivalent_Stress;


%% Critical Location - Hip Plate 2 (Equation 91 to 96, and
   102, 103 in Analysis Report)
Total_Shear = sqrt(Forces_x(3)^2+Forces_y(3)^2);
Total_Moment = sqrt(Moment_x(5)^2+Moment_y(5)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(3))/(pi*D^3);
Shear_Stress = (4/3)*obj.CF_S(3)*Total_Shear/((pi*D^2)/4);
Torsion_Stress = (16*obj.CF_T(3)*Torsion(5)/(pi*D^3));
Axial_Stress = (4*f_friction*obj.CF_A(3))/(pi*D^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
   Torsion_Stress^2);
SafetyFactor(3) = obj.M_sy/equivalent_Stress;


%% Critical Location - Step 2 (Equation 91 to 96, and 102, 103
    in Analysis Report)
Total_Shear = sqrt(Forces_x(4)^2+Forces_y(4)^2);
Total_Moment = sqrt(Moment_x(7)^2+Moment_y(7)^2);

Bending_Stress = (32*Total_Moment*obj.CF_B(4))/(pi*d^3);
Shear_Stress = (4/3)*obj.CF_S(4)*Total_Shear/((pi*d^2)/4);
Torsion_Stress = (16*obj.CF_T(4)*Torsion(6)/(pi*d^3));
Axial_Stress = (4*f_friction*obj.CF_A(4))/(pi*d^2);
equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
```

```
            Torsion_Stress^2);
        SafetyFactor(4) = obj.M_sy/equivalent_Stress;


        %% Critical Location - Hub (Equation 91 to 96, and 102, 103 in
            Analysis Report)
        Total_Shear = sqrt(Forces_x(6)^2+Forces_y(6)^2);
        Total_Moment = sqrt(Moment_x(8)^2+Moment_y(8)^2);

        Bending_Stress = (32*Total_Moment*obj.CF_B(5))/(pi*D^3);
        Shear_Stress = (4/3)*obj.CF_S(5)*Total_Shear/((pi*D^2)/4);
        Torsion_Stress = (16*obj.CF_T(5)*Torsion(6)/(pi*D^3));
        Axial_Stress = (4*f_friction*obj.CF_A(5))/(pi*D^2);
        equivalent_Stress = sqrt((Bending_Stress + Axial_Stress)^2+3*
            Torsion_Stress^2);
        SafetyFactor(5) = obj.M_sy/equivalent_Stress;
    end
    obj.D_big = D;
    obj.d_small = d;
end


% Print function to TXT file
function printTXT(obj)
    filePath = '..\Solidworks\Equations\HipShaft.txt';
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"D_big"= ',num2str(obj.D_big),'\n'));
    fprintf(fid,strcat('"d_small"= ',num2str(obj.d_small),'\n'));
    fprintf(fid,strcat('"L_mid"= ',num2str(obj.L_mid),'\n'));
    fprintf(fid,strcat('"L_bearing"= ',num2str(obj.L_bearing),'\n'));
    fprintf(fid,strcat('"Plane_keyplates"= ',num2str(
        obj.Plane_keyplates),'\n'));
    fprintf(fid,strcat('"w_keyhub"= ',num2str(obj.w_keyhub),'\n'));
    fprintf(fid,strcat('"h_keyhub"= ',num2str(obj.h_keyhub),'\n'));
    fprintf(fid,strcat('"w_keyplates"= ',num2str(obj.w_keyplates),'\n'
        ));
    fprintf(fid,strcat('"h_keyplates"= ',num2str(obj.h_keyplates),'\n'
        ));
    fprintf(fid,strcat('"Plane_keyhub"= ',num2str(obj.Plane_keyhub),'\
        n'));
    fclose(fid);
```

```
        end

    end
end
```

C.13 Bearing Hip Knee

```
classdef Bearing_KneeHip
    properties
        d_shafthipknee  % [mm] Diameter
    end
    methods

        % Bearing Knee Hip Constructor
        function obj = Bearing_KneeHip(d_shaft_KneeHip)
            obj = obj.setParameters(d_shaft_KneeHip);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, d_shaft_KneeHip)
            obj.d_shafthipknee = d_shaft_KneeHip;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipKneeBearing.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_shafthipknee"= ',num2str(obj.d_shafthipknee
               ),'\n'));
            fclose(fid);
        end

    end
end
```

C.14 Key Hip Knee Pulley

```
classdef Key_KneeHip_Pulley
    properties
```

```
        h    % [mm] Height
        w    % [mm] Width
    end
    methods

        % Key Knee Hip Pulley Constructor
        function obj = Key_KneeHip_Pulley(shaft_KneeHip)
            obj = obj.setParameters(shaft_KneeHip);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_KneeHip )
            obj.h = shaft_KneeHip.h_keypulley*2;
            obj.w = shaft_KneeHip.w_keypulley;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Keys\HipKneeKeyPulley.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"h"= ',num2str(obj.h),'\n'));
            fprintf(fid,strcat('"w"= ',num2str(obj.w),'\n'));
            fclose(fid);
        end

    end
end
```

C.15 Key Hip Knee Hub

```
classdef Key_KneeHip_Hub
    properties
        h    % [mm] Height
        w    % [mm] Width
    end
    methods

        % Key Knee Hip Hub Constructor
        function obj = Key_KneeHip_Hub(shaft_KneeHip)
            obj = obj.setParameters(shaft_KneeHip);
```

```
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_KneeHip )
            obj.h = shaft_KneeHip.h_keyhub*2;
            obj.w = shaft_KneeHip.w_keyhub;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Keys\
               HipKneeKeyFlangeCollar.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"h"= ',num2str(obj.h),'\n'));
            fprintf(fid,strcat('"w"= ',num2str(obj.w),'\n'));
            fclose(fid);
        end

    end
end
```

C.16 Spacer Hip Knee

```
classdef Spacer_KneeHip
    properties
        D_shafthipknee  % [mm] Inner diameter of spacer
        D_o             % [mm] Diameter of spring
        L               % [mm] Length of spacer
    end
    methods

        % Spacer Knee Hip Constructor
        function obj = Spacer_KneeHip(shaft_KneeHip,L1,d_o_spring,t_bolt_head)
            obj = obj.setParameters(shaft_KneeHip,L1,d_o_spring,t_bolt_head);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_KneeHip,L1,d_o_spring,
           t_bolt_head )
            obj.D_shafthipknee = shaft_KneeHip.D_big;
```

```
            obj.D_o = d_o_spring-3;
            obj.L = L1;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Spacers\
               HipKneeSpacerSpring.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_shafthipknee"= ',num2str(obj.D_shafthipknee
               ),'\n'));
            fprintf(fid,strcat('"D_o"= ',num2str(obj.D_o),'\n'));
            fprintf(fid,strcat('"L"= ',num2str(obj.L),'\n'));
            fclose(fid);
        end

    end
end
```

C.17 Bearing Knee

```
classdef Bearing_Knee
    properties
        d_shaftknee % [mm] Diameter
    end
    methods

        % Bearing Knee Constructor
        function obj = Bearing_Knee(shaft_Knee_d)
            obj = obj.setParameters(shaft_Knee_d);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_Knee_d)
            obj.d_shaftknee = shaft_Knee_d;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\KneeBearing.txt';
```

```
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_shaftknee"= ',num2str(obj.d_shaftknee),'\n'
                ));
            fclose(fid);
        end

    end
end
```

C.18 Spacer Knee

```
classdef Spacer_Knee
    properties
        D_shaftknee % [mm] Inner diameter of spacer
        L           % [mm] Length of spacer
    end
    methods

        % Spacer Knee Constructor
        function obj = Spacer_Knee(shaft_Knee, t_pulley)
            obj = obj.setParameters(shaft_Knee, t_pulley);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_Knee, t_pulley)
            obj.D_shaftknee = shaft_Knee.D_big;
            obj.L = (shaft_Knee.L_mid-t_pulley-20)/2;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Spacers\KneeSpacer.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_shaftknee"= ',num2str(obj.D_shaftknee),'\n'
                ));
            fprintf(fid,strcat('"L"= ',num2str(obj.L),'\n'));
            fclose(fid);

        end
```

```
    end
end
```

C.19 Bearing Hip

```
classdef Bearing_Hip
    properties
        d_shafthip  % [mm] Diameter
    end
    methods

        % Bearing Hip Constructor
        function obj = Bearing_Hip(d_shaft_Hip)
            obj = obj.setParameters(d_shaft_Hip);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,d_shaft_Hip)
            obj.d_shafthip = d_shaft_Hip;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipBearing.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_shafthip"= ',num2str(obj.d_shafthip),'\n'))
                ;
            fclose(fid);

        end

    end
end
```

C.20 Key Hip Plates

```
classdef Key_Hip_Plates
    properties
        h   % [mm] Height
```

```
        w   % [mm] Width
    end
    methods

        % Key Hip Plates Constructor
        function obj = Key_Hip_Plates(shaft_Hip)
            obj = obj.setParameters(shaft_Hip);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_Hip)
            obj.h = shaft_Hip.h_keyplates*2;
            obj.w = shaft_Hip.w_keyplates;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Keys\HipKeyPlates.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"h"= ',num2str(obj.h),'\n'));
            fprintf(fid,strcat('"w"= ',num2str(obj.w),'\n'));
            fclose(fid);

        end

    end
end
```

C.21 Key Hip Collar

```
classdef Key_Hip_Collar
    properties
        h   % [mm] Height
        w   % [mm] Width
    end
    methods

        % Key Hip Collar Constructor
        function obj = Key_Hip_Collar(shaft_Hip)
            obj = obj.setParameters(shaft_Hip);
```

```
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,shaft_Hip)
            obj.h = shaft_Hip.h_keyhub*2;
            obj.w = shaft_Hip.w_keyhub;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Keys\HipKeyFlangeCollar.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"h"= ',num2str(obj.h),'\n'));
            fprintf(fid,strcat('"w"= ',num2str(obj.w),'\n'));
            fclose(fid);

        end

    end
end
```

C.22 Spacer Hip Mid

```
classdef Spacer_Hip_Mid
    properties
        D_shafthip  % [mm] Inner diameter of spacer
        L           % [mm] Length of spacer
    end
    methods

        % Spacer Hip Mid Constructor
        function obj = Spacer_Hip_Mid(D_shaft_Hip, L_mid_KneeHip)
            obj = obj.setParameters(D_shaft_Hip, L_mid_KneeHip);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,D_shaft_Hip, L_mid_KneeHip)
            obj.D_shafthip = D_shaft_Hip;
            obj.L = L_mid_KneeHip+4;
        end
```

```
        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Spacers\HipSpacerMid.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_shafthip"= ',num2str(obj.D_shafthip),'\n'))
               ;
            fprintf(fid,strcat('"L"= ',num2str(obj.L),'\n'));
            fclose(fid);

        end

    end
end
```

C.23 Spacer Hip Spring

```
classdef Spacer_Hip_Spring
    properties
        D_shafthip  % [mm] Inner diameter of spacer
        L           % [mm] Length of spacer
        D_o         % [mm] Diameter of spring
    end
    methods

        % Spacer Hip Spring Constructor
        function obj = Spacer_Hip_Spring(D_shaft_Hip, w_spring_hip,d_o_spring)
            obj = obj.setParameters(D_shaft_Hip, w_spring_hip,d_o_spring);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,D_shaft_Hip, w_spring_hip,d_o_spring)
            obj.D_shafthip = D_shaft_Hip;
            obj.L = w_spring_hip;
            obj.D_o = d_o_spring-3; % d_o_spring is Di_h from torsionSpring.m
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\Spacers\HipSpacerSpring.txt';
```

```
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_shafthip"= ',num2str(obj.D_shafthip),'\n'))
               ;
            fprintf(fid,strcat('"L"= ',num2str(obj.L),'\n'));
            fprintf(fid,strcat('"D_o"= ',num2str(obj.D_o),'\n'));
            fclose(fid);

        end

    end
end
```

C.24 Tibia Tube

```
classdef TibiaTube
    properties
        D                       % [mm] Outer Diameter of the tube
        d                       % [mm] Inner Diameter of the tube
        sy = 250;               % [MPa] Yield Strength
        density = 2.7/(10^6);   % [kg/mm^3] density of aluminum
        height                  % [mm] Height between ground and r2
        r2_partial              % [mm] Distance of upper tibia before press
           fit
        r_bend                  % [mm] Radius of bend
        r3_minusfoot            % [mm] Length of lower tibia witout foot
        m2                      % [kg] mass of upper tibia
        m3                      % [kg] mass of lower tibia
    end
    methods

        % Tibia Tube Constructor
        function obj = TibiaTube(forces, leg, D_pulley)
            obj = obj.calculate(forces,leg ,D_pulley);
        end

        % Parametric Calculation function (Section 4.2 of Analysis Report)
        function obj = calculate(obj,forces, leg,D_pulley)

            % Variable Assignment
            obj.height = leg.r3*sind(leg.curveAngle);
```

```
precision =0.5; % Precision of loop
obj.D = 4 - precision; % Starting outter diameter
n=0; % Starting safety factor
phi = leg.Psy;
FN = forces.FN;
FF = forces.FF;
ff = forces.ff;
theta = 360-leg.Alpha;
r2 = leg.r2;
r3 = leg.r3;

%% Determines optimal outter diameter with constant thickness for
   a safety factor of 2
while(n<=2)

    %% Geometric Calculation
    obj.D=obj.D+precision;
    obj.d = obj.D - 3.175;
    A = pi/4 * (obj.D^2-obj.d^2);
    I = pi/64*(obj.D^4-obj.d^4);
    m2 = A*r2*obj.density;
    m3 = A*r3*obj.density;
    obj.m2 = m2;
    obj.m3 = m3;

    %% Joint Forces (Section 4.2.6 of Analysis Report, equation :
       25, 26, 31, 32)
    B = [FF,FN-9.81*m3];
    Bm = FN*r3*cosd(theta)-FF*r3*sind(theta)-m3*9.81*r3/2*cosd(
       theta);
    Cm = B(1)*sind(phi)*r2 + B(2)*cosd(phi)*r2+Bm-m2*9.81*r2/2*
       cosd(phi);

    %% Internal Forces  (Section 4.2.6 of Analysis Report,
       equation : 36, 37, 41)
    Axial = -B(1)*cosd(phi)+B(2)*sind(phi)-m2*9.81*sind(phi);
    Radial = sqrt((B(1)*sind(phi)+B(2)*cosd(phi)-9.81*m2*cosd(phi)
       )^2+ ff^2);
    Bending = sqrt((Cm)^2 + (ff*(r3+r2))^2);

    %% Internal Stresses (Section 4.2.6 of Analysis Report,
```

```
                    equation : 42, 43, 44, 45, 46, 47)
                r_avg = obj.D/4+obj.d/4;
                Px = Axial/A;
                sigma_x = Px + Bending*obj.D/(2*I);
                Shear = Radial*2/A;
                Torsion = ((sqrt(leg.x_walking_max^2+obj.height^2 -r2^2))*ff)
                    /(2*pi*r_avg^2*(obj.D-obj.d)/2);
                tau_xy = Shear + Torsion;
                e_equivalent = (sigma_x^2+3*tau_xy^2)^(1/2);
                n = obj.sy/e_equivalent;
            end
            %% Geometric Calculation Bend
            obj.r2_partial = r2 - 3 - (0.5*D_pulley+5);
            obj.r_bend = (obj.r2_partial-17)*tand(leg.curveAngle/2);
            obj.r3_minusfoot = r3-100;
        end

       % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\TibiaTube.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_i"= ',num2str(obj.d),'\n'));
            fprintf(fid,strcat('"d_o"= ',num2str(obj.D),'\n'));
            fprintf(fid,strcat('"r_bend"= ',num2str(obj.r_bend),'\n'));
            fprintf(fid,strcat('"r2_partial"= ',num2str(obj.r2_partial),'\n'))
                ;
            fprintf(fid,strcat('"r3_minusfoot"= ',num2str(obj.r3_minusfoot),'\
                n'));
            fclose(fid);
        end

    end
end
```

C.25 Foot Cap

```
classdef FootCap
    properties
        D_sock      % [mm] Diameter of the cap
        do_tibia    % [mm] Large diameter of the tibia tube
```

```
    end
    methods

        % Foot Cap Constructor
        function obj = FootCap(d_tibia_tube,D_tibia_tube)
           obj = obj.setParameters(d_tibia_tube,D_tibia_tube);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, d_tibia_tube,D_tibia_tube)
            obj.D_sock =d_tibia_tube+8; % 2*4 mm of silicone thickness
            obj.do_tibia =D_tibia_tube;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\FootCap.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_sock"= ',num2str(obj.D_sock),'\n'));
            fprintf(fid,strcat('"do_tibia"= ',num2str(obj.do_tibia),'\n'));
            fclose(fid);
        end

    end
end
```

C.26 Foot Rod

```
classdef FootRod
    properties
        di_tibia    % [mm] Small Diameter Tibia Tube
        do_tibia    % [mm] Large Diameter Tibia Tube
    end
    methods

        % Foot Rod Constructor
        function obj = FootRod(d_tibia_tube,D_tibia_tube)
            obj = obj.setParameters(d_tibia_tube,D_tibia_tube);
        end
```

```
        % Properties and parameters calculation function
        function obj = setParameters(obj, d_tibia_tube,D_tibia_tube)
            obj.di_tibia =d_tibia_tube;
            obj.do_tibia =D_tibia_tube;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\FootRod.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"di_tibia"= ',num2str(obj.di_tibia),'\n'));
            fprintf(fid,strcat('"do_tibia"= ',num2str(obj.do_tibia),'\n'));
            fclose(fid);
        end

    end
end
```

C.27 Foot Silicone

```
classdef FootSilicone
    properties
        di_tibia    % [mm] Small diameter of Tibia Tube
    end
    methods

        % Foot Silicone Constructor
        function obj = FootSilicone(d_tibia_tube)
            obj = obj.setParameters(d_tibia_tube);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, d_tibia_tube)
            obj.di_tibia =d_tibia_tube;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\FootSilicone.txt';
            fid = fopen(filePath,'wt');
```

```
            fprintf(fid,strcat('"di_tibia"= ',num2str(obj.di_tibia),'\n'));
            fclose(fid);
        end

    end
end
```

C.28 Tibia Pulley Holder

```
classdef TibiaPulleyHolder
    properties
        D_kneeshaft             % [mm] Large Diameter of Knee Shaft
        do_tibia                % [mm] Small Diameter of Tibia Tube
        D_bellowholder          % [mm] Diameter of bellow holding cylinder
        D_pulley                % [mm] Diameter of knne pulley
        L_plate                 % [mm] Length of structural plates
        L_shaft                 % [mm] Length of shaft ---
        d_bolts                 % [mm] Diameter of bolt holes
        L_bolts                 % [mm] Length of bolt holes
        Fillet                  % [mm] Radius of fillet
        mass                    % [kg] Mass of Tibia Pulley Holder Piece
        density = 2.7/(10^6);   % [kg/mm^3] Density of Aluminum
    end
    methods

        % Tibia Pulley Holder Constructor
        function obj = TibiaPulleyHolder(shaft_Knee, do_tibia, D_pulley)
           obj = obj.setParameters(shaft_Knee, do_tibia, D_pulley);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, shaft_Knee, do_tibia, D_pulley)
            h_thigh_plate = D_pulley +4;
            obj.D_kneeshaft = shaft_Knee.D_big;
            obj.do_tibia = do_tibia;
            obj.D_bellowholder = sqrt((h_thigh_plate)^2 + (
               shaft_Knee.L_total_knee_shaft)^2)+4;
            obj.D_pulley = D_pulley;
            obj.L_plate = D_pulley+5;
            obj.L_shaft = 0.5*D_pulley +5;
```

```
            obj.Fillet = D_pulley*0.5-0.5;
            obj.d_bolts = shaft_Knee.D_big;
            obj.L_bolts = shaft_Knee.D_big;
            obj.mass = pi*((obj.D_bellowholder/2)^2)*20*obj.density;
        end

        % Bolt calculation function
        function obj = calculateBolts(obj, R, D_pulley_min, t_pulley, t_tibia,
            FF, FN)
            obj.d_bolts = fasteners_knee(R, D_pulley_min, obj.D_kneeshaft,
               t_pulley, t_tibia, FF, FN);
            obj.L_bolts = (D_pulley_min - obj.D_kneeshaft)/4+0.5*
               obj.D_kneeshaft;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\TibiaPulleyHolder.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_kneeshaft"= ',num2str(obj.D_kneeshaft),'\n'
               ));
            fprintf(fid,strcat('"do_tibia"= ',num2str(obj.do_tibia),'\n'));
            fprintf(fid,strcat('"D_bellowholder"= ',num2str(obj.D_bellowholder
               ),'\n')); % 4 mm of bellow groove
            fprintf(fid,strcat('"D_pulley"= ',num2str(obj.D_pulley),'\n'));
            fprintf(fid,strcat('"L_plate"= ',num2str(obj.L_plate),'\n')); % 5
               mm clearance
            fprintf(fid,strcat('"L_shaft"= ',num2str(obj.L_shaft),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fprintf(fid,strcat('"Fillet"= ',num2str(obj.Fillet),'\n'));
            fclose(fid);
        end

    end
end
```

C.29 Harmonic Drives

```
classdef Harmonic_Drive
```

```
properties
    filePath;
    name;                       % 'Hip' or 'Knee'
    gear_ratio = 100;           % always
    torque_rated;               % [Nm]
    torque_repeat_limit;        % [Nm]
    speed_avg_input_limit;      % [rpm]
    inertia_I;                  % [kgm^2]
    inertia_J;                  % [kgfms^2]
    torque_backdriving;         % [Nm]
    torque_starting;            % [Nm]
    mass;                       % [kg]
    spline_outer_diameter;      % [mm]
    thickness;                  % [mm] (thickness of entire HD)
    spline_thickness;           % [mm]
    spline_mounting_diameter;   % [mm]
    spline_num_bolts;           % [qty]
    spline_bolt_diameter;       % [mm]
    flex_outer_diameter;        % [mm]
    flex_mounting_diameter;     % [mm]
    flex_num_bolts;             % [qty]
    flex_bolt_diameter;         % [mm]
    shaft_diameter;             % [mm]
    key_width;                  % [mm]
    key_height;                 % [mm]
    key_length;                 % [mm]
    % Polyfit functions from generate_hd_functions.m
    T = [0.105039984693498  18.0800657612682;
        1                   8.31777767726056e-15;
        2.0676453497642     12.5829417537992;
        -2.84250955770586   3660.41143310669;
        0.0264563259788898  -0.887074575970917;
        0.026982497617217   -0.904619392487767;
        0.198435069605609   0.650071324047861;
        0.00166096629380689 0.00471147845928719;
        0.00567155506266099 -0.0259402692101916;
        0.350091945307601   63.0494201599388;
        0.0657187501107241  13.104629153926;
        0.0264860948223631  5.45233335577342;
        0.30887865699617    56.2916958683074;
        0.00874277635854066 9.99399786701059;
```

```
        0.0107686734010552  3.22755131398525;
        0.267665368684739   49.533971576676;
        0.130782899235738   21.9512477279411;
        0.00610754234976068 9.13055879362087;
        0.0239809837970755  3.74058813108318;
        0.124895919329065   22.7953095492023;
        6.22181530996435e-18    4;
        0.010132585487877   2.83407716319495];
    spline_fasteners;           % spline fastener specs
    flex_fasteners;             % flex fastener specs
end

methods
    % Constructor
    function obj = Harmonic_Drive(torque_max,hd_name)
        % torque_max [Nm]
        obj = obj.setParameters(torque_max);
        obj.name = hd_name;
        obj.filePath = strcat('..\Solidworks\Equations\HD',hd_name,'.txt')
            ;
    end

    % Properties and parameters calculation
    function obj = setParameters(obj,torque_max)
        % torque_max [Nm]
        obj.torque_rated = obj.T(2,1)*torque_max + obj.T(2,2);
        obj.torque_repeat_limit = obj.T(3,1)*torque_max + obj.T(3,2);
        obj.speed_avg_input_limit = obj.T(4,1)*torque_max + obj.T(4,2);
        obj.inertia_I = obj.T(5,1)*torque_max + obj.T(5,2);
        obj.inertia_J = obj.T(6,1)*torque_max + obj.T(6,2);
        obj.torque_backdriving = obj.T(7,1)*torque_max + obj.T(7,2);
        obj.torque_starting = obj.T(8,1)*torque_max + obj.T(8,2);
        obj.mass = obj.T(9,1)*torque_max + obj.T(9,2);
        obj.spline_outer_diameter = obj.T(10,1)*torque_max + obj.T(10,2);
        obj.thickness = obj.T(11,1)*torque_max + obj.T(11,2);
        obj.spline_thickness = obj.T(12,1)*torque_max + obj.T(12,2);
        obj.spline_mounting_diameter = obj.T(13,1)*torque_max + obj.T
            (13,2);
        obj.spline_num_bolts = obj.T(14,1)*torque_max + obj.T(14,2);
        obj.spline_bolt_diameter = obj.T(15,1)*torque_max + obj.T(15,2);
        obj.flex_outer_diameter = obj.T(16,1)*torque_max + obj.T(16,2);
```

```
        obj.flex_mounting_diameter = obj.T(17,1)*torque_max + obj.T(17,2)
            ;;
        obj.flex_num_bolts = obj.T(18,1)*torque_max + obj.T(18,2);
        obj.flex_bolt_diameter = obj.T(19,1)*torque_max + obj.T(19,2);
        %% Tweak values
        % no negative inertia
        if obj.inertia_I < 0.021
            obj.inertia_I = 0.021;
        end
        % convert from x10^-4 kgm^2 to kgm^2
        obj.inertia_I = obj.inertia_I/(10^4);
        % no negative inertia
        if obj.inertia_J < 0.021
            obj.inertia_J = 0.021;
        end
        % convert from x10^-5 kgfms^2 to kgfms^2
        obj.inertia_J = obj.inertia_J/(10^5);
        % no negative mass
        if obj.mass < 0.06
            obj.mass = 0.06;
        end
        % round number of spline bolts to integer
        obj.spline_num_bolts = ceil(obj.spline_num_bolts);
        % round spline bolt diameter to nearest mm
        obj.spline_bolt_diameter = ceil(obj.spline_bolt_diameter);
        % round number of flex bolts to integer
        obj.flex_num_bolts = ceil(obj.flex_num_bolts);
        % round flex bolts to nearest mm
        obj.flex_bolt_diameter = ceil(obj.flex_bolt_diameter);
        % Values below are approximated from HD website
        obj.shaft_diameter = obj.flex_bolt_diameter*2;
        obj.key_width = obj.flex_bolt_diameter*0.6;
        obj.key_height = obj.flex_bolt_diameter*0.2;
        obj.key_length = obj.spline_thickness;
        obj.spline_fasteners = Fastener(obj.spline_bolt_diameter);
        obj.flex_fasteners = Fastener(obj.flex_bolt_diameter);
    end

    % From Analysis Report, Section 3.5.1 Harmonic Drive Efficiency
    function [torque_input,rpm_input] = getInputs(obj,torque_output,
        rad_s_output)
```

```
    %% getInputs
    % INPUTS
    % torque_output = torque at output of HD [Nm]
    % rad_s_output = speed output of HD [rad/s]
    % OUTPUTS
    % rpm_input = rpm at input of HD [rpn]
    % torque_output = torque at input of HD [mNm]

    rpm_output = rad_s_output*30./pi;
    rpm_input = rpm_output*obj.gear_ratio;

    %% SPEED EFFICIENCY
    % Equation 13
    eta_r = (4.848*(10^(-9)))*(rpm_input.^2) + (-5.879*(10^(-5)))*(
        rpm_input) + 0.8367;
    if eta_r > 0.81
        eta_r = 0.81;
    elseif eta_r < 0.69
        eta_r = 0.69;
    end

    %% TORQUE EFFICIENCY
    % Equations 14, 15
    alpha = torque_output./obj.torque_rated;
    if alpha > 1
        alpha = 1;
    end
    k_e = (-1.481*(alpha.^4))+(4.312*(alpha.^3))-(5.013*(alpha.^2))+(3
        .159.*alpha)-0.02076;
    if k_e < 0.3
        k_e = 0.3;
    end

    %% MOTOR TORQUE
    % Equations 16, 17
    eta_HD = eta_r.*k_e;
    % Input torque in mNm
    torque_input = torque_output.*1000./(obj.gear_ratio.*eta_HD);
end

% Print function to TXT file
```

```
        function printTXT(obj)
            fid = fopen(obj.filePath,'wt');
            fprintf(fid,strcat('"spline outer diameter"=',num2str(
                obj.spline_outer_diameter),'\n'));
            fprintf(fid,strcat('"thickness total"=',num2str(obj.thickness),'\n
                '));
            fprintf(fid,strcat('"thickness spline"=',num2str(
                obj.spline_thickness),'\n'));
            fprintf(fid,strcat('"spline mounting diameter"=',num2str(
                obj.spline_mounting_diameter),'\n'));
            fprintf(fid,strcat('"spline bolts"=',num2str(obj.spline_num_bolts)
                ,'\n'));
            fprintf(fid,strcat('"spline bolt diameter"=',num2str(
                obj.spline_bolt_diameter),'\n'));
            fprintf(fid,strcat('"flex outer diameter"=',num2str(
                obj.flex_outer_diameter),'\n'));
            fprintf(fid,strcat('"flex mounting diameter"=',num2str(
                obj.flex_mounting_diameter),'\n'));
            fprintf(fid,strcat('"flex bolts"=',num2str(obj.flex_num_bolts),'\n
                '));
            fprintf(fid,strcat('"flex bolt diameter"=',num2str(
                obj.flex_bolt_diameter),'\n'));
            fprintf(fid,strcat('"input shaft diameter"=',num2str(
                obj.shaft_diameter),'\n'));    % Approximately
            fprintf(fid,strcat('"key width"=',num2str(obj.key_width),'\n'));
                                % Approximately
            fprintf(fid,strcat('"key height"=',num2str(obj.key_height),'\n'));
                               % Approximately
            fclose(fid);
        end

    end
end
```

C.30 Motors

```
classdef Motor
    properties
        filePath;
        name;                       % 'Hip' or 'Knee'
```

```
torque_nominal;             % [mNm]
power_nominal;              % [W]
speed_nominal;              % [RPM]
current_nominal;            % [A]
resistance;                 % [Ohms]
torque_const;               % [mNm/A]
speed_const;                % [RPM/V]
mass;                       % [kg]
inertia;                    % [gcm^2]
diameter_outer;             % [mm]
thickness;                  % [mm] (exluding shaft diameter)
mounting_diameter;          % [mm] (always 3)
bolt_diameter;              % [mm]
bolt_depth;                 % [mm]
efficiency_max;             % [frac]
shaft_diameter;             % [mm]
shaft_length;               % [mm]
key_width;                  % [mm]
key_height;                 % [mm]
key_length;                 % [mm]
% Polyfit functions from generate_motor_functions.m
T = [1  -4.47388539068991e-14;
    0.371251182445849   -3.96043280431109;
    -0.890576993210452  3157.56845966845;
    0.00436690058974527 1.12390912813721;
    -0.00636435419784517    7.19743966864434;
    0.146918265140937   19.7946858605065;
    -0.0490022633920436 113.413595477954;
    1.74501207080679e-18    3;
    0.0115249147411334  33.1160654320377;
    0.682852347817259   47.2171719217119;
    3.71306346334785    -278.889925136164;
    0.0474433809398074  35.5646211593142;
    0.00930449005014656 27.1840445766206;
    0.0382290835378379  21.3960316237073;
    0.0020249467122444  2.73644608019762;
    0.00142808192465724 4.25529396274844;
    0.000179149339976652    0.680570335542018];
fasteners;  % fastener object for mounting
adapter_thickness;          % [mm] (stored here since it depends on
   the thickness of the bolt head)
```

```
end
methods

    % Constructor
    function obj = Motor(torque_max,HD,motor_name)
        %% INPUTS
        % torque_max    = max torque seen by motor [mNm]
        % HD            = paired harmonic drive
        obj = obj.setParameters(torque_max,HD);
        obj.name = motor_name;
        obj.filePath = strcat('..\Solidworks\Equations\Motor',motor_name,'
            .txt');
    end

    % Properties and parameters calculation
    function obj = setParameters(obj,torque_max,HD)
        obj.torque_nominal = obj.T(1,1)*torque_max + obj.T(1,2);
        obj.power_nominal = obj.T(2,1)*torque_max + obj.T(2,2);
        obj.speed_nominal = obj.T(3,1)*torque_max + obj.T(3,2);
        obj.current_nominal = obj.T(4,1)*torque_max + obj.T(4,2);
        obj.resistance = obj.T(5,1)*torque_max + obj.T(5,2);
        obj.torque_const = obj.T(6,1)*torque_max + obj.T(6,2);
        obj.speed_const = obj.T(7,1)*torque_max + obj.T(7,2);
        obj.mass = obj.T(10,1)*torque_max + obj.T(10,2);
        obj.inertia = obj.T(11,1)*torque_max + obj.T(11,2);
        obj.diameter_outer = obj.T(12,1)*torque_max + obj.T(12,2);
        obj.thickness = obj.T(13,1)*torque_max + obj.T(13,2);
        obj.mounting_diameter = obj.T(14,1)*torque_max + obj.T(14,2);
        obj.bolt_diameter = obj.T(15,1)*torque_max + obj.T(15,2);
        obj.bolt_depth = obj.T(16,1)*torque_max + obj.T(16,2);
        obj.efficiency_max = obj.T(16,1)*torque_max + obj.T(16,2);
        obj.shaft_diameter = HD.shaft_diameter;
        obj.key_width = HD.key_width;
        obj.key_height = HD.key_height;
        obj.key_length = HD.key_length;
        %% Tweak values
        % Make sure Watts is positive (round up to 3W if
        % necessary)
        if obj.power_nominal < 3
            obj.power_nominal = 3;
        end
```

```
        % Make sure RPM doesn't round down to 0
        if obj.speed_nominal < 1500
            obj.speed_nominal = 1500;
        end
        % Terminal Resistance
        if obj.resistance < 0.1
            obj.resistance = 0.1;
        end
        % Speed Constant is defined by Maxon as the following:
        obj.speed_const = 30000/(pi*(obj.torque_const));
        % Mass: g -> kg
        obj.mass = obj.mass/1000;
        % Rotor inertia
        if obj.inertia < 3
            obj.inertia = 3;
        end
        % Round screw diameter to closest integer
        obj.bolt_diameter = ceil(obj.bolt_diameter);
        % Make sure maximum efficiency is below 0.9
        if obj.efficiency_max > 0.9
            obj.efficiency_max = 0.9;
        end
        obj.fasteners = Fastener(obj.bolt_diameter);
        % Determined experimentally
        obj.adapter_thickness = max(obj.fasteners.thickness_head*3,
           HD.spline_thickness);
        obj.shaft_length = obj.adapter_thickness + HD.spline_thickness;

        % Cheating: At a certain point, the motor outgrows the harmonic
           drive and their bolts intefere
        % in order to solve this, we will artificially constrain the motor
            diameter to that of the harmonic drive
        % (in reality, would need to use a different motor or additional
           reducer)
        if obj.diameter_outer >  HD.spline_outer_diameter
            obj.diameter_outer = HD.spline_outer_diameter;
            obj.mounting_diameter = obj.diameter_outer/2;    % Lazy way to
               guesstimate

        end
    end
```

```
% Analysis Report, Section 3.5.2 Motor Power Consumption
function current = getPower(obj,torque,rpm)
    %% getPower
    % INPUTS:
    % torque = torque at output of motor [mNm]
    % rpm = speed at output of motor [rpm]
    % OUTPUTS:
    % current into motor [A]
    % Equation 18
    U = (1./obj.speed_const).*(((30000.*obj.resistance.*torque)./(pi.*
        obj.torque_const.^2))+rpm);
    a = obj.resistance;
    b = -U;
    c = (pi.*rpm.*torque)./(30000);
    I_upper = (-b+sqrt((b.^2)-(4.*a.*c)))./(2.*a);
    I_lower = (-b-sqrt((b.^2)-(4.*a.*c)))./(2.*a);
    % When speed is 0, I_lower = 0
    if I_lower <= 0
        current = I_upper;
    else
        current = I_lower;
    end
end

% Print function to TXT file
function printTXT(obj)
    fid = fopen(obj.filePath,'wt');
    fprintf(fid,strcat('"outer diameter"=',num2str(obj.diameter_outer)
        ,'\n'));
    fprintf(fid,strcat('"motor thickness"=',num2str(obj.thickness),'\n
        '));
    fprintf(fid,strcat('"mounting diameter"=',num2str(
        obj.mounting_diameter),'\n'));
    fprintf(fid,strcat('"screw diameter"=',num2str(obj.bolt_diameter),
        '\n'));
    fprintf(fid,strcat('"screw depth"=',num2str(obj.bolt_depth),'\n'))
        ;
    fprintf(fid,strcat('"shaft length"=',num2str(obj.shaft_length),'\n
        '));
    fprintf(fid,strcat('"shaft diameter"=',num2str(obj.shaft_diameter)
```

```
            ,'\n'));  % Approximately
        fprintf(fid,strcat('"key width"=',num2str(obj.key_width),'\n'));
                    % Approximately
        fprintf(fid,strcat('"key height"=',num2str(obj.key_height),'\n'));
                  % Approximately
        fprintf(fid,strcat('"key length"=',num2str(obj.key_length),'\n'));
                  % spline thickness
        fclose(fid);
    end

  end
end
```

C.31 Power Consumption

```
% Inputs
% Leg_Object            = object containing r1, R, etc.
% HD_hip, HD_knee       = objects of Harmonic Drives
% motor_hip,motor_knee  = objects of motors
% theta                 = [rad] range of theta over walking cycle
% phi                   = [rad] range of phi over walking cycle
% thetad,phid           = [rad/s] range of angular velocities over walking
   cycle
% torque_theta          = [Nm] range of torques at hip over walking cycle
% torque_phi            = [Nm] range of torques at knee over walking cycle

% Outputs
% current_consumed      = [A] current consumed on average over walking cycle
% walking_speed         = [mm] walking speed of robot

function [current_consumed,walking_speed] = Power_Consumption(Leg_Object,
   HD_hip,HD_knee,motor_hip,motor_knee,theta,phi,thetad,phid,torque_theta,
   torque_phi)
    % Calculated the power the robot consumed over a cycle
    % Similar to Section 3.5.3 of Analysis Report, with theta_max = 28degrees

    r1 = Leg_Object.r1;
    x_walking_min = Leg_Object.x_walking_min;
    x_walking_max = Leg_Object.x_walking_max;
    thetad_mean = mean(abs(thetad));
```

```
phid_mean = mean(abs(phid));
ed = r1*0.5;            % velocity in x is multiple of leg length
dd =0;
ddd=0;
edd=0;
steps = 1000;
e = linspace(x_walking_min,x_walking_max,steps);
gear_ratio=100;
% Time required per phase
phase_time = [(x_walking_max-x_walking_min)/ed,(max(phi)-min(phi))/(mean(
    abs(phid))*180/pi),(max(theta)-min(theta))/(mean(abs(thetad))*180/pi)
    ];
step_time = phase_time./steps;
walking_speed = (x_walking_max-x_walking_min)/sum(phase_time);
% Total power consumption
coulombs_consumed = 0;

%--------------------------%
%% POWER CONSUMPTION
%--------------------------%
% Phase 1: leg pulls body forward
[torque_motor_theta,rpm_motor_theta] = HD_hip.getInputs(abs(torque_theta
    (1:1000)),abs(thetad(1:1000)));
[torque_motor_phi,rpm_motor_phi] = HD_knee.getInputs(abs(torque_phi
    (1:1000)),abs(phid(1:1000)));
coulombstheta = motor_hip.getPower(torque_motor_theta,rpm_motor_theta);
coulombsphi = motor_knee.getPower(torque_motor_phi,rpm_motor_phi);
coulombs_consumed = coulombs_consumed + sum(coulombstheta+coulombsphi)*
    step_time(1);

% Phase 2: phi increases
[torque_motor_theta,rpm_motor_theta] =  HD_hip.getInputs(abs(torque_theta
    (1001:2000)),0);
[torque_motor_phi,rpm_motor_phi] =  HD_knee.getInputs(abs(torque_phi
    (1001:2000)),abs(phid_mean));
coulombstheta = motor_hip.getPower(torque_motor_theta,rpm_motor_theta);
coulombsphi = motor_knee.getPower(torque_motor_phi,rpm_motor_phi);
coulombs_consumed = coulombs_consumed + sum(coulombstheta+coulombsphi)*
    step_time(2);

% Phase 3: theta decreases
```

```
    [torque_motor_theta,rpm_motor_theta] =  HD_hip.getInputs(abs(torque_theta
        (2001:3000)),abs(thetad_mean));
    [torque_motor_phi,rpm_motor_phi] =  HD_knee.getInputs(abs(torque_phi
        (2001:3000)),0);
    coulombstheta = motor_hip.getPower(torque_motor_theta,rpm_motor_theta);
    coulombsphi = motor_knee.getPower(torque_motor_phi,rpm_motor_phi);
    coulombs_consumed = coulombs_consumed + sum(coulombstheta+coulombsphi)*
        step_time(3);

    current_consumed = (sum(coulombs_consumed)*5)/(phase_time(1) + phase_time
        (2)*5 + phase_time(3)*5);

    % Approximately for NVIDIA Jetson TX2, Raspberry Pi, sensors, etc.
    % Wattage is given in Table 1 of Literature Review
    % Pi runs at 5V (1A for 5W) and Jetson runs at 7.5V (1A for 7.5W). Other
        electronics approximately 0.5A together
    amps_other_electronics = 2.5;
    current_consumed = current_consumed + amps_other_electronics;
end
```

C.32 Battery

```
classdef Battery
    properties
        filePath = '..\Solidworks\Equations\Battery.txt';
        length;                 % [mm]
        height;                 % [mm]
        width;                  % [mm]
        mass;                   % [kg]
        voltage_cell = 3.6;     % [V]
        Ah_cell = 3.4;          % [Ah]
        mass_cell = 0.05;       % [kg]
        height_cell = 65.08;    % [mm]
        diameter_cell = 18.63;  % [mm]
        cells_parallel;
        cells_series;
        cells_total;
        cells_per_battery;
        cells_per_layer;
        running_hours_per_day;  % [h]
```

```
    case_width;             % [mm]
    case_length;            % [mm]

end
methods

    % Constructor
    function obj = Battery(days,active_hours,power_solar,current_consumed)
        obj = obj.setParameters(days,active_hours,power_solar,
            current_consumed);
    end

    % Properties and parameters calculation
    % From Analysis Report, section 3.5 Power Consumption
    function obj = setParameters(obj,days,active_hours,power_solar,
        current_consumed)
        % motors run at 48 volts
        voltage = 48;
        % Equation 19
        power_consumed = voltage*current_consumed;
        % Equation 21
        obj.running_hours_per_day = power_solar*active_hours/
            power_consumed;
        % Number of hours of work the battery can hold
        hours_in_battery = obj.running_hours_per_day*days;
        % Equations 22, 23, 24
        obj.cells_series = ceil(voltage/obj.voltage_cell);
        obj.cells_parallel = ceil(current_consumed*hours_in_battery/
            obj.Ah_cell);
        obj.cells_total = obj.cells_parallel*obj.cells_series;
        obj.mass = obj.cells_total*obj.mass_cell;

        % Dimensions of battery packs (geometric, not based on reports)
        % 2 batteries, one on either side of robot
        obj.cells_per_battery = round(obj.cells_total/2,1);
        % 1 layers of cells
        obj.height = obj.height_cell;
        cells_wide = round(sqrt(obj.cells_per_battery));
        obj.width = cells_wide*obj.diameter_cell;
        obj.length = obj.width;
        % dimensions of plates that will sandwich the batteries
```

```
            obj.case_width = obj.width + obj.diameter_cell;
            obj.case_length = obj.length + obj.diameter_cell;
        end

        % Print function to TXT file
        function printTXT(obj)
            fid = fopen(obj.filePath,'wt');
            fprintf(fid,strcat('"height"=',num2str(obj.height),'\n'));
            fprintf(fid,strcat('"width"=',num2str(obj.width),'\n'));
            fprintf(fid,strcat('"length"=',num2str(obj.length),'\n'));
            fprintf(fid,strcat('"diameter of cell"=',num2str(obj.diameter_cell
                ),'\n'));
            fprintf(fid,strcat('"height of cell"=',num2str(obj.height_cell),'\
                n'));
            fprintf(fid,strcat('"case width"=',num2str(obj.case_width),'\n'));
            fprintf(fid,strcat('"case_length"=',num2str(obj.case_length),'\n')
                );
            fclose(fid);
        end
    end
end
```

C.33 Adapter Knee

```
classdef Adapter_Knee
    properties
        filePath = '..\Solidworks\Equations\AdapterKnee.txt';
        tube_diameter;          % [mm] diameter of tube surrounding HD
        mounting_diameter;      % [mm] mounting diameter for adapter on hip
            plate
        outer_diameter;         % [mm] outer diameter of adapter
        bolt_angle;             % [deg] angle between horizontal and
            supporting bolts (above and below parallel)
        bolt_diameter;          % [mm] diameter of bolts connecting to hip
            plates
        hip_plate_to_HD;        % [mm] distance from hip plate to HD
            flexspline face
        hip_plate_height;       % [mm] height of hip plate
        height_outer_diameter;  % [mm] height of mounting section of adapter
        density = 0.0000027;    % [kg/mm^3] approximately: https://
```

```
        www.engineeringclicks.com/6061-t6-aluminum/
    mass;                   % [kg] approximately
end

methods

    % Constructor
    function obj = Adapter_Knee(HD_knee,motor_knee,hip_plate_to_HD,
        max_diameter_pulley)
        obj = obj.setParameters(HD_knee,motor_knee,hip_plate_to_HD,
            max_diameter_pulley);
    end

    % Properties and parameters calculation
    % Purely geometric, not in reports
    function obj = setParameters(obj,HD_knee,motor_knee,hip_plate_to_HD,
        max_diameter_pulley)
        obj.hip_plate_height = max_diameter_pulley +4;
        obj.tube_diameter = HD_knee.spline_outer_diameter + (
            HD_knee.spline_thickness*2);
        obj.mounting_diameter = obj.tube_diameter + ((
            HD_knee.spline_outer_diameter-HD_knee.spline_mounting_diameter
            ));
        obj.outer_diameter = obj.mounting_diameter + ((
            HD_knee.spline_outer_diameter-HD_knee.spline_mounting_diameter
            ));
        obj.height_outer_diameter = min(obj.hip_plate_height,
            obj.tube_diameter);
        obj.bolt_angle = asind((obj.height_outer_diameter/2)/(
            obj.outer_diameter/2))/2;
        obj.bolt_diameter = HD_knee.spline_bolt_diameter;
        obj.hip_plate_to_HD = hip_plate_to_HD;
        % approximately
        obj.mass = obj.density * ((motor_knee.adapter_thickness*pi*(
            obj.outer_diameter/2)^2) + ((((obj.outer_diameter-
            HD_knee.spline_outer_diameter)/2)^2)*pi*obj.hip_plate_to_HD));
    end

    % Print function to TXT file
    function printTXT(obj,HD_knee,motor_knee)
        fid = fopen(obj.filePath,'wt');
```

```
        fprintf(fid,strcat('"adapter thickness"=',num2str(
            motor_knee.adapter_thickness),'\n'));
        fprintf(fid,strcat('"spline outer diameter"=',num2str(
            HD_knee.spline_outer_diameter),'\n'));
        fprintf(fid,strcat('"HD thickness"=',num2str(HD_knee.thickness),'\
            n'));
        fprintf(fid,strcat('"spline thickness"=',num2str(
            HD_knee.spline_thickness),'\n'));
        fprintf(fid,strcat('"spline mounting diameter"=',num2str(
            HD_knee.spline_mounting_diameter),'\n'));
        fprintf(fid,strcat('"spline bolt quantity"=',num2str(
            HD_knee.spline_num_bolts),'\n'));
        fprintf(fid,strcat('"spline bolts diameter"=',num2str(
            HD_knee.spline_bolt_diameter),'\n'));
        fprintf(fid,strcat('"motor shaft diameter"=',num2str(
            HD_knee.shaft_diameter),'\n'));
        fprintf(fid,strcat('"motor mounting diameter"=',num2str(
            motor_knee.mounting_diameter),'\n'));
        fprintf(fid,strcat('"motor bolt quantity"=',num2str(3),'\n'));
                    % always 3 bolts
        fprintf(fid,strcat('"motor bolt diameter"=',num2str(
            motor_knee.bolt_diameter),'\n'));
        fprintf(fid,strcat('"from hd output to hip plate"=',num2str(
            obj.hip_plate_to_HD),'\n'));
        fprintf(fid,strcat('"hip plate height"=',num2str(
            obj.hip_plate_height),'\n'));
        fprintf(fid,strcat('"adapter tube diameter"=',num2str(
            obj.tube_diameter),'\n'));
        fprintf(fid,strcat('"adapter mounting diameter"=',num2str(
            obj.mounting_diameter),'\n'));
        fprintf(fid,strcat('"adapter outer diameter"=',num2str(
            obj.outer_diameter),'\n'));
        fprintf(fid,strcat('"adapter secondary bolt angle"=',num2str(
            obj.bolt_angle),'\n'));
        fprintf(fid,strcat('"height of outer diameter"=',num2str(
            obj.height_outer_diameter),'\n'));
        %fprintf(fid,strcat('""=',num2str(),'\n'));
        fclose(fid);
    end

end
```

```
end
```

C.34 Hip Bracket

```
classdef Hip_Bracket
    properties
        filePath = '..\Solidworks\Equations\HipBracket.txt';
        thickness;                      % [mm]
        height;                         % [mm] from center of shaft to base
        bracket_width;                  % [mm] equal to outer diameter of HD
        bolt_diameter;                  % [mm]
        bolt_ell;                       % [mm] from bolt edge to bracket edge
        bushing_diameter;               % [mm]
        spline_thickness;               % [mm] harmonic drive spline thickness
        distance_between_drives;        % [mm] from edge of hip HD to edge of
            knee HD adapter
        distance_between_shafts;        % [mm] from center of knee shaft to
            center of hip shaft
        density = 0.0000027;            % [kg/mm^3] approximately: https://
            www.engineeringclicks.com/6061-t6-aluminum/
        mass;                           % [kg] approximate mass
    end

    methods
        % Hip Bracket Constructor
        function obj = Hip_Bracket(HD_hip,adapter_knee,leg,shaft_length,
            bushing_diameter,pulley_diameter,Fx,Fy,Fz)
             obj = obj.setParameters(HD_hip,adapter_knee,leg,shaft_length,
                bushing_diameter,pulley_diameter,Fx,Fy,Fz);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,HD_hip,adapter_knee,leg,shaft_length,
            bushing_diameter,pulley_diameter,Fx,Fy,Fz)
             %% INPUTS
             % HD_hip            = hip harmonic drive object
             % adapter_knee      = knee adapter object
             % shaft_length      = length of shaft between hip brackets [mm]
             % bushing_diameter  = outer diameter of bushing [mm]
             % pulley_diameter   = outer diameter of pulley [mm]
```

```
% Fx, Fy, Fz        = forces along fastener x (friction), y (
    friction) and z axes (normal) [N]

% maximum hip angle
theta_max = deg2rad(28);
% alpha = distance between edge of knee adapter and hip bracket (
    so they don't interfere when at 28 degrees)
alpha = ceil((adapter_knee.outer_diameter/2)*((1/cos(theta_max))
    -1));
beta = alpha;
obj.distance_between_drives = alpha;
obj.distance_between_shafts = HD_hip.spline_outer_diameter/2 +
    adapter_knee.outer_diameter/2 + alpha;
plate_L_end = (adapter_knee.outer_diameter/2 + 5);
plate_height = pulley_diameter+4;
% bracket needs to be tall enough so that hip plates and knee
    adapters don't touch chassis
% three cases: either hip plate is taller than adapter_knee, hip
    plate is less tall but
% is closer to ground at 28degrees, and hip plate is smaller and
    adapter_knee closer to
% ground at 28degrees
if (plate_height > adapter_knee.outer_diameter)
    max_diameter = plate_height;
    obj.height = obj.distance_between_shafts*sin(theta_max) +
        max_diameter/2 + (plate_L_end*sin(theta_max)) - ((
        max_diameter/2)*(1-cos(theta_max)));
else
    if (((plate_L_end)*sin(theta_max)+(plate_height/2)*cos(
        theta_max)) > (adapter_knee.tube_diameter/2))
        % if vertical projection of hip plate is higher than
            radius of knee adapter
        obj.height = obj.distance_between_shafts*sin(theta_max) +
            ((plate_L_end)*sin(theta_max)+(plate_height/2)*cos(
            theta_max));
    else
        % if radius of knee adapter is higher than vertical
            projection of hip plate
        obj.height = obj.distance_between_shafts*sin(theta_max) +
            adapter_knee.tube_diameter/2;
    end
```

```
            end
            obj.bracket_width = HD_hip.spline_outer_diameter;
            % Same thickness as the bushings
            obj.thickness = 10;

            % Get dimensions of bolts that fasten to chassis
            [d_bolt,ell] = fasteners_hip(leg.r1,leg.R,obj.thickness,obj.height
                ,obj.bracket_width,shaft_length,Fx,Fy,Fz);
            obj.bolt_diameter = d_bolt;
            obj.bolt_ell = ell;

            obj.bushing_diameter = bushing_diameter;
            obj.spline_thickness = HD_hip.spline_thickness;

            % mass of mounting and adapter section + mass of section that
                attaches to hip base (approximately)
            obj.mass = obj.density*((obj.height+HD_hip.spline_outer_diameter
                /2) + (obj.bolt_diameter+obj.bolt_ell*2))*
                HD_hip.spline_outer_diameter*obj.thickness;
        end

        % Print function to TXT file
        function printTXT(obj)
            fid = fopen(obj.filePath,'wt');
            fprintf(fid,strcat('"bracket height"=',num2str(obj.height),'\n'));
            fprintf(fid,strcat('"bracket width"=',num2str(obj.bracket_width),'
                \n'));
            fprintf(fid,strcat('"bracket thickness"=',num2str(obj.thickness),'
                \n'));
            fprintf(fid,strcat('"bushing diameter"=',num2str(
                obj.bushing_diameter),'\n'));
            fprintf(fid,strcat('"ell"=',num2str(obj.bolt_ell),'\n'));
            fprintf(fid,strcat('"hip bolt diameter"=',num2str(
                obj.bolt_diameter),'\n'));
            fprintf(fid,strcat('"adapter thickness"=',num2str(
                obj.spline_thickness),'\n'));
            fclose(fid);
        end
    end
end
```

C.35 Adapter Hip

```
classdef Adapter_Hip
    properties
        filePath = '..\Solidworks\Equations\AdapterHip.txt';
        width;                  % [mm] equal to outer diameter of HD
        thickness;              % [mm] thickness between HD and motor
        mount_thickness;        % [mm] thickness of section that connects
           adapter to base
        height;                 % [mm] height from base to center of motor
           shaft
        bracket_bolt_diameter;  % [mm] bolt diameter of brackets holding up
           leg
        bracket_ell;            % [mm] distance from bolt edge to plate edge
           of brackets holding up leg
        density = 0.0000027;    % [kg/mm^3] approximately: https://
           www.engineeringclicks.com/6061-t6-aluminum/
        mass;                   % [kg] approximately
    end

    methods

        % Constructor
        function obj = Adapter_Hip(HD_hip,motor_hip,hip_bracket)
            obj = obj.setParameters(HD_hip,motor_hip,hip_bracket);
        end

        % Properties and parameters calculation
        % Purely geometric, not in reports
        function obj = setParameters(obj,HD_hip,motor_hip,hip_bracket)
            obj.width = HD_hip.spline_outer_diameter;
            obj.mount_thickness = HD_hip.spline_thickness;
            obj.thickness = motor_hip.adapter_thickness;
            obj.height = hip_bracket.height;
            obj.bracket_bolt_diameter = hip_bracket.bolt_diameter;
            obj.bracket_ell = hip_bracket.bolt_ell;
            % approximately
            obj.mass = obj.density*((obj.height+(HD_hip.spline_outer_diameter
               /2)) + (obj.bracket_bolt_diameter+(obj.bracket_ell)))*
               obj.thickness*HD_hip.spline_outer_diameter;
```

```
        end

        % Print function to TXT file
        function printTXT(obj,HD_hip,motor_hip)
            fid = fopen(obj.filePath,'wt');
            fprintf(fid,strcat('"bracket height"=',num2str(obj.height),'\n'));
            fprintf(fid,strcat('"bracket width"=',num2str(obj.width),'\n'));
            fprintf(fid,strcat('"bracket thickness"=',num2str(
                obj.mount_thickness),'\n'));
            fprintf(fid,strcat('"adapter thickness"=',num2str(obj.thickness),'
                \n'));
            fprintf(fid,strcat('"spline mounting diameter"=',num2str(
                HD_hip.spline_mounting_diameter),'\n'));
            fprintf(fid,strcat('"spline bolt quantity"=',num2str(
                HD_hip.spline_num_bolts),'\n'));
            fprintf(fid,strcat('"spline bolts diameter"=',num2str(
                HD_hip.spline_bolt_diameter),'\n'));
            fprintf(fid,strcat('"motor shaft diameter"=',num2str(
                motor_hip.shaft_diameter),'\n'));
            fprintf(fid,strcat('"motor mounting diameter"=',num2str(
                motor_hip.mounting_diameter),'\n'));
            fprintf(fid,strcat('"motor bolt quantity"=',num2str(3),'\n'));
                        % always 3 bolts
            fprintf(fid,strcat('"motor bolt diameter"=',num2str(
                motor_hip.bolt_diameter),'\n'));
            fprintf(fid,strcat('"bracket bolt diameter"=',num2str(
                obj.bracket_bolt_diameter),'\n'));
            fprintf(fid,strcat('"bracket bolt ell"=',num2str(obj.bracket_ell),
                '\n'));
            %fprintf(fid,strcat('""=',num2str(),'\n'));
            fclose(fid);
        end

    end
end
```

C.36 Hip Base

```
classdef Hip_Base
    properties
```

```
    filePath = '..\Solidworks\Equations\HipBase.txt';
    width;                              % [mm] width of base
    thickness;                          % [mm] thickness of base
    distance_between_hip_bracket_bolts; % [mm]
    hip_bracket_bolt_to_adapter_bolt;   % [mm]
    bolt_diameter;                      % [mm] bolt diameter
    bolt_ell;                           % [mm] edge distance of bolt
    hip_plate_to_HD;                    % [mm] distance from hip plate to
       HD output
    L_shaft;                            % [mm] distance between hip
       brackets
    bracket_thickness;                  % [mm] thickness of hip bracket
    density = 0.0000027;                % [g/mm^2] approximately: https://
       www.engineeringclicks.com/6061-t6-aluminum/
    mass;                               % [kg] approximately
end

methods

    % Hip Base Constructor
    function obj = Hip_Base(HD_hip,hip_bracket,L_shaft,hip_plate_to_HD)
        obj = obj.setParameters(HD_hip,hip_bracket,L_shaft,hip_plate_to_HD
           );
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,HD_hip,hip_bracket,L_shaft,
       hip_plate_to_HD)
        obj.width = HD_hip.spline_outer_diameter;
        obj.thickness = HD_hip.spline_thickness;
        obj.bolt_diameter = hip_bracket.bolt_diameter;
        obj.bolt_ell = hip_bracket.bolt_ell;
        obj.hip_plate_to_HD = hip_plate_to_HD;
        obj.L_shaft = L_shaft;
        obj.bracket_thickness = hip_bracket.thickness;
        obj.distance_between_hip_bracket_bolts = L_shaft - (
           obj.bolt_diameter+(obj.bolt_ell*2));
        obj.hip_bracket_bolt_to_adapter_bolt = (obj.bolt_diameter/2) +
           obj.bolt_ell + hip_bracket.thickness + hip_plate_to_HD +
           HD_hip.thickness - (obj.bolt_ell + (obj.bolt_diameter/2));
        obj.mass = obj.density*(obj.thickness*(
```

```
            obj.distance_between_hip_bracket_bolts+
            obj.hip_bracket_bolt_to_adapter_bolt+obj.bolt_diameter+(
            obj.bolt_ell*2)+(obj.bracket_thickness*2)))*obj.thickness;
        end

        % Print function to TXT file
        function printTXT(obj,HD_hip)
            fid = fopen(obj.filePath,'wt');
            fprintf(fid,strcat('"shaft length"=',num2str(obj.L_shaft),'\n'));
            fprintf(fid,strcat('"thickness hip plate"=',num2str(
                obj.bracket_thickness),'\n'));
            fprintf(fid,strcat('"hip plate to HD"=',num2str(
                obj.hip_plate_to_HD),'\n'));
            fprintf(fid,strcat('"bracket bolt diameter"=',num2str(
                obj.bolt_diameter),'\n'));
            fprintf(fid,strcat('"bracket ell"=',num2str(obj.bolt_ell),'\n'));
            fprintf(fid,strcat('"adapter width"=',num2str(obj.width),'\n'));
            fprintf(fid,strcat('"adapter thickness"=',num2str(obj.thickness),'
                \n'));
            fprintf(fid,strcat('"HD thickness"=',num2str(HD_hip.thickness),'\n
                '));
            fprintf(fid,strcat('"HD spline thickness"=',num2str(
                HD_hip.spline_thickness),'\n'));
            fprintf(fid,strcat('"from hip bracket bolt to hip bracket bolt"=',
                num2str(obj.distance_between_hip_bracket_bolts),'\n'));
            fprintf(fid,strcat('"from bracket bolt to adapter bolt"=',num2str(
                obj.hip_bracket_bolt_to_adapter_bolt),'\n'));
            %fprintf(fid,strcat('""=',num2str(),'\n'));
            fclose(fid);
        end
    end
end
```

C.37 Plates Hip 1

```
classdef Plates_Hip_1
    properties
        d_knee                  % [mm] Small diameter of the knee shaft
        d_hip                   % [mm] Large diameter of the hip shaft
        d_hipknee               % [mm] Small diameter of the hip knee shaft
```

```
    height                  % [mm] Height of the hip plate
    curve                   % [mm] Fillet of the plates
    r1                      % [mm] Length of the thigh
    L_hipknee               % [mm] Distance between hip and hip knee
       shafts
    L_end                   % [mm] Length of the end of the plates
    w_k                     % [mm] Width of key
    h_k                     % [mm] Height of key
    r_springHip             % [mm] Radius spring mounting for hip
    Rnom_springHip          % [mm] Radius of spring for hip
    dHolder_springHip       % [mm] Size of spring holder for hip
    d_springHip             % [mm] Spring wire diameter for hip
    r_springKnee            % [mm] Radius spring mounting for knee
    Rnom_springKnee         % [mm] Radius of spring for knee
    dHolder_springKnee      % [mm] Size of spring holder for knee
    d_springKnee            % [mm] Spring wire diameter for knee
    HipSpacing              % [mm] Distance between hip plates plus
       alignment pin penetration distance
    d_boltsAdapter          % [mm] Size of bolt for knee adapter
    L_boltsAdapter          % [mm] Distance of bolt from center of shaft
    angle                   % [mm] Angle separating bolts
end
methods

    % Plates Hip 1 Constructor
    function obj = Plates_Hip_1(d_knee_shaft,D_hip_shaft,shaft_hipKnee,
       max_diameter_pulley,leg_r1,distance_between_shafts,adapter_Knee,
       key_Hip_Plates,spring_Torsion_Hip,spring_Torsion_Knee)
        obj = obj.setParameters(d_knee_shaft,D_hip_shaft,shaft_hipKnee,
           max_diameter_pulley,leg_r1,distance_between_shafts,
           adapter_Knee,key_Hip_Plates,spring_Torsion_Hip,
           spring_Torsion_Knee );
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,d_knee_shaft,D_hip_shaft,
       shaft_hipKnee,max_diameter_pulley,leg_r1,distance_between_shafts,
       adapter_Knee,key_Hip_Plates,spring_Torsion_Hip,spring_Torsion_Knee
       )
        obj.d_knee = d_knee_shaft + 4;
        obj.d_hip = D_hip_shaft;
```

```
    obj.d_hipknee = shaft_hipKnee.d_small +4;
    obj.height = max_diameter_pulley +4;
    obj.curve = obj.height /2;
    obj.r1 = leg_r1;
    obj.L_hipknee = distance_between_shafts; %from hip bracket
    obj.L_end = adapter_Knee.outer_diameter/2+5;
    obj.w_k = key_Hip_Plates.w;
    obj.h_k = key_Hip_Plates.h/2;
    obj.r_springHip = spring_Torsion_Hip.r;
    obj.Rnom_springHip =  spring_Torsion_Hip.bigR;
    obj.d_springHip = spring_Torsion_Hip.smalld;
    obj.dHolder_springHip = obj.d_springHip * 4;
    obj.r_springKnee = spring_Torsion_Knee.r;
    obj.Rnom_springKnee = spring_Torsion_Knee.bigR;
    obj.d_springKnee = spring_Torsion_Knee.smalld;
    obj.dHolder_springKnee = obj.d_springKnee *4;
    obj.HipSpacing = shaft_hipKnee.L_mid + 4 + 4; %Bearing flange +
       Alignment Pin
    obj.d_boltsAdapter = adapter_Knee.bolt_diameter;
    obj.L_boltsAdapter = adapter_Knee.mounting_diameter/2;
    obj.angle = adapter_Knee.bolt_angle;
end

% Print function to TXT file
function printTXT(obj)
    filePath = '..\Solidworks\Equations\HipPlate1.txt';
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"d_knee"= ',num2str(obj.d_knee),'\n'));
    fprintf(fid,strcat('"d_hip"= ',num2str(obj.d_hip),'\n'));
    fprintf(fid,strcat('"d_hipknee"= ',num2str(obj.d_hipknee),'\n'));
    fprintf(fid,strcat('"height"= ',num2str(obj.height),'\n'));
    fprintf(fid,strcat('"curve"= ',num2str(obj.curve),'\n'));
    fprintf(fid,strcat('"r1"= ',num2str(obj.r1),'\n'));
    fprintf(fid,strcat('"L_hipknee"= ',num2str(obj.L_hipknee),'\n'));
    fprintf(fid,strcat('"L_end"= ',num2str(obj.L_end),'\n'));
    fprintf(fid,strcat('"w_k"= ',num2str(obj.w_k),'\n'));
    fprintf(fid,strcat('"h_k"= ',num2str(obj.h_k),'\n'));
    fprintf(fid,strcat('"r_springHip"= ',num2str(obj.r_springHip),'\n'
       ));
    fprintf(fid,strcat('"Rnom_springHip"= ',num2str(obj.Rnom_springHip
       ),'\n'));
```

```
            fprintf(fid,strcat('"dHolder_springHip"= ',num2str(
                obj.dHolder_springHip),'\n'));
            fprintf(fid,strcat('"d_springHip"= ',num2str(obj.d_springHip),'\n'
                ));
            fprintf(fid,strcat('"r_springKnee"= ',num2str(obj.r_springKnee),'\
                n'));
            fprintf(fid,strcat('"Rnom_springKnee"= ',num2str(
                obj.Rnom_springKnee),'\n'));
            fprintf(fid,strcat('"dHolder_springKnee"= ',num2str(
                obj.dHolder_springKnee),'\n'));
            fprintf(fid,strcat('"d_springKnee"= ',num2str(obj.d_springKnee),'\
                n'));
            fprintf(fid,strcat('"HipSpacing"= ',num2str(obj.HipSpacing),'\n'))
                ;
            fprintf(fid,strcat('"d_boltsAdapter"= ',num2str(obj.d_boltsAdapter
                ),'\n'));
            fprintf(fid,strcat('"L_boltsAdapter"= ',num2str(obj.L_boltsAdapter
                ),'\n'));
            fprintf(fid,strcat('"angle"= ',num2str(obj.angle),'\n'));
            fclose(fid);
        end

    end
end
```

C.38 Plates Hip 2

```
classdef Plates_Hip_2
    properties
        d_knee                  % [mm] Small diameter of the knee shaft
        d_hip                   % [mm] Large diameter of the hip shaft
        d_hipknee               % [mm] Small diameter of the hip knee shaft
        height                  % [mm] Height of the hip plate
        curve                   % [mm] Fillet of the plates
        r1                      % [mm] Length of the thigh
        L_hipknee               % [mm] Distance between hip and hip knee
            shafts
        L_end                   % [mm] Length of the end of the plates
        w_k                     % [mm] Width of key
        h_k                     % [mm] Height of key
```

```
    r_springHip             % [mm] Radius spring mounting for hip
    Rnom_springHip          % [mm] Radius of spring for hip
    dHolder_springHip       % [mm] Size of spring holder for hip
    d_springHip             % [mm] Spring wire diameter for hip
    r_springKnee            % [mm] Radius spring mounting for knee
    Rnom_springKnee         % [mm] Radius of spring for knee
    dHolder_springKnee      % [mm] Size of spring holder for knee
    d_springKnee            % [mm] Spring wire diameter for knee
end
methods

    % Plates Hip 2 Constructor
    function obj = Plates_Hip_2(d_knee_shaft,D_hip_shaft,shaft_hipKnee,
        max_diameter_pulley,leg_r1,distance_between_shafts,adapter_Knee,
        key_Hip_Plates,spring_Torsion_Hip,spring_Torsion_Knee)
         obj = obj.setParameters(d_knee_shaft,D_hip_shaft,shaft_hipKnee,
            max_diameter_pulley,leg_r1,distance_between_shafts,
            adapter_Knee,key_Hip_Plates,spring_Torsion_Hip,
            spring_Torsion_Knee);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,d_knee_shaft,D_hip_shaft,
        shaft_hipKnee,max_diameter_pulley,leg_r1,distance_between_shafts,
        adapter_Knee,key_Hip_Plates,spring_Torsion_Hip,spring_Torsion_Knee
        )
         obj.d_knee = d_knee_shaft + 4;
         obj.d_hip = D_hip_shaft;
         obj.d_hipknee = shaft_hipKnee.d_small +4;
         obj.height = max_diameter_pulley +4;
         obj.curve = obj.height /2;
         obj.r1 = leg_r1;
         obj.L_hipknee = distance_between_shafts;
         obj.L_end = adapter_Knee.outer_diameter/2+5;
         obj.w_k = key_Hip_Plates.w;
         obj.h_k = key_Hip_Plates.h/2;
         obj.r_springHip = spring_Torsion_Hip.r;
         obj.Rnom_springHip =  spring_Torsion_Hip.bigR;
         obj.d_springHip = spring_Torsion_Hip.smalld;
         obj.dHolder_springHip = obj.d_springHip * 4;
         obj.r_springKnee = spring_Torsion_Knee.r;
```

```
        obj.Rnom_springKnee = spring_Torsion_Knee.bigR;
        obj.d_springKnee = spring_Torsion_Knee.smalld;
        obj.dHolder_springKnee = obj.d_springKnee *4;

    end

    % Print function to TXT file
    function printTXT(obj)
        filePath = '..\Solidworks\Equations\HipPlate2.txt';
        fid = fopen(filePath,'wt');
        fprintf(fid,strcat('"d_knee"= ',num2str(obj.d_knee),'\n'));
        fprintf(fid,strcat('"d_hip"= ',num2str(obj.d_hip),'\n'));
        fprintf(fid,strcat('"d_hipknee"= ',num2str(obj.d_hipknee),'\n'));
        fprintf(fid,strcat('"height"= ',num2str(obj.height),'\n'));
        fprintf(fid,strcat('"curve"= ',num2str(obj.curve),'\n'));
        fprintf(fid,strcat('"r1"= ',num2str(obj.r1),'\n'));
        fprintf(fid,strcat('"L_hipknee"= ',num2str(obj.L_hipknee),'\n'));
        fprintf(fid,strcat('"L_end"= ',num2str(obj.L_end),'\n'));
        fprintf(fid,strcat('"w_k"= ',num2str(obj.w_k),'\n'));
        fprintf(fid,strcat('"h_k"= ',num2str(obj.h_k),'\n'));
        fprintf(fid,strcat('"r_springHip"= ',num2str(obj.r_springHip),'\n'
            ));
        fprintf(fid,strcat('"Rnom_springHip"= ',num2str(obj.Rnom_springHip
            ),'\n'));
        fprintf(fid,strcat('"dHolder_springHip"= ',num2str(
            obj.dHolder_springHip),'\n'));
        fprintf(fid,strcat('"d_springHip"= ',num2str(obj.d_springHip),'\n'
            ));
        fprintf(fid,strcat('"r_springKnee"= ',num2str(obj.r_springKnee),'\
            n'));
        fprintf(fid,strcat('"Rnom_springKnee"= ',num2str(
            obj.Rnom_springKnee),'\n'));
        fprintf(fid,strcat('"dHolder_springKnee"= ',num2str(
            obj.dHolder_springKnee),'\n'));
        fprintf(fid,strcat('"d_springKnee"= ',num2str(obj.d_springKnee),'\
            n'));

        fclose(fid);
    end

end
```

```
end
```

C.39 Plates Hip Knee

```
classdef Plates_KneeHip
    properties
        D_hipkneeshaft  % [mm] Large diameter of the hip knee shaft
        D_o             % [mm]
        d_bolts         % [mm]
        L_bolts         % [mm]
    end
    methods

        function obj = Plates_KneeHip(D_hipKnee_shaft,d_interior_pulley,
            L_interior_pulley,spring_nominal_d,spring_wire_d)
            obj = obj.setParameters(D_hipKnee_shaft,d_interior_pulley,
                L_interior_pulley,spring_nominal_d,spring_wire_d);
        end

        function obj = setParameters(obj,D_hipKnee_shaft,d_interior_pulley,
            L_interior_pulley,spring_nominal_d,spring_wire_d)
            obj.D_hipkneeshaft = D_hipKnee_shaft;
            obj.d_bolts = d_interior_pulley;
            obj.L_bolts = L_interior_pulley;
            obj.D_o = spring_nominal_d + 4*spring_wire_d;
        end

        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipKneePulleyPlate.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"D_hipkneeshaft"= ',num2str(obj.D_hipkneeshaft
                ),'\n'));
            fprintf(fid,strcat('"D_o"= ',num2str(obj.D_o),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fclose(fid);
        end

    end
end
```

C.40 Collar Hip

```
classdef Collar_Hip
    properties
        d_hipshaft  % [mm] Small Diameter
        D_o         % [mm] Large Diameter
        d_bolts     % [mm] Diameter of bolt holes
        L_bolts     % [mm] Length of bolt holes
        w_k         % [mm] Width of key
        h_k         % [mm] Height of key
        n_bolts     % [quantity] Number of bolts
    end
    methods

        % Collar Hip Constructor
        function obj = Collar_Hip(d_hip_shaft, key_Hip_Collar,
            d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,n_bolts_HD_flexSpline)
            obj = obj.setParameters(d_hip_shaft, key_Hip_Collar,
                d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,
                n_bolts_HD_flexSpline);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, d_hip_shaft, key_Hip_Collar,
            d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,n_bolts_HD_flexSpline)
            obj.d_hipshaft = d_hip_shaft;
            obj.d_bolts = d_bolt_HD_flexSpline;
            obj.L_bolts = L_bolt_HD_flexSpline/2;
            obj.w_k = key_Hip_Collar.w;
            obj.h_k = key_Hip_Collar.h/2;
            obj.n_bolts = n_bolts_HD_flexSpline;
            obj.D_o = (obj.L_bolts +1.5*obj.d_bolts)*2;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipCollarOutput2.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_hipshaft"= ',num2str(obj.d_hipshaft),'\n'))
                ;
```

```
            fprintf(fid,strcat('"D_o"= ',num2str(obj.D_o),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fprintf(fid,strcat('"w_k"= ',num2str(obj.w_k),'\n'));
            fprintf(fid,strcat('"h_k"= ',num2str(obj.h_k),'\n'));
            fprintf(fid,strcat('"n_bolts"= ',num2str(obj.n_bolts),'\n'));
            fclose(fid);
        end

    end
end
```

C.41 Collar Hip Knee

```
classdef Collar_KneeHip
    properties
        d_hipkneeshaft  % [mm] Small Diameter
        D_o             % [mm] Large Diameter
        d_bolts         % [mm] Diameter of bolt hole
        L_bolts         % [mm] Length of bolt hole
        w_k             % [mm] Width of key
        h_k             % [mm] Height of key
        n_bolts         % [quantity] Number of bolts
    end
    methods

        % Collar Knee Hip Constructor
        function obj = Collar_KneeHip(d_hipKnee_shaft, key_KneeHip_Hub,
            d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,n_bolts_HD_flexSpline)
            obj = obj.setParameters(d_hipKnee_shaft, key_KneeHip_Hub,
                d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,
                n_bolts_HD_flexSpline);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj, d_hipKnee_shaft, key_KneeHip_Hub,
            d_bolt_HD_flexSpline,L_bolt_HD_flexSpline,n_bolts_HD_flexSpline)
            obj.d_hipkneeshaft = d_hipKnee_shaft;
            obj.d_bolts = d_bolt_HD_flexSpline;
            obj.L_bolts = L_bolt_HD_flexSpline/2;
```

```
            obj.w_k = key_KneeHip_Hub.w;
            obj.h_k = key_KneeHip_Hub.h/2;
            obj.n_bolts = n_bolts_HD_flexSpline;
            obj.D_o = (obj.L_bolts +1.5*obj.d_bolts)*2;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\HipKneeCollarOutput2.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"d_hipkneeshaft"= ',num2str(obj.d_hipkneeshaft
                ),'\n'));
            fprintf(fid,strcat('"D_o"= ',num2str(obj.D_o),'\n'));
            fprintf(fid,strcat('"d_bolts"= ',num2str(obj.d_bolts),'\n'));
            fprintf(fid,strcat('"L_bolts"= ',num2str(obj.L_bolts),'\n'));
            fprintf(fid,strcat('"w_k"= ',num2str(obj.w_k),'\n'));
            fprintf(fid,strcat('"h_k"= ',num2str(obj.h_k),'\n'));
            fprintf(fid,strcat('"n_bolts"= ',num2str(obj.n_bolts),'\n'));
            fclose(fid);
        end

    end
end
```

C.42 Timing Belt

```
classdef TimingBelt
    properties
        R_i     % [mm] Inner Radius
        R_o     % [mm] Outer Radius
        C       % [mm] Distancee center to center of pulleys
    end
    methods

        % Timing Belt Constructor
        function obj = TimingBelt(belt_system)
            obj = obj.setParameters(belt_system);
        end

        % Properties and parameters calculation function
```

```
        function obj = setParameters(obj, belt_system)
            obj.R_i = belt_system.pulley_pitch_dia/2;
            obj.R_o = belt_system.Pulley_Belt_Total_Dia/2;
            obj.C = belt_system.C;
        end

        % Print function to TXT file
        function printTXT(obj)
            filePath = '..\Solidworks\Equations\TimingBelt.txt';
            fid = fopen(filePath,'wt');
            fprintf(fid,strcat('"R_i"= ',num2str(obj.R_i),'\n'));
            fprintf(fid,strcat('"R_o"= ',num2str(obj.R_o),'\n'));
            fprintf(fid,strcat('"C"= ',num2str(obj.C),'\n'));
            fclose(fid);
        end

    end
end
```

C.43 Chassis

```
classdef Chassis
    properties
        Hip_HD_D                % [mm]   Harmonic Drive outer diameter
        Hip_plate_width         % [mm]   Hip plates total width
        Hip_plate_height        % [mm]   Hip plates heigth
        theta_max = 28;         % [deg]  Highest angular position of thigh
        t_plate                 % [mm]   Thickness of mounting plate
        L_bracket_hole          % [mm]   L-bracket hole position from plate
        BoxArea                 % [mm^2] Litter box surface area
        Hip_shaft_shortside_L   % [mm]   Shaft length from pulley to end
        Hip_plate_L             % [mm]   Dist b/w hip shaft & end of plate
        Total_hip_L             % [mm]   Total hip plates length
        Chassis_width           % [mm]   Chassis outer width
        Chassis_depth           % [mm]   Chassis outer depth
        B_D                     % [mm]   Depth of plate for WR2B
        B_W                     % [mm]   Width of plate for WR2B
        Chassis_height          % [mm]   Chassis height
        t_wall                  % [mm]   Chassis wall thickness
        B_t_plate               % [mm]   Thickness of plate for WR2B
```

```
    D_i                     % [mm]   Inner diameter of leg hole
    Hole_vert_P             % [mm]   Leg hole vertical position
    Leg_spacing             % [mm]   Distance between leg mid planes
    Hole_front_horiz_P      % [mm]   Front leg holes horiz position
    l_flange                % [mm]   Length of flange to mount bellow
    Leg_spacing_half        % [mm]   Half the distance between legs
    fillet_r                % [mm]   Inner chassis fillet radius
    fillet_r_outside        % [mm]   Outer chassis fillet radius
    Lip_offset              % [mm]   Lip offset distance from wall
    Lip_inner_r             % [mm]   Lip inner fillet radius
    Lip_outer_r             % [mm]   Lip outer fillet radius
    Lip_width               % [mm]   Lip width
    L                       % [mm]   Variable used in Analysis Sec4.9.6
    mass                    % [mm]   Approximated mass of chassis
    area                    % [mm]   Approximated chassis cover area
    L_hipknee               % [mm]   Length between hip and knee shaft
    R_pulley_max            % [mm]   Maximum pulley radius
end

methods

    % Chassis Constructor
    function obj = Chassis(shaft_HipKnee,spring_Torsion_Hip,plates_Hip_1,
        BoxArea,hip_base,hip_bracket,motor_hip,HD_hip,belt_System)
        obj = obj.setParameters(shaft_HipKnee,spring_Torsion_Hip,
            plates_Hip_1,BoxArea,hip_base,hip_bracket,motor_hip,HD_hip,
            belt_System);
    end

    % Properties and parameters calculation function
    function obj = setParameters(obj,shaft_HipKnee,spring_Torsion_Hip,
        plates_Hip_1,BoxArea,hip_base,hip_bracket,motor_hip,HD_hip,
        belt_System)
        obj.Hip_plate_width = shaft_HipKnee.L_mid+2*2+2*10;
        obj.Hip_plate_height = plates_Hip_1.height;
        obj.Hip_shaft_shortside_L = (shaft_HipKnee.L_mid+2*2)/2+10+
            spring_Torsion_Hip.L+2+10;
        obj.Hip_plate_L = plates_Hip_1.L_hipknee + plates_Hip_1.L_end;
        obj.L_hipknee = plates_Hip_1.L_hipknee;
        obj.R_pulley_max = belt_System.Pulley_Belt_Total_Dia/2;
        obj.BoxArea = BoxArea;
```

```
        obj.Hip_HD_D = HD_hip.spline_outer_diameter;
        obj.t_plate = hip_base.thickness;
        obj.L_bracket_hole = hip_bracket.height + hip_base.thickness;
        obj.Total_hip_L = hip_base.distance_between_hip_bracket_bolts +
            hip_base.hip_bracket_bolt_to_adapter_bolt + (
            hip_base.bolt_diameter+(hip_base.bolt_ell*2)) +
            hip_bracket.thickness + motor_hip.adapter_thickness +
            motor_hip.thickness;
        obj = obj.calculate();
    end

    function obj = calculate(obj)
        %% Constants
        obj.fillet_r = 5;
        obj.t_wall = 5;
        obj.B_t_plate = 2*obj.t_wall;
        obj.fillet_r_outside = obj.fillet_r + obj.t_wall;
        obj.l_flange = 12;
        spacing = 5;
        obj.Lip_offset = obj.t_wall/2;
        obj.Lip_inner_r = obj.fillet_r + obj.Lip_offset;
        obj.Lip_width = 2*obj.t_wall - 2*obj.Lip_offset;
        obj.Lip_outer_r = obj.Lip_inner_r + obj.Lip_width;

        %% Minimum Leg Hole Dimensions
        obj.L = obj.Hip_HD_D/2 + obj.fillet_r + obj.t_wall + obj.l_flange;
        H = obj.L*tand(obj.theta_max)+obj.Hip_plate_height/(2*cosd(
            obj.theta_max));
        R_i_min = sqrt(H^2+(obj.Hip_plate_width/2)^2);
        R_i = R_i_min + 2.5; % for precautions
        obj.D_i = 2*R_i;
        D_o = obj.D_i + 2*obj.t_wall;

        %% Litter Box Dimensions
        BoxRatio = 7/6;
        Box_width = sqrt(obj.BoxArea/BoxRatio);
        Box_depth = BoxRatio*Box_width;
        ExtraSpaceRatio = 1.1;
        Box_W = Box_width*ExtraSpaceRatio;
        Box_D = Box_depth*ExtraSpaceRatio;
        Hand_space = 50;
```

```
Arm_space = 120;
obj.B_W = Box_W;
obj.B_D = Box_D + Hand_space + Arm_space;

%% Other Chassis Dimensions

obj.Hole_vert_P          = obj.t_wall + obj.L_bracket_hole;
Hole_front_horiz_P1      = obj.B_W/2 + obj.t_wall +
   obj.Hip_shaft_shortside_L +1;
Hole_front_horiz_P2      = obj.B_W/2 + obj.t_wall + obj.fillet_r +
    D_o/2 +1;
obj.Hole_front_horiz_P   = max(Hole_front_horiz_P1,
   Hole_front_horiz_P2);

Motor_side_hip_L      = obj.Total_hip_L -
   obj.Hip_shaft_shortside_L;
obj.Leg_spacing       = 2*Motor_side_hip_L + spacing;
obj.Leg_spacing_half  = obj.Leg_spacing/2;

Chassis_back_width1   = 2*obj.Leg_spacing + 2*
   obj.Hip_shaft_shortside_L + 2*obj.t_wall + 2*obj.fillet_r + 2*
   spacing;
Chassis_back_width2   = 2*obj.Leg_spacing + D_o + 2*obj.t_wall +
   2*obj.fillet_r;
Chassis_front_width1  = 2*obj.Total_hip_L + 4*obj.t_wall + obj.B_W
    + 4*obj.fillet_r;
Chassis_front_width2  = 2*(obj.Total_hip_L - Motor_side_hip_L) +
   4*obj.t_wall + obj.B_W + 4*obj.fillet_r + D_o;
obj.Chassis_width     = max([Chassis_back_width1,
   Chassis_back_width2,Chassis_front_width1,Chassis_front_width2
   ]);

Chassis_side_depth    = 2*(obj.L - obj.l_flange + obj.Hip_plate_L)
    + spacing;
Chassis_center_depth1 = (obj.L - obj.l_flange + obj.Hip_plate_L +
   obj.t_wall) + 2*spacing + obj.B_D;
Chassis_center_depth2 = (obj.L - obj.l_flange + obj.L_hipknee +
   obj.R_pulley_max + obj.t_wall) + 2*spacing + obj.B_D;
obj.Chassis_depth     = max([Chassis_side_depth,
   Chassis_center_depth1,Chassis_center_depth2]);
```

```
        Chassis_height1       = obj.Hole_vert_P + D_o/2 + 3*spacing;
        Chassis_height2       = obj.t_wall + obj.fillet_r + D_o + 3*
            spacing;
        obj.Chassis_height    = max(Chassis_height1,Chassis_height2);

        % Chassis mass
        material_density = 1.18E-6; % [kg/mm^3] for acrylic
        V_o = obj.Chassis_width * obj.Chassis_depth * obj.Chassis_height;
        V_i = (obj.Chassis_width - 2*obj.t_wall) * (obj.Chassis_depth - 2*
            obj.t_wall) * (obj.Chassis_height - 2*obj.t_wall);
        obj.mass = (V_o - V_i) * material_density;

        % Solar cell available area
        obj.area = (obj.Chassis_width * obj.Chassis_depth) - (Box_W *
            Box_D);

    end

    % Print function to TXT file
    function printTXT(obj)
        filePath = '..\Solidworks\Equations\Chassis.txt';
        fid = fopen(filePath,'wt');
        fprintf(fid,strcat('"Width"=',num2str(obj.Chassis_width),'\n'));
        fprintf(fid,strcat('"Depth"=',num2str(obj.Chassis_depth),'\n'));
        fprintf(fid,strcat('"2B_depth"=',num2str(obj.B_D),'\n'));
        fprintf(fid,strcat('"2B_width"=',num2str(obj.B_W),'\n'));
        fprintf(fid,strcat('"Chassis_height"=',num2str(obj.Chassis_height)
            ,'\n'));
        fprintf(fid,strcat('"Wall_thickness"=',num2str(obj.t_wall),'\n'));
        fprintf(fid,strcat('"2B_plate_thickness"=',num2str(obj.B_t_plate),
            '\n'));
        fprintf(fid,strcat('"Leg_hole_inner_D"=',num2str(obj.D_i),'\n'));
        fprintf(fid,strcat('"Leg_hole_vertical_P"=',num2str(
            obj.Hole_vert_P),'\n'));
        fprintf(fid,strcat('"Leg_spacing"=',num2str(obj.Leg_spacing),'\n')
            );
        fprintf(fid,strcat('"Front_leg_horizontal_P"=',num2str(
            obj.Hole_front_horiz_P),'\n'));
        fprintf(fid,strcat('"Flange_length"=',num2str(obj.l_flange),'\n'))
            ;
        fprintf(fid,strcat('"Flange_thickness"=',num2str(obj.t_wall),'\n')
```

```
            );
        fprintf(fid,strcat('"Half_leg_spacing"=',num2str(
            obj.Leg_spacing_half),'\n'));
        fprintf(fid,strcat('"fillet_r"=',num2str(obj.fillet_r),'\n'));
        fprintf(fid,strcat('"fillet_r_outside"=',num2str(
            obj.fillet_r_outside),'\n'));
        fprintf(fid,strcat('"Lip_offset"=',num2str(obj.Lip_offset),'\n'));
        fprintf(fid,strcat('"Lip_inner_r"=',num2str(obj.Lip_inner_r),'\n')
            );
        fprintf(fid,strcat('"Lip_outer_r"=',num2str(obj.Lip_outer_r),'\n')
            );
        fprintf(fid,strcat('"Lip_width"=',num2str(obj.Lip_width),'\n'));
        fclose(fid);
    end
  end
end
```

C.44 Bellow

```
classdef Bellow
    properties
        bellow_inner_length     % [mm] Bellow inner length excluding flanges
        bellow_thickness        % [mm] Bellow wall thickness
        large_fold_length       % [mm] Bellow large fold length
        Bellow_small_inner_d    % [mm] Bellow tibia inner flange diameter
        Bellow_large_inner_D    % [mm] Bellow chassis inner flange diameter
        R_chassis_flange        % [mm} Bellow tibia inner flange radius
        R_knee_flange           % [mm] Bellow chassis inner flange radius
    end

    methods

        % Bellow Constructor
        function obj = Bellow(plates_Hip_1,chassis,tibiaPulleyHolder)
            obj = obj.setParameters(plates_Hip_1,chassis,tibiaPulleyHolder);
        end

        % Properties and parameters calculation function
        function obj = setParameters(obj,plates_Hip_1,chassis,
            tibiaPulleyHolder)
```

```
    obj.bellow_inner_length = plates_Hip_1.r1 - (chassis.L - 2) + (
        tibiaPulleyHolder.L_shaft + 5);
    obj.Bellow_small_inner_d = tibiaPulleyHolder.D_bellowholder - 4;
    obj.Bellow_large_inner_D = chassis.D_i + 2*chassis.t_wall;
    obj = obj.calculate();
end

% Parametric Calculation function
function obj = calculate(obj)

    %% Parameterized Bellow Dimensions
    obj.R_chassis_flange = obj.Bellow_large_inner_D/2;
    obj.R_knee_flange = obj.Bellow_small_inner_d/2;
    % Possible scenarios for fold length
    large_fold_length1 = (obj.R_chassis_flange-obj.R_knee_flange)/2;
    large_fold_length2 = obj.bellow_inner_length/5;
    % Picks the largest value for the fold length
    obj.large_fold_length = max(large_fold_length1,large_fold_length2)
        ;

    %% Constant bellow dimensions
    obj.bellow_thickness = 2;

end

% Print function to TXT file
function printTXT(obj)
    filePath = '..\Solidworks\Equations\Bellow.txt';
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"R_chassis_flange"=',num2str(
        obj.R_chassis_flange),'\n'));
    fprintf(fid,strcat('"R_knee_flange"=',num2str(obj.R_knee_flange),'
        \n'));
    fprintf(fid,strcat('"bellow_thickness"=',num2str(
        obj.bellow_thickness),'\n'));
    fprintf(fid,strcat('"large_fold_length"=',num2str(
        obj.large_fold_length),'\n'));
    fprintf(fid,strcat('"bellow_inner_length"=',num2str(
        obj.bellow_inner_length),'\n'));
    fclose(fid);
end
```

```
    end
end
```

C.45 Solar Energy

```
% Parameters
% Area              % [mm^2] Total area of solar panels

%Outputs
% Watts_produced    % [W] Watt produced by solar panels
function [Watts_produced] = solarEnergy(Area)
    averageWattsPerArea = 130.78; % [W/m^2]
    Watts_produced = averageWattsPerArea * Area/(1000^2);
```

C.46 Linkages

```
% Inputs
% x_reach       = [mm] desired reach in x
% y_reach       = [mm] desired reach in y
% Outputs
% r1            = [mm] length of thigh linkage
% r2            = [mm] length of upper tibia
% r3            = [mm] length of lower tibia
% R             = [mm] virtual length from knee to foot
% phi_min       = [rad] minimum angle between knee and foot from horizontal
% phi_max       = [rad] maximum angle between knee and foot from horizontal
% d             = walking height from hip
% x_walking_min = [mm] minimum distance of foot from hip while walking
% x_walking_max = [mm] maximum distance of foot from hip while walking

function [r1,r2,r3,R,phi_min,phi_max,d,x_walking_min,x_walking_max] = linkages
    (x_reach,y_reach)
    % Reference values are given by workspace.m using r1=r2=100mm, r3 = 300mm
       and alpha = 69 degrees, theta_max = 28 degrees.
    % If the user gives a x or y reach higher than the reference, then the
       lengths will scale accordingly
    % The smallest desired reach will be respected, and the other will have
       higher range
```

```
    % Maximum reach in x and y (not necessarily at walking height)
    x_reference = 223.9456;
    y_reference = 52.5435;
    % walking x range and walking ground height
    x_walking_min_reference = 105.4968;
    x_walking_max_reference = 234.3159;
    ground_height_reference = 232.5574;

    % The ratio of x_reach to y_reach must be conserved
    if (x_reach/y_reach) < 4.2621
        x_reach = y_reach*4.2621;
    end

    r1 = 100*(x_reach/x_reference);
    r2 = 120*(x_reach/x_reference);
    r3 = 300*(x_reach/x_reference);
    % Geometric derivations similar to those in workspace.m
    R = sqrt(r2^2 + r3^2 - (2*r2*r3*cos(deg2rad(69))));
    beta = acos((r3^2 - R^2 - r2^2)/(-2*R*r2));
    phi_min = deg2rad(-22.5) - beta;
    phi_max = deg2rad(22.5) - beta;
    d = ground_height_reference*(x_reach/x_reference);
    x_walking_min = x_walking_min_reference*(x_reach/x_reference);
    x_walking_max = x_walking_max_reference*(x_reach/x_reference);
end
```

C.47 Keys

```
% Parameters
% D                 [mm] Large Diameter of component
% Length_component  [mm] Length of component
% Torque            [Nmm] Torque on the component

% Outputs
% key_width         [mm] Width of key
% key_height        [mm] Height of key
function[key_width, key_height, worked]= keys(D, Torque, Length_component,
    name)
m_Sy = 205; %   [MPa] Yield Strength
%% Dimensions
```

```
worked = true;
key_width = D/4;
key_height = D/3;
length_shear = (8*Torque)/(0.58*m_Sy*D^2);
if (Length_component <  length_shear)
    %disp("Key Too Short") % Warning that the key is too short
    worked = false;
end
```

C.48 Dynamic Equation

```
% Parameters
% m1                mass of knee [kg]
% m2                mass of foot [kg]
% l                 simplified length of tibia [m]
% L                 length of thigh [m]
% theta             angle of thigh [deg]
% phi               angle of simplified tibia [deg]
% thetadd           angular acceleration of tibia [rad/s]
% phidd             angular acceleration of thigh [rad/s]
% N                 Normal force [N]
% f                 Friction force [N]
% k_hip             Spring constant hip [Nmm]
% k_knee            Spring constant knee [Nmm]
% Correction_hip    Angle of correction hip [deg]
% Correction_knee   Angle of correction knee [deg]

% Outputs
% torque_hip        Torque at hip [Nm]
% torque_knee       Torque at knee [Nm]

% Dynamic Equation in matrix format (Section 3.2 of Analysis Report)
function [torque_hip,torque_knee] = dynamic_equation(m1, m2, l, L, theta, phi,
    thetadd, phidd, N, f,k_hip,k_knee,Correction_hip,Correction_knee)

    M1M2_hip = ((m1+m2).*L.^2 .* thetadd) + (m2.*l.*L.*cosd(theta-phi) .*
       phidd);
    M1M2_knee = (m2.*l.*L.*cosd(theta-phi) .* thetadd) + (m2.*l .* phidd);
    M3_hip = (m1+m2).*L.*cosd(theta)+m2.*l.*cosd(theta);
    M3_knee = m2.*l.*cosd(phi);
```

```
    M4_hip = L.*cosd(theta)+l.*cosd(phi);
    M4_knee = l.*cosd(phi);
    M5_hip = L.*sind(theta) + l.*sind(phi);
    M5_knee = l.*sind(phi);

    M6_hip = M1M2_hip - 9.81*M3_hip + N.*M4_hip - f.*M5_hip;
    M6_knee = M1M2_knee - 9.81*M3_knee + N.*M4_knee - f.*M5_knee;

    % Torque corrected due to spring
    torque_hip = M6_hip - (2*k_hip*(pi/180)*(-theta + Correction_hip)/1000);
    torque_knee= M6_knee - (2*k_knee*(pi/180)*(phi-theta + Correction_knee)
       /1000);
end
```

C.49 Dynamics

```
% Parameters
% r1            [mm] Length of thigh
% r2            [mm] Length of upper tibia
% r3            [mm] Length of lower tibia
% e_max         [mm] Max distance in x
% d             [mm] Height of the thigh motor
% alpha         [deg] Angle of the bend in the tibia
% R             [mm] Length of simplified tibia

% Outputs
% Theta         [deg] Angle between horizontal and thigh
% Phi           [deg] Angle between horizontal and simplified tibia
% Psy           [deg] Angle between horizontal and upper tibia
% Alpha         [deg] Angle between horizontal and lower tibia
% small_phi     [deg] Angle between Phi and Psy

% Geometric Calculations Function (Section 2.1 of Design Dossier)
function[Theta, Phi, Psy, Alpha,small_phi] = Dynamics(r1,r2,r3,e_max,d,alpha,R
   )
C = sqrt(e_max^2+d^2);
c = acosd((C^2-r1^2-R^2)/(-2*r1*R));
b = acosd((R^2-r1^2-C^2)/(-2*r1*C));
small_phi = acosd((r3^2-r2^2-R^2)/(-2*r2*R));
Theta = b + atand(e_max/d)-90;
```

```
Phi = Theta + c -180;
Psy = Phi + small_phi;
Alpha = Psy + alpha + 180;
```

C.50 Leg Angles

```
% Parameters
% r1            [mm] Length of thigh
% x             [mm] Max distance in x
% y             [mm] Height of the thigh motor
% R             [mm] Length of simplified tibia

% Outputs
% theta         [deg] Angle between horizontal and thigh
% phi           [deg] Angle between horizontal and simplified tibia

% Geometric Calculations Function (Section 2.1 of Design Dossier)
function [theta, phi] = leg_angles(r1, R, y, x)
    C = sqrt(y.^2 + x.^2);
    c = acos((C.^2 -r1^2 - R^2)/(-2*r1*R))*180/pi;
    b = acos((R^2 - r1^2 - C.^2)./(-2*r1*C))*180/pi;
    theta = b + atan(x/y)*180/pi - 90;
    phi = theta + 180 + c;
end
```

C.51 Leg Angular Velocity

```
% Parameters
% r1            [mm] Length thigh
% R             [mm] Length simplified tibia
% theta         [deg] Angle of tibia
% phi           [deg] Angle of splified tibia
% dd            [mm/s] Vertical velocity
% ed            [mm/s] Horizontal velocity

% Ouputs
% thetad        [rad/s] Angular velocity of Theta
% phid          [rad/s] Angular velocity of Phi
```

```
% Calculation Function (Section 2.1 of Design Dossier)
function[thetad, phid] = leg_angles_dot(r1,R,theta,phi,dd,ed)
    m1=-r1.*sind(theta);
    m2=-R.*sind(phi);
    m3=r1.*cosd(theta);
    m4=R.*cosd(phi);

    det = 1./(m4.*m1 - m2.*m3);

    m1i = det.*m4;
    m2i = -det.*m2;
    m3i= -det.*m3;
    m4i = det.*m1;

    thetad = (m1i.*ed + m2i.*dd);
    phid = (m3i.*ed + m4i.*dd);
end
```

C.52 Leg Angular Acceleration

```
% Parameters
% r1            [mm] Length thigh
% R             [mm] Length simplified tibia
% theta         [deg] Angle of tibia
% phi           [deg] Angle of splified tibia
% ddd           [mm/s^2] Vertical acceleration
% edd           [mm/s^2] Horizontal acceleration
% thetad        [rad/s] Angular velocity of Theta
% phid          [rad/s] Angular velocity of Phi

% Outputs
% thetadd       [rad/s^2] Angular acceleration of theta
% phidd         [rad/s^2] Angular acceleration of phi

% Calculation Function (Section 2.1 of Design Dossier)
function[thetadd, phidd] = leg_angles_ddot(r1,R,theta,thetad,phi,phid,ddd,edd)

    m1=-r1.*sind(theta);
    m2=-R.*sind(phi);
    m3=r1.*cosd(theta);
```

```
    m4=R.*cosd(phi);

    det = 1./(m4.*m1 - m2.*m3);

    m1i = det.*m4;
    m2i = -det.*m2;
    m3i= -det.*m3;
    m4i = det.*m1;

    M1 = edd+r1.*cosd(theta).*(thetad.^2)+R.*cosd(phi).*(phid.^2);
    M2 = ddd+r1.*sind(theta).*(thetad.^2)+R.*sind(phi).*(phid.^2);

    thetadd = (m1i.*(M1) + m2i.*M2);
    phidd = (m3i.*(M1) + m4i.*M2);
end
```

C.53 Torque Angles

```
% Inputs
% leg_Object            = object containing r1, R, m1, m2
% force_Object          = object containing forces
% k_hip, k_knee         = [Nmm] spring constants of hip and knee
% correction_hip,knee   = [rad] geometric property of springs

% Outputs
% max_torque_theta      = [Nm] max torque experienced at hip
% max_torque_phi        = [Nm] max torque experienced at knee
% max_thetad            = [rad/s] max angular velocity at hip
% max_phid              = [rad/s] max angular velocity at knee
% theta                 = [theta] range of thetas over walking cycle
% phi                   = [theta] range of thetas over walking cycle
% thetad, phid          = [rad/s] range of angular velocities over walking
   cycle
% thetadd, phidd        = [rad/s^2] range of angular accelerations over
   walking cycle
% torque_theta          = [Nm] range of torques at hip over walking cycle
% torque_phi            = [Nm] range of torques at knee over walking cycle

% Calculate all torques, angles, and derivatives over the regular walking
   cycle
```

```
function [max_torque_theta,max_torque_phi,max_thetad,max_phid,theta, phi,
    thetad, phid,thetadd, phidd,torque_theta,torque_phi] = Torques_Angles(
    leg_Object, force_Object, k_hip, k_knee, correction_hip, correction_knee)
    % Makes use of the dynamic equation from Section 3.2 Dynamic Equation of
        Analysis Report
    % Calculates the torques and velocities necessary for Power_Consumption to
         run (first part of Section 3.5.3 Robot Power Consumption of Analysis
        Report)

    r1 = leg_Object.r1;
    R = leg_Object.R;
    d = leg_Object.d;
    m1 = force_Object.m_foot;
    m2 = force_Object.m_knee;
    N = force_Object.FN;
    f = force_Object.FF;
    x_walking_min = leg_Object.x_walking_min;
    x_walking_max = leg_Object.x_walking_max;

    % velocity in x is constant multiple of leg length
    ed = r1*0.5;
    dd =0;
    ddd=0;
    edd=0;

    % Simulation properties
    steps = 1000;
    e = linspace(x_walking_min,x_walking_max,steps);
    gear_ratio=100;

    % 3 phases
    torque_theta = zeros(1,steps*3);
    torque_phi = zeros(1,steps*3);

    %--------------------------------------%
    %% TORQUES
    %--------------------------------------%
    % Phase 1: leg pulls body forward
    [theta, phi] = leg_angles(r1,R,d,e);
    [thetad, phid] = leg_angles_dot(r1,R,theta,phi,dd,ed);
    [thetadd, phidd] = leg_angles_ddot(r1,R,theta,thetad,phi,phid,ddd,edd);
```

```
    % Dynamic equation works in [N] and [m]
    [torque_theta(1:1000),torque_phi(1:1000)] = dynamic_equation(m1,m2,R/1000,
        r1/1000, theta, phi,thetadd, phidd, N,f,k_hip,k_knee,correction_hip,
        correction_knee);
    % Phase 2: phi increases
    [torque_theta(1001:2000),torque_phi(1001:2000)] = dynamic_equation(m1,m2,R
        /1000,r1/1000, theta(1), phi,0, 0, 0,0,k_hip,k_knee,correction_hip,
        correction_knee);
    % Phase 3: theta decreases
    [torque_theta(2001:3000),torque_phi(2001:3000)] = dynamic_equation(m1,m2,R
        /1000,r1/1000, theta, phi(steps),0, 0, 0,0,k_hip,k_knee,correction_hip
        ,correction_knee);

    max_torque_theta = max(abs(torque_theta(:)));
    max_torque_phi = max(abs(torque_phi(:)));
    max_thetad = max(abs(thetad));
    max_phid = max(abs(phid));
end
```

C.54 Fasteners General

```
function [t_head,d_head,t_nut,d_nut,t_washer,od_washer,id_washer] =
    fasteners_general(d)
    % Functions extracted using polyfit(x,y,1)
    % Bolt dimensions taken from McMaster Grade 8.8 metric bolts
    % Nuts dimensions following ISO 4032, taken from: https://www.amesweb.info
        /Fasteners/Nut/Metric-Hex-Nut-Sizes-Dimensions-Chart.aspx

    %% INPUTS
    % d         = bolt diameter [mm]

    %% OUTPUTS
    % t_head    = thickness of head [mm]
    % d_head    = diameter of head [mm]
    % t_nut     = thickness of nut [mm]
    % d_nut     = diameter of nut [mm]
    % t_washer  = thickness of washer [mm]
    % od_washer = outer diameter of washer [mm]
    % id_washer = inner diameter of washer [mm]
```

```
    t_head = 0.60433*d + 0.39;
    d_head = 1.5617*d + 0.5761;
    t_nut = 0.8584*d - 0.0455;
    d_nut = 1.5045*d + 0.9042;
    t_washer = 0.2254*d - 0.1096;
    od_washer = 1.8473*d + 1.2575;
    %id_washer = 1.038*d + 0.1222; We'll simplify by just using d
    id_washer = d;
end
```

C.55 Fasteners Hip

```
%% INPUTS
% r1            = length of thigh member [mm]
% R             = virtual length of tibia (straight line from knee to foot) [
    mm]
% t             = thickness of hip brackets and plate [mm]
% H_bracket     = height of the hip bracket (from center of shaft to
% W_bracket     = width of bracket [mm]
% L_shaft       = lengt of shaft between brackets [mm]
%Fx,Fz,Fz       = forces along fastener x, y and z [N]
%% OUTPUTS
% d             = [mm] bolt diameter
function [d,ell] = fasteners_hip(r1,R,t,H_bracket,W_bracket,L_shaft,Fx,Fy,Fz)
    d = linspace(1,20,20);
    % recommended distance from edge is 1.5 times the diameter; we're using 1
        .25 times since SF is so high [mm]
    % https://www.engineeringexpress.com/wiki/steel-bolt-edge-distance-
        requirements/
    ell = d.*1.25;

    %% Material Properties
    Sp = 650;       % [MPa] Proof strength
    Sy_b = 720;     % [MPa] Yield strength bolt
    Sut = 900;      % [MPa] Ultimate Tensile Strength bolt
    Se = 140;       % [MPa] Endurance strength bolt
    Sy_alu = 276;   % [MPa] Yield strength Aluminium 6061
    SF = 2.5;       % Safety factor for all but tension
    Eb = 200;       % [MPa] Elastic Modulus bolt
    Em = 276;       % [MPa] Elastic Modulus Aluminium 6061
```

```
%% GEOMETRIC PROPERTIES
% theta = angle between thigh and robot horizontal plane [rad]. 17
% degrees was found to give the largest moment
% phi = angle between tibia and robot horizontal plane [rad]. -51
% degrees was found to give the largest moment
theta = deg2rad(17.1887);
phi = deg2rad(-51.8913);

% Since the bolts are not necessarily all the same distance from the
    centroid, we will use the smallest distance for shear and largest
    distance for tension (gives lowest SF)
% From centroid to center of bolt
% Based on Figure 45 of Analysis Report
rx = (W_bracket/2) - (ell+(d./2));
ry = (L_shaft/2) - (ell+(d./2));
r_shear = min(rx,ry);
r_tension = max(rx,ry);
% From centroid to edge of plate
rpx = (W_bracket/2);
rpy = (L_shaft) + t;

% Number of bolts
n = 4;
% thickness of both members
g = t+t;
% Bolt diameter
Ab = pi*(d.^2)/4;
% tensile strength area, Polyfit with MATLAB, below equation 155
At = 0.7023*(d.^2) - 2.669*d + 9.963;

%% GEOMETRIC DERIVATIONS
x = r1.*cos(theta) + R.*cos(phi);
y = 0;
z = r1.*sin(theta) + R.*sin(phi) - (H_bracket+t);
rho = 0;                              %rho = atan2(y,x);
epsilon = atan2(z,x);
P = sqrt(r1.^2 + R.^2 - (2*r1.*R.*cos(phi - pi - theta)));

%% FORCES AND MOMENTS
% Pure moments applied to leg are assumed to be zero
```

```
% Equations 138, 139, 140, 141, 142
Vx = Fx;
Vy = Fy;
Ft = Fz;
My = (Fz*cos(epsilon) - Fx*sin(epsilon))*P + 0;
Mz = (Fy*cos(rho) - Fx*sin(rho))*P + 0;

%% PURE SHEAR
% Equations 143 to 148
Fsh_primary = sqrt(Vx^2 + Vy^2)/n;
Fsh_secondary = Mz./(n*r_shear);
Fsh = sqrt(Fsh_primary.^2 + Fsh_secondary.^2);
SF_s = (0.58*Sy_b*(pi*(d.^2)./4))./Fsh;

%% BEARING FORCES
% Equations 149 to 151
SF_b = t*Sy_b*d./Fsh;

%% EDGE SHEARING
% Equations 152, 153
SF_e = 0.577*Sy_alu*t*ell./Fsh;

%% TENSION
% Equations 155 to 164
Fi = 0.7*Sp*At;
T = 0.2*Fi.*d;
kb = Eb*Ab./g;
Am = d.^2 + 0.68*g*d + 0.065*(g^2);
km = (Em*Am)./g;
Fb = Fi + (((kb)/(kb+km)).*(1/n).*(Ft+(My./(rpx+r_tension))));
Fm = Fi - (((km)/(kb+km)).*(1/n).*(Ft+(My./(rpx+r_tension))));
sigma_b = Fb*4./(pi*(d.^2));
SF_t = Sy_b./sigma_b;

%% RESULTS
% Final d and ell
% See parametrization flowcharts for vectorized parts
[~,cols] = find(SF_s>2.5 & SF_b>2.5 & SF_e>2.5 & SF_t>2);
d = min(d(cols));
ell = d*1.25;
[t_head,d_head,t_nut,d_nut,t_washer,od_washer,id_washer] =
```

```
    fasteners_general(d);
    L_bolt = g + (t_nut*1.2) + 2*t_washer;      % Approximately a couple extra
        threads extending beyond nut

    %-------------------------------------%
    %% PRINT TO FILE
    %-------------------------------------%
    % FastenersHip.txt contains dimensions for the bolts and nuts
    filePath = '..\Solidworks\Equations\FastenersHip.txt';
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"diameter bolt"=',num2str(d),'\n'));
    fprintf(fid,strcat('"ell"=',num2str(ell),'\n'));
    fprintf(fid,strcat('"head thickness"=',num2str(t_head),'\n'));
    fprintf(fid,strcat('"head diameter"=',num2str(d_head),'\n'));
    fprintf(fid,strcat('"nut thickness"=',num2str(t_nut),'\n'));
    fprintf(fid,strcat('"nut diameter"=',num2str(d_nut),'\n'));
    fprintf(fid,strcat('"washer thickness"=',num2str(t_washer),'\n'));
    fprintf(fid,strcat('"washer outer diameter"=',num2str(od_washer),'\n'));
    fprintf(fid,strcat('"bolt length"=',num2str(L_bolt),'\n'));
    fclose(fid);
end
```

C.56 Fasteners Knee

```
%% INPUTS
% R             = [mm] virtual length of tibia (straight line from knee to
   foot)
% D_pulley      = [mm] outer diameter of pulley (from inside of teeth)
% D_shaft       = [mm] diameter of shaft the pulley is mounted on
% t_pulley      = [mm] thickness of pulley
% t_tibia       = [mm] thickness of tibia (one side of it)
% Fx, Fz        = [N] forces along fastener x (friction), z (normal)
%% OUTPUTS
% d*1           = [mm] bolt diameter
function [d,ell] = fasteners_knee(R,D_pulley,D_shaft,t_pulley,t_tibia,Fx,Fz)
    d = linspace(1,20,20);
    % recommended distance from edge is 1.5 times the diameter; we're using 1
       .25 times since SF is so high [mm]
    % https://www.engineeringexpress.com/wiki/steel-bolt-edge-distance-
       requirements/
```

```
ell = d.*1.25;

% Material Properties
Sp = 650;       % [MPa] Proof strength
Sy_b = 720;     % [MPa] Yield strength bolt
Sut = 900;      % [MPa] Ultimate Tensile Strength bolt
Se = 140;       % [MPa] Endurance strength bolt
Sy_pulley = 276;% [MPa] Yield strength Aluminium 6061
Sy_tibia = 276; % [MPa] Yield strength Aluminium 6061
SF = 2.5;       % Safety factor for all but tension
Eb = 200;       % [MPa] Elastic Modulus bolt
Em = 276;       % [MPa] Elastic Modulus Aluminium 6061

%% GEOMETRIC PROPERTIES
% phi = angle between tibia and robot horizontal plane [rad]. -51
% degrees was found to give the largest moment
phi = deg2rad(-51.8913);

% Tension is neglected as the force is assumed to be taken by
% the pulley and shaft
% Bolt is mounted in center of pulley (center of inner and outer diameter)
r = D_shaft/2 + (D_pulley-D_shaft)/4;
n = 2;
% Since contact on both sides
n_contact = n*2;
% Bolt diameter
Ab = pi*(d.^2)/4;
% tensile strength area, Polyfit with MATLAB, below equation 155
At = 0.7023*(d.^2) - 2.669*d + 9.963;

%% FORCES AND MOMENTS
% Pure moments applied to leg are assumed to be zero
% Tension is assumed to be carried by the pulley and shaft
% (Vy = My = 0)
% Equations 138, 139, 140, 141, 142
Vx = Fx;
Vy = 0;
Vz = Fz;
My = (Fz*cos(phi) - Fx*sin(phi))*R + 0;
Mz = 0;
```

```
%% PURE SHEAR
% Equations 143 to 148
Fsh_primary_pulley = sqrt(Vz^2 + Vy^2)/n_contact;
Fsh_secondary_pulley = My./(n_contact*r);
Fsh_pulley = sqrt(Fsh_primary_pulley.^2 + Fsh_secondary_pulley.^2);
SF_s_pulley = (0.58*Sy_b*(pi*(d.^2)./4))./Fsh_pulley;

Fsh_primary_tibia = sqrt(Vz^2 + Vy^2)/n_contact;
Fsh_secondary_tibia = My./(n_contact*r);
Fsh_tibia = sqrt(Fsh_primary_tibia.^2 + Fsh_secondary_tibia.^2);
SF_s_tibia = (0.58*Sy_b*(pi*(d.^2)./4))./Fsh_tibia;

%% BEARING FORCES
% Equations 149 to 151
SF_b_pulley = t_pulley*Sy_b*d./Fsh_pulley;

SF_b_tibia = t_tibia*Sy_b*d./Fsh_tibia;

%% EDGE SHEARING
% Equations 152, 153
SF_e_pulley = 0.577*Sy_pulley*t_pulley*ell./Fsh_pulley;

SF_e_tibia = 0.577*Sy_tibia*t_tibia*ell./Fsh_tibia;

%% TENSION
% Neglected

%% RESULTS
% Final d and ell
% See parametrization flowcharts for vectorized parts
[~,cols] = find(SF_s_pulley>2.5 & SF_b_pulley>2.5 & SF_e_pulley>2.5 &
    SF_s_tibia>2.5 & SF_b_tibia>2.5 & SF_e_tibia>2.5);
d = min(d(cols));
if (d < 2)
    d = 2;
end
ell = d*1.25;
[t_head,d_head,t_nut,d_nut,t_washer,od_washer,id_washer] =
    fasteners_general(d);
% Approximately a couple extra threads extending beyond nut
L_bolt = t_pulley + (t_tibia*2) + (t_nut*1.2) + 2*t_washer;
```

```
    %--------------------------------------%
    %% PRINT TO FILE
    %--------------------------------------%
    % FastenersKnee.txt contains dimensions for the bolts and nuts
    filePath = '..\Solidworks\Equations\FastenersKnee.txt';
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"diameter bolt"=',num2str(d),'\n'));
    fprintf(fid,strcat('"ell"=',num2str(ell),'\n'));
    fprintf(fid,strcat('"head thickness"=',num2str(t_head),'\n'));
    fprintf(fid,strcat('"head diameter"=',num2str(d_head),'\n'));
    fprintf(fid,strcat('"nut thickness"=',num2str(t_nut),'\n'));
    fprintf(fid,strcat('"nut diameter"=',num2str(d_nut),'\n'));
    fprintf(fid,strcat('"washer thickness"=',num2str(t_washer),'\n'));
    fprintf(fid,strcat('"washer outer diameter"=',num2str(od_washer),'\n'));
    fprintf(fid,strcat('"bolt length"=',num2str(L_bolt),'\n'));
    fclose(fid);

end
```

C.57 Fasteners Hip Knee

```
% From Analysis Report, Section 4.8 Fasteners
%% INPUTS
% D_pulley      = [mm] outer diameter of pulley (from inside of teeth)
% D_shaft       = [mm] diameter of shaft the pulley is mounted on
% t_pulley      = [mm] thickness of pulley
% t_plates      = [mm] thickness of tibia (one side of it)
% Torque        = [Nmm] torque seen at knee hip
%% OUTPUTS
% d             = [mm] bolt diameter
function [d,ell] = fasteners_kneeHip(D_pulley,D_shaft,t_pulley,t_plates,Torque
    )
    d = linspace(1,20,20);
    % recommended distance from edge is 1.5 times the diameter; we're using 1
        .25 times since SF is so high [mm]
    % https://www.engineeringexpress.com/wiki/steel-bolt-edge-distance-
        requirements/
    ell = d.*1.25;
```

```
% Material Properties
Sp = 650;       % [MPa] Proof strength
Sy_b = 720;     % [MPa] Yield strength bolt
Sut = 900;      % [MPa] Ultimate Tensile Strength bolt
Se = 140;       % [MPa] Endurance strength bolt
Sy_tibia = 276; % [MPa] Yield strength Aluminium 6061
Sy_pulley = 276;% [MPa] Yield strength Aluminium 6061
SF = 2.5;       % Safety factor for all but tension
Eb = 200;       % [MPa] Elastic Modulus bolt
Em = 276;       % [MPa] Elastic Modulus Aluminium 6061

%% GEOMETRIC PROPERTIES
% Tension is neglected as the force is assumed to be taken by the pulley
   and shaft
% Bolt is mounted in center of pulley (center of inner and outer diameter)
r = D_shaft/2 + (D_pulley-D_shaft)/4;
n = 2;
% Since contact on both sides
n_contact = n*2;
% Bolt diameter
Ab = pi*(d.^2)/4;
% tensile strength area, Polyfit with MATLAB, below equation 155
At = 0.7023*(d.^2) - 2.669*d + 9.963;

%% FORCES AND MOMENTS
% Pure moments applied to leg are assumed to be zero
% Tension is assumed to be carried by the pulley and shaft
% (Vy = My = 0)
% Equations 138, 139, 140, 141, 142
Vx = 0;
Vy = 0;
Vz = 0;
My = Torque;
Mz = 0;

%% PURE SHEAR
% Equations 143 to 148
Fsh_primary_pulley = sqrt(Vz^2 + Vy^2)/n_contact;
Fsh_secondary_pulley = My./(n_contact*r);
Fsh_pulley = sqrt(Fsh_primary_pulley.^2 + Fsh_secondary_pulley.^2);
SF_s_pulley = (0.58*Sy_b*(pi*(d.^2)./4))./Fsh_pulley;
```

```
Fsh_primary_tibia = sqrt(Vz^2 + Vy^2)/n_contact;
Fsh_secondary_tibia = My./(n_contact*r);
Fsh_tibia = sqrt(Fsh_primary_tibia.^2 + Fsh_secondary_tibia.^2);
SF_s_tibia = (0.58*Sy_b*(pi*(d.^2)./4))./Fsh_tibia;

%% BEARING FORCES
% Equations 149 to 151
SF_b_pulley = t_pulley*Sy_b*d./Fsh_pulley;

SF_b_tibia = t_plates*Sy_b*d./Fsh_tibia;

%% EDGE SHEARING
% Equations 152, 153
SF_e_pulley = 0.577*Sy_pulley*t_pulley*ell./Fsh_pulley;

SF_e_tibia = 0.577*Sy_tibia*t_plates*ell./Fsh_tibia;

%% TENSION
% Neglected

%% RESULTS
% Final d and ell
% See parametrization flowcharts for vectorized parts
[~,cols] = find(SF_s_pulley>2.5 & SF_b_pulley>2.5 & SF_e_pulley>2.5 &
    SF_s_tibia>2.5 & SF_b_tibia>2.5 & SF_e_tibia>2.5 );
d = min(d(cols));
if (d < 2)
    d = 2;
end
ell = d*1.25;
[t_head,d_head,t_nut,d_nut,t_washer,od_washer,id_washer] =
    fasteners_general(d);
% Approximately a couple extra threads extending beyond nut
L_bolt = t_pulley + (t_plates*2) + (t_nut*1.2) + 2*t_washer;

%--------------------------------------%
%% PRINT TO FILE
%--------------------------------------%
% FastenersKneeHip.txt contains dimensions for the bolts and nuts
filePath = '..\Solidworks\Equations\FastenersKneeHip.txt';
```

```
    fid = fopen(filePath,'wt');
    fprintf(fid,strcat('"diameter bolt"=',num2str(d),'\n'));
    fprintf(fid,strcat('"ell"=',num2str(ell),'\n'));
    fprintf(fid,strcat('"head thickness"=',num2str(t_head),'\n'));
    fprintf(fid,strcat('"head diameter"=',num2str(d_head),'\n'));
    fprintf(fid,strcat('"nut thickness"=',num2str(t_nut),'\n'));
    fprintf(fid,strcat('"nut diameter"=',num2str(d_nut),'\n'));
    fprintf(fid,strcat('"washer thickness"=',num2str(t_washer),'\n'));
    fprintf(fid,strcat('"washer outer diameter"=',num2str(od_washer),'\n'));
    fprintf(fid,strcat('"bolt length"=',num2str(L_bolt),'\n'));
    fclose(fid);

end
```

C.58 Forces

```
% Parameters
% P1    [mm] Position of point 1
% P2    [mm] Position of point 2
% P3    [mm] Position of point 3
% CG    [mm] Position of centre of gravity
% W     [kg] Weight of robot
% Slope [deg] Slope of ground

% Outputs
% N1    [N] Normal Force at P1
% N2    [N] Normal Force at P2
% N3    [N] Normal Force at P3

% Normal Force Calculation Function (Section 3.1.1 of Analysis Report)
function[N1,N2,N3] = forces(P1, P2, P3, CG, W, Slope)

FW = W*9.81;
matrixForce = [FW*cosd(Slope); (FW*cosd(Slope)*(CG(2)-P1(2))); FW*(cosd(Slope)
    *(CG(1)-P2(1))-sind(Slope)*(CG(3)-P2(3)))];
matrixDistance = [1 1 1; 0 (P2(2)-P1(2)) (P3(2)-P1(2)); (P1(1)-P2(1)) 0 (P3(1)
    -P2(1))];
matrix = transpose(inv(matrixDistance)*matrixForce);

N1=matrix(1);
```

```
N2=matrix(2);
N3=matrix(3);
```

C.59 Friction Forces

```
% Parameters
% A     [mm] Position of point 1
% C     [mm] Position of point 2
% D     [mm] Position of point 3
% CG    [mm] Position of centre of gravity
% W     [kg] Weight of robot
% Slope [deg] Slope of ground

%Ouputs
% ff1   [N] Friction force at foot 1
% ff2   [N] Friction force at foot 2

%Friction Calculation (Section 3.1.2 of Analysis Report)
function[ff1,ff2] = friction_Forces(A, C, D, CG, W, Slope)

FW = W*9.81;
ff1 = FW*sind(Slope)*abs(A(1)-CG(1))/abs(A(1)-D(1));
ff2 = FW*sind(Slope)-ff1;
```

C.60 Foot Position

```
function [x,y] = foot_position(theta,phi,r1,R)
    x = (r1*cos(theta)) + (R*cos(theta+phi));
    y = (r1*sin(theta)) + (R*sin(theta+phi));
end
```

C.61 Generated Harmonic Drives Function

```
clc;clear;
%% IMPORT DATA

hd_specs = importdata('HDSpecs.csv',',',1);
% Starts with largest denominator first, so must flip
```

```
data = flipud(hd_specs.data);

%% GENERATE LINEAR FUNCTIONS FOR PARAMETERS
% Col 1 = size
% Col 2 = torque (Nm)
% Col 3 = repeat torque limit (Nm)
% Col 4 = average input speed limit (rpm)
% Col 5 = moment of inertia I (x10^-4 kgm^2)
% Col 6 = moment of inertia J (x10^-5 kgfms^2)
% Col 7 = backdriving torque (Nm)
% Col 8 = starting torque (Nm)
% Col 9 = mass (kg)
% Col 10 = spline ring outer diameter (mm)
% Col 11 = total thickness (mm)
% Col 12 = spline ring thickness (mm)
% Col 13 = spline ring mounting diameter (mm)
% Col 14 = spline ring number of bolts
% Col 15 = spline ring bolt diameter (mm)
% Col 16 = flexspline outer diameter (mm)
% Col 17 = flexspline mounting diameter (mm)
% Col 18 = flexspline number of bolts
% Col 19 = flexspline bolt diameter (mm)
% Col 20 = bearing mounting diameter (mm)
% Col 21 = bearing number of bolts (mm)
% Col 22 = bearing bolt diameter (mm)

functions = zeros(size(data,2),2);
for i=1:1:size(data,2)
    P = polyfit(data(:,2),data(:,i),1);
    functions(i,1) = P(1);
    functions(i,2) = P(2);
end

%% DATA STORAGE
T = table(functions(:,1),functions(:,2),'VariableNames',{'Poly1','Poly2'},'
   RowNames',hd_specs.colheaders');
writetable(T,'hd_spec_functions.csv','WriteRowNames',true)

%% PLOTTING
% This is optional and solely for illustrative purposed
```

```
torque = data(:,2);
for i=1:1:size(hd_specs.data,2)
    figure(i)
    hold on
    plot(torque,data(:,i),'o');
    line([0,350],[functions(i,1)*0+functions(i,2), functions(i,1)*350+
        functions(i,2)],'Color','red');
    xlabel('Torque (mNm)');
    ylabel(hd_specs.colheaders(i));
    title(strcat(hd_specs.colheaders(i),' as a function of torque'));
    legend('Actual values', 'Linear Approximation');
end
```

C.62 Generated Motor Functions

```
clc;clear;
%% IMPORT DATA
motor_specs = importdata('MotorSpecs.csv',',',1);
% Starts with largest denominator first, so must flip
data_flipped = flipud(motor_specs.data);

%% GENERATED FUNCTIONS FOR PARAMETERS
% Col 2 = torque (mNm)
% Col 3 = Power (W)
% Col 4 = RPM (rpm)
% Col 5 = Current (A)
% Col 6 = Motor resistance (Ohms)
% Col 7 = Torque constant (mNm/A)
% Col 8 = Poles (#)
% Col 9 = Voltage (V)
% Col 10 = Mass (g)
% Col 11 = Rotor inertia (gcm^2)
% Col 12 = Outer Diameter (mm)
% Col 13 = Thickness without shaft (mm)
% Col 14 = Mounting Diameter (mm)
% Col 15 = Screw diameter (M--)
% Col 16 = Screw depth (mm)
% Col 17 = Maximum efficiency (fraction)

functions= zeros(size(motor_specs.data,2),2);
```

```
for i=1:1:size(motor_specs.data,2)
    P = polyfit(data_flipped(:,1),data_flipped(:,i),1);
    functions(i,1) = P(1); functions(i,2) = P(2);
end

%% DATA STORAGE
T = table(functions(:,1),functions(:,2),'VariableNames',{'Poly1','Poly2'},'
   RowNames',motor_specs.colheaders');
writetable(T,'motor_spec_functions.csv','WriteRowNames',true)

%% PLOTTING
% This is optional and solely for illustrative purposed

torque = data_flipped(:,1);
for i=1:1:size(motor_specs.data,2)
    figure(i)
    hold on
    plot(torque,data_flipped(:,i),'o');
    line([0,1500],[functions(i,1)*0+functions(i,2), functions(i,1)*1500+
       functions(i,2)],'Color','red');
    xlabel('Torque (mNm)');
    ylabel(motor_specs.colheaders(i));
    title(strcat(motor_specs.colheaders(i),' as a function of torque'));
    legend('Actual values', 'Linear Approximation');
end
```

C.63 Workspace

```
%% Purpose
% This script calculated the workspace for a given leg configuration, the
% largest achievable x and r range for movement, as well as the range in x
% during "regular walking". This is used to generate the reference values used
    in `linkages.m`

%% Leg configuration
% This was determined experimentally from Analysis Report, Section 3.4
alpha = deg2rad(69);
r1 = 100;
r2 = 100;
r3 = 300;
```

```
theta_min = 0;
% Higher thetas doesn't fit the chassis
theta_max = deg2rad(28);
% relative to linkage r1
psi_min = deg2rad(-22.5);
psi_max = deg2rad(22.5);

% Geometric properties derived following convention in workspace_variables.png
R = sqrt(r2^2 + r3^2 - (2*r2*r3*cos(alpha)));
beta = acos((r3^2 - R^2 - r2^2)/(-2*R*r2));
% Works in phi relative to thigh linkage, not horizontal
phi_min = psi_min - beta;
phi_max = psi_max - beta;

% Ground height is determined (quasi-arbitrarily) when theta is max and phi is
    min
[~,ground_height] = foot_position(theta_max,phi_min,r1,R);

%% Simulation Properties
steps = 1000;
precision = 1;

% Generate values of theta and phi over all 4 phases:
% phase 1: theta = 28, phi goes from min to max
% phase 2: theta goes from 28 to 0, phi constant
% phase 3: theta = 0, phi goes from max to min
% phase 4: theta goes from 0 to 28, phi constant (closes cycle)
theta1 = ones(1,steps)*theta_max;
theta2 = linspace(theta_max,theta_min,steps);
theta3 = ones(1,steps)*theta_min;
theta4 = linspace(theta_min,theta_max,steps);
phi1 = linspace(phi_min,phi_max,steps);
phi2 = ones(1,steps)*phi_max;
phi3 = linspace(phi_max,phi_min,steps);
phi4 = ones(1,steps)*phi_min;

[x1,y1] = foot_position(theta1,phi1,r1,R);
[x2,y2] = foot_position(theta2,phi2,r1,R);
[x3,y3] = foot_position(theta3,phi3,r1,R);
[x4,y4] = foot_position(theta4,phi4,r1,R);
```

```
% All x, y, theta and phi over all cycles
x = [x1,x2,x3,x4];
y = [y1,y2,y3,y4];
theta = [theta1,theta2,theta3,theta4];
phi = [phi1,phi2,phi3,phi4];

x_min = min(x);
x_max = max(x);
y_min = min(y);
y_max = max(y);

% Generate range of possible values of x and y in workspace
x_range = linspace(x_min,x_max,steps);
y_range = linspace(y_min,y_max,steps);

% y_max - y_min
y_reach = 0;
% value of x at location of y_reach
x_at_y_reach = 0;
y_min = 0;
y_max = 0;
% x_max - x_min
x_reach = 0;
% value of y at location of x_reach
y_at_x_reach = 0;
x_min = 0;
x_max = 0;

% Iterate over x, find maximum y_range
for i=1:length(x_range)
    [~,cols] = find((x>(x_range(i)-precision)) & (x<(x_range(i)+precision)));
    yVals = y(cols);
    yVals1 = repmat(yVals,[length(yVals),1]);
    yVals2 = repmat(yVals',[1,length(yVals)]);
    y_reach_i = max(abs(yVals1 - yVals2),[],'all');
    if (y_reach_i > y_reach)
        y_reach = y_reach_i;
        x_at_y_reach = x_range(i);
        y_min = min(yVals1,[],'all');
        y_max = max(yVals1,[],'all');
    end
```

```
end
% Iterate over y, find maximum x_range
for i=1:length(y_range)
    [~,cols] = find((y>(y_range(i)-precision)) & (y<(y_range(i)+precision)));
    xVals = x(cols);
    xVals1 = repmat(xVals,[length(xVals),1]);
    xVals2 = repmat(xVals',[1,length(xVals)]);
    x_reach_i = max(abs(xVals1 - xVals2),[],'all');
    if (x_reach_i > x_reach)
        x_reach = x_reach_i;
        y_at_x_reach = y_range(i);
        x_min = min(xVals1,[],'all');
        x_max = max(xVals1,[],'all');
    end
end

% Find x_range for the (pretty much arbitrarily) chosen walking height
% y=d=-ground_height [mm]; although x_range can be higher, this is the
   distance that'll
% be used for regular walking conditions. -ground_height is
% the foot height when theta = 28degrees, phi = min
[~,cols] = find((y>(ground_height-precision)) & (y<(ground_height+precision)))
   ;
WxVals = x(cols);
WxVals1 = repmat(WxVals,[length(WxVals),1]);
WxVals2 = repmat(WxVals',[1,length(WxVals)]);
% Wx_reach is the distance the leg can reach in x over any y in the workspace
Wx_reach = max(abs(WxVals1 - WxVals2),[],'all');
Wx_min = min(WxVals1,[],'all');
Wx_max = max(WxVals1,[],'all');

% This is what we use in linkages as x_reference and y_reference
fprintf('Reach in x: %f at y=%f\n',x_reach,y_at_x_reach);
fprintf('Reach in y: %f at x=%f\n',y_reach,x_at_y_reach);

figure;
title('Workspace and range visualization');
xlabel('x (mm)');
ylabel('y (mm)');
hold on;
plot(x1,y1,'-b','LineWidth',1.5);
```

```
plot(x2,y2,'-b','LineWidth',1.5,'HandleVisibility','off');
plot(x3,y3,'-b','LineWidth',1.5,'HandleVisibility','off');
plot(x4,y4,'-b','LineWidth',1.5,'HandleVisibility','off');
plot([x_at_y_reach x_at_y_reach],[y_min y_max],'--r','LineWidth',1);
plot([x_min x_max],[y_at_x_reach y_at_x_reach],'--r','LineWidth',1,'
    HandleVisibility','off');
plot([Wx_min Wx_max],[ground_height ground_height],'-.k','LineWidth',1);
plot(0,0,'b*','LineWidth',3,'HandleVisibility','off');
legend('Workspace','Maximum x-y range','Walking x range');
```

D Meeting Minutes

Group Minutes

Attendees:	**Absent:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault (MAA) Alexane Lahaie (AL) Mathieu Carroll (MC) Joshua O'Reilly (JO)		2019-09-04	CBY C08

Minute taker: Who is filling out this form? Marc-Andre Arsenault	**Chairperson:** Who is organising the meeting? Marc-Andre Arsenault

	Task What has to be done?	**Action** What action is required to get it done?	**Who** Who is responsible?	**Duration** How long will it take to complete?	**Status** Has the task been completed?
1	Group Contract	Finish typing contract	MC	1 Day	No
2	Picture	Take picture while being well dressed (2019-09-08)	All	1 Day	No
3	Literature Review - Existing Solution	Research existing solutions	All	1 Week	No
4	Literature Review - Standards, Code & Rules Review	Research standards, code and rules	All	1 Week	No
5	Literature Review - Relevant Sub-Systems	Research relevant sub-systems	All	1 Week	No

Next meeting **Chairperson:**	**Minute taker:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-09-08	CBY C08

Minutes	
The remaining sections of the literature review will be completed after the first week of research.	
Previous Friday lab attendance N/A	**Previous lecture attendance** N/A

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Joshua O'Reilly Alexane Lahaie Mathieu Carroll Marc-Andre Arsenault	None	2019-09-09	CRX-040

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Alexane Lahaie

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Research	- Pneumatic System and Actuators (Available solutions) - Hydraulic system equipments and drawings - Foot design - More legs design - Human interactions standards - More joint water proctection (rubber bellows)	all		not complete
2	Writing	see attached file red name is the task owner	red name next to subject		not complete
3	Dress nice	Take Picture	all		complete
4	Draft Report Review	The report must be substantially complete for friday for revision with Prof. Send email to prof to confirm.	all		not complete
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Alexane Lahaie	Marc-Andre Arsenault	2019-09-11	CBY B02

Minutes

A word document saved on the team's shared drive was created to separate the report by sections and tasks, and assigned ownership.
A picture was taken.
Multiple questions were asked to the teacher, the answers are in red in the report draft document
Adding references to zotero has changed, must be saved locally before uploading to overleaf.

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Mathieu Carroll Alexane Lahaie Joshua O'Reilly	N/A	2019-09-16	CRX 030

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Mathieu Carroll	Alexane Lahaie

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Identify Requirement Identify Constraints Identify Criteria	Identify - completed during the meeting	All	1h	Complete
2	Brainstorming Session	Brainstorm in team during the lab session	All	1h	Complete
3	Meeting tuesday (tomorrow)	Complete self-brainstorming session by designging different systems	All	5h	Incomplete
4	Meeting Professor Friday	Substantially complete design comparison	All	20h	Incomplete
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Joshua O'Reilly	Mathieu Carroll	2019-09-18	CBY-C08

Minutes

Discussion with prof. E. Lanteigne to confirm the following:
- Designing three complete solution is an option for the comparitive analysis
- Reviewed Requirements, Constraints, Criteria
- Reviewed no continuous joint constraint and bellow mounting possibilities
- Discussed salt damages

Meeting will be held on Tuesday at 1pm to brainstorm, each member is required to come prepared with drawings
Meeting will be held on Friday to show drawings to prof

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Alexane Lahaie Mathieu Carroll Joshua O'Reilly	N/A	2019-09-23	CBY-C011

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Joshua O'Reilly

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Concept Drawings	Complete Drawings Complete Annotation Complete	All	5h	Incomplete
2	Decision Analysis	Weighing of every category	All	15 min	Complete
3	Review Drawing	Review all drawing and give comment and recommended change	All	30 min	Complete
4	Separate task	Separate all remaining task	All	15 min	Complete
5	Complete given task		All	Variable	Incomplete

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-09-30	CBY-C011

Minutes

Weighing:
Power Concuption will determined by counting the number of total motors
Operating time is not required and was removed as it is partially included in solar panel and power consuption
Every criteria to be explained in the discussion
All criteria's were reviewed and approved by the team
Scoring was completed by the team during the meeting
Solar panel weighing an scoring was completed by the team during the meeting

To Do List:
All:
Write report explanation

Marc-Andre Arsenault:
L bracket drawing
Bearing on connecting rod
Rajouter des trous pour bellows
Annotation
3D Components

Alexane Lahaie:
Explanation, Discussion
Decision analysis criteria and weight explanation and discussion
Joshua O'Reilly:
Add component to design concept drawing
U-Bracket - L Brackt Design cut view
Electronic Cost
Battery Selection and Cost
Mathieu Carroll:
Slap in place solar panel's design and drawings
Annotation
Clamp for case design

Previous Friday lab attendance	Previous lecture attendance
All	All

Group Minutes

Attendees:	**Absent:**	**Date & Time:**	**Venue:**

Minute taker: Who is filling out this form?	**Chairperson:** Who is organising the meeting?

	Task What has to be done?	**Action** What action is required to get it done?	**Who** Who is responsible?	**Duration** How long will it take to complete?	**Status** Has the task been completed?
1					
2					
3					
4					
5					

Next meeting **Chairperson:**	**Minute taker:**	**Date & Time:**	**Venue:**

Minutes	
Previous Friday lab attendance	**Previous lecture attendance**

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Mathieu Carroll Joshua O'Reilly Alexane Lahaie	N/A	2019-09-30	CBY-C011

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Marc-Andre Arsenault

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Discuss work completed	discuss	All	15min	yes
2	Complete work	Complete work as described by the work separation list in word doc	All	5h	no
3					
4					
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Joshua O'Reilly	Marc-Andre Arsenault	2019-10-07	CBY-C011

Minutes

Marc-Andre Arsenault:
Discussed equation
Dicsussed graph
No feedback provided
Mathieu Carroll:
Created multiple FBDs' to simplify drawings
Assumption Negligle les poids desjambes
Feedback provided
Others work were changed and modified during the meeting

List of what to do:
1. See word doc

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Group Minutes

Attendees:	**Absent:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault Joshua O'Reilly Mathieu Carroll Alexane Lahaie	N/A	2019-10-07	CBY-C02

Minute taker: Who is filling out this form? Marc-Andre Arsenault	**Chairperson:** Who is organising the meeting? Marc-Andre Arsenault

	Task What has to be done?	**Action** What action is required to get it done?	**Who** Who is responsible?	**Duration** How long will it take to complete?	**Status** Has the task been completed?
1	Add drawings to word doc	Complete drawing and add them to the word doc	All	30 min	Complete
2	Review Word doc	Review word doc	All	1h	Complete
3	Discuss what's left to do	Discuss what's left to do	All	1h	Complete
4	Review report guidelines	Review and fix report guidelines	All	15min	Complete
5	Complete task list	All have to complete the task list in word doc	All	5 h	Incomplete

Next meeting **Chairperson:**	**Minute taker:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-10-14	CBY-C02

Minutes	
Everyone's task was distributed and noted in the word doc, these finalization are to be completed for wednesday	
Previous Friday lab attendance All	**Previous lecture attendance** All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Joshua O'Reilly Alexane Lahaie Mathieu Carroll	N/A	2019-10-21	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Alexane Lahaie

	Task (What has to be done?)	Action (What action is required to get it done?)	Who (Who is responsible?)	Duration (How long will it take to complete?)	Status (Has the task been completed?)
1	Update Discussion	Everyone to discuss what they did and where they are at with their task	All	30min	Yes
2	Ask question to prof	Ask question to prof	All	30 min	Yes
3	Split of task	Split task depending on priorities	All	2h	Yes
4					
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Mathieu Carroll	Marc-Andre Arsenault	2019-10-28	CBY-C02

Minutes

During the meeting:
1. Discuss work accomplished since the last meeting.
2. Discuss tasks not-completed since the last meeting.
3. Review action items and tasks to be completed after the meeting

Meeting minutes content:
1. Summarize completed work
2. List previous tasks that have not been completed in the prescribed timeline
3. Specify task reassignments
4. List additional tasks completed but not listed in previous minutes
5. Specify additional out-of-class meeting attendance

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Mathieu Carroll Alexane Lahaie Joshua O'Reilly	N/A	2019-10-28	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Alexane Lahaie

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Battery Math Parametrization	Talk to prof regarding issue and possible solution	All	15 min	Complete
2	SolidWorks	Should we start sketches in solidworks	All	15min	Complete
3	Task/Analysis Priority	Devide task and analysis, and coordinate	All	1h	Complete
4	Assigned Task	Finish assigned task	All	1 Week	Complete
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-11-04	CBY-C02

Minutes	
Question Prof: 1. Bearing details (size only) 2. Battery Usage 3. Details in Solidworks (Bellow, Gear teeth)	
Previous Friday lab attendance N/A	**Previous lecture attendance** N/A

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Joshua O'Reilly Alexane Lahaie Mathieu Carroll	N/A	2019-11-04	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Marc-Andre Arsenault

	Task (What has to be done?)	Action (What action is required to get it done?)	Who (Who is responsible?)	Duration (How long will it take to complete?)	Status (Has the task been completed?)
1	Progress report	- Discuss progress - Redistribute task	All	2h	Complete
2					
3					
4					
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-11-11	CBY-C02

Minutes	
Good progress.	
Previous Friday lab attendance All	**Previous lecture attendance** All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Alexane Lahaie Mathieu Carroll Joshua O'Reilly	N/A	2019-11-11	CBY-C11

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Marc-Andre Arsenault

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Discuss Updates	Everyone share what was done	All	1h	Yes
2	Ask question to prof	Ask question to prof	All	15 min	Yes
3	Battery weight and usage	All determine and confirm equation	All	1h	Yes
4					
5					

Next meeting

Chairperson:	Minute taker:	Date & Time:	Venue:
Alexane Lahaie	Marc-Andre Arsenault	2019-11-18	CBY-C11

Minutes

Alexane:
- Shaft analysis is complete, missing example
- Bellows Safety Factor de 3

Josh:
- Modelling section
- Ratio Leg

Mathieu:
- Priorite sur forces poulies,
- Bolts and Hip Plates
- Springs in Matlab

Marc-Andre:
- Look at solar power sun
- Transfer limb to square area
- Torsion

Questions for prof:
- Bolts torque: whatever is good
- Calcul, parametrization, ordre: pas de paraetrization
- battery: every day is independent

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Group Minutes

Attendees:	Absent:	Date & Time:	Venue:
Marc-Andre Arsenault Mathieu Carroll Joshua O'Reilly Alexane Lahaie	N/A	2019-11-18	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Joshua O'Reilly

	Task What has to be done?	Action What action is required to get it done?	Who Who is responsible?	Duration How long will it take to complete?	Status Has the task been completed?
1	Analysis Report	What is left? Redistribute Task	All	30 min	yes
2	Capstone Report	Where to start Redistribute Task	All	30 min	yes
3					
4					
5					

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Mathieu Carroll	Marc-Andre Arsenault	2019-11-25	CBY-C02

Minutes

Prof Comments:
- Bellow clamp
- remove material in bellow holder

The report was read beginning to end together to correct mistake and identify missing information
Questions were answered by the prof
The remaining task were distributed

Previous Friday lab attendance	**Previous lecture attendance**

Group Minutes

Attendees:	**Absent:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault Mathieu Carroll Alexane Lahaie Joshua O'Reilly	N/A	2019-11-25	CBY-C02

Minute taker: Who is filling out this form?	Marc-Andre Arsenault	**Chairperson:** Who is organising the meeting?	Marc-Andre Arsenault

	Task What has to be done?	**Action** What action is required to get it done?	**Who** Who is responsible?	**Duration** How long will it take to complete?	**Status** Has the task been completed?
1	Harmonic Drive and Flange Collar	Review and resolve, ask prof Use the parametrized HD and create new adaptor	All	30 min	Yes
2	Spring	Ask prof	All	30 min	Yes
3	Bushin/Hip plate, length	Decide parametrization	All	15 min	Yes
4					
5	Next Steps	Discuss next step and schedule	All	30 min	Yes

Next meeting **Chairperson:**	**Minute taker:**	**Date & Time:**	**Venue:**
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-12-02	CBY-C02

Minutes

Ask prof:
1. Harmonic Drive/hubs
Use custom parts,
2. Torsion Spring (Pulley Spacer)
Besoin d'un spacey/pulley type
3. SW directory path:
No exper, reccomend to trial before hand
4. SW parametrized and non parametrized in the submission file
N'ouvre pas le SW non parametrize

Meeting Tuesday Morning

Previous Friday lab attendance	**Previous lecture attendance**
All	All

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
Joshua O'Reilly Team Waterfront 2 B	Alexane Lahaie Mathieu Carroll Marc-Andre Arsenault	2019-09-04	CBY B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Joshua O'Reilly

Minutes

- Joshua met with Team Waterfront 2 B to obtain contact information
- Patrick Richer (Prof) confirmed team meetings are held on mondays and wednesdays during the laboratory period as their are no team contract with meeting obligations. Presence are taken during laboratories which will reflect attendence to Waterfront team meetings

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Joshua O'Reilly	Marc-Andre Arsenault	2019-09-08	CBY C011

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR-2A WR-2B WR-3A		2019-09-11	CBY B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Abel Jacob

Minutes

- Use the facebook group for team communication
- Added the team lead of WR-3A so that she can add the remainder of her team members.
- Power requirements : solar
- Concepts will be discussed in the following meeting
- All meetings are on the Wednesday lab

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Abel Jacob	Marc-Andre Arsenault	2019-09-18	CBY B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR-2A WR-2B WR-3A	N/A	2019-09-18	CBY-B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Abel Jacob

Minutes

Designs:
- Most likely a robotic arm mounted top or side plate

Litter:
- Mounting on top of the body
- Sensors for litter and wiring required for locomotion teams
- Sensors basic dimensions required for locomotion teams
- Probably height parametrization for litter size, not wide
- Hatch door type for litter compartement
- Tank will be exposed for easy removal of trash from human

Arm:
- Litter picker from a relative frontal position
- Reach and weight required for locomotion teams

Locomotion:
- Wide body
- Small height (close to the ground)

Next meeting

Chairperson:	Minute taker:	Date & Time:	Venue:
Alexane Lahaie	Marc-Andre Arsenault	2019-09-25	CBY-B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A	None	2019-09-25	CBY B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Alexane Lahaie

Minutes

1. Show Concepts

WR2B:
Bin with lid
Arm mounted on top (flat surface)
Lid and arm mounted separately
Agreement : WR2B to built their own mount and groove mechanism, will be bolted to the case of WR2A and WR3A for easy integration.

WR2A:
Showed concept - no issues determined by WR2B so far

WR3A:
Showed concept - no issues determined by WR2B so far

2. Concept properties
Weight: Not known
Location: top surface (parrallel to the ground), the bin is located behind the arm
Size: Not known for arm, 10x20x25 cm for the bin

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Ahmed Taimah	Marc-Andre Arsenault	2019-10-02	CBY-B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A		2019-10-02	CBY B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Ahmed Taimah

Minutes

Important Updates:

WR2A : Confirmed None

WR2B : Dimension changed : 20x25x10 (cm), max height from the ground to top of chassis 20 cm

WR3A : Confirmed None

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Abel Jacob	Marc-Andre Arsenault	2019-10-09	CBY B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A	N/A	2019-10-09	CBY-B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Abel Jacob

Minutes

WR2B:
- Increase litter tank size 30x35x15(H) cm
- Confirmed robot arm mounting 6x6 cm
- Weight: 5.4 kg for arm, and litter tank bin :1 kg - 2kg, garbage weight : max is 5kg
- Power analysis not complete (Need current drawn)
- Arm reach 39 cm below its mounting point

WR2A:
- Nothing to say

WR3A:
- Nothing to say

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Mathieu Carroll	Marc-Andre Arsenault	2019-10-23	CBY-B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A		2019-10-23	CBY-B02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Mathieu Carroll

Minutes

WR2A:

WR2B:
Litter box parametrization: 30 x 35 x15
Relation to be shared, most likely a ratio of height
Arm Length Change: 10 cm + 10 cm = 20 cm total
Same maximum reach
End Effector: 15 cm
Vertical : 10 cm
Moving: 10 cm + 10 cm
Arms Power Consumtion(Preliminary): 0.003 HP x 4 (motors), Voltage 12 DC, 0.1 Amps (on full load) x 4 (motors)
Might use two more motor
Hinge motors: https://hobbyking.com/en_us/power-hd-lw-20mg-servo.html
End effector motor: 0.4 Amp (most likely)
Maximum 8 motors total total
Worm gear so none backdriveable
WR3A:

Next meeting

Chairperson:	Minute taker:	Date & Time:	Venue:
Eleni Sabourin	Marc-Andre Arsenault	2019-10-30	CBY-B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR-2A WR-2B WR-3A	N/A	2019-10-30	CBY-C011

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Eleni Sabourin

Minutes

WR-2B:
- No change

WR-2A:
- Mounting Box size: 5 cm distance between arm and litter
- Mounting Arm size: 10cm x 10cm

WR- 3A
- No chane

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Abel Jacob	Marc-Andre Arsenault	2019-11-06	CBY-B02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A		2019-11-06	CBY-C011

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Abel Jacob

Minutes

WR2A: no change
WR2B: Nothing Change, no power calculation, not backdriveable
WR3A: no change

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-11-13	CBY-C011

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR-2A WR-2B WR-3A	None	2019-11-13	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Marc-Andre Arsenault

Minutes

No Change
No Question
Nothing to discuss

Meeting complete

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Marc-Andre Arsenault	Marc-Andre Arsenault	2019-11-20	CBY-C02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A	N/A	2019-11-20	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Marc-Andre Arsenault

Minutes

WR2B:
- Everything stay the same
- To provide the parametrizatio for the chassis holding mount
- Min 25 x 20 x 10 , L X W X H

Other comment: Poster will be completed as a team. Poster December 4th TBD.

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Galadrielle Michaud	Marc-Andre Arsenault	2019-11-27	CBY-C02

Team/Partner Minutes

Attendees:	Absent:	Date & Time:	Venue:
WR2A WR2B WR3A	Kareem	2019-11-27	CBY-C02

Minute taker: (Who is filling out this form?)	Chairperson: (Who is organising the meeting?)
Marc-Andre Arsenault	Galadrielle

Minutes

Next Wednesday schedule is not a monday schedule

WR3A: questions were answered on the FB group chat

Changes from WR2B:
Maximum height off the ground: 20 cm off the ground
WR2A and WR3A must allow space for clip system and mounting plate
Minimum limb length 10 cm, 10 cm, 5 cm
End effector is 30 cm length

Poster Day is Friday at 2:01pm
Arm cad due date Tuesday

Next meeting Chairperson:	Minute taker:	Date & Time:	Venue:
Abel Jacob	Marc-Andre Arsenault	2019-12-04	CBY-C02

E Data Sheets

Ordering Code

CSD - 20 - 100 - 2A - GR - SP

Table 063-1

Series	Size	Ratio*				Model	Special specification
CSD	14	50	80	100	—	2A-GR = component type (2A-R for Size 14, 17)	Blank= Standard product SP= Special specification code BB= Big Bore
	17	50	80	100	120		
	20	50	80	100	120		
	25	50	80	100	120		
	32	50	80	100	120		
	40	50	80	100	120		
	50	50	80	100	120		

* The reduction ratio value is based on the following configuration:
Input: wave generator, fixed: circular spline, output: flexspline

Technical Data

CSD-2A Component Set

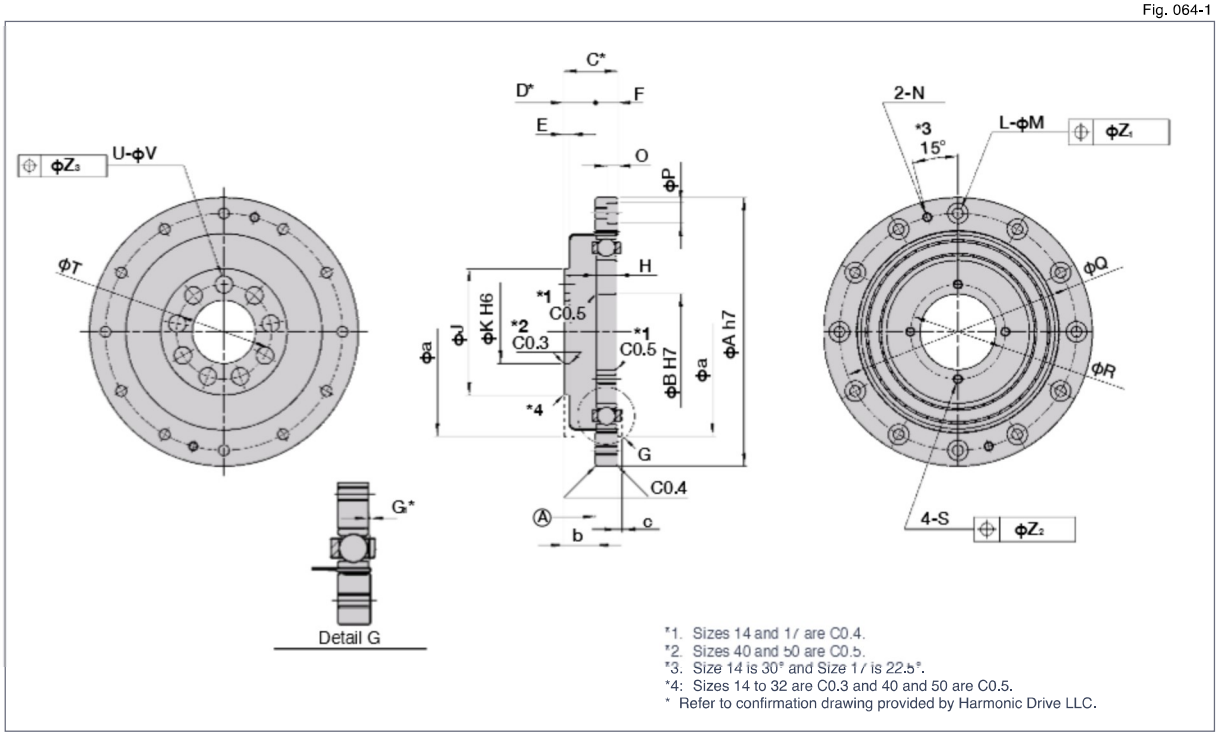
Size	Gear ratio	Rated torque at input speed 2000rpm		Limit for repeated peak torque		Limit for average torque		Limit for momentary peak torque		Maximum input speed (rpm)		Limit for average input speed (rpm)		Moment of inertia	
		Nm	kgfm	Nm	kgfm	Nm	kgfm	Nm	kgfm	Oil	Grease	Oil	Grease	I x 10^{-4}kgm^2	J x 10^{-3}kgfms^2
14	50	3.7	0.38	12	1.2	4.8	0.49	24	2.4	14000	8500	6500	3500	0.021	0.021
	80	5.4	0.55	16	1.6	7.7	0.79	31	3.2						
	100	5.4	0.55	19	1.9	7.7	0.79	31	3.2						
17	50	11	1.1	23	2.3	18	1.8	48	4.9	10000	7300	6500	3500	0.054	0.055
	80	15	1.5	29	3.0	19	1.9	55	5.6						
	100	16	1.6	37	3.8	27	2.8	55	5.6						
	120	16	1.6	37	3.8	27	2.8	55	5.6						
20	50	17	1.7	39	4.0	24	2.4	69	7.0	10000	6500	6500	3500	0.090	0.092
	80	24	2.4	51	5.2	33	3.4	76 (65)	7.7 (6.6)						
	100	28	2.9	57	5.8	34	3.5	76 (65)	7.7 (6.6)						
	120	28	2.9	60	6.1	34	3.5	76 (65)	7.7 (6.6)						
25	50	27	2.8	69	7.0	38	3.9	127	13	7500	5600	5600	3500	0.282	0.288
	80	44	4.5	96	9.8	60	6.1	152 (135)	15 (14)						
	100	47	4.8	110	11	75	7.6	152 (135)	15 (14)						
	120	47	4.8	117	12	75	7.6	152 (135)	15 (14)						
32	50	53	5.4	151	15	75	7.6	268	27	7000	4800	4600	3500	1.09	1.11
	80	83	8.5	213	22	117	12	359 (331)	37 (34)						
	100	96	9.8	233	24	151	15	359 (331)	37 (34)						
	120	96	9.8	247	25	151	15	359 (331)	37 (34)						
40	50	96	9.8	281	29	137	14	480	49	5600	4000	3600	3000	2.85	2.91
	80	144	15	364	37	198	20	685 (580)	70 (59)						
	100	185	19	398	41	260	27	694 (580)	71 (59)						
	120	205	21	432	44	315	32	694 (580)	71 (59)						
50	50	172	18	500	51	247	25	1000	102	4500	3500	3000	2500	8.61	8.78
	80	260	27	659	67	363	37	1300	133						
	100	329	34	686	70	466	48	1440 (1315)	147 (134)						
	120	370	38	756	77	569	58	1441	147 (134)						

1. Moment of inertia: I = $^1/_4$ GD^2
2. *The maximum allowable momentary torque value marked by an asterisk(*) is restricted by the tightening torque of the flexspline.
3. The parenthesized value indicates the value when the bore of the flexspline has the maximum value (BB type).
4. See “Rating Table Definitions” on Page 12 for details of the terms.
5. When the max allowable momentary torque is expected to be applied, see “Bolt tightening of the flexspline” on p. 75.

Outline Dimensions

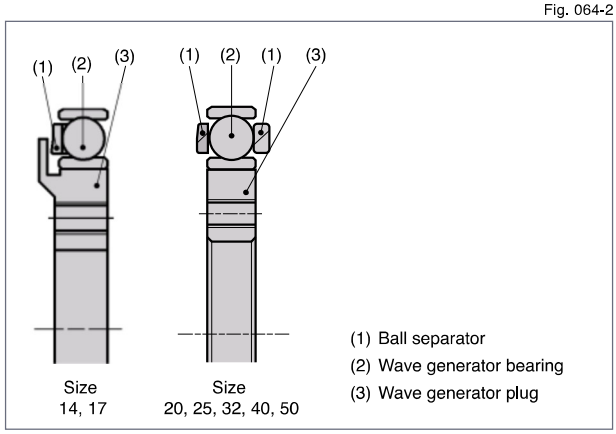
You can download the CAD files from our website: harmonicdrive.net

Fig. 064-1

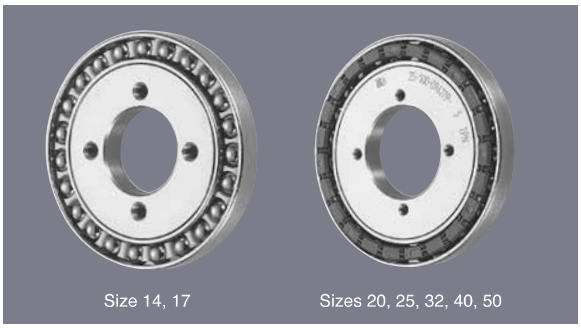


■ Structure and shape of the wave generator

Fig. 064-2



There is a difference in appearance of the the ball separator depending on the size.



Dimensions

Table 065-1
Unit : mm

Symbol \ Size		14	17	20	25	32	40	50
ϕA h7		50 $^{0}_{-0.025}$	60 $^{0}_{-0.030}$	70 $^{0}_{-0.030}$	85 $^{0}_{-0.035}$	110 $^{0}_{-0.035}$	135 $^{0}_{-0.040}$	170 $^{0}_{-0.040}$
ϕB H7		11 $^{+0.018}_{0}$	15 $^{+0.018}_{0}$	20 $^{+0.021}_{0}$	24 $^{+0.021}_{0}$	32 $^{+0.025}_{0}$	40 $^{+0.025}_{0}$	50 $^{+0.025}_{0}$
C＊		11	12.5	14	17	22	27	33
D＊		6.5 $^{+0.2}_{0}$	7.5 $^{+0.2}_{0}$	8 $^{+0.3}_{0}$	10 $^{+0.3}_{0}$	13 $^{+0.3}_{0}$	16 $^{+0.3}_{0}$	19.5 $^{+0.3}_{0}$
E		1.4	1.7	2	2	2.5	3	3.5
F		4.5	5	6	7	9	11	13.5
G_1＊		0.3 $^{+0.2}_{0}$	0.3 $^{+0.2}_{0}$	0.3 $^{+0.2}_{0}$	0.4 $^{+0.2}_{0}$	0.5 $^{+0.2}_{0}$	0.6 $^{+0.2}_{0}$	0.8 $^{+0.2}_{0}$
H		4 $^{0}_{-0.1}$	5 $^{0}_{-0.1}$	5.2 $^{0}_{-0.1}$	6.3 $^{0}_{-0.1}$	8.6 $^{0}_{-0.1}$	10.3 $^{0}_{-0.1}$	12.7 $^{0}_{-0.1}$
ϕJ		23	27.2	32	40	52	64	80
ϕK H6	Standard	11 $^{+0.011}_{0}$	11 $^{+0.011}_{0}$	16 $^{+0.011}_{0}$	20 $^{+0.013}_{0}$	30 $^{+0.013}_{0}$	32 $^{+0.016}_{0}$	44 $^{+0.016}_{0}$
	BB spec.	11 $^{+0.011}_{0}$	11 $^{+0.011}_{0}$	20 $^{+0.013}_{0}$	24 $^{+0.013}_{0}$	32 $^{+0.016}_{0}$	40 $^{+0.016}_{0}$	50 $^{+0.016}_{0}$
L		6	8	12	12	12	12	12
ϕM		3.4	3.4	3.4	3.4	4.5	5.5	6.6
N		M3	M3	M3	M3	M4	M5	M6
O		—	—	3.3	3.3	4.4	5.4	6.5
ϕP		—	—	6.5	6.5	8	9.5	11
ϕQ		44	54	62	75	100	120	150
ϕR		17	21	26	30	40	50	60
S		M3	M3	M3	M3	M4	M5	M6
ϕT	Standard	17	19.5	24	30	41	48	62
	BB spec.	17	19.5	26	32	42	52	65
U	Standard	9	8	9	9	11	10	11
	BB spec.	9	8	12	12	14	14	14
ϕV	Standard	3.4	4.5	4.5	5.5	6.6	9	11
	BB spec.	3.4	4.5	3.4	4.5	5.5	6.6	9
ϕZ_1		0.2	0.2	0.2	0.2	0.25	0.25	0.3
ϕZ_2		0.25	0.25	0.2	0.2	0.25	0.25	0.3
ϕZ_3	Standard	0.2	0.25	0.25	0.25	0.3	0.5	0.5
	BB spec.	0.2	0.25	0.2	0.25	0.25	0.3	0.5
Minimum housing clearance	ϕa	38	45	53	66	86	106	133
	b	6.5	7.5	8	10	13	16	19.5
	c	1	1	1.5	1.5	2	2.5	3.5
Mass (kg)		0.06	0.10	0.13	0.24	0.51	0.92	1.9

(Note) Standard dimension for size 14 and 17 is the maximum bore.

- Surface A is the recommended mounting surface.
- The following dimensions can be modified to accommodate customer-specific requirements.

  Wave Generator: B
  Flexspline: U and V
  Circular Spline: L and M

- *C, D and G_1 values indicate relative position of individual gearing components (wave generator, flexpline, circular spline). Please strictly adhere to these values when designing your housing and mating parts.
- Due to the deformation of the Flexspline during operation, it is necessary to provide a minimum housing clearance, dimensions ϕa, b, c

The wave generator, flexspline, and circular spline are not assembled when delivered.

Positional accuracy

See "Engineering data" for a description of terms.

Table 066-1

Ratio		14	17	20	25	32	40	50
Positional Accuracy	$\times 10^{-4}$rad	4.4	4.4	2.9	2.9	2.9	2.9	2.9
	arc min	1.5	1.5	1.0	1.0	1.0	1.0	1.0

Hysteresis loss

See "Engineering data" for a description of terms.

Table 066-2

Ratio \ Size		14	17	20	25	32	40	50
50	$\times 10^{-4}$rad	7.3	5.8	5.8	5.8	5.8	5.8	5.8
	arc min	2.5	2.0	2.0	2.0	2.0	2.0	2.0
80 or more	$\times 10^{-4}$rad	5.8	2.9	2.9	2.9	2.9	2.9	2.9
	arc min	2.0	1.0	1.0	1.0	1.0	1.0	1.0

Torsional stiffness

See "Engineering data" for a description of terms.

Table 066-3

Symbol \ Size			14	17	20	25	32	40	50
T_1		Nm	2.0	3.9	7.0	14	29	54	108
		kgfm	0.2	0.4	0.7	1.4	3.0	5.5	11
T_2		Nm	6.9	12	25	48	108	196	382
		kgfm	0.7	1.2	2.5	4.9	11	20	39
Reduction ratio 50	K_1	$\times 10^4$Nm/rad	0.29	0.67	1.1	2.0	4.7	8.8	17
		kgfm/arc min	0.085	0.2	0.32	0.6	1.4	2.6	5.0
	K_2	$\times 10^4$Nm/rad	0.37	0.88	1.3	2.7	6.1	11	21
		kgfm/arc min	0.11	0.26	0.4	0.8	1.8	3.4	6.3
	K_3	$\times 10^4$Nm/rad	0.47	1.2	2.0	3.7	8.4	15	30
		kgfm/arc min	0.14	0.34	0.6	1.1	2.5	4.5	9
	θ_1	$\times 10^{-4}$rad	6.9	5.8	6.4	7.0	6.2	6.1	6.4
		arc min	2.4	2.0	2.2	2.4	2.1	2.1	2.2
	θ_2	$\times 10^{-4}$rad	19	14	19	18	18	18	18
		arc min	6.4	4.6	6.6	6.1	6.1	5.9	6.2
Reduction ratio 80 or more	K_1	$\times 10^4$Nm/rad	0.4	0.84	1.3	2.7	6.1	11	21
		kgfm/arc min	0.12	0.25	0.4	0.8	1.8	3.2	6.3
	K_2	$\times 10^4$Nm/rad	0.44	0.94	1.7	3.7	7.8	14	29
		kgfm/arc min	0.13	0.28	0.5	1.1	2.3	4.2	8.5
	K_3	$\times 10^4$Nm/rad	0.61	1.3	2.5	4.7	11	20	37
		kgfm/arc min	0.18	0.39	0.75	1.4	3.3	5.8	11
	θ	$\times 10^{-4}$rad	5.0	4.6	5.4	5.2	4.8	4.9	5.1
		arc min	1.7	1.6	1.8	1.8	1.7	1.7	1.7
	θ	$\times 10^{-4}$rad	16	13	15	13	14	14	13
		arc min	5.4	4.3	5.0	4.5	4.8	4.8	4.6

* The values in this table are reference values. The minimum value is approximately 80% of the displayed value.

Starting torque

See "Engineering data" for a description of terms. Please use as reference values; the values vary based on use conditions.

Table 067-1
Unit: Ncm

Ratio \ Size	14	17	20	25	32	40	50
50	3.7	5.7	7.3	14	28	50	94
80	2.7	3.8	4.8	8.8	19	32	63
100	2.4	3.3	4.3	7.9	18	29	56
120	—	3.1	3.8	7.2	16	27	53

Backdriving torque

See "Engineering data" for a description of terms. Please use as reference values; the values vary based on use conditions.

Table 067-2
Unit: Nm

Ratio \ Size	14	17	20	25	32	40	50
50	2.5	3.8	4.4	8.3	17	30	57
80	2.6	3.7	4.9	8.8	19	32	62
100	3.1	4.1	5.2	9.6	21	35	67
120	—	4.5	5.7	11	22	38	74

Ratcheting torque

See "Engineering data" for a description of terms.

Table 067-3
Unit: Nm

Ratio \ Size	14	17	20	25	32	40	50
50	60	105	150	315	685	1260	2590
80	75	140	245	475	980	1960	3780
100	55	110	180	350	700	1470	2870
120	—	80	165	325	685	1330	2660

Buckling torque

See "Engineering data" for a description of terms.

Table 067-4
Unit: Nm

Size	14	17	20	25	32	40	50
All ratios	190	330	560	1000	2200	4300	8000

No-load running torque

No-load running torque is the torque which is required to rotate the input side (high speed side), when there is no load on the output side (low speed side).

Measurement condition

Table 068-1

Ratio 100:1			
Lubricant	Grease lubrication	Name	Harmonic Grease SK-1A (size 20 or larger)
			Harmonic Grease SK-2 (size 14, 17)
		Quantity	Recommended quantity (See page 71)
Torque value is measured after 2 hours at 2000rpm input.			

* Contact us for oil lubrication.

■ Compensation value in each ratio

No load running torque of the gear varies with ratio.
The graphs indicate a value for ratio 100.
For other gear ratios, add the compensation values from table on the right.

Compensation coefficient for no-load running torque

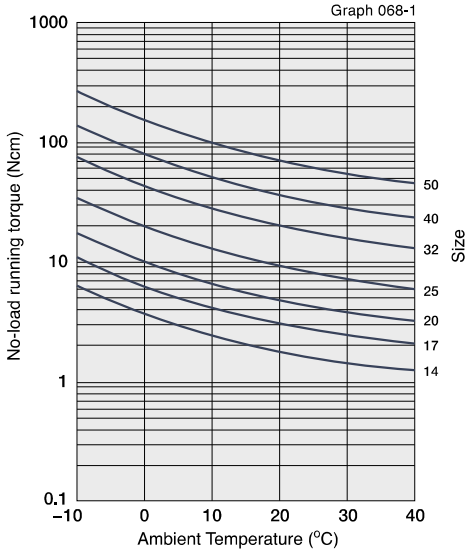
Table 068-2
Unit: Ncm

Size \ Ratio	50
14	+0.56
17	+0.95
20	+1.4
25	+2.6
32	+5.4
40	+9.6
50	+18

■ No-load running torque for a reduction ratio of 100

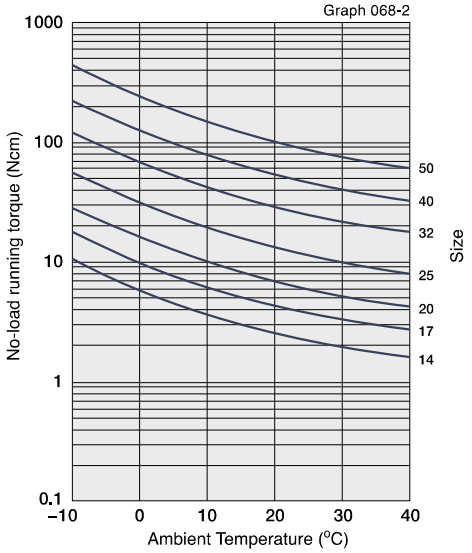
Input rotational speed: 500rpm

Graph 068-1



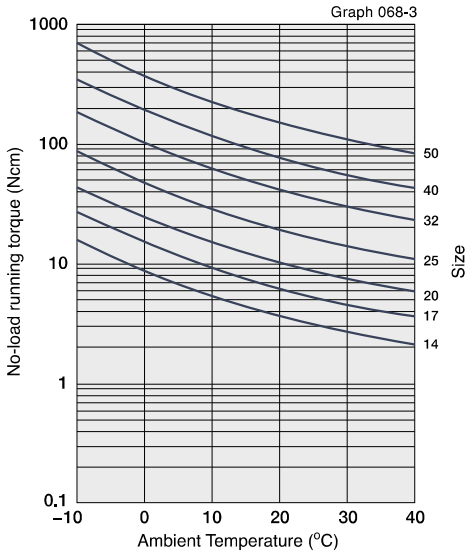
Input rotational speed: 1000rpm

Graph 068-2



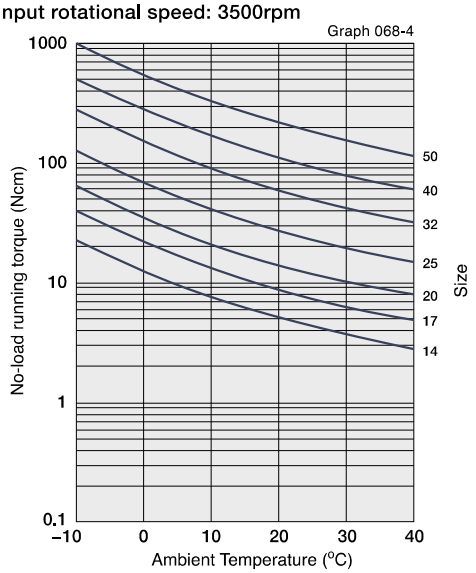
Input rotational speed: 2000rpm

Graph 068-3



Input rotational speed: 3500rpm

Graph 068-4



* The values in this graph are average value "$\bar{X}$".

Efficiency

The efficiency varies depending on the following conditions.

- Reduction ratio
- Input rotational speed
- Load torque
- Temperature
- Lubrication (Type and quantity)

Measurement condition

Table 069-1

Installation	Based on recommended tolerance		
Load torque	The rated torque shown in the rating table (see page 63)		
* When load torque is smaller than rated torque, the efficiency value is lowered. See efficiency compensation coefficient below.			
Lubricant	Grease lubrication	Name	Harmonic Grease SK-1A (size 20 or larger)
			Harmonic Grease SK-2 (size 14, 17)
		Quantity	Recommended quantity (see page 71)

* Contact us for oil lubrication.

■ Efficiency compensation coefficient

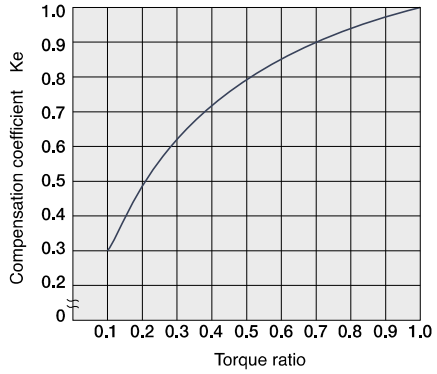
If the load torque is lower than the rated torque, the efficiency value decreases. Calculate the compensation coefficient Ke from Graph 069-1 to calculate the efficiency using the following calculation example.

* Efficiency Compensation coefficient Ke=1 holds when the load torque is greater than the rated torque.

Efficiency compensation coefficient

Graph 069-1

Torque ratio α is the value of load torque/rated torque (Rating table: page 063).

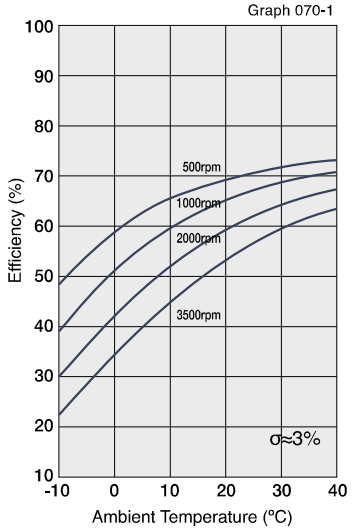


■ Efficiency at rated torque

Reduction ratio 50:1

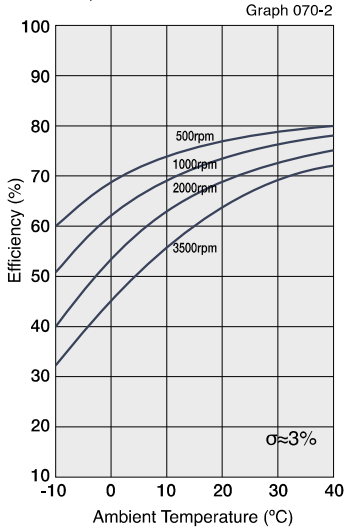
Size 14

Graph 070-1



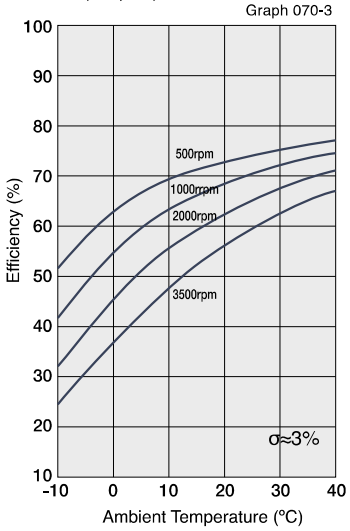
Size 17, 20

Graph 070-2



Size 25, 32, 40, 50

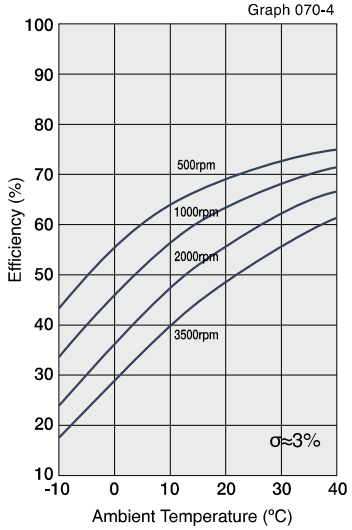
Graph 070-3



Reduction ratio 80, 100, 120:1

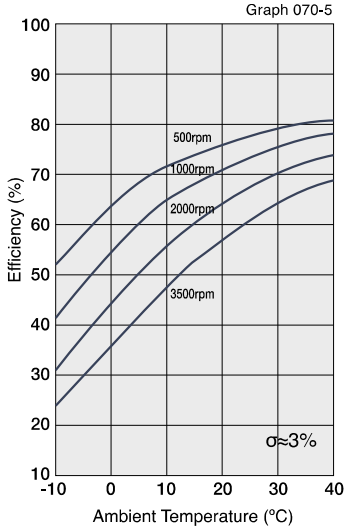
Size 14

Graph 070-4



Size 17, 20, 25, 32, 40, 50

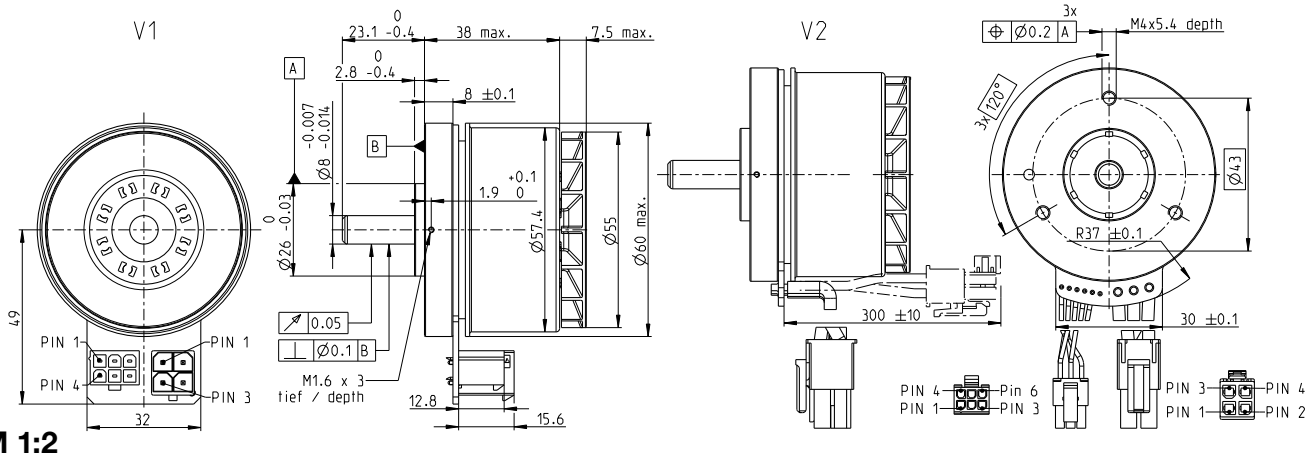
Graph 070-5



EC 60 flat ∅60 mm, brushless, 200 Watt

Ventilated

NEW



M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers			
V1 with Hall sensors	625860	**614949**	625861
V2 with Hall sensors and cables	647696	642221	647697

Motor Data					
Values at nominal voltage					
1 Nominal voltage	V	12	24	48	
2 No load speed	rpm	3760	4300	4020	
3 No load current	mA	815	497	224	
4 Nominal speed	rpm	2790	3240	3020	
5 Nominal torque (max. continuous torque)	mNm	492	536	577	
6 Nominal current (max. continuous current)	A	15.1	9.28	4.6	
7 Stall torque[1]	mNm	3340	4300	4870	
8 Stall current	A	111	81.9	43.2	
9 Max. efficiency	%	83.8	85.2	86.3	
Characteristics					
10 Terminal resistance phase to phase	Ω	0.108	0.293	1.11	
11 Terminal inductance phase to phase	mH	0.0911	0.279	1.28	
12 Torque constant	mNm/A	30	52.5	113	
13 Speed constant	rpm/V	318	182	84.8	
14 Speed/torque gradient	rpm/mNm	1.14	1.01	0.837	
15 Mechanical time constant	ms	9.95	8.83	9.29	
16 Rotor inertia	gcm²	832	832	832	

Specifications

Thermal data

17	Thermal resistance housing-ambient	1.22 K/W
18	Thermal resistance winding-housing	0.843 K/W
19	Thermal time constant winding	9.19 s
20	Thermal time constant motor	44 s
21	Ambient temperature	-40…+100°C
22	Max. winding temperature	+125°C

Mechanical data (preloaded ball bearings)

23	Max. speed		6000 rpm
24	Axial play at axial load	< 12.0 N	0 mm
		> 12.0 N	0.14 mm
25	Radial play		preloaded
26	Max. axial load (dynamic)		12 N
27	Max. force for press fits (static)		170 N
	(static, shaft supported)		8000 N
28	Max. radial load, 5 mm from flange		112 N

Other specifications

29	Number of pole pairs	7
30	Number of phases	3
31	Weight of motor	360 g

Values listed in the table are nominal.

Connection V1		**V2** (sensors, AWG 24)
Pin 1	Hall sensor1	Hall sensor1
Pin 2	Hall sensor 2	Hall sensor 2
Pin 3	V_{Hall} 4.5…24 VDC	Hall sensor 3
Pin 4	Hall sensor 3	GND
Pin 5	GND	V_{Hall} 4.5…24 VDC
Pin 6	N.C.	N.C.
		V2 (Motor, AWG 14)
Pin 1	Motor winding 1	Motor winding 1
Pin 2	Motor winding 3	Motor winding 2
Pin 3	Motor winding 2	Motor winding 3
Pin 4		N.C.

Wiring diagram for Hall sensors see p. 47

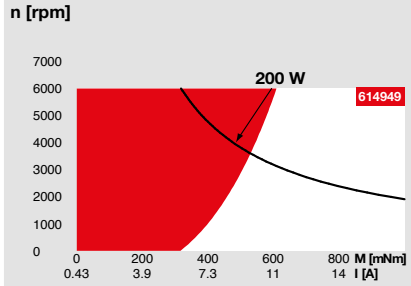
Connector	**Part number**	
Molex	46015-0606	43025-0600
Molex	76829-0104	171692-0104

Connection cable for V1

Connection cable Universal, L = 500 mm **651900**

[1]Calculation does not include saturation effect (p. 57/162)

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

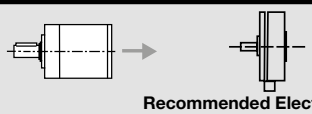
Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Details on catalog page 36

Planetary Gearhead
∅52 mm
4 - 30 Nm
Page 367



Recommended Electronics:

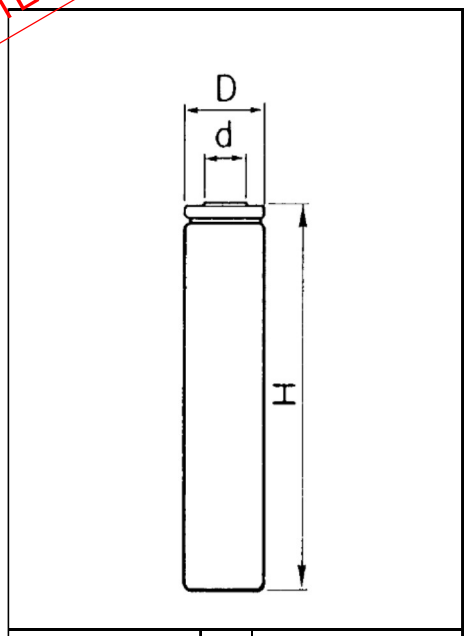
Notes	Page **36**
ESCON Module 50/5	455
ESCON Mod. 50/8 (HE)	456
ESCON 50/5	457
ESCON 70/10	457
DEC Module 50/5	459

High Capacity type

TENTATIVE

Cell Type NCR18650B

Specification



Dimensions(Typ.) of Bare Cell		
	H	64.93mm
	D	18.2mm
	d	7.9 mm

Discharged State after Assembling

Rated Capacity *at 20deg.C		Min.3200mAh
Nominal Capacity *at 25deg.C		Min.3250mAh
		Typ.3350mAh
Nominal Voltage		3.6V
Charging Method		Constant Current -Constant Voltage
Charging Voltage		4.2V
Charging Current		Std.1625mA
Charging Time		4.0hrs.
Ambient Temperature	Charge	10～+45℃
	Discharge	-20～+60℃
	Storage	-20～+50℃
Weight (Max.)		47.5g
Dimensions (Max.)*	(D)	18.25mm *without tube
		18.50mm *with tube
	(H)	65.10mm *without tube
		65.30mm *with tube
Volumetric Energy Density		676Wh/l
Gravimetric Energy Density		243Wh/kg

*Maximum size without tube

Panasonic ideas for life

6S–16S (22.2V–74V, Adjustable) 10(max. BMS Battery Management Syst for Lithium-ion Battery Pack with Balancing and Communication

100A

Availability: **In Stock**

Email to a Friend

Notes

* Port(s)

-- Please Select --

* Maximum continuous discharge current (A)

* Peak discharge current (A/sec)

* Number of cells in series

Options

- [ ] Communication Interface
- [ ] Electrical Switch

- [ ] Temperature Switch
- [ ] Thermistor
- [ ] Heat-Sink
- [ ] Diode
- [ ] Other

* Required Fields

Qty: 2

A (22.2 V–59.2 V) battery management system (BMS) for use with l ion batteries

DESCRIPTION

SPECIFICATIONS

SPECIFICATIONS TABLE

BASIC	Options	Balancer, CANbus, I2C, RS232, RS485, Bus
	Country of Manufacture	China
	Series (compatible with)	6S, 7S, 8S, 9S, 10S, 11S, 12S, 13S, 14S, S, 16S
SIZE	Length, max.	134.50 mm
	Width, max.	140.50 mm
	Height, max.	5.00 mm
VOLTAGE	Voltage (compatible with)	22.2V, 25.9V, 29.6V, 33.3V, 37V, 40.7V, 4 48.1V, 51.8V, 55.5V, 59.2V
CURRENT	Max. continuous discharge current	100.00 A
PROTECTION	Protection	over-charge, over-current, over-discharg ver-temperature, short-circuit

µIMU, µAHRS, µINS+RTK, µINS-Dual

Calibrated Inertial Systems with Onboard GPS

INERTIALSENSE
micro navigation systems

Overview

The **µIMU**™ is a miniature calibrated sensor module consisting of an Inertial Measurement Unit (IMU), magnetometer, barometer, and onboard L1 GPS (GNSS) receiver. Data out includes angular rate, linear acceleration, magnetic field, barometric altitude, and GPS.

The **µAHRS**™ is an Attitude Heading Reference System (AHRS) that includes all functionality of the µIMU™ and fuses IMU and magnetometer data to estimate roll, pitch, and heading.

The **µINS+RTK**™ is a GPS (GNSS) aided Inertial Navigation System (GPS-INS) module that includes all functionality of the µAHRS™ and provides orientation, velocity, and position. Sensor data from MEMs gyros, accelerometers, magnetometers, barometric pressure, and GPS/GNSS is fused to provide optimal estimation.

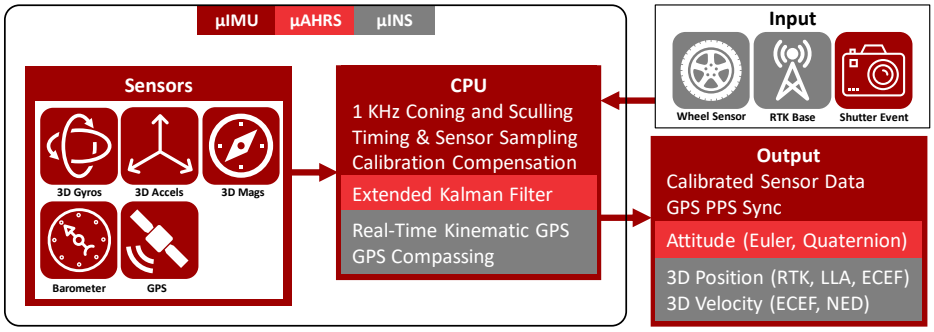
The **µINS Dual**™ is a GPS (GNSS) aided Inertial Navigation System (GPS-INS) module that includes all functionality of the µAHRS™ and provides orientation, velocity, and position. By utilizing Dual GPS antennas, accurate heading can be determined in environments that are challenging for a magnetometer.

Applications

- Drone Navigation
- Unmanned Vehicle Payloads
- Aerial Survey
- Stabilized Platforms
- Antenna and Camera Pointing
- First Responder and Personnel Tracking
- Health, Fitness, and Sport Monitors
- Robotics and Ground Vehicles
- Maritime

Features

- ***NEW*** – Rugged Enclosure
- ***NEW*** – Precision RTK GNSS
- ***NEW*** – Dual GNSS Compassing
- Up to 1KHz IMU, 500Hz INS Update Rate
- Attitude (Roll, Pitch, Yaw, Quaternions), Velocity, and Position UTC Time Synchronized
- Dual Redundant IMUs Calibrated for Bias, Scale Factor, and Cross-Axis Alignment
- -40°C to 85°C Sensor Temperature Calibration
- On-Board u-Blox L1 GPS (GNSS) Receiver(s)
- Onboard World Magnetic and Gravity Models
- Binary and NMEA ASCII Protocol
- Barometric Pressure and Humidity
- Strobe In/Out Data Sync (Camera Shutter Event)
- Fast Integration with SDK and Example Software
- Data Logging (SDK and Application Software)



Specifications

Performance (µINS, µAHRS)	Typ
Roll/Pitch (RMS)	0.1°
Static Heading w/magnetometer (RMS)	2.0°
Static Heading w/Dual Compass (RMS)	0.3°
µINS Dynamic Heading** (RMS)	0.3°

*Position is stationary. **Requires GPS lock with periodic >0.8 m/s² acceleration and >2 m/s velocity.

Performance	Typ	RTK-GPS
Horizontal Position (w/ SBAS)	2.5 m (2.0 m)	3 cm
Vertical Position	2.5 m	5 cm
Velocity (GPS and INS)	0.05 m/s	
Angular Resolution	0.05°	
Operation Limits		
Velocity	500 m/s	
Altitude (GPS)	50 Km	
Altitude (Barometric)	10 Km	
Startup Time	0.8 sec	
GPS Lock Time		
Hot Start	1 sec	10 sec
Cold Start	30 sec	2-4 min
GNSS Receiver Sensitivity		
Tracking & Navigation	-164 dBm	
Cold Start	-147 dBm	
Hot Start	-156 dBm	
GPS Update Rate	5 Hz	
Max Output Data Rate (IMU, INS)	1 KHz, 500 Hz	
GPS_PPS Time Sync. Pulse (10% duty cycle)	1 Hz	
RMS Accuracy	30 ns	
99% Accuracy	60 ns	
IMU signal latency	4 ms	
Humidity Sensor Relative Accuracy	±3 %	

Absolute Maximum Ratings	MAX	
Acceleration	10,000 g	
Storage Temperature (µINS)	-45 to 85 °C	Barometer limitation
Overpressure	600 kPa	
ESD rating	± 2 kV	Human body model
Soldering Temperature	Hand Solder ONLY. Do NOT solder reflow.	

Sensors	IMU - Gyros	IMU - Accels	Mags	Pressure
Operating Range	±2000 °/sec	±16 g	±4800 µT	30–120 kPa
Bias Repeatability	< 0.2 °/sec	< 5 mg		
In-Run Bias Stability	< 10 °/hr	< 40 µg		
Random Walk	0.15 °/√hr	0.07 m/s/√hr		
Non-linearity	< 0.1 % FS	< 0.5 % FS		
Noise Density	0.01 °/s/√Hz	300 µg/√Hz		Pa/√Hz
Bias Error over -40C to 85C	0.7 °/s RMS	0.4 m/s² RMS		
Max Output Rate	1 KHz	1 KHz	100 Hz	50 Hz
Bandwidth	250 Hz	218 Hz	50 Hz	5 Hz
Alignment Error	0.05°	0.05°	0.05°	
Sampling Rate	8 KHz	4 KHz	100 Hz	250 Hz
Resolution	*0.0076 °/sec	*122 µg	0.6 µT	0.0016 kPa
*1KHz resolution after oversampling				(13 cm)

Data Output	µIMU™	µAHRS™	µINS™
GPS, GPS Raw, UTC Time	•	•	•
IMU (Gyro & Accelerometer)	•	•	•
Magnetometer & Barometer	•	•	•
Attitude (Quaternions, Euler, DCM)		•	•
Inertial Velocity & Position			•

Electrical (µINS, µAHRS, µIMU)	Min	Typ	Max	Units
Power Draw (w/o GPS ant.)				
µIMU @ 1KHz		340		mW
µINS, µAHRS @ 250Hz		412		mW
Supply Voltage (Vcc)	3.0	3.3	3.6	V
GPS VBAT Voltage	1.4	3.3	3.6	V
GPS VBAT Current @ 3.0V		15		µA
GPS Antenna Supply w/o load (2.8V w/ 10mA load)*		2.9		V
GPS Antenna Supply Current*			300	mA
I/O Pin MAX Voltage Range	-0.5		3.6	V
Total Output Current, All Pins			120	mA
I/O Pin Input low-level	0.99			V
I/O Pin Input high-level	2.31	3.3	3.6	V
I/O Pin Output high-level		3.3		V
STROBE pulse duration	1			ms
STROBE pulse period	5			ms
Rising Slope of VIN**	2.4			V/ms

*A 10 Ohm current limiting resistor sits inline between voltage supply and antenna.
**The supply rising slope must be higher than minimum rating for proper function.

Electrical (µINS with Rugged/EVB)	Min	Typ	Max	Units
Supply Voltage (VIN)	4.0		20	V
µINS with Rugged or EVB				
Current Draw @ 5V, 250Hz*		125		mA
Power Consumption @250Hz*		625		mW
Power Consumption @100Hz*		575		mW
Power Consumption – Dual		1100		mW

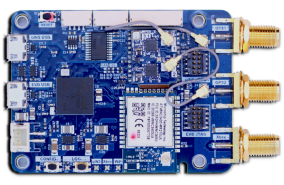
*Navigation filter update rate.

Mechanical (µINS, µAHRS, µIMU)		
µINS		**Units**
Size	16.5 x 12.6 x 4.6	mm
Weight	1.3	grams

Mechanical (Rugged µINS)		Units	Conditions
Size	25.4 x 25.4 x 11.2 35.9 x 25.4 x 11.2	mm	W/o mounting tabs W/ mounting tabs
Distance Between Mounting Tab Holes	30.836	mm	
Weight	10.5	grams	
Connectors	Main: Harwin# G125-MV11205L1P, GPS A/B: MMCX		

Communications	
Interface	TTL, SPI
Rugged Interface (IS-RUG-1.x)	USB, TTL, RS232, RS485, CAN*
Max Baud Rate:	
TTL, RS422, RS485	3 Mbps
RS232	500 Kbps

*Available in future firmware update.



Development Kits available on our website.

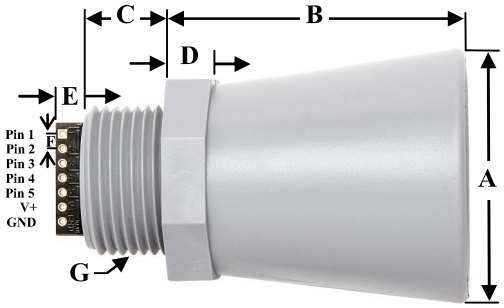
Performance Changes when Selecting a Non-Full Horn Package

When selecting a HRXL-MaxSonar-WR sensor without the full horn the sensor will experience the following performance changes:

- The sensor will have a wider beam shape for the first meter.
- The sensor may be less accurate by an additional +/- 0.5%.
- The sensor may have a dead zone from 0mm–500mm.
- The sensor may have worse performance to small or soft targets.
- The sensor may experience decreased noise immunity when ranging to small, soft, angled, or distant targets.

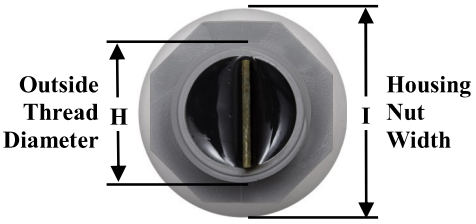
Mechanical Dimensions

Full Horn

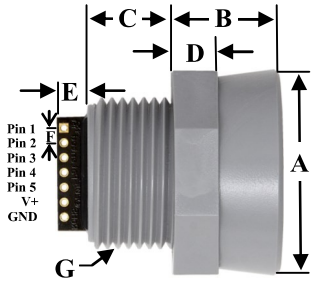


A	1.72” dia.	43.8 mm dia.
B	2.00”	50.7 mm
C	0.58”	14.4 mm
D	0.31”	7.9 mm
E	0.23”	5.8 mm
F	0.1”	2.54 mm
G	3/4”-14 NPS	
H	1.032” dia.	26.2 mm dia.
I	1.37”	34.8 mm
Weight, 1.76 oz., 50 grams		

Values Are Nominal

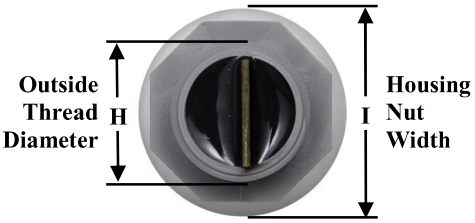


Compact Housing

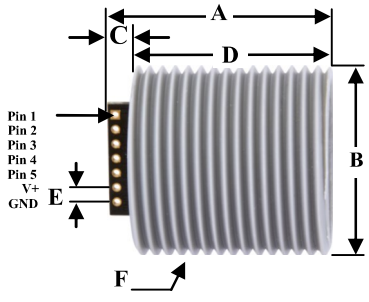


A	1.37” dia.	34.7 mm dia.
B	0.70”	17.9 mm
C	0.57”	14.4 mm
D	0.31”	7.9 mm
E	0.23”	5.8 mm
F	0.1”	2.54 mm
G	3/4”-14 NPS	
H	1.032” dia.	26.2 mm dia.
I	1.37”	34.8 mm
Weight, 1.23 oz., 32 grams		

Values Are Nominal

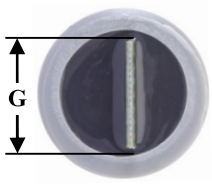


1” NPS Pipe Threading



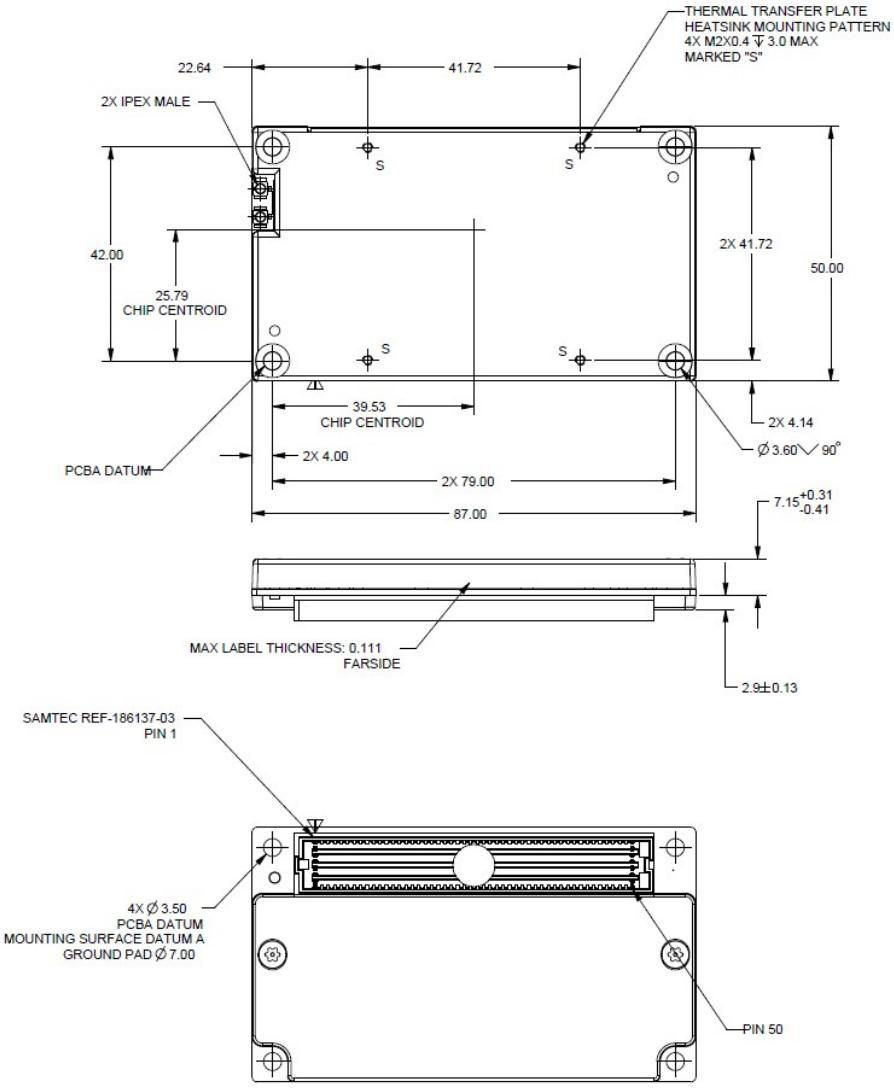
A	1.52”	38.5 mm
B	1.29” dia.	33.0 mm dia.
C	0.22”	5.5 mm
D	1.30”	33.1 mm
E	0.10”	2.54 mm
F	1” - NPS	
G	0.78”	19.81 mm
Weight, 1.23 oz., 35 grams		

Values Are Nominal



6.5 Package Drawings and Dimensions

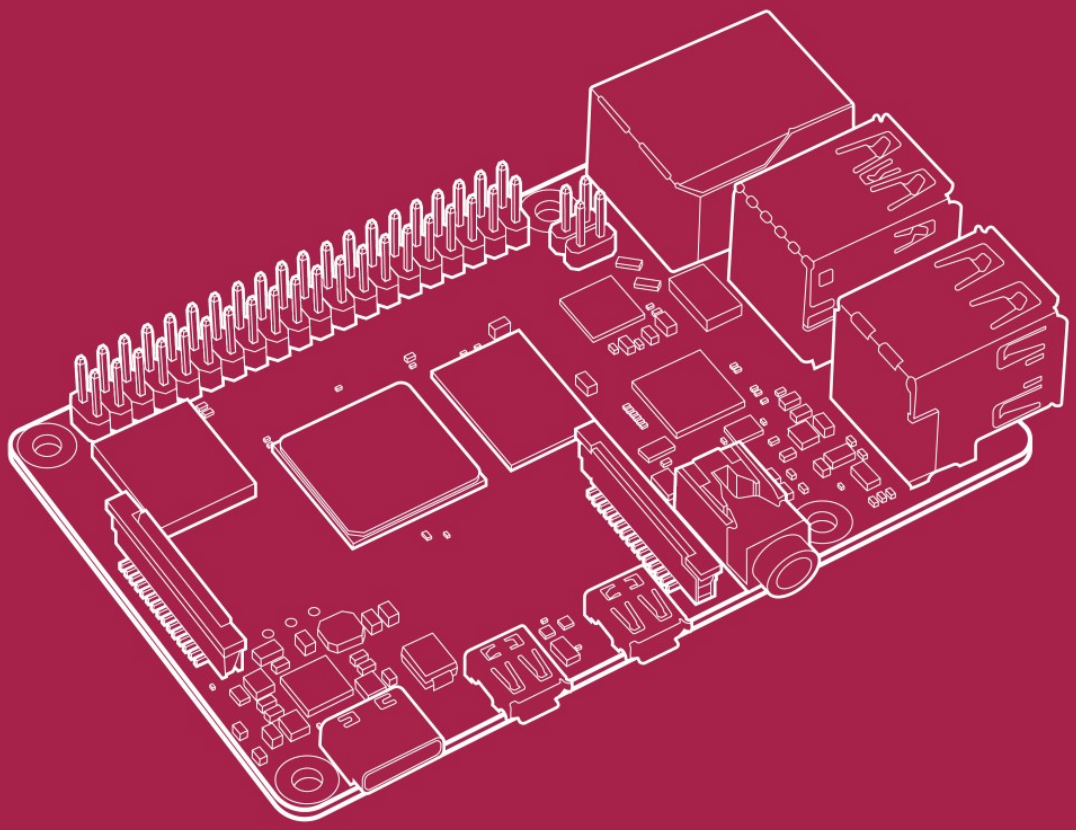
Figure 7 Jetson TX2 Module Package Outline with Dimensions



NOTES

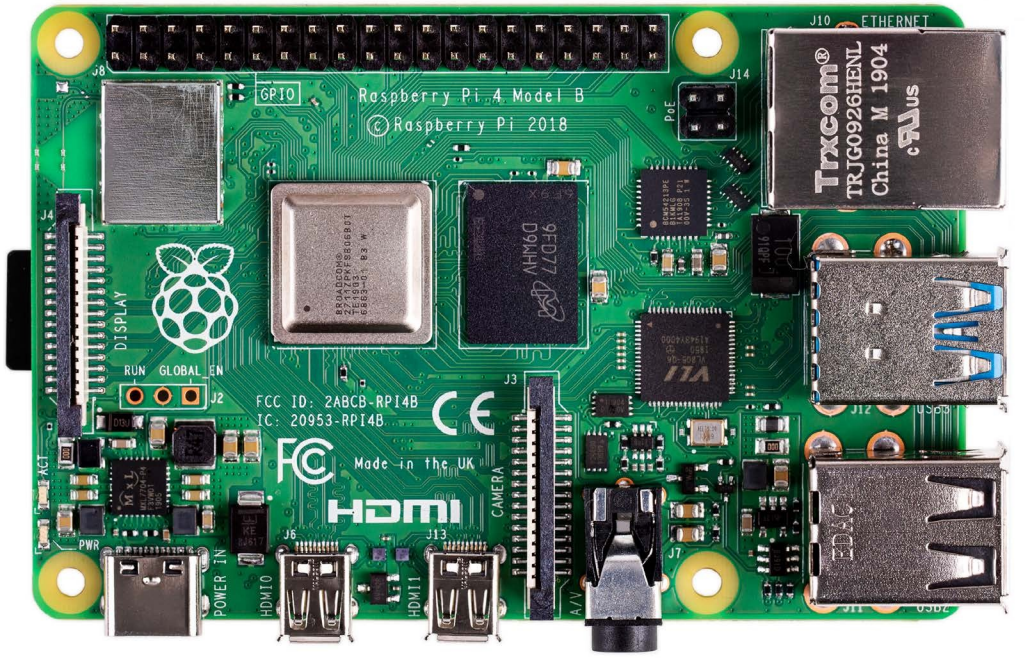
- Carrier Board connector location & mounting holes should match the module dimensions shown in figure above.
- Carrier Board components limited to 2.5mm under outline of the module. This assumes the use of A SEARAY mating connector, “SAMTEC REF-186138-02” (SEAM-50-02.0-S-08-2-A-K-TR) or Molex receptacle (PN:45970-001). If the connector used is taller, the max component height may change accordingly.
- Keepout area on Carrier Board for standoffs depends on diameter of standoffs used. The Jetson module carrier board uses 6MM diameter round keepout areas surrounding the four mounting holes. These areas on the PCB should be GND with no soldermask. See the Jetson TX2 Carrier Board layout for reference.
- All dimensions are in millimeters unless otherwise specified.
- Tolerances are: .X ± 0.25, .XX ± 0.10, Angles ± 1°
- Mass: 88 ±1.7% Grams
- Thermal transfer plate and bottom stiffener finish: Clear Chemfilm per MIL-C-5541-E Class 3

Raspberry Pi 4 Computer
Model B



Published in June 2019
by Raspberry Pi Trading Ltd.

Overview



Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

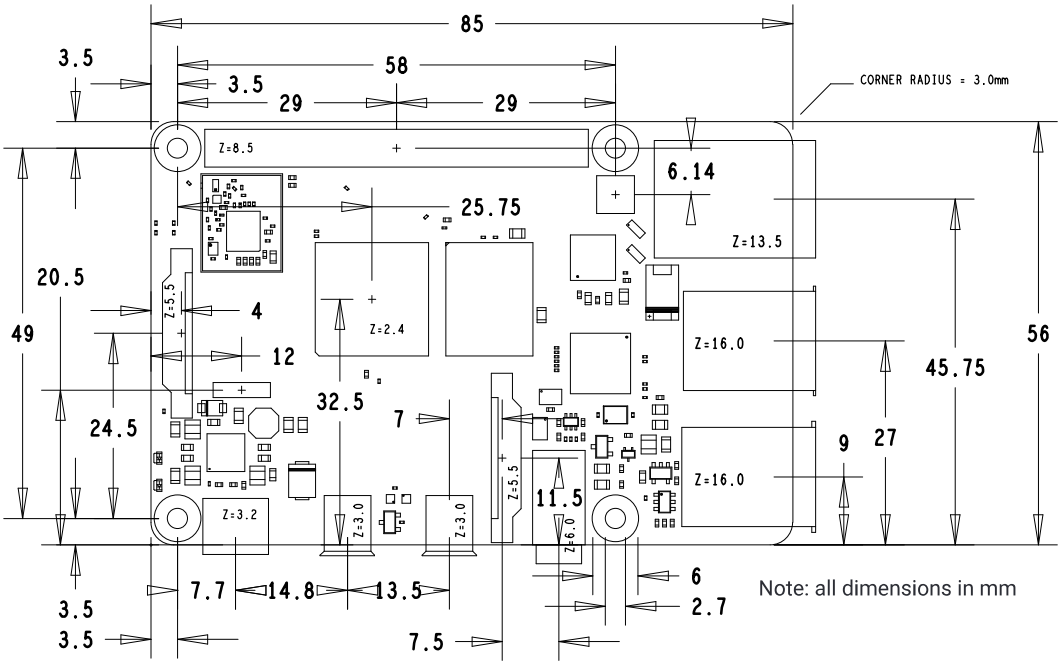
This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

Specification

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	1GB, 2GB or 4GB LPDDR4 (depending on model)
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD card support:	Micro SD card slot for loading operating system and data storage
Input power:	5V DC via USB-C connector (minimum 3A[1]) 5V DC via GPIO header (minimum 3A[1]) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature 0–50ºC
Compliance:	For a full list of local and regional product approvals, please visit https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md
Production lifetime:	The Raspberry Pi 4 Model B will remain in production until at least January 2026.

Physical Specifications



WARNINGS

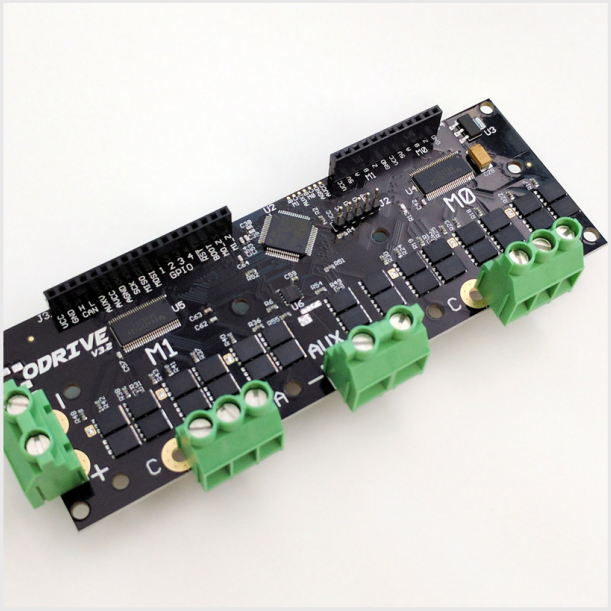
- This product should only be connected to an external power supply rated at 5V/3A DC or 5.1V/ 3A DC minimum[1]. Any external power supply used with the Raspberry Pi 4 Model B shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- This product should be placed on a stable, flat, non-conductive surface in use and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance and result in damage to the unit and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors and mice when used in conjunction with the Raspberry Pi.
- Where peripherals are connected that do not include the cable or connector, the cable or connector must offer adequate insulation and operation in order that the relevant performance and safety requirements are met.

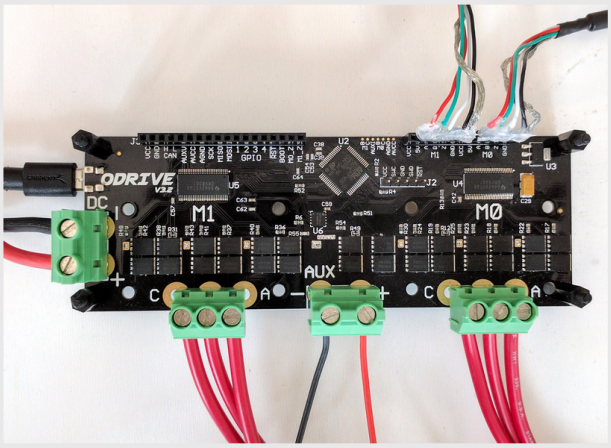
SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product please observe the following:

- Do not expose to water, moisture or place on a conductive surface whilst in operation.
- Do not expose it to heat from any source; Raspberry Pi 4 Model B is designed for reliable operation at normal ambient room temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Avoid handling the printed circuit board whilst it is powered and only handle by the edges to minimise the risk of electrostatic discharge damage.

[1] A good quality 2.5A power supply can be used if downstream USB peripherals consume less than 500mA in total.





KEY SPECS

- **Controls two motors.**
- 24V and 48V versions available.
- Peak current >100A per motor.
- Continuous current depends on cooling: Details.
- Encoder feedback for arbitrarily precise movements.
- Supports two braking modes:
  - Brake resistor.
  - Regenerative braking.
- Optional use of a battery means you can achieve very high peak power output with only a modest power supply.
- Open source: Hardware, Software

INTERFACES

- USB -- Custom protocol, open source
  - PC, RaspberryPi, etc.
  - ROS node (coming soon).
- Step/direction -- Existing motion controllers
- UART -- Arduino (with library), mBed, etc.
- Servo PWM/PPM -- RC Recievers, Arduino, etc.
- CAN -- Synchronise multiple ODrives (coming soon)
- Some general purpose digital and analogue pins

PROTOCOLS

- Many types of command modes
  - Goto (position control with trajectory planning)
  - Position commands
  - Velocity command
  - Torque command

Figure 25: oDrive informal specs

F Recommendations for Improving the Course

1. Do the course over two semesters.
2. Give projects of smaller scope so that students can perfect it more and build the design. This could also reduce stress levels by making the project less initially intimidating.
3. Make students build the project.
4. The grading value of the Analysis Report should match more closely the ratio of time allocated for that deliverable compared to others.