KINEMATIC ARRANGEMENT OPTIMIZATION OF A QUADRUPED ROBOT WITH GENETIC ALGORITHMS

by
MEHMET MERT GÜLHAN

Submitted to the Graduate School of Engineering and Natural Sciences in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Sabanci University
July 2018

KINEMATIC ARRANGEMENT OPTIMIZATION OF A QUADRUPED ROBOT WITH GENETIC ALGORITHMS

APPROVED BY:

Assoc. Prof. Dr. Kemalettin ERBATUR ……..
(Thesis Supervisor)

Prof. Dr. Kürşat ŞENDUR ……..

Assoc. Prof. Dr. Selim BALCISOY ……..

Assist. Prof. Dr. Özkan BEBEK ……..

Assist. Prof. Dr. Sıddık Murat YEŞİLOĞLU ……..

DATE OF APPROVAL: 31/ 07 / 2018

KINEMATIC ARRANGEMENT OPTIMIZATION OF A QUADRUPED ROBOT WITH GENETIC ALGORITHMS

Mehmet Mert GÜLHAN

Mechatronics Engineering, Ph.D. Thesis, 2018

Thesis Supervisor: Assoc. Prof. Dr. Kemalettin ERBATUR

Keywords: Quadruped robots, kinematic arrangement, optimization, genetic algorithms, quadruped trotting

ABSTRACT

Quadruped robots are capable of performing a multitude of tasks like walking, running carrying and jumping. As research on quadruped robots grows, so does the variety of the designs available. These designs are often inspired by nature and finalized around technical constraints that are different for each project. A load carrying robot design will take its inspiration from a mule, while a running robot will use a cheetah-like design. However, this technique might be too broad when approaching a designing process for a quadruped robot aimed to accomplish certain tasks with varying degrees of importance. In order to reach an efficient design with precise link lengths and joint positions, for some specific task at hand, a complex series of problems have to be solved.

This thesis proposes to use genetic algorithms to handle the designing process. An approach that mimics the evolutionary process of living beings, genetic algorithms can be used to reach quadruped designs which are optimized for a given task. The task-specific nature of this process is expected to result in more efficient designs than simply mimicking

animal structures, since animals are evolved to be efficient in a bigger variety of tasks. To explore this, genetic algorithms are used to optimize the kinematic structure of quadruped robots designed for the tasks of vertical jumping and trotting. The robots are optimized for these two tasks separately and then together. Algorithm results are compared to a relatively more conventional quadruped design.

DÖRT BACAKLI ROBOTLARDA GENETİK ALGORİTMALARLA KİNEMATİK DÜZEN OPTİMİZASYONU

Mehmet Mert GÜLHAN

MekatronikMühendisliğiProgramı, DoktoraTezi, 2018

TezDanışmanı: Doç. Dr. Kemalettin ERBATUR

AnahtarKelimeler: Dörtbacaklırobotlar, kinematikdüzen, optimizasyon, genetikalgoritmalar, dörtbacaklıtırıs

ÖZET

Dört bacaklı robotlar yürümek, koşmak, yük taşımak ve zıplamak gibi çeşitli görevleri yerine getirme kapasitesine sahiplerdir. Dört bacaklı robot alanındaki araştırmalar arttıkça, bu robotların tasarımlarındaki çeşitlilik de artmaktadır. Bu tasarımların gelişiminde genellikle doğadaki hayvanlardan esinlenilir ve proje bazlı değişen teknik kısıtlamalar göz önünde bulundurularak sonuçlandırılırlar. Yük taşıyan bir robot tasarımı için bir katırın yapısından, koşmak için tasarlanan bir robot içinse çitanın yapısından yola çıkılır. Belli görevleri belli önem derecelerine uyarak yerine getirmek üzerine tasarlanması planlanan bir dört bacaklı robot için, doğadaki hayvanların yapılarını kullanmak verimlilik açısından fazla basit bir yöntem olabilir. Bu görevleri en başarılı şekilde yerine getirecek uygun bağlantı uzunluklarına ve eklem pozisyonlarına sahip bir robotun tasarlanması için birçok karmaşık problemin çözülmesi gereklidir.

Bu tezde dört bacaklı robotların kinematik yapılarının tasarımı için genetik algoritmaların kullanılması önerilmektedir. Genetik algoritmalar canlı varlıkların doğal

seleksiyon ile evrimlerini taklit eden bir yöntemdir ve belli bir görevi yerine getirmek için geliştirilen dört bacaklı robotların tasarım optimizasyonunda kullanılabilirler. Bu yöntem, bir robotun görev bazlı verimliliğini iyileştirmek için kullanılacağından direkt olarak göreve özel başarılı bir hayvanın yapısını taklit etmekten daha iyi sonuç alınması beklenir. Bunun sebebi doğadaki hayvanların türlü sebeplerden çeşitli birbirinden farklı görevleri yerine getirmek üzere evrimleşmiş olmalarıdır.Bunu araştımak amacıyla bu çalışmada genetik algoritmalar dört bacaklı robot kinematik yapısını dikey zıplama ve tırıs hareketleri için optimize etmek üzerine kullanılmışlarıdır. Optimizasyon bu iki görev için öncelikle ayrı ayrı yapılmış, sonrasında da iki görevi birden yerine getirmek üzerine uygulanmıştır. Algoritma sonuçları birbirleriyle ve önceden tasarlanmış bir dört bacaklı robotla kıyaslanıp incelenmiştir.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincerest gratitude to my advisor Prof. Kemalettin Erbatur for his selfless time and for his ability to keep me motivated without stress. He sets an example not only as an advisor but as a person that I would like to follow.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Kürşat Şendur, Prof. Selim Balcısoy, Prof. Özkan Bebek and Prof. Sıddık Murat Yeşiloğlu for their constructive attitude and contributions.

The financial support of TÜBİTAK BİDEB (2211) Doctoral Scholarship Program for this PhD study is gratefully acknowledged. The financial support of TÜBİTAK through project 114E618 “Quadruped Robot Design, Construction and Control” is also acknowledged.

I thank the rest of the project team, and the people of the Mechatronics Lab, for their help over the years.

Last but not the least; I would like to thank my family; my parents and my brothers for their endless support and motivation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\hat{f}$	:	Spatial force vector
$\hat{I}^A$	:	Spatial articulated inertia
$\hat{a}$	:	Spatial acceleration
$\hat{Z}^A$		Spatial articulated zero-acceleration force
$\boldsymbol{\omega}$		Angular velocity
$\mathbf{v}$		Linear velocity
$\mathbf{R}$		Rotation matrix
$\mathbf{r}$		The vector that connects joint i-1 axis to joint i
$\dot{q}_i$		Scalar velocity of joint i
$\mathbf{u}_i$		Rotation axis vector for joint i
$\hat{c}_i$		Coriolis term for joint i
v_i		Vector velocity of joint i
$\mathbf{d}_i$		The vector that connects joint i-1 axis to link i center of mass
$\mathbf{g}$		Gravity vector
$_{\mathbf{i-1}}\hat{\mathbf{X}}_{\mathbf{i}}$		Spatial transformation matrix from frame i origin to frame i-1
$\hat{\mathbf{s}}_{\mathbf{i}}$		Spatial joint axis vector of i
Q_i		Scalar joint force/torque
$\ddot{q}_i$		Scalar joint acceleration
Γ		Joint torque vector
A		Matrix that relates contact forces to joint accelerations
J		Jacobian that transforms joint accelerations to contact ones
Λ		Matrix that relates contact forces to contact accelerations
v_c		Contact velocity vector
dt		Simulation time step
M		Inertia coefficient
B		Damping coefficient
K		Spring coefficient

LIST OF ABBREVIATIONS

COM	:	Center of Mass
ZMP	:	Zero Moment Point
ABM	:	Articulated Body Method
DOF	:	Degrees of Freedom
GA		Genetic Algorithms
LCP		Linear Complementary Problem
PID		Proportional Integral Derivative

Chapter 1

1. INTRODUCTION

The application of genetic algorithms is reported in the field of structural optimization where the problem is geometrical and topological design (Grierson and Pak, 1993, Kane et al., 1996). The challenge of designing efficient proportions for individual elements in a complex structure is a multilevel optimization task that has been tackled with genetic algorithms (Deb et al., 2001; Erbatur et al., 2000). It is similarly used in optimizing the kinematic arrangement of mechanisms with links and joints (Roston, G. 1994) and in optimization of robotic manipulator designs (Kim et al., 1992, Chen et al., 1995). The purpose of this work is to take this a step further and optimize the design for the kinematic structure of a complex free-fall manipulator, a quadruped robot. The optimization is carried out within a certain scale range and with defined available power via full dynamic simulations. This quadruped robot is planned to be used as a research platform for experimenting on tasks such as walking and running while mostly focusing on jumping. Although there are multiple successful quadruped robots that exist in literature, they use different kinematic structures and the tasks they focus on performing vary (Raibert et al., 2008; Semini, 2010; Seok et al., 2013). When it comes to how the design of these quadrupeds' kinematic structures the most common explanation is taking the nature and its quadruped animals as inspiration for these robots' designs (Yamazaki, 1999; Semini, 2010; Kahn et al., 2015). This work aims instead to mimic the way nature came up with these limb structures and arrangements. Using genetic algorithms, a process similar to natural selection is achieved. Mimicking the process rather than the result is expected to get more efficient outcomes. In any robot design, there will be tasks that are expected for the robot to undertake that are limited compared to its natural counterpart which handles additional tasks like hunting, resting, eating, climbing etc. Even if a task is common for the robot and the animal it is mimicking (like walking, running, jumping), the weight of this task as a criterion for the optimization process might differ. The goal in this work is to

explore the significance of optimizing a quadruped robot kinematic structure based on the specific tasks it is designed to perform.

Genetic algorithms build “populations” of solutions/designs and go through iterations by eliminating the individuals in the population unfit for the task at hand and end up with a final population filled with fit individuals (quadruped designs in this case) (Beasley et al., 1993). The iterations are termed as generations in analogy to living beings. The way the fitness decision is made is by taking each individual through a simulation and evaluating the results using a fitness function.

In this thesis, the application of genetic algorithms is carried out in the context of an ongoing quadruped project (TUBITAK 114E618 “Quadruped Robot Design, Construction and Control”) where the primary objectives include locomotion and high jump altitudes. Hence the focus was on two tasks: Vertical jumping and trotting. The kinematic parameters in the two cases are optimized and results of these optimizations are presented. Quadruped robot design parameters and the use of the algorithms are discussed. The kinematic arrangement of a robot is defined as the relative locations and orientation of the joint axes with respect to each other. A brief parametrization technique of the arrangement is by means of the Denavit-Hartenberg variables (Denavit and Hartenberg, 1955). Among the Denavit-Hartenberg variables are the link lengths which are optimized in our work along with quadruped main body dimensions. Hydraulic cylinder actuators, as in the aforementioned TUBITAK project work carried out by the authors are assumed for all joints.

The thesis is organized as follows: Chapter 2 is a literature survey on genetic algorithms and quadruped robots and the chapter also mentions the earlier work on Sabancı University on these subjects. Chapter 3 states the problem definition and elaborates on the work’s aim. Chapter 4 explains in detail how our implementation of the genetic algorithms works and explores the importance of its parameters. Chapter 5 goes into our dynamic simulation of quadrupeds and describes the Articulated Body Method (ABM), Gauss Seidel like method used for the contact model, Zero Moment Point (ZMP) based trajectory generation method, control algorithms, and hydraulic actuator torque model. Chapter 6 presents results for the optimization of the vertical jumping task, trotting task, and the combined optimization of both. It displays plots of the data and has a discussion through their analysis.

Chapter 2

2. LITERATURE SURVEY

2.1. Survey of GA Literature Related to Design Applications

Genetic algorithm itself doesn't differ greatly between applications. Fittest individuals of a population move on to the next generation which is followed by cross-over and mutation. Fitness is decided by a value calculated through a fitness function. Usually a problem involves constraints for the values that certain variables can take. These are also dealt in the fitness function through the use of penalties.

Since the algorithm itself stays more or less the same, this literature survey focuses on the applications genetic algorithm is used on. These applications are kinematic arrangement and structure optimization related similar to the one proposed here.

One of the fields where genetic algorithm sees use is structural topology optimization. In Grierson and Pak 1993, optimization of topology for different skeletal building frameworks are explored. Example structures are shown in Figure 2.1. The variables for this problem are column lengths and widths and there are constraints to the values they can take. Algorithm is used to optimize the weight and elastic strain under certain loading.

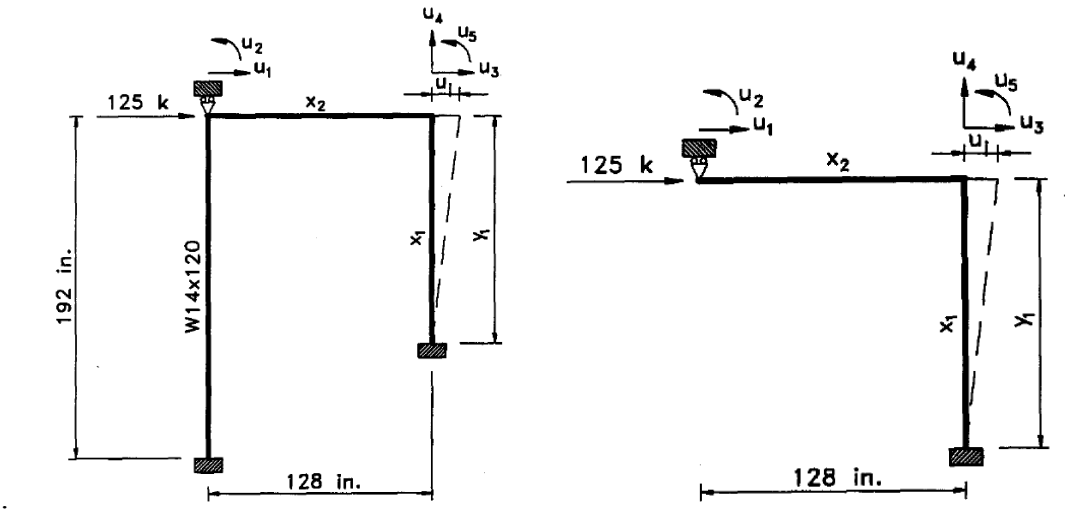


Figure 2.1: Example skeletal building frameworks optimized with GA

Genetic algorithm is preferred for this problem mainly because of the discrete nature of the design space. Materials with standard commercial use are not available at every size, making the variables discrete for structure optimization problems. This works with genetic algorithms since variable values are first generated randomly within the design space and contents of later generations come from parent populations.

Another advantage of genetic algorithms is that the optimization only requires values of the fitness function. Constraint penalties are tuned by the programmer, so all there is needed is a simulation that solves for the number that is going to be optimized. No other equation for the effect of each variable is required. In Kane et al. 1996, a finite element method is used to numerically simulate and solve for the mechanical behavior cantilever plates. The relationships of each variable with the solution are not explicitly available and therefore cannot be used for the optimization. Genetic algorithm is preferred because the result of the simulation can be used as the fitness function. This means that the simulation has to run for each individual in each population, which is a slow process and a major drawback of genetic algorithms.

The same work also analyzes ways to handle the constraint penalties. An individual might land out of the design space making it infeasible even though it contains optimal variables. Strict penalties would mean that this information is lost in the next generation which is undesirable. An adaptive coefficient is used which grows with each generation, gradually enforcing the penalization. This way, infeasible but high fitness information is kept

for early generations while only feasible results are allowed in the final population. This also shows that a genetic algorithm's result might change with the tuning of constraint penalties.

The definition of variables can also be used to manipulate the optimization problem. In a similar topology optimization problem to the one above where dimensions like height and width are used as variables, one can take the 2D cross section of a beam as pixels and use the information if there is a material or not in a pixel as a variable (Jekiela et al. 2000). The topology might start as an I-beam and end differently as a result as seen in Figure 2.2.

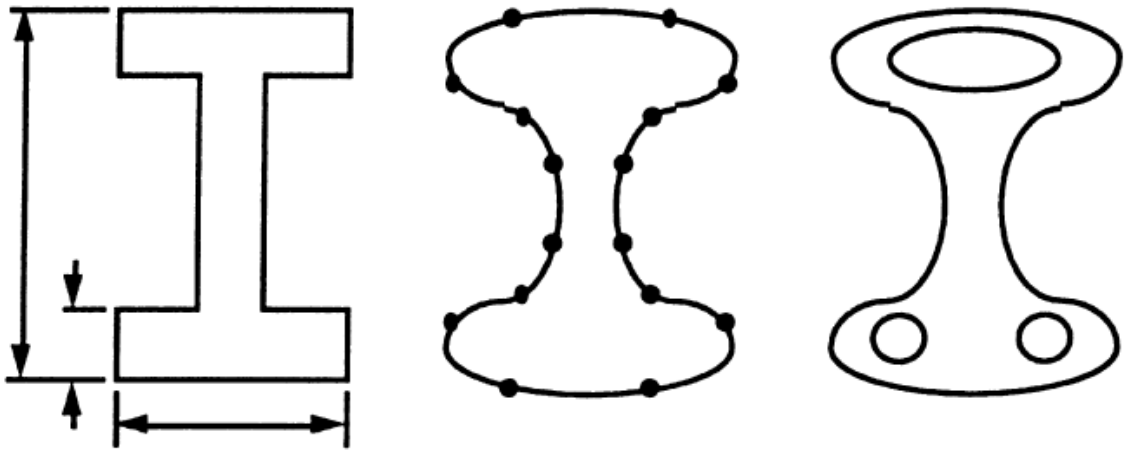


Figure 2.2 :2D cross section of a beam going through the pixel optimization process

GAs are also used in the design of more complex structures such as trusses. In a structure like this the optimization problem includes finding optimal cross-sectional size, topology, and configuration. Solving for optimal size can be modeled as a nonlinear programming problem (NLP). Another NLP would be formed to find the optimal configuration of trusses (example configurations on Figure 2.3). This turns the entire design into a multi-level optimization problem. It is explained that in multi-level optimization methods, you go through the problems one by one, keeping the configuration fixed while solving for the optimal size and vice versa (Deb et al. 2001). This would mean that depending on the initial configuration the method might solve for a different result which would not need to be the globally best design. This is not necessarily a problem with GAs because you can solve for all the required variables together by simulating the resulting truss's mechanical performance. Forming a fitness function from the simulation results such elastic deflection and combining it with other considerations like weight of the truss and any given constraints

the design space can be searched for a global optimum. A design problem involving multiple NLPs can be solved simultaneously by using GA.

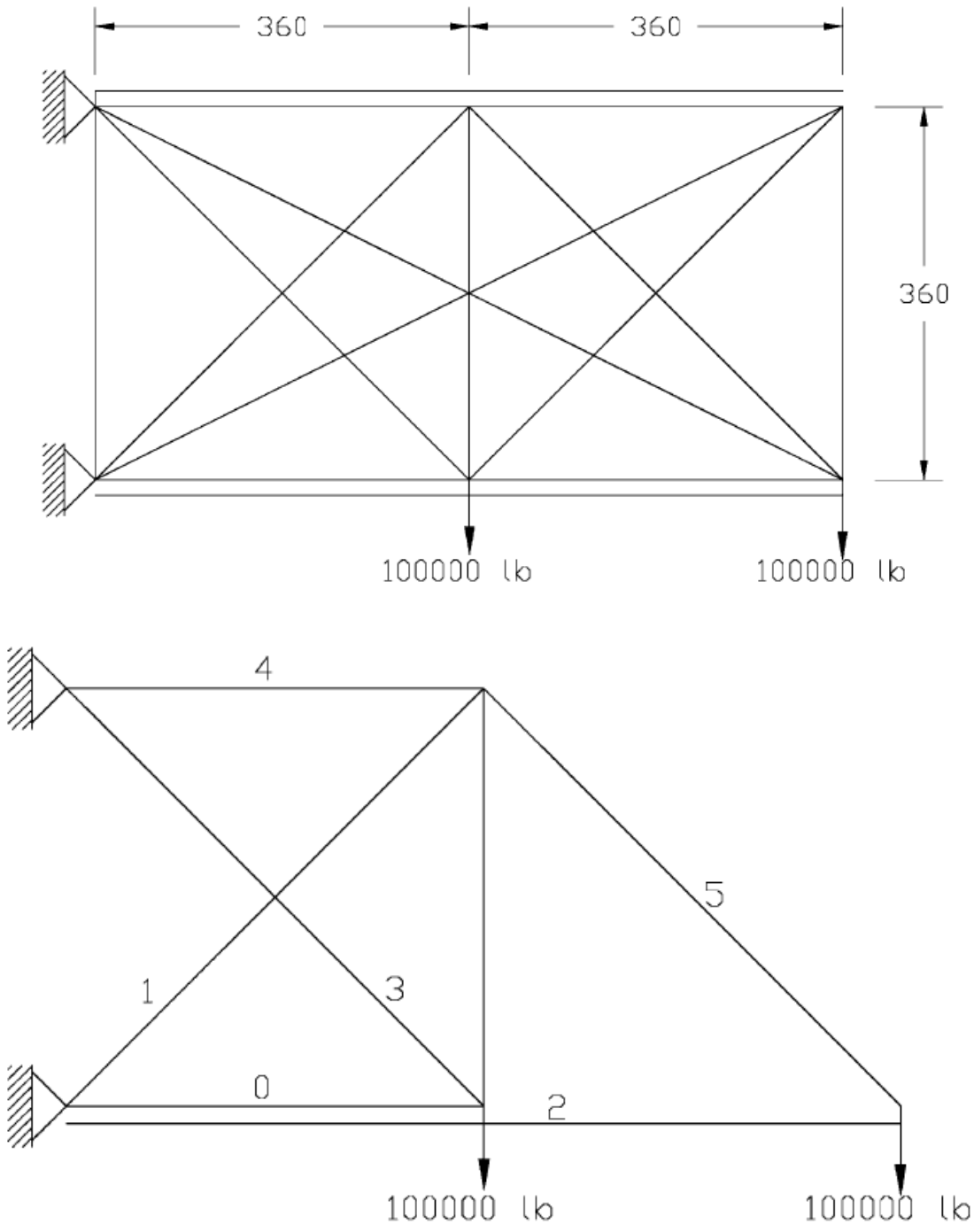


Figure 2.3 : Truss configuration examples.

In addition to its other advantages, the success of GAs in actually finding a global optimum is also addressed. In Erbatur et al. 2000, results from GA for various mechanical

design optimizations are compared to other discrete and continuous optimization algorithms. Non-GA algorithm solutions for the design problems are taken from previous literature. The problems themselves are optimization of different structures like cantilever beams and 3-D trusses. In all 6 different problems tested, the GA used finds either similar or better results when compared to other optimization algorithms used in previous literature for the same problems. This shows that there is no significant drawback performance-wise when GAs are used for optimization.

As the design application for GAs get more complex, it gets harder to form a system to store variable information in an individual. When optimizing the cross-section area of a beam, variables are numbers representing the dimensions such as height and width. Information is stored as a 2-bit string in an individual's chromosome. When variables are numbers they are converted to their 2-bit versions by a user-defined system. On other applications like in the truss topology example a link is not there simply if its cross-section area is smaller than a threshold (Deb et al. 2001). The configuration difference between the two examples in Figure 2.3 is that on the right one some links do not exist. Simulation algorithm for the mechanical behavior of the system handles the fact that a link with a small enough cross-section area equals to no link in that spot.

When optimizing a kinematic arrangement handling the design variables is more complicated. In Rao 2003, a gear kinematic train has to be represented in a string so that the design can be optimized by a GA. In order to do this, gear arrangements are first turned into graphs just as in Figure 2.4. These graphs then can be represented by two matrices; one for single connections with values as 1 if there is a link and 0 if there is none and one for double connections with the same format. Placing the rows of these matrices next to each other one can get a string of 1 and 0s which can be used in chromosomes.

The way variable representation is handled in GAs has an effect on how to formulate pairing of individuals in order to carry the information to the next generation and how crossover works. This is because individuals of the next generation should also be physically possible arrangements. This brings a layer of difficulty to using GAs in robotic applications where kinematic arrangement is part of the optimization process.

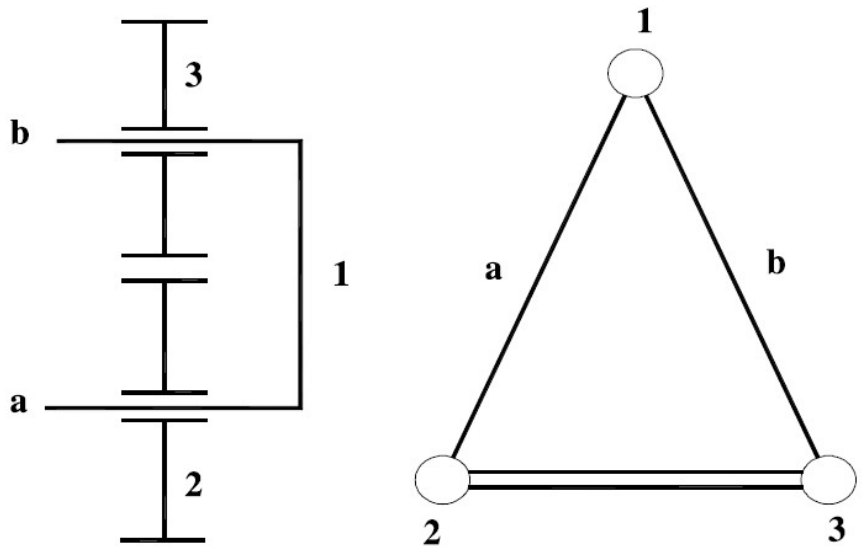


Figure 2.4 : Different representations of a gear train

A particularly complicated case of kinematic arrangement representation is analyzed (Roston, 1994). In this work, planar mechanism configurations have to be represented in order to be used in a GA. A six-bar mechanism for example can have different number of joints on each link, resulting in considerable number of possibilities. A link can be binary, ternary or quaternary which corresponds to two, three or four other links attached to it through joints. Figure 2.5 shows an example six-bar mechanism. A system like this needs an algorithm developed just to convert the planar mechanism into a string representation suitable for GAs. One of the results of this work reveals that the representation can actually bias the outcome of the GA.

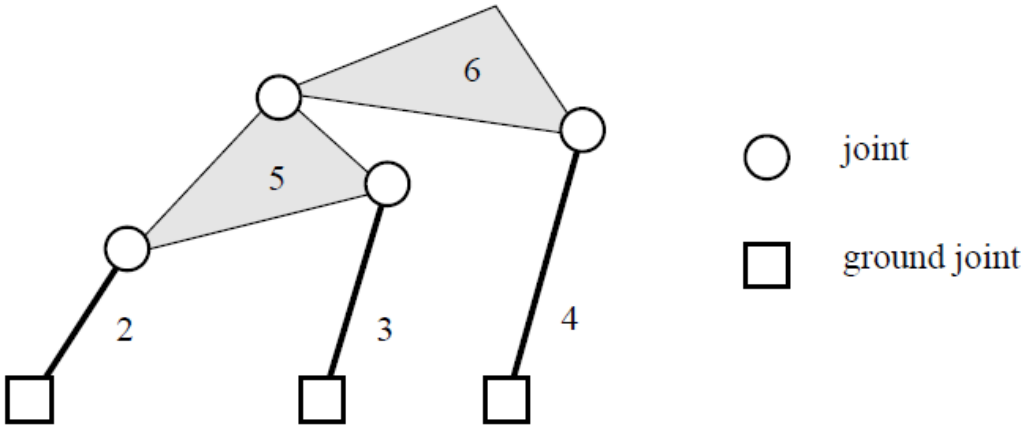


Figure 2.5 : A six-bar mechanism

There are a few examples of GAs being used in robotics applications in literature. In Kim et al. 1992, task based design of a 3DOF manipulator is optimized. The manipulator arm has three revolute joints with a stationary base. Given desired end effector position, the design variables are link lengths and joint angles. As seen in Figure 2.6, four variables x_1 , y_1 , x_2 , and y_2 are enough to completely define a design. After these four are found by the GA and since x_3 and y_3 are already given as the end effector position, all the link lengths and joint angles can be calculated.

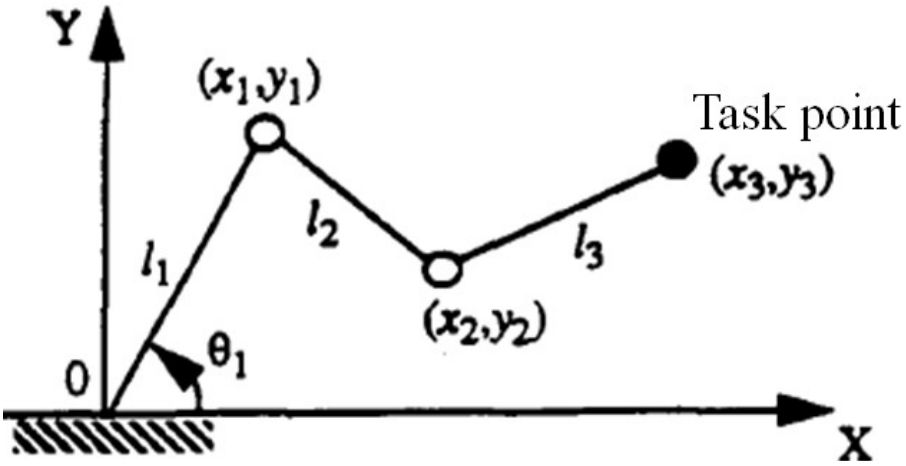


Figure 2.6 : Manipulator arm with three revolute joints

In order to evaluate the efficiency of a manipulator with GA there needs to be a single number that represents the fitness of an individual design, which is the purpose of a fitness function. In the above work, fitness function uses a dexterity measure calculated from a relative manipulability term. This term is calculated from the Jacobian matrix of the manipulator and a higher number indicates a bigger effect of joint angles on the end effector position. The rest of the fitness function consists of various constraint penalties like maximum and minimum allowable link lengths and joint angles, maximum difference between link lengths relative to each other and task specific constraints like some *x* and *y* values being undesirable due to obstacle avoidance. This fitness function is then used in a GA successfully exemplifying the algorithm's effectiveness in robotic applications.

A similar yet much more complex robotics application of GA is task based optimization of modular robot assembly configurations (Chen et al. 1995). Similar to the earlier example, requirement is to optimize a manipulator arm's kinematic configuration for specific tasks. The additional complexity of the problem comes from the relative size of design space. Each link in the design is picked from two options with separate sizes and different number of joint ports (Figure 2.7). Each port can either be empty or occupied by one of the four different available joint types which are revolute, prismatic, helical or cylindrical joints. Even if there are ways to simplify the problem like by limiting the number of links or number of DOF, considerable number of possible designs is left in the design space.

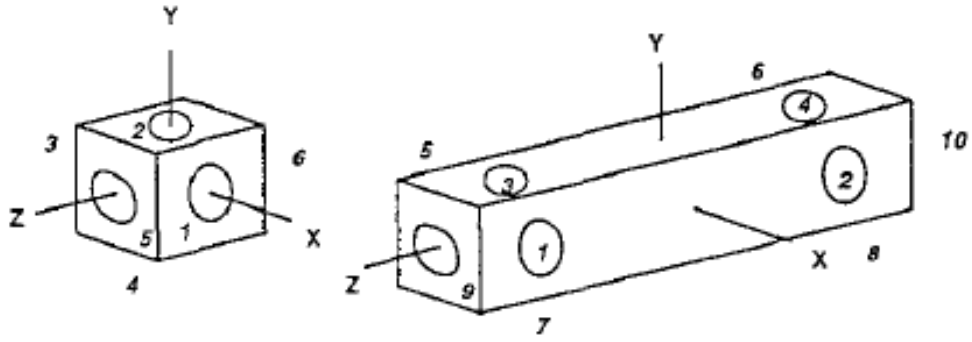


Figure 2.7 : The two link designs with their joint ports

The paper states its unique matrix representation for a given assembly configuration, describing where the links are connected and by which joint type they are joined. This is then transformed in to a bit string in order to be used with GA. The effectiveness of a design is evaluated by a fitness function which also makes use of the manipulability term, except these are calculated for multiple task points forming a desired trajectory. Design constraints for the problem include redundant joints and minimum link interference. In short, it is an example of kinematic arrangement optimization with a certain degree of complexity where it is stated that the genetic algorithm method is found to be well suited.

Another example of task based modular robot manipulator design is done in Chung et al. 1997. Joint and link module options for this manipulator are much more limited and therefore the assembly arrangement is decided through kinematic relations. After robot configuration is determined, a GA is used to optimize link lengths. If one can handle the first step like the previous example, these two steps could be combined and simultaneously

optimized using GAs, resulting in a more accurate global optimum. A more challenging project such as the one we propose should aim to make use of these examples.

Manipulability enhancement techniques by the genetic tuning of kinematic parameters is also reported in (Khatami and Sassani, 2002), (Merlet, 2003) and (Liu et al., 2015). Again, the optimization is based on static objective functions in these studies, without assessing performance in a dynamic task.

It should be noted that, in the robotics literature, GAs can be found in applications which are not related to kinematics parameter optimization. Path planning (Lee and Kim, 2016, Song et al., 2016, Bakdi et al., 2017, and Elhoseny et al., 2017) and solutions for the inverse kinematics problem (Momani, 2016) are among such.

To summarize, multiple uses of GAs in optimization of various structural designs are studied. It is seen that although GAs are computationally heavy, they are advantageous in applications involving discrete design spaces, multiple simultaneous optimizations and use of simulation data for measuring effectiveness. It has also been noted in several works that representation of design variables and tuning of the fitness function have an effect on the solutions. This is especially true in kinematic configuration problems where each work has its unique bit string representation. All of these apply to our work since we aim to reach to a novel design for a quadruped robot through exploring possible kinematic configurations and link dimensions with GAs.

2.2. Review of Legged Robot Literature

Studies on legged robots have started in mid-20th century and there have been increasingly more research on the subject since. It has been important for many applications that these mobile robots can autonomously traverse terrain. Response to danger in natural events such as earthquakes and fires, finding and defusing bombs or landmines in urban or natural areas, reconnaissance and load bearing in military missions are all examples to these applications.

Mobile land robots can be designed with wheels, treads or legs. Depending on the application, legged robots might have advantages over the other two. In objectives such as climbing steps, going over barriers, traversing on rocky terrain, and passing over holes legged robots are favored. Robots with wheels or treads require constant contact with a surface to

move while legged robots can step on surfaces separated a certain distance apart from each other, accomplishing otherwise difficult tasks. There are legged robots with one, two, four or more legs. With respect to one or two legged kinematic arrangements, four legged robots are more balanced.

When compared to robots with even more legs, four legged robots have a simpler and cheaper production process. There are many four legged animals observed in nature with the ability to run at considerably high speeds which have weights that range from tens to hundreds of kilograms. These fast movement and load carrying capabilities are among the desired attributes in mobile robot applications like the aforementioned examples like firefighting and military cases. A lot of four legged robot work and research are present in literature born from these motivations.

One of the first legged robot project ever completed is a human operated one done by General Electric (Mosher, 1968). McGhee from USA (McGhee, 1985) and Gurfinkel from USSR (Gurfinkel, 1981) are the first scientists to apply computerized control to legged robots. In 1984 Hirose built a computer controlled, pantograph leg designed machine capable of climbing stairs (Hirose, 1984). One thing these three pioneer machines have in common is that ground projection of their center of gravities fall into the area defined by the space between the legs in contact with the ground. This area is called the support polygon and the type of walk which stays in this polygon is defined as static walk (Raibert et al., 1986). Static walk is well balanced but it results in a locomotion slower than optimum. A different category named dynamic walk involves center of mass of the robot leaving the support polygon at certain intervals. First examples of research on this field are Kato et al. 1981 and Miura et al. 1984. Various different robot examples are presented also in other works by Raibert (Raibert, 1986).

Raibert's work starts with a single legged jumping robot with hydraulic actuators (Raibert et al., 1986) and he goes on to use the same jumping principle with two legged (Playter et al., 1992) and four legged robots (Raibert, 1986)(Raibert, 1990). Raibert later founded Boston Dynamics and the company built some of the leading examples of legged robots designed for military use, namely Bigdog (Raibert et al., 2008), Alphadog-LS3 Littledog (Kolter and Andrew, 2011), Cheetah ,WildCat, and Rhex (Boston Dynamics, 2014). With Google showing interest and buying Boston Dynamics in 2013, research on legged robots promises to become its own sector in the robotics industry.

A byproduct of Boston Dynamics' work is researchers' growing interest to the field of hydraulic actuated mobile robots. Research on four legged robots increased all around the world in recent years. An example robot is HyQ; inspired from Boston Dynamics' BigDog and built by IIT (Italian Technology Institute). While information on Boston Dynamics' robots are limited due to the military nature of their applications, HyQ's research group had published papers including detailed information and research results about hydraulic actuated robots (Semini, 2010, Focchi, 2013, and Boaventura, 2013).

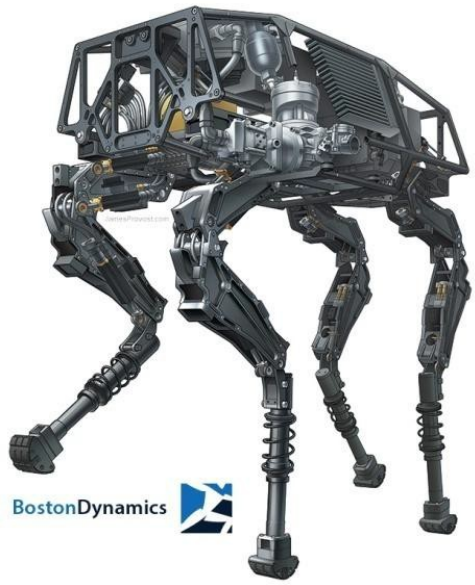


Figure 2.8: BigDog

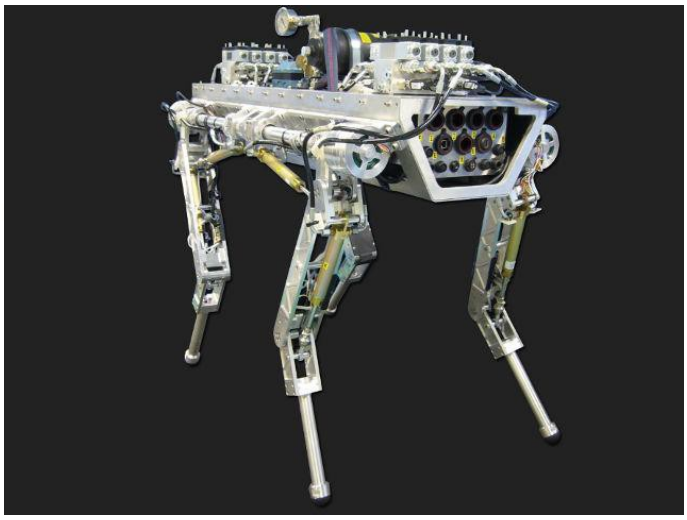


Figure 2.9: HyQ

Quadrupeds in the literature have varying capabilities. Kotetsuis able to walk on even ground (Maufroy et al., 2010). Patrush runs on even ground and it can climb up to 12 degrees of inclination (Kimura et al., 1999). Tekken can climb slopes 10 degrees while moving forwards and 5 degrees while moving sideways (Fukuoka et al., 2003). It can also traverse gravel with 0.6 m/s speed. PAW climbs 16 degree slopes and it can jump over 166mm obstacles (Smith, 2006). Scout can climb steps as high as 45% of its leg length (Buehler et al., 1998).Mrwallspect can walk on 35% slopes both uphill and downhill (Kim et al., 2005). Kolt can reach up to a speed of 1.1 m/s while running on even ground (Estremera and Waldron, 2008). Hydraulic actuated BigDog can run with 3.1m/s speed, climb 35% inclinations, and jump as high as 1.1 meters (Raibert et al., 2008). HyQ can run with 1.7m/s speed, and jump 0.2 meters (Boaventura, 2013). The size of the robots and their actuators play a role on all

these numbers. The effect of preferred control methods and sensor systems is also considerable. BigDog sets itself apart from the other quadruped robots by its high mobility on slippery or uneven surfaces and its robustness to disturbances. The effectiveness of hydraulic actuators can be observed on various video footages available for the robot (Boston Dynamics, 2014).

Aforementioned quadrupeds also show diversity in their kinematic arrangements and number of DOFs. Some of them have joints solely on their shoulders and hips, absorbing ground contact forces through springs located on their legs (Yamazaki, 1999). This type of robot is best suited for walking with bound gaits. Some robots have wheels or treads attached to the end of their legs, allowing them to reconfigure from legged locomotion to wheeled locomotion (Schenker et al., 2000) (Takahashi at al., 2006) (Nagatani et al., 2011).Some of the most important quadrupeds have legs with multiple revolute joints similar to their natural equivalents. BigDog, LS3, HyW and StarIETH are all examples to these walking gaits (Hutter et al., 2012). HyQ robot, BigDog 2005, and BigDog 2006 use 3 DOF leg structures with revolute joints. BigDog 2010 adds one more joint to this structure, improving its kinematic arrangement. The added DOF lets the robot place its feet to the ground with any desired angle on the sagittal plane. Our research group (TUBITAK 114E618) uses a similar kinematic arrangement for their quadruped simulations.

The survey shows that quadruped robots keep progressing and evolving in terms of the actuators they carry, the kinematic arrangement they have and the control algorithms they use. Parameters like link sizes and actuator power also contribute to their effectiveness in accomplishing the various tasks they perform. There is reason to believe that putting all these through a genetic algorithm might result in a novel effective quadruped design for a hydraulic actuated jumping robot.

2.3. Previous Work on SU on Legged Robotics and Genetic Algorithms

There are two examples where GAs was previously used for robotics applications in our Mechatronics Research Laboratory. These are bipedal humanoid robot (SURALP) walking reference tuning and tuning of a central pattern generator for quadruped locomotion.

2.3.1. GA assisted gait tuning in biped robot SURALP

SURALP is a full-body humanoid robot platform designed for bipedal walking (Figure 2.10). It has 29-DOF, with 6-DOF on each leg. Its walking reference is generated using a method which keeps the ZMP (zero moment point) of the body in the supporting polygon defined by foot or feet touching the ground. The ZMP criterion is a commonly used stability condition in biped robotics.

Genetic algorithms are used to optimize the walking reference of SURALP. There are many variables involved while generating SURALP's walking reference. Among these, double support period and single support period are the variables chosen for optimization. Double support period is defined as the time spent where both feet are on the ground while single support period is when a single foot is touching the ground. Both these numbers are turned into 8-bit binary representations to be used in an individual's chromosome.

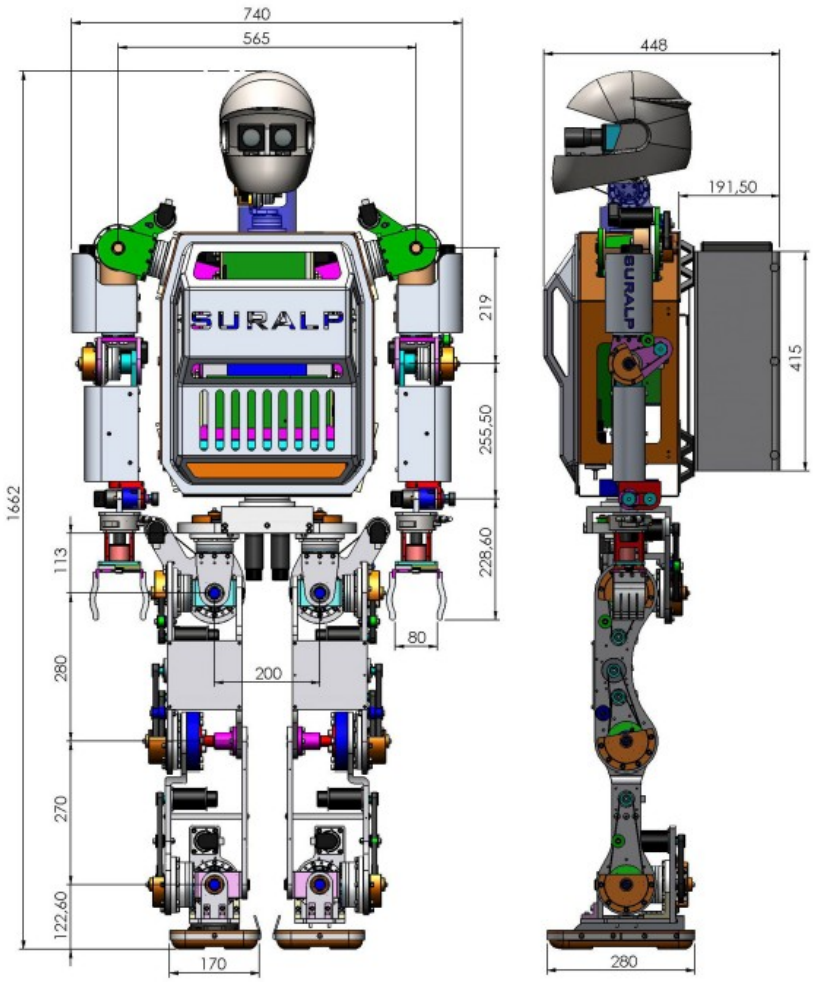


Figure 2.10: SURALP

Walking references resulting from randomly generated double support and single support periods are than animated using a Newton-Euler based dynamics simulation. The simulation results have to give a number which can be used in a fitness function evaluating effectiveness of individuals. This work implemented virtual torsion spring-damper systems to resist deviations in robot's body orientation. The more imbalanced a walking reference gets the more deflections on the springs are observed; giving an indication of effectiveness. These deflection values are then used in the fitness function and combined into a single number defining an individual's fitness.

After tuning the remaining variables in a GA like population size and mutation probability, the algorithm is executed. Results successfully find stable solutions in high velocities that were previously not reached by SURALP.

2.3.2. GA assisted gait tuning in a quadruped robot

The quadruped used for this work has 4-DOF on each leg which consists of 2 on each hip, 1 on each knee, and 1 on each ankle, making a total of 16-DOF (Figure 2.11). Central pattern generation is used for creating walking references for the robot. CPG can be considered as nonlinear coupled oscillators which produce rhythmic signals with different phase relationships. Coupling of the oscillators can be tuned to produce different types of gaits.

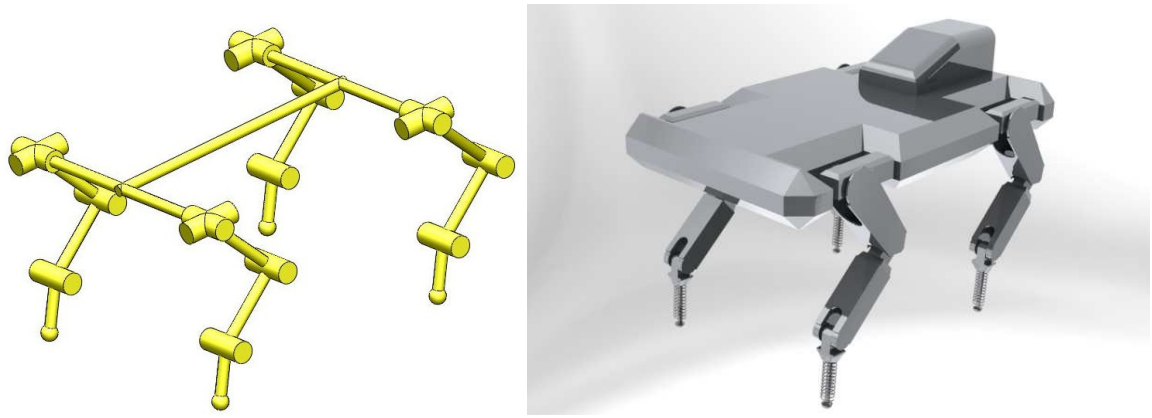


Figure 2.11: 16-DOF quadruped

Although the variables involved in a CPG are mostly wave parameters such as phase, amplitude and frequency; for GA's tuning variables step height and step frequency of the robot are found to be more convenient. Given an average velocity value, randomly generated step frequency and step height values are used to produce a population. To evaluate the fitness of individuals, a Newton-Euler based dynamic simulation similar to SURALP's case is used. The simulation employs an adaptive penalty based method for its ground contact model.
Virtual torsion springs are implemented again to keep the robot balanced while walking. With most randomly generated individual, the quadruped fails to keep its balance while walking and it makes it difficult to determine a fitness value for such cases. The springs keep the system walking while spring torque numbers can be used in the fitness function. Multiple successful walking references for various average walking velocities were found as a result of the GA used for this work.

2.4 Contributions of the Thesis

This work contributes to the literature in robotics in the following ways:

- The thesis poses the quadruped kinematic design task as a GA search problem. Success measures in dynamic tasks of a complicated robot structure are formulated for a GA optimization problem. This problem is more complex from the ones reported in the robotics literature in the following aspects:

  - The quadruped is a multi-DoF manipulator compared to the optimized structures in the literature which are at most of 6 DoF's. The robot in our consideration is a 16-DoF one.
  - The quadruped is a free-fall manipulator. The simulation model is structured in such a way to handle the free-fall dynamics. No GA study addresses free-fall robot structure optimization.
  - The optimization is based on dynamic performance. The studies reported in the literature focus on reaching space maximization, solely working on static relations.
  - High level tasks are considered as performance measures in kinematic parameter optimization. Jumping and running are sophisticated tasks with newly introduced task specific fitness indicators. This is also in contrast with the existing literature where manipulability is investigated and improved, via the use of static relations.
  - A co-design of trajectory and kinematics parameters is presented for specific tasks.

- The thesis presents an efficient design methodology for dynamically performing free-fall manipulators. The work in this paper stresses that the base inspiration from the nature does not necessarily end up with an efficient quadruped (or other legged) design. This work proposes that a "to the point" design for specific tasks is viable and kinematic parameters can be significantly different for different tasks.

- An exact multi-body dynamics and contact modeling technique is adapted for the quadruped jump. It should be noted that the contact model in a jumping robot scenario is of extreme significance. Contact models also available in commercial dynamics simulation software packages are mostly relying on penalty based contact force computation which, requiring low simulation cycle times, are computationally inefficient. Such models lose

realism in the case of longer cycle times. For a search method with simulation based design evaluation, simulation cycle times are of great practical importance. The GAs presented in this work are such algorithms. Jumping and other dynamic movements are where the penalty-based contact model loses realism, and no reliable locomotion result can be obtained without a proper exact model. This work presents an exact model framework implemented for the jumping and fast locomotion dynamic movements.

- The dynamic motion of quadruped robots, although promising for various all-terrain applications, is not intensively documented on. Many studies rely on simplified dynamic models and non-dynamic tasks. This work is one of few reported studies which are related to a robotic structure with a full-dynamics model.

Chapter 3

3. PROBLEM DEFINITION

The aim of this work is to tune the kinematic arrangement of a quadruped robot design so that it is successful in performing two tasks: Vertical jumping and trotting. These two tasks are chosen in accordance with our TUBITAK quadruped project's goals. The variables - referred here as design parameters – that define the kinematic arrangement and the task at hand are required to be optimized through a process. This is done by the use of genetic algorithms.

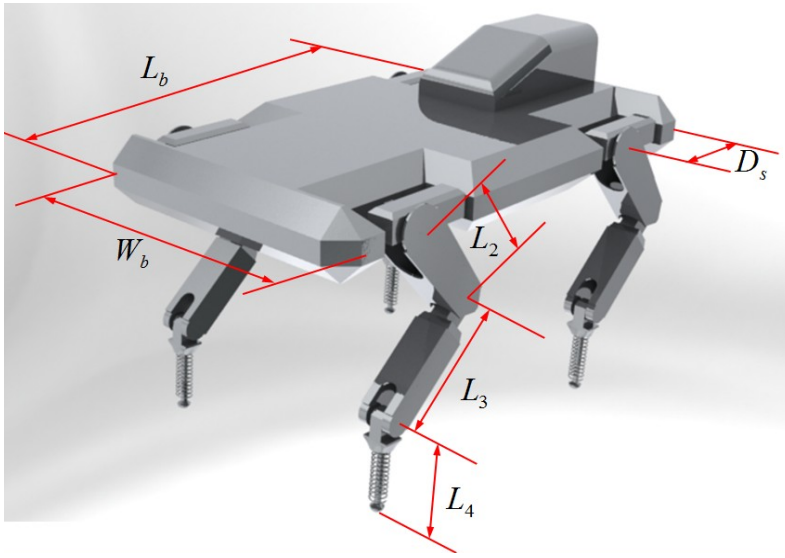


Figure 3.1. An example quadruped design with link lengths (L_2, L_3 and L_4) and the body dimensions (L_b and W_b) as kinematic parameters displayed

Figure 3.1 shows an example quadruped design. The kinematic parameters are the three link lengths on the legs (L_2, L_3 and L_4 on the figure), the body dimensions (L_b and W_b on the figure) and the shoulder attachment positions on the body. L_1 is omitted since it is a link with no length and exists due to our adaptation of Denavit-Hartanberg convention. When changing the lengths of the leg links, it is assumed that their masses scale linearly. When changing the body dimensions there is a constraint keeping its volume fixed. The space required to fit the necessary equipment is thus always allocated. The equipment includes various circuit boards and hydraulic units. There is flexibility in their placement and, to some degree, shapes. Still, a volume requirement does exist.

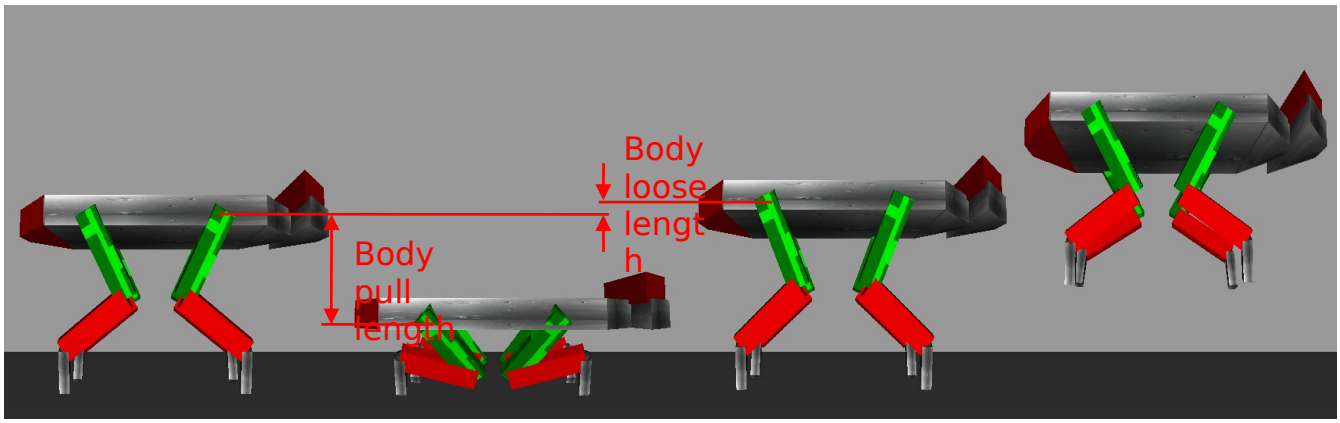


Figure 3.2. Animation frames showing the task of vertical jump. Task specific parameters are displayed.

The first of the two tasks used for the optimization algorithm is vertical jumping. In order to achieve this motion robot pulls itself down and then pushes the ground rapidly, resulting in a vertical jump as shown in Figure 3.2. The robot then lands on the ground, regaining its balance. For the purposes of improving the optimization of kinematic variables, some task specific variables had to be added to the algorithm. The task specific variables for the jumping task are body pull length and body loose length as shown in Figure 3.2. The robot picture on the left represents a “neutral” standing posture. Actions are taken to deviate from this posture to facilitate the jump. From this original standing; the body is lowered to the pose

shown in the second picture from the left. Inverse kinematics for the legs, which can be regarded as “shoulder/hip-based manipulator arms”, is solved for with a continuously tightening height variable. The result is a lowered robot body close to the ground. The amount of vertical deviation from the original posture is termed as “body pull length”. The standing posture is defined by joint angle references, which are fixed for all kinematic parameter sets. The joint angle references are inherited from trotting simulations with the model of a “base” design. They define the initial and final standing posture in simulations, which were well-balanced and taken as a stable/balanced posture for the genetic tuning studies in this thesis. The joint angle references translate to a body height which is different for different link lengths. However, they constitute a standardized starting point for each fitness-test run of the simulations. The jump is created by a fast climb from the lower-most body height (defined by the original height and the body pull length) to a posture above the original standing posture. The amount of rise over the standing posture is called “body loose length” in this dissertation. When the climb reaches the height defined by the standing pose and the body loose length, the joint references are frozen. Then the resulting motion can be a successful take off (the right-most picture in Figure 3.2), depending on the body pull and loose lengths, or it can fail. A fitness function evaluated the success of the jump. The speed of the climb is dictated by the available torque (also called the torque capacity) of the joints. The torque capacities are assigned by considering typical hydraulic cylinder actuator diameters and pressures. A taller, heavier robot can in exchange afford bigger pull and push distances, therefore these task specific variables should also be included in the optimization. The animation visuals are simplified for computational purposes.

The second task is performing a trotting gait. This starts by the quadruped lifting its left front and right hind feet, moving them forward, landing them (it can also start by the quadruped doing the same with its right front and left hind feet). The robot is then supported by its four feet before the next step. This is then followed by the robot lifting the right front and left hind feet and landing them some distance forward. The motion is planned by using a ZMP (Zero Moment Point) based trajectory generation (Akbas et al., 2012). Like with the first task, here it is also assumed that as the lengths and positions of the design change, the optimal trajectory variables change with it. For this reason, task specific trajectory parameters which are step size, double support period, and quad support period are added to the design parameters. Step size is simply the length of each step during trotting, which is the distance a

foot covers from lifting off the ground and landing back on it. Double support period determines the time spent on two legs while taking a step. Quad support period determines the time spent on four legs. Table 3.1 shows the values that each of these parameters can take.

Table 3.1. Design parameter value ranges.

Parameter	Value range
L_2 Shoulder to elbow link ( )	0.2-0.5 m
L_3 Elbow to ankle link ( )	0.2-0.5 m
L_4 Foot link ( )	0.15-0.35 m
L_b Body length ( )	1-1.5 m
W_b Body width ( )	0.4-1.0 m
Body edge to shoulder distance	0.1-0.3 m
Task Specific Parameters	
Body pull length for jumping task	0.1-0.4 m
Body loose length for jumping task	0.05-0.2 m
Step size for trotting task	0.05-0.25 m
Step height for trotting task	0.05-0.25 m
Quad support period for trotting	0.01-0.11 sec
Double support period for trotting	0.2-0.6 sec

Applying these ranges sets borders on the search space in the quadruped design and hence has practical reasons. It can also be stated that they do not pose drastic limitations to lead to a conservative design. The body dimensions are limited from below for equipment carrying purposes. Link minimum values result from the minimum sizes of the hydraulic actuator hose attachment and mechanical bearing structures. Upper limits of the link lengths and body dimensions originate from the fact that a mule size animal structure is put as an upper limit for the TUBITAK project mentioned above. The mule size is reasonable from the ease of transportation on medium sized off-road vehicles like pickups. This can allow fast transportation of a quadruped to the task/operation area on roads and tracks leaving the true off-road/all-terrain task to the quadruped when it is landed on the ground. The mule size is versatile since there are many animals which run and jump and which display body and link lengths equal or shorter than a mule.

A simulation that handles the dynamics of the robot, the trajectory of the planned tasks, and the control of its motion is required. It should be prepared in a way that any change in aforementioned design variables can be applied and the results can be gathered. In this work, the preferred method of assessing these results and optimizing the design parameters accordingly is genetic algorithms.

With this optimization tool, first step taken was to solve the problem of vertical jumping in isolation. The design variables are tuned with GAs in order to get the best performing quadruped design for the first task only. The second step was to do the same for the trotting problem. This allows us to research and observe the kinematic qualities that make a quadruped robot successful in each motion separately. It also draws a picture of how a true jumper and a trotter would look like.

The final step is to work towards a quadruped design that is efficient in both the jumping and the trotting motions. This requires a way to assess the combined fitness of the robot including the two tasks. It also requires a decision to be made regarding the weights of each task, assigning relative importance to them. All the resulting data is used to validate the need for such work in designing a task specific robot. The data also enables analysis of the simulation and optimization tools used.

Chapter 4

4. GENETIC ALGORITHMS

4.1. Search Algorithms

The problem of quadruped kinematic design can be approached in multiple ways. The dynamic equations, trajectory generation and control of the robot are contained coded in a MATLAB Simulink environment. This defines the relation between the parameter values we aim to optimize and the resulting performance of the quadruped. One can attempt to convert this relation in to an equation or a transfer function and go for a direct solution. Such a direct relation may exist in robot kinematic parameter optimization problems with static nature, for example, for manipulation space improvement. However, for the assessment of quality of a dynamic motion like jumping and running of a highly nonlinear multi-DoF free fall manipulator, finding such a direct "formula" of assessment is not straightforward if ever possible. No such assessment technique is reported in the literature. However, the performance of the robot can be observed via simulations. In this work, a full-dynamics simulation method with a contact modeling technique suitable to simulate jumping and running motion is employed to model the robot's movements. Thus, the observations of the simulated robot's trajectories can serve as indicators of performance of a certain kinematic parameter set. Search algorithms employ the simulation results as performance criteria in order to find optimum values. Genetic algorithms (GAs) are methods based on the natural selection process of biology and they pose a robust technique suitable for the work at hand.

The quadruped simulation handles the dynamics of the 16 DoF robot (with four DoFs per leg) and the control algorithm for its trajectory tracking. We can treat the simulation like a black box system where we only work with the design variables as inputs and the motion data constitute the outputs.

Search algorithms are techniques used to solve optimization problems. They try to maximize an objective function termed a fitness function in the context of GAs in order to find optimal values for given variables. Genetic algorithms present a relatively sophisticated method among many existing search and optimization techniques. Some examples of these techniques and the strengths of GAs are described below. A more detailed source on this is in reference Beasley et. al., 1993.

Examples of search and optimization techniques:

- Random search: The most basic strategy which searches the design space randomly. This method is computationally ineffective for large search spaces or when the computation of the objective function is heavy.
- Gradient methods: Searching towards the direction in design space where the results are improving by calculating the gradient. This is referred to as "hillclimbing". Having multiple local maximas will make this method incorrect. It is also unusable when the design variables are discontinuous.
- Iterated search: Combines random search with gradient search. Moves to a random point at the end of every maximum find. Hard to define when to settle on a result and increases the computation time.
- Simulated annealing: Introduces a probability approach when deciding to move up or down in gradient methods' hillclimbing process. This brings a way to escape local maxima. The probability of moving down decreases at every move to finally settle at a maximum. Information about past solutions are not preserved and it's hard to conclude if the resulting solution is the best one found in the entire run.

The quadruped problem includes 8-12 parameters to be optimized depending on the task at hand (The jumping problem has 8 parameters and the trot problem is defined by 12 parameters). Multiple local maxima are assumed, as a robot can have similar leg lengths or weights with different link dimensions. The parameters, especially some link dimensions and resulting trajectory parameters, are expected to be partially coupled. These describe a high level optimization problem where the above listed approaches will be unsatisfactory. Working with an 8-bit resolution (this is the precision implemented in this study) on the design variables also makes the computation times significantly long with a brute force approach.

GAs are advanced search techniques that our research team already has practice with. They can be used to solve multilevel optimization problems with manageable computation times. Next section explains how they work to give more insight on their strengths (Adak, 2013, Adak et al., 2015, Akbas, 2012).

4.2. Working Principles

Genetic algorithms take their inspiration from natural selection of living organisms. Every parameter that is to be optimized is represented in binary. These are called genes. When all the genes are attached next to each other they form a chromosome. In our work for example, each quadruped, more specifically a set of kinematic design and trajectory parameters that defines a quadruped, is a chromosome. Every variable in it is a gene. Multiple chromosomes form a population. The purpose is to carry the desired genes from population to population through iterations called generations and reach a final population. The solution chromosome with the optimum genes will be in this final population.

The process starts with a population of chromosomes containing randomly generated genes. In order to decide which chromosomes carry the desired genes, a fitness function must be devised. Given a particular chromosome, the fitness function returns a single numerical value that is supposed to indicate the merit or effectiveness of the chromosome. The algorithm's end goal is to maximize this fitness value.

The fitness function is the most crucial aspect of any GA. As it is discussed further on, there are multiple parameters in a GA. However, research shows that GAs are robust enough methods and these parameter values are not critical (Beasley et al., 1993). The success of the algorithm mainly depends of the strength of its fitness function.

"The general rule in constructing a fitness function is that it should reflect the value of the chromosome in some "real" way." (Beasley et al., 1993). What is referred to here as "real" is a value that has a physical meaning. Since we are dealing with a robot's motion this is ideal for us and contributes to the decision process of using GAs for our work. For the task of vertical jump, the fitness function is simply chosen as the peak height of the robot body. For the task of trotting motion, the velocity is employed as the fitness function. Additional considerations like robot balance during movement is also incorporated using body orientation data.

Each individual quadruped design defined by a chromosome goes through the simulation and the results are assessed by the fitness function. Fittest individuals have the highest fitness function values. The next step is to form the next generation. This process involves multiple steps. The first step is to pick a number of fittest individuals from the current population and directly moving them on to the new population. These are called elite individuals and they have the highest fitness function values of the current population. Then a number of the fittest individuals are picked and paired up for the cross-over process as shown in Figure 3.1. In cross-over, each individual in a pair of chromosomes are split into two parts of “head” and “tail”. Then the “head” part of the first individual is combined with the “tail” part of the second individual and vice versa to form up two new chromosomes. The bit where this split happens is chosen randomly for each pair of chromosomes. These new individuals are then added to the new population.

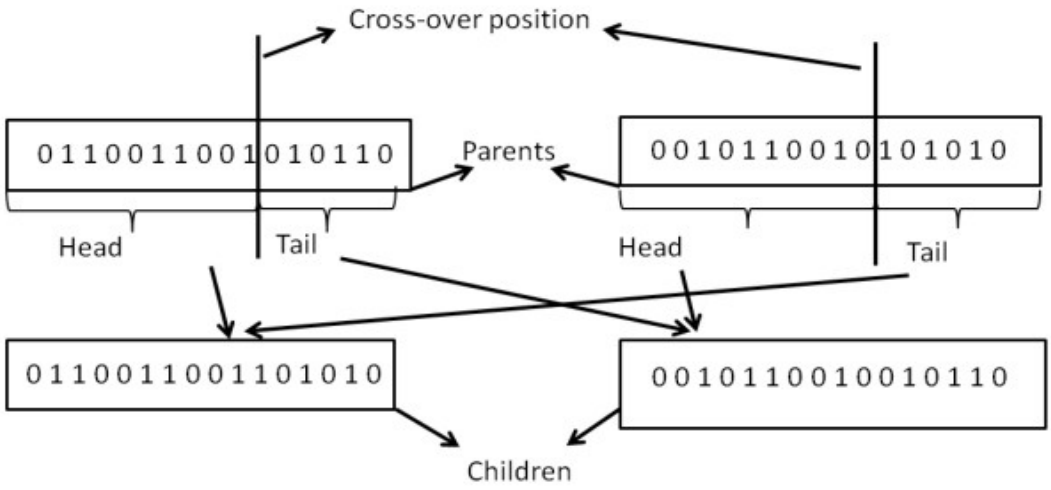


Figure 4.1. Visual description of the cross-over process.

The third step is called mutation. In the mutation step, a number of individuals are chosen from the current population and a random bit in their chromosome is changed from “0” to “1” or from “1” to “0”, whichever applies. This is displayed in Figure 4.2. These individuals are also chosen randomly and after they are mutated they are placed into the new population. This population now has elite individuals, cross-over individuals, and mutated individuals in it. The remaining spots are filled with randomized individuals similar to the first population.

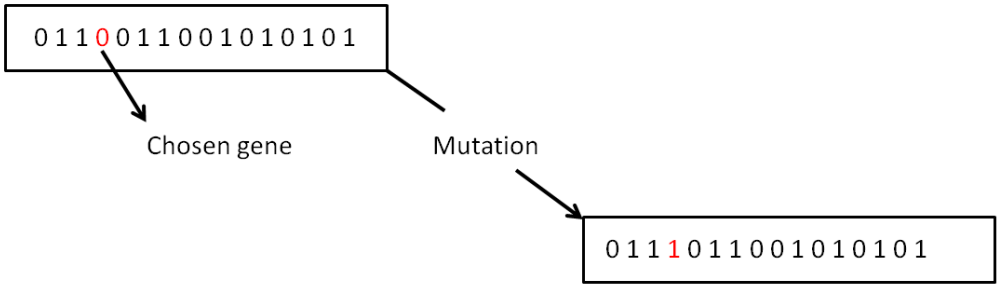


Figure 4.2. Visual description of the mutation process.

After a new population is formed it goes through the assessment step with the fitness function again. New generations are continued to be created until the iteration is done a sufficient amount of times to get a convergence. The fittest individual of the final population is the solution found by the algorithm. The values chosen for the mentioned genetic algorithm parameters are displayed in Table 4.1.

Table 4.1. Genetic algorithm parameter values.

Parameter	Value
Individuals per population	20
Number of generations	16
Elite individuals per generation	2
Cross-over individuals per generation	8
Mutations per generation	2

In order to efficiently reach a global maximum, a search algorithm uses two techniques: exploration and exploitation. If the search space is the NxN imaginary space where N is the number of design variables you have, a successful exploration is investigating the areas of this space (Beasley et al., 1993). All the random elements of the GAs; generating random individuals in populations, random cross-over cutoff bits, and the mutation process

contributes to exploration. Regardless of the fitness information this allows the handling of unknown areas in the search space. This is for preventing the solution from converging to a local maximum.

Exploitation technique is to get better results using the previous ones. The algorithm is meant to make use of the information gathered from visiting a point on the search space to make decision on where to check next. This is achieved in GAs with the transfer of genes to the future generations in the form of elite individuals and cross-over individuals. Successful genes in a population have higher chances to survive through generations and this results in getting fitter and fitter until they reach a peak.

Using a search algorithm allows us to use simulation results as design evaluation criteria. GAs are suitable methods to be used for this purpose. They can be used in our multilayer optimization problem with potential discrete search space. This is called a multilayer optimization because multiple optimization problems are solved concurrently. Having physical values involving the motion of the robot means that strong fitness functions can be employed. Computational times for the simulations and for the overall GA routines are also suitable for the scope of a thesis work.

4.3 The Chromosomes

The chromosomes structured for the three optimization problems; jumping, trotting, and combined jumping and trotting are illustrated in Figure 4.3. In this figure, D_s stands for the leg attachment distance. H_p and H_l are the jump pull length and jump loose height, respectively. ΔX_s and Z_s are totting spatial parameters. The former stands for step size and the latter is the step height. T_q and T_d are the quad support and dual support periods, respectively. All the parameters above are quantized into 8-bit binary values. The 8-bits provide adequate resolution, in the order of mms in the case of spatial parameters and 0.005 seconds in the temporal ones.

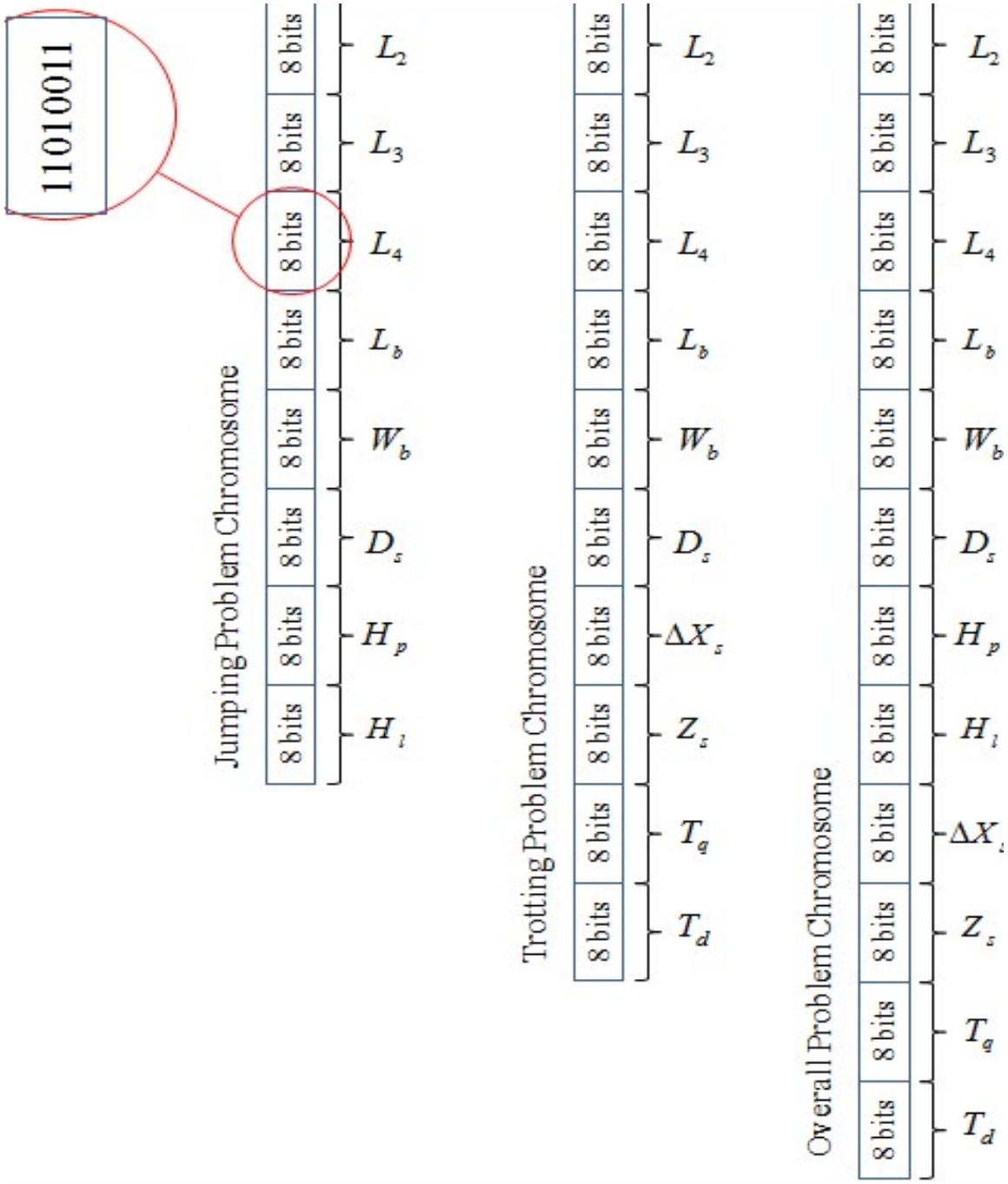


Figure 4.3. The chromosomes of the three optimization problems

The trajectory parameters step height and step length specific to the trotting task are added to the GA tuned design parameters (the trotting chromosome) so that optimum stepping trajectories are generated to get the full benefit out of the changes occurring on the link sizes of the robot. For the overall fittest design, since both tasks are involved in this optimization,

all the task specific trajectory parameters which are jump pull length, jump loose height, step height, step length, double support period, and quad support period are included in the algorithm.

4.4 Fitness Functions

For the vertical jumping task, the obvious fitness indicator is the peak jump height. This is defined in our work as the distance between the initial height of the robot body center of mass (before it pulls itself down) to the maximum height the robot body center of mass reaches during its whole motion. As mentioned above, a fitness function that takes values with physical meaning is strong and makes the genetic algorithm reliable. The peak jump height distance a quadruped design can reach in a simulation is one such value. Other than the jump height, a value that can be considered for fitness evaluation is the balance of the robot. For this task, we decided that a jump is satisfactory if the robot stabilizes on its four feet after the landing. A robot that loses its balance and falls can be identified by its body center of mass position and body orientation. This is reflected on the fitness function by harsh penalties causing the individual to not be considered for the next generation. The overall fitness function for this task is

$$f = \begin{cases} \text{if successful landing} & h \\ \text{if loss of balance} & h/2 \end{cases} \tag{4.1}$$

Where f is the fitness value and h is the peak jump height. The decision for loss of balance is made by checking if the robot body orientation flipped upside down or not.

The second task is planned to observe the locomotion capabilities of the quadruped robot through a motion of trotting gait. Fitter individual designs for this motion are expected to move faster without losing balance significantly. Going by what we know of animals in nature or even by human athletes, the key to faster locomotion is having bigger strides in ones steps. This is limited physically by the distance a foot can cover from the point it takes off the ground to the furthest point it can reach on the ground. Having longer legs contributes to having a bigger stride. Aside from the distance a single step can cover, the speed that the robot

can move its legs is also a main factor that determines the horizontal (locomotion) velocity. The torque restrictions exist for this task as well and they determine the speed the robot can swing its legs.

It is intuitive to consider that the balance of the robot also plays a role in locomotion speed. The more stable a quadruped is in motion, the longer it can afford to stay on its two feet. A balanced robot will also spend less time on its four legs between steps to position its body and regain balance. Attaching the legs farther apart on the body should increase the size of the support polygon and make for a more stable structure. In order to exploit this change in robot structure trajectory parameters double support period and quad support period are added as trotting task specific design parameters to the algorithm.

The main focus of this task is to maximize the physical velocity of the quadruped. This is evaluated in the fitness function by taking the distance the robot body center of mass covers on the horizontal axis after trotting five seconds. Similar to the jumping task, this metric makes a strong fitness function for genetic algorithms since it represents a physical value. For the quadruped robot, it is also important that the trotting motion is smooth, meaning that the robot body moves in a stable fashion. It is undesirable if the robot orientation oscillates a lot. This is evaluated in this work by adding all the absolute robot rotation on the roll axis (walking direction) during the entire trotting motion. This value is used to penalize the corresponding quadruped design in its fitness function. The resulting fitness function is

$$f = d_{total} + \frac{K}{\int |\gamma(t)|} dt \tag{4.2}$$

where d_{total} is what is covered on the horizantal axis in five seconds and γ is the roll angle data around the trotting axis, measured in radians. K is a gain for weighting the covered distance and roll angle integral in the fitness function. Its manually tuned value is 1.

Both the vertical jumping and the trotting capabilities of the robot are optimized in the overall performance GA tuning and therefore each individual goes through the jumping simulation first and then the trotting simulation. The variables used in the fitness function of this optimization problem are the peak height calculated from initial body position for the

vertical jumping task and the distance covered in five seconds for the trotting task. These are combined to formulate the following fitness function

$$f = W \times h + d + \frac{K}{\int |\gamma(t)|} dt \tag{4.3}$$

where h is the peak height of the jumping simulation, d is distance covered on the horizantal axis in five seconds during the trotting simulation, and γ is the body angle about the locomotion axis for the trotting simulation. W is introduced as a weighting parameter and determines the relative importance of the jumping task compared to the trotting task. Increasing this weight will result in individuals who jump higher but trot slower, while decreasing it will result in individuals who jump lower but trot faster.

Chapter 5

5. SIMULATION

Given the initial conditions and the kinematic and trajectory parameters of the robot described in previous sections, the simulation yields the complete trajectories of the robot. When evaluating a quadruped design with certain kinematic properties, the data from the vertical jumping or the trotting trajectory (the peak height reached and velocity) are employed. The simulation data are to be assessed by a fitness function and put to use in the GA. The nature of the optimization process constantly changes the kinematic design of the robot which means that the simulation needs to be robust in all of its building blocks; dynamics, trajectory planning and control.

Proper modeling of the contact forces is especially important for the jumping simulation of the quadruped robot. For this case, penalty based contact models are not suitable due to the drawbacks they have in calculating impact forces. It is necessary to increase the simulation durations (because of the primary need of decreasing simulation cycle times) in order to get sufficiently accurate results using this contact modeling method. The evaluation of quadruped designs using search algorithms is already a computationally complex task, which makes the simulation durations important. In a work where computationally heavy genetic algorithms are in use and where dynamic movements quadruped jumping and trotting tasks are involved, exact contact models instead of the penalty based contact models are required to be used.

A number of contact modeling techniques are reported recently rely upon penalty techniques (Diolaiti et al., 2005 and Duan et al., 2018). There are inadequate for the task in the thesis because of the arguments mentioned in the above paragraph. More sophisticated techniques for contact force computation are reported too. However, they are application specific being coupled with the particular task studied. (Herzog et al., 2015, Herzog et al., 2016, and Mason et al., 2017)

The free body dynamics of the quadruped is handled in this thesis by a technique called Articulated Body Method (ABM) (Mirtich, 1996; Featherstone, 1999) which is suitable for the computation of exact contact forces and which presents a computationally efficient integration algorithm. The environmental contact force calculations are done by a Gauss-Seidel like exact solution method (Jorden et al, 1998). The ABM and Gauss-Seidel algorithms together form a framework of dynamics and exact contact force computation. Embedded in a numerical integration environment, this framework provides the robustness required for the changing parameters and dynamic motion in the GA simulation iterations.

Zero Moment Point (ZMP) based generation is used for the trajectory planning of the trotting motion (Akbas et al., 2012). Joint motion is controlled by a combination of a joint space PID method and a task space admittance-based force control scheme.

5.1. Articulated Body Method (ABM)

ABM is an algorithm similar to the Newton Euler algorithm that is used to solve the forward dynamics of a robot manipulator. It belongs to a family of methods called structurally recursive algorithms. Given the initial positions, velocities, and external forces of a body, the aim is to get the joint and body accelerations. Once the accelerations are known, the input information for the next time frame can be calculated by integration and the process can be repeated.

Newton Euler algorithm builds the inertia matrix of the manipulator and then inverts it to find the solution for accelerations. For an n link manipulator, this has a computational complexity of $O(n^3)$. The primary difference of ABM compared to similar solutions is the fact that the inverse of the inertia matrix is not involved in the calculations, lowering the complexity to O(n). The computational complexity and the resulting running time of the simulation is important for this work considering that a simulation run for each individual in each population for multiple generations have to be performed. This is the motivation behind using ABM as the dynamic solution method in this thesis.

In a serial linkage, if a link is picked and the subchain of connected links is considered, the link is associated with a handle number and the link chain is referred to as an articulated body in ABM. The algorithm works by starting from an initial link alone, solving for it. Then its inboard link (the next link up in the serial linkage towards the base link) is added to the calculations. This is now an articulated body consisting of two links. Successively adding all the inboard joints until the entire linkage defines the articulated body completes the solution. The result is the elements defining the relation between spatial accelerations of all links and the spatial forces applied to them. This relation is given in the equation below:

$$\hat{f}_i^I = \hat{I}_i^A \hat{a}_i + \hat{Z}_i^A \tag{5.1}$$

where i is the handle for the particular articulated body. $\hat{f}_i^I$ is the spatial force given by the 6x1 matrix $\left[f_{ix}^I f_{iy}^I f_{iz}^I \tau_{ix}^I \tau_{iy}^I \tau_{iz}^I\right]^T$ and $\hat{a}_i$ is the spatial acceleration given by the 6x1 matrix $\left[a_{ix}^I a_{iy}^I a_{iz}^I \alpha_{ix}^I \alpha_{iy}^I \alpha_{iz}^I\right]^T$. $\hat{I}_i^A$ is called the spatial articulated inertia (6x6 matrix) of link i. Here, the term articulated means that it is an inertia matrix describing the entire subchain of link i, not the inertia of the link itself. It does not depend on joint velocities or accelerations.

$\hat{Z}_i^A$ is called the spatial articulated zero-acceleration force (6x6 matrix) of link i. It is the force required to keep link i without acceleration, when exerted from the inboard joint. It does not depend on joint acceleration. The fundamental idea of the method is to start from a single link to make these calculations and then build up the I and Z matrices by adding in board links one by one.

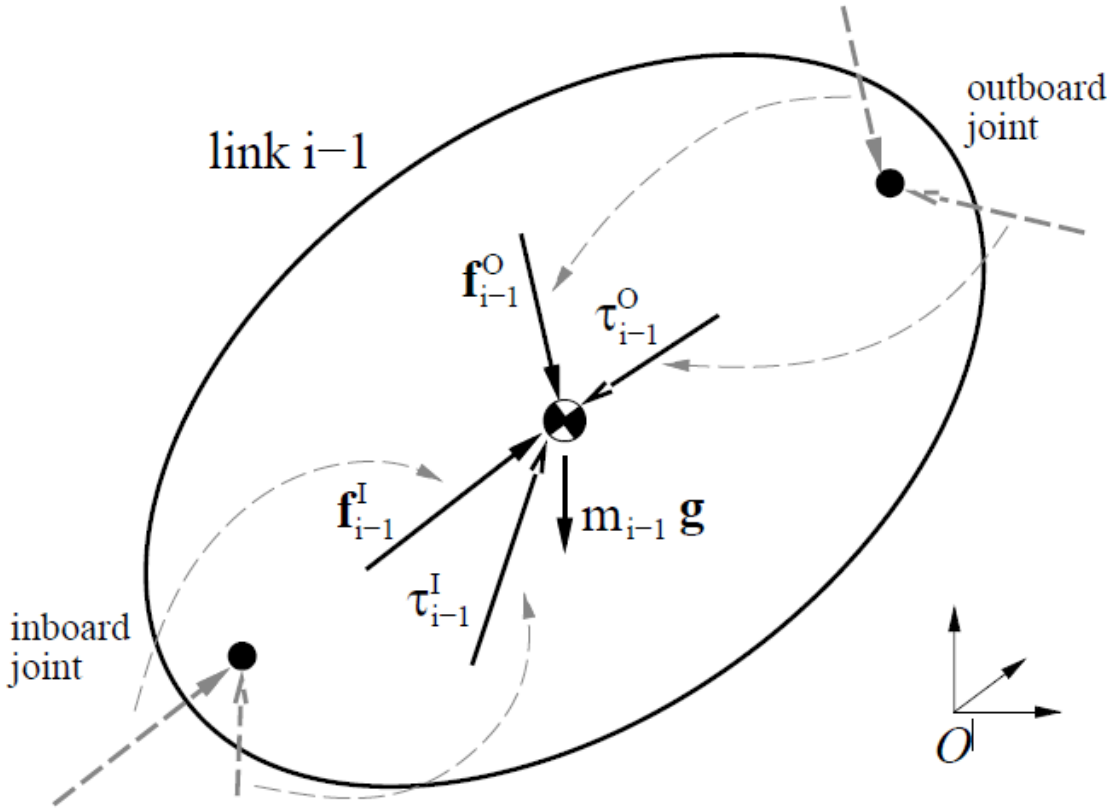


Figure 5.1. The free body diagram of a link in the serial linkage.

In Figure 5.1, all forces and torques acting on a link in the serial linkage is shown. For a given link $i-1$, $m_{i-1}g$ is the gravity acting on the link. f^{I}_{i-1} and τ^{I}_{i-1} are the force and torque vectors acting on the link from its inboard joint. f^{O}_{i-1} and τ^{O}_{i-1} are force and torque vectors acting on the link from its outboard joint. All the vectors pass through the origin of the link frame. For the end link (tool, hand or foot side is referred to as "end" link) of the serial linkage, the outboard joint (next joint towards the end link, so not connected in this case) forces and torques are zero.

ABM involves four main steps while computing the dynamics of a serial linkage:

1) A forward recursion step, where it starts from the base link and computes linear and angular velocities of all links.
2) Same forward recursion step, calculating the Coriolis vector for each link. For each link, articulated inertia matrix and articulated zero acceleration force vector are

calculated. This is done isolated for each link considering only the inertia, gravity and velocity information.

3) A backwards recursion step, where it starts from the tip link and builds the articulated inertia matrix and the articulated zero acceleration force vector for the whole articulated body. All the joint forces and their effects on each link are considered here.
4) Second forward recursion step, again starting from the base and this time calculating all the joint accelerations and the spatial acceleration vectors for all the links.

The entire process consists of two forward recursions and a backwards one. At the end we end up not only computing the joint and link accelerations, but also building a matrix and a vector that describes the relationship between the external forces on the system and the joint and link accelerations (articulated inertia matrix and articulated zero acceleration force vector). The rest of this section explains these steps in detail with the associated equations.

The first step starts from the base link. If this is a fixed link, all the velocity and accelerations will be zero. In a floating base link like a walking robot body, these are either initial conditions that need to be supplied or they are the outputs of the previous iteration of the algorithm. Either way they are known for the base link. The equations for the following links are given below. They are calculated for i = 1 to n where n is the total number of links.

If the joint in between links i and $i-1$ is prismatic we have

$$\boldsymbol{\omega}_i = \mathbf{R}\boldsymbol{\omega}_{i-1} \tag{5.2}$$

$$\mathbf{v}_i = \mathbf{R}\mathbf{v}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{r} + \dot{q}_i \mathbf{u}_i \tag{5.3}$$

If the joint in between links i and $i-1$ is revolute the expressions are

$$\boldsymbol{\omega}_i = \mathbf{R}\boldsymbol{\omega}_{i-1} + \dot{q}_i \mathbf{u}_i \tag{5.4}$$

$$\mathbf{v}_i = \mathbf{R}\mathbf{v}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{r} + \dot{q}_i (\mathbf{u}_i \times \mathbf{d}_i) \tag{5.5}$$

In the above equations, i identifies the current link and $i-1$ is for the inboard link. $\boldsymbol{\omega}_i$ and $\mathbf{v}_i$ are the angular velocity and the linear velocity of link i respectively and they are the outputs of the computation step. $\dot{q}_i$ is the scalar velocity of the joint in between the two links and it's the input. $\mathbf{R}$ is the rotation matrix from link $i-1$'s coordinate frame to link i's coordinate frame. The vector that connects the origin of link $i-1$ to the origin of link i is the vector **r**. The vector that denotes the axis of the joint is $\mathbf{u}_i$. The vector $\mathbf{d}_i$ connects this axis to the origin of link i. These are all defined based on the current configuration of the articulated body.

The second step of the algorithm is calculating the Coriolis vectors and isolated articulated inertia matrix and articulated zero acceleration force vector for all links starting with i = 1 and ending with n. This can be done in the same forward recursion as the one in step one. The Coriolis vector calculations are given in equations (5.6) and (5.7).

$$\hat{c}_i = \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_i) + 2\boldsymbol{\omega}_{i-1} \times v_i \end{bmatrix} \tag{5.6}$$

$$\hat{c}_i = \begin{bmatrix} \boldsymbol{\omega}_{i-1} \times v_i \\ \boldsymbol{\omega}_{i-1} \times (\boldsymbol{\omega}_{i-1} \times \mathbf{r}_i) + 2\boldsymbol{\omega}_{i-1} \times (v_i \times \mathbf{d}_i) + v_i \times (v_i \times \mathbf{d}_i) \end{bmatrix} \tag{5.7}$$

Equation (5.6) is the expression for the Coriolis vector for link i for the case when joint i is prismatic and equation (5.7) is for the case when joint i is revolute. v_i is joint

velocity vector. The equations for isolated articulated zero acceleration force vector and articulated inertia matrix are as follows:

$$\hat{\mathbf{Z}}_i^A = \begin{bmatrix} \mathbf{Z}_f \\ \mathbf{Z}_T \end{bmatrix} = \begin{bmatrix} -m_i \mathbf{g} \\ \boldsymbol{\omega}_i \times \mathbf{I}_i \boldsymbol{\omega}_i \end{bmatrix} \tag{5.8}$$

$$\hat{\mathbf{I}}_i^A = \begin{bmatrix} 0 & M_i \\ \mathbf{I}_i & 0 \end{bmatrix} \tag{5.9}$$

where M_i is the mass of link i and $\mathbf{I}_i$ is its inertia tensor.This means that $\hat{\mathbf{I}}_i^A$ depends only on the mass properties of the link. $\hat{\mathbf{Z}}_i^A$ depends only on the gravitational force and the angular velocity since it is calculated isolated for the particular link only, with no outboard links attached. Since the end link of the articulated body does not have any outboard link attached, these calculations yield the complete articulated inertia matrix and articulated zero acceleration force vector for link n .

The third step is to complete these two elements for the entire articulated body. Starting from the end link where there are no outboard joints and therefore the calculation is straightforward, $\hat{\mathbf{I}}_i^A$ and $\hat{\mathbf{Z}}_i^A$ are calculated. At each step, the articulated inertia matrix and articulated zero acceleration force vector for an articulated body with link i as handle are obtained. When i is equal to 1, the whole serial linkage is covered. The equations for these calculations are presented below.

$$\hat{\mathbf{I}}_{\mathbf{i-1}}^{\mathbf{A}} = \hat{\mathbf{I}}_{\mathbf{i-1}}^{\mathbf{A}} + {}_{\mathbf{i-1}}\hat{\mathbf{X}}_{\mathbf{i}} \left[\hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} - \frac{\hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{s}}_{\mathbf{i}} \hat{\mathbf{s}}_{\mathbf{i}}' \hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}}}{\hat{\mathbf{s}}_{\mathbf{i}}' \hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{s}}_{\mathbf{i}}} \right]_i \hat{\mathbf{X}}_{i-1} \tag{5.10}$$

$$\hat{\mathbf{Z}}_{\mathbf{i-1}}^{\mathbf{A}} = \hat{\mathbf{Z}}_{\mathbf{i-1}}^{\mathbf{A}} + {}_{\mathbf{i-1}}\hat{\mathbf{X}}_{\mathbf{i}} \left[\hat{\mathbf{Z}}_{\mathbf{i}}^{\mathbf{A}} + \hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{c}}_{\mathbf{i}} + \frac{\hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{s}}_{\mathbf{i}} \left[Q_i - \hat{\mathbf{s}}_{\mathbf{i}}' (\hat{\mathbf{Z}}_{\mathbf{i}}^{\mathbf{A}} + \hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{c}}_{\mathbf{i}}) \right]}{\hat{\mathbf{s}}_{\mathbf{i}}' \hat{\mathbf{I}}_{\mathbf{i}}^{\mathbf{A}} \hat{\mathbf{s}}_{\mathbf{i}}} \right] \tag{5.11}$$

Here, Q_i is scalar joint force/torque. $\hat{\mathbf{s}}_\mathbf{i}$ is the spatial joint axis of joint i. ${}_{\mathbf{i-1}}\hat{\mathbf{X}}_\mathbf{i}$ is the spatial transformation matrix from the origin of frame i to $i-1$. These are defined below in equations (5.12) to (5.14).

For a revolute joint i,

$$\hat{\mathbf{s}}_\mathbf{i} = \begin{bmatrix} \mathbf{0} \\ \mathbf{u}_\mathbf{i} \end{bmatrix} \tag{5.12}$$

If the joint is prismatic, the equation becomes

$$\hat{\mathbf{s}}_\mathbf{i} = \begin{bmatrix} \mathbf{u}_\mathbf{i} \\ \mathbf{u}_\mathbf{i} \times \mathbf{d}_\mathbf{i} \end{bmatrix} \tag{5.13}$$

The transformation matrix is calculated as

$${}_{\mathbf{i-1}}\hat{\mathbf{X}}_\mathbf{i} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\mathbf{r}\,\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ -\mathbf{rR} & \mathbf{R} \end{bmatrix} \tag{5.13}$$

where r is the vector from the origin of frame i to the origin of frame $i-1$ and R is the rotation matrix from frame i to frame $i-1$.

In the above equations, Q_i is an input to the system, $\hat{\mathbf{I}}_\mathbf{i}^\mathbf{A}$ and $\hat{\mathbf{Z}}_\mathbf{i}^\mathbf{A}$ come from previous calculations, and the rest is defined for the current configuration of the system. Spatial articulated inertia matrix and the spatial articulated zero acceleration vector for the whole body are calculated at the end of these calculations.

The fourth and final step is to calculate the joint and link accelerations. It is a forward recursion starting from the base link. Below are the equations for these calculations:

$$\ddot{q}_i = \frac{Q_i - \hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{a}}_{i-1} - \hat{\mathbf{s}}_i'(\hat{\mathbf{Z}}_i^A + \hat{\mathbf{I}}_i^A \hat{\mathbf{c}}_i)}{\hat{\mathbf{s}}_i' \hat{\mathbf{I}}_i^A \hat{\mathbf{s}}_i} \tag{5.14}$$

$$\hat{\mathbf{a}}_i = {}_i\hat{\mathbf{X}}_{i-1}\hat{\mathbf{a}}_{i-1} + \hat{\mathbf{c}}_i + \ddot{q}_i \hat{\mathbf{s}}_i \tag{5.15}$$

Here $\ddot{q}_i$ is the acceleration of joint i and $\hat{\mathbf{a}}_i$ is the spatial acceleration of link i origin. As it can be seen in (5.15), $\hat{\mathbf{a}}_i$ uses $\hat{\mathbf{a}}_{i-1}$ in its calculations. If the base link is fixed, $\hat{\mathbf{a}}_1$ is simply zero, but in our robotic application the base link is the body which is a floating linkage. For a floating linkage, $\hat{\mathbf{a}}_1$ is calculated as follows:

$$\hat{\mathbf{a}}_1 = -(\hat{\mathbf{I}}_1^A)^{-1}\hat{\mathbf{Z}}_1^A \tag{5.13}$$

In the above equation, $\hat{\mathbf{I}}_1^A$ is a 6x6 spatial inertia matrix belonging to the first link and thus the inversion of it does not change the O(n) computational complexity of the algorithm.

A final modification has to be done on the algorithm of ABM for our application. All the calculations so far are for a serial linkage. A quadruped robot is a treelike linkage where the four legs branch out from the robot body. When doing the steps involving forward recursions, the only change this makes is repeating the process four times for the four legs starting from the body and going to the foot link. For the backwards recursion, a slight modification needs to be carried out when calculating the articulated inertia matrix and the articulated zero acceleration vector for the body link. In (5.10) and (5.11), the right side of the equation has to be added four times where i changes to the base link of each leg for each of these additions. This way, the inertia and force information of all four legs are included in the calculations.

To summarize, the inputs of ABM are joint and link velocities, joint and link positions, and joint torques. It is assumed that the system kinematic arrangement is known. The algorithm is of $O(n)$ complexity and goes through two forward and one backwards recursions. The outputs of the algorithm are joint accelerations and link spatial accelerations. During this process $\hat{I}_i^A$ and $\hat{Z}_i^A$ in equation (5.1) are also calculated. These will be useful when calculating the contact dynamics in the next section.

5.2. Contact Forces

Calculations presented in the previous section solve the dynamics of a body when all the external forces are known. For them to be applicable to our work we need a way to know the ground contact forces. A computationally light and simple way to model this can be done by using penaly based methods. These model the ground interaction as a spring damper system that can be solved with related mechanical calculations. This method loses accuracy when the simulation cycle times are not very short (in the orders of tens of microseconds) and does not function properly when impact forces are involved (Erbatur, K & Kawamura, A, 2003). Impact forces are a significant part of the vertical jumping motion present in the framework of this thesis. Therefore, penalty based methods are not suitable and alternative methods that seek exact solutions have to be used.

The basis of the exact solutions is defining and satisfying the physical conditions the system is in. It starts by defining the contact points on the body. For our work, these are on the tips of the four feet of the robot. Point contacts are considered. Foot contact points are satisfactory to simulate the jumping and trotting motions and the robots contact with the ground during them. Since the dynamics of the robot body will be solved at every timestep of the simulation, calculating if a point is going to be in contact with a surface is straightforward.

Once a contact point is in contact with the ground, there are physical constraints that need to be applied on the system. If a point is in contact, its normal velocity to the contact

surface is zero since the feet of the robot can not penetrate the ground. The contact surface can only push the robot not pull it, and therefore the normal contact force is positive which also means that the normal acceleration is positive. For the tangential directions, friction rules apply. These contraints are displayed in following equations

$$v_k = 0 \tag{5.14}$$

$$a_k \geq 0 \tag{5.15}$$

$$f_k \geq 0 \tag{5.16}$$

where v_k is the relative velocity and a_k is the relative acceleration of k^{th} contact point with respect to the contact surface. f_k is the contact force acting on k^{th} contact point.

From the constraints, our problem definition is to find the contact forces that will make contact velocities zero. For this, the equation that relates the contact forces to the contact velocities is required. In the previous section, dynamics of the system were being solved for joint and link accelerations. Adding the effect of contact forces to these and integrating them over a timestep will yield the expressions of the velocities we are looking for. We can start the calculations from the equation of joint accelerations

$$\ddot{q} = \ddot{q}_{f=0} + \ddot{q}_{f\neq 0} \tag{5.17}$$

where $\ddot{q}_{f=0}$ are the joint accelerations without contact forces and $\ddot{q}_{f\neq 0}$ are the joint accelerations caused by only contact forces. The acceleration term $\ddot{q}_{f=0}$ is a result of joint torques and gravity. It's defined by the equation

$$\ddot{q}_{f=0} = I^{-1}(q)[\Gamma - b(q,\dot{q}) - g(q)] \tag{5.18}$$

where I is the inertia, Γ is the joint torque vector, b represents the Coriolis effects, and g is gravity (Ruspini and Khatib, 2000). This is explained in the previous section and calculated by ABM while omitting the existance of the contact forces.

In order to define $\ddot{q}_{f\neq 0}$, we assume a system with no joint torques, no joint velocities and no gravity, so that all the $\ddot{q}_{f=0}$ terms disappear. For this step, we aim to construct a matrix that relates contact forces to joint acceleration. ABM can be used for this task. If we apply a unit force on one axis (x, y or z) of one contact point and calculate the resulting joint accelerations through ABM, and then repeat this process for each axis on each contact point; placing the joint acceleration vectors next to each other will construct a matrix. This matrix multiplied by the contact forces will yield joint accelerations caused by them. ABM needs to be run three times the number of contact points for this matrix to be calculated. This is configuration dependent and needs to be performed at every simulation iteration.

With these new information, (5.17) can be written as

$$\ddot{q} = I^{-1}(q)[\Gamma - b(q,\dot{q}) - g(q)] + Af \tag{5.19}$$

where A is the newly calculated matrix. The next step is to write the same relation for contact point accelerations. Both joint accelerations and joint velocities contribute to the linear accelerations of a point on the body. For the part of the equation that corresponds to $\ddot{q}_{f=0}$, link accelerations are also already calculated by ABM. The only additional calculation is to carry the accelerations from feet origin to where the contact point is. The rest of the equation omits joint velocities and leaves only the joint accelerations in the calculations. There exist a Jacobian matrix J which transforms the joint accelerations to contact point accelerations. This is configuration dependent and can be solved by kinematic relations. The complete equation for contact point accelerations is

$$a_c = JAf + a_{f=0} \tag{5.20}$$

The product JA is also denoted by Λ.

The velocity equation is the above acceleration equation integrated over timestep dt as shown below:

$$v_c^t = \Lambda f dt + a_{f=0} dt + v_c^{t-dt} \tag{5.21}$$

If a contact point is in contact, the normal velocity vector component belonging to that point in v_c^t is zero. If it is not in contact, it is unknown before the calculations. Similarly, the contact forces corresponding to points not in contact are zero, and unknown for the rest. The positive force and no penetration constraints apply here. This poses a linear complimentary problem (LCP) and there are solvers to tackle it. The method employed in this thesis is an iterative Gauss Seidel approach (Chardonnet et al., 2006).

The method works by initially assigning estimated values to all unknowns and then solving for the unknowns one by one, updating the values, and repeating the process until a certain convergence is reached. The steps implemented in our application are outlined below.

1) Calculate the Λ matrix in (5.21) by running the algorithm of ABM a total of 12 times (3 axes times 4 feet contact points).
2) Initialize all the contact forces as zero. (5.21) can be solved now since Λ is calculated and $a_{f=0}$ is known from ABM.
3) Solve the equation for the first contact force using the estimated values for the other contact forces.
4) Apply the constraints. If the normal contact force is not positive set it to zero.
5) Limit the tangential forces to the maximum possible friction if they exceed it. We use Coulomb friction as an upper limit in these calculations.
6) Follow steps 3, 4, and 5 for the other contact forces. Always use the most recently calculated values for the rest of the contact forces in the equations.

7) Check if the desired convergence criteria is met. If not, return to Step 3 with the updated contact force values. If convergence criteria is met, the process is complete.

In Step 7, taking the norm of the contact force vector and comparing it to the norm from the previous iteration can be used as a convergence metric. After the contact forces are calculated, new joint accelerations of the system can be found through equation (5.19). This completes the dynamic solution of the system.

5.3. Trajectory Generation

For the vertical jumping task, the trajectory is generated for the robot body height. The height of the body is lowered according to the trajectory parameter body pull length and then it is pushed with maximum power to the body loose height. The form of the body height curve in the lowering and increasing this variable is shown in Figure 5.2. As can be seen from this figure, the motion features two "ramp" sections for the body height measured from the ground level. The lowering period T_p, the waiting period T_w and the rising period T_r are two time variables influencing the speed of the motion. H_b stands for the height of the body frame measured from the ground level in the figure. The rising period is assigned very low to "operate" the robot at the limits of its actuators' torque capacity. In other words, this body height ramp is steep, very close to a step reference (the rising period employed is 0.1s in this work). The ramp nature provides, however, some degree of smoothness when compared with step references. This has positive effects on the balance of the jumping motion. The lowering period has no speeding effect on the jumping motion, and hence it is chosen longer (0.5s). With this larger period the speeding up upwards motion starts with a well settled posture and the effects of the lowering motion on the jump are thus isolated. Once the body height curve

is planned the joint trajectories are generated through the application of inverse kinematics at every simulation cycle.

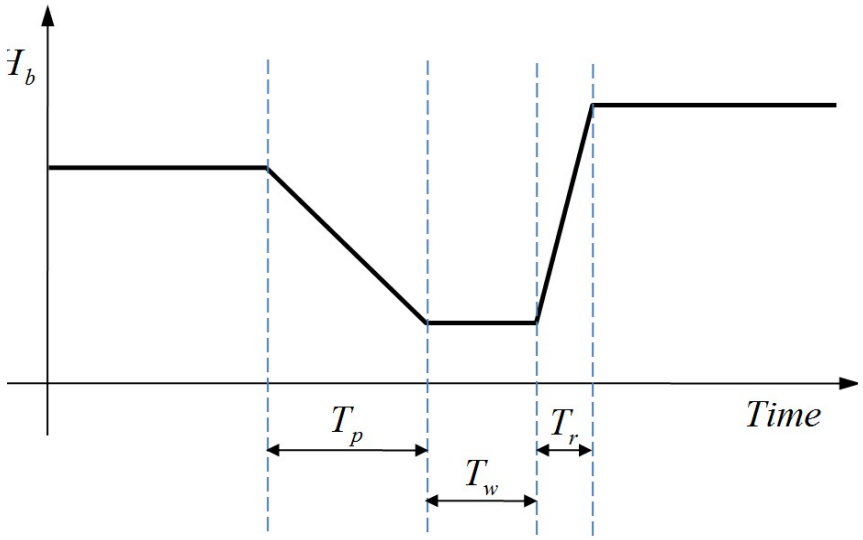


Figure 5.2. The body height reference position curve for the jumping task

The inverse kinematics routine is implemented for the four legs independently. The x, y, z positions of the foot tip and the angle the foot link makes with an axis normal to the ground enter the inverse kinematics algorithm as inputs. The joint angles of the leg are obtained as outputs (Fidan et al., 2011 and Adak, 2013).

The second task, trotting, is generated by a method based on the Zero Moment Point (ZMP) concept (Fidan et al., 2011) and the preview control technique (Akbas et al., 2012). ZMP method generates a trajectory to the center of mass (COM) of the robot body. The ZMP is defined as a point on the ground where no horizontal torque exists. In physical terms this means that the torque created on this point by the gravitational force of the robot mass is perfectly balanced by an equal force created on the opposite side by the inertial force from the robot acceleration. If this point lies on the support polygon formed by the robot feet on the ground, then the robot movement is stable. The position of the ZMP is planned for the trotting motion. Since the trotting is a known linear cyclic motion, future trajectory information can be utilized. This is handled by preview control. Once the ZMP trajectory is generated, the COM position reference of the robot body can be calculated. At this point, inverse kinematics routines are applied similar to the vertical jumping case and the references for the joints are generated.

5.4. Control

The generated trajectory is tracked by the use of PID control (proportional-integral-derivative controller) on all 16 joints independently. This control strategy is effective and achieves good control performance (low joint position errors) when external forces on the system are minimal or change steadily, like in the case of jumping motion. This kind of smooth contact force profiles, however, are not observed during the trotting motion of the quadruped. Trotting requires robot feet to constantly leave and land on the ground, creating frequent changes on the contact forces. In an attempt to minimize its trajectory error, the quadruped might push or land on the ground with its foot in a way that it jerks and shakes the body. To smooth this phenomenon, the method of force admittance control is incorporated to the system.

Admittance control is carried out by modeling a mechanical relation between the force applied to a surface and the resulting motion (admitted motion) with a desired impedance. This relation is expressed as

$$F = Ma + Bv + Kx \tag{5.22}$$

This equation can be regarded as a "force in-distance-out" relation. It is applied for the four legs independently. F stands for the vertical component of the contact force exerted on the ground by one particular foot. In the admittance ontrol scheme, the foot tip responds to this force by deviating from its vertical direction Cartesian position reference (as described in the selected shoulder frame). The amount of deviation is denoted by x in (5.22). v (for velocity) is the derivative of this displacement variable and a (for acceelration) is its second derivative. M stands for the desired mass, B is the desired viscous friction

coefficient and K is teh desired stiffness constant. M , B , and K define a desired smooth interaction behaviour for teh contacting bodies, in this case the foot and the ground. With the above mentioned relations between x , v , and a , (5.22) can be rewritten as

$$F = M\ddot{x} + B\dot{x} + Kx \tag{5.23}$$

Through the application of the Laplace transform with zero initial conditions as

$$F(s) = (Ms^2 + Bs + K)X(s) \tag{5.24}$$

The following transfer function can be obtained for the linear system in (5.22)

$$X(s) = F(s)\frac{1}{Ms^2 + Bs + K} \tag{5.25}$$

Note that this is a second order behaviour and its dynamics can be tuned via the undamped natural frequency and dampin coefficient parameters. Embedded in its characteristic equation as

$$\omega_n = \sqrt{\frac{K}{M}} \tag{5.26}$$

and

$$\zeta = \frac{B}{2\sqrt{MK}} \tag{5.27}$$

(5.25) is used to compute the desired deviation from the foot vertical position reference and the modified version of the reference is computed as

$$Z_{ref_\mu} = Z_{ref} - x \tag{5.28}$$

In this work the values of M , B and K are tuned by hand with the parameters of the base design.

The contact forces that apply on the feet are calculated as in Section 5.2. With the application of the admittance control, the robot will "fight" against the ground less and the potential impact on the body from the shoulder of that foot will be smoothed, resulting in a steadier motion overall.

5.5. Hydraulic Actuators

A hydraulic cylinder is connected to a link it is actuating at a specific point and the angle of actuation will change depending on the angle between the link and the cylinder. This means that with the same force applied through a certain pressure, the torque on the system will vary since this angle is not constant and it changes during the motion of the robot.

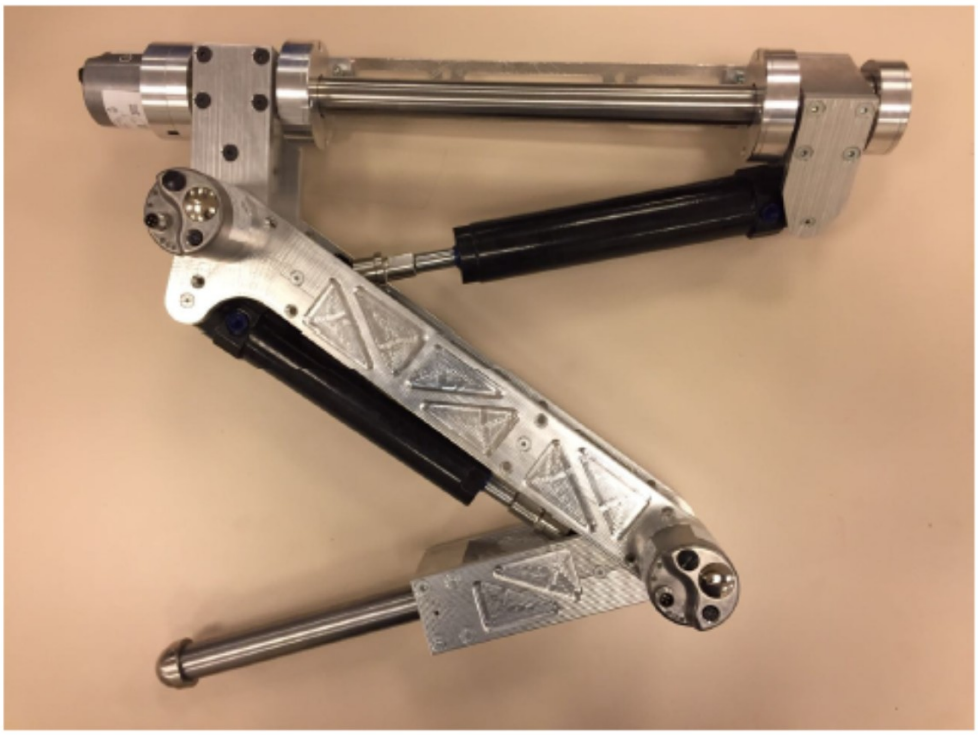


Figure 5.3. Quadruped leg hip and knee hydraulic actuation system. The leg belongs to the quadruped prototype of the TUBITAK 114E618 project

In Figure 5.3, black hydraulic cylinders can be seen on a prototype quadruped leg. For our simulations it is necessary to limit the available torque on each joint in order the accurately model our dynamic limitations. In order to correctly calculate the maximum torque available for a certain joint at a certain configuration, the angle between the actuating cylinder rod and the link needs be known. The position of the point where the cylinder is attached to the link is designed to be 1/5th of the link length away from the joint axis. For every iteration of the simulation, all the joint angles and the orientation of the robot are obtained by the simulation algorithm. With this knowledge what we have is a geometry problem which has a direct solution. After calculating the angle, the torque limit of each joint actuator is modified at every iteration accordingly.

Chapter 6

6. RESULTS

Quadruped robot research began at Sabancı University before this work and an initial robot design (Fidan et al., 2011) was obtained without the GA optimization which is the subject of this thesis. Similar to the other quadrupeds in literature, the design guidelines were inspiration from the nature and considerations of the sizes of various equipments to be housed by the robot body. This design was used in tests of our simulation environment and it was successful in simulation at performing both tasks at hand. Its results can be used as a means of comparison for our current research of kinematic arrangement optimization presented in this thesis. We are terming this initial design as the base design. It should also be noted that the link and body proportions of this base design are quite similar to the ones of the other hydraulically actuated quadrupeds robots reported in literature (Boston Dynamics, 2014 and Semini 2010). Hence the genetic tuning results have the potential of providing insight for modifications in other robots' kinematic parameters too.

The aim of this work is to explore what kind of performance improvement can be achieved by involving an optimization step in the kinematic design of a quadruped. The GA steps described in Chapter 4 are followed for the three tasks and results are recorded over generations. This chapter presents and evaluates the obtained data. The results shown in this section are of three designs; fittest jumper, fittest trotter, and fittest overall design. The final results for the three designs, together with the base design are shown with their animation window screenshots in Figure 6.1. Figure 6.2-6.6 presents drawings obtained by a CAD package. The CAD package images are to scale, and provide a means for qualitative comparison of the designs. Detailed analysis of the data of GA generations which converge to the optimized results in Figures 6.1 and 6.2 is presented in the sections which follow.

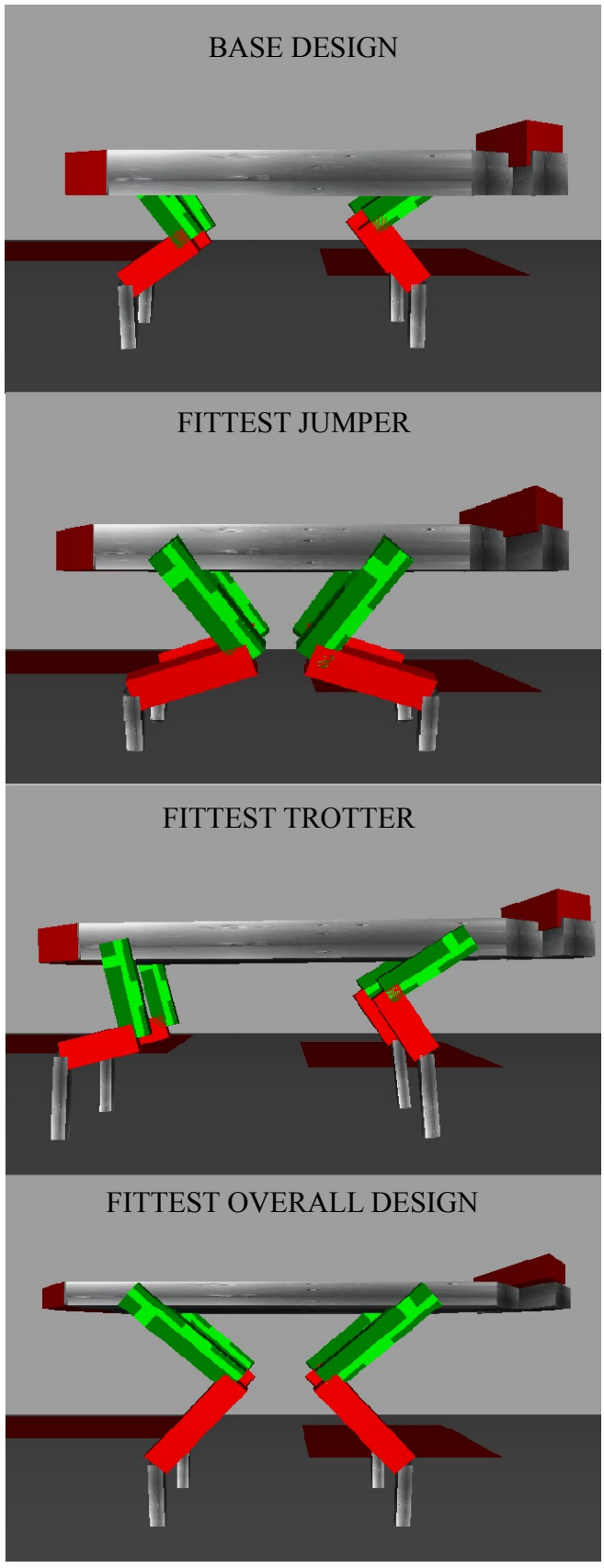


Figure 6.1. Animation platform look of the designs in order from top to bottom; base design, fittest jumper, fittest trotter, and fittest overall

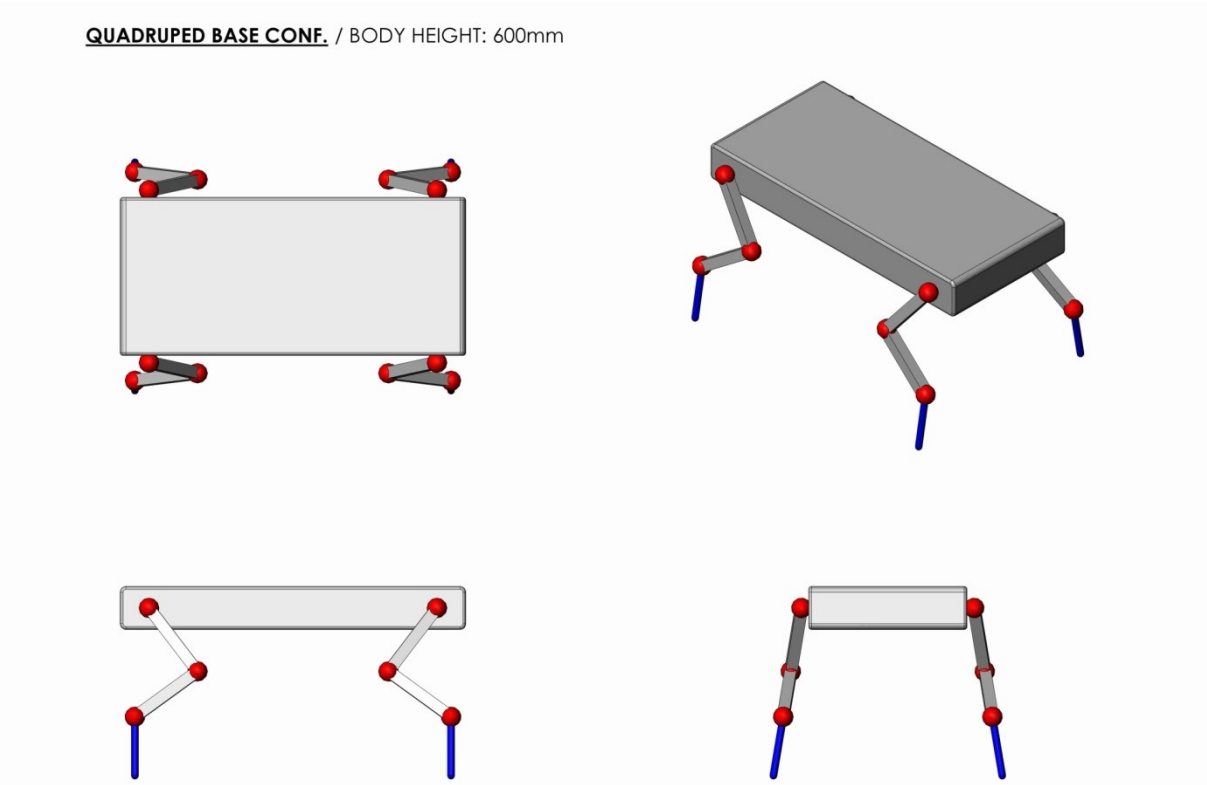


Figure 6.2. Base design views

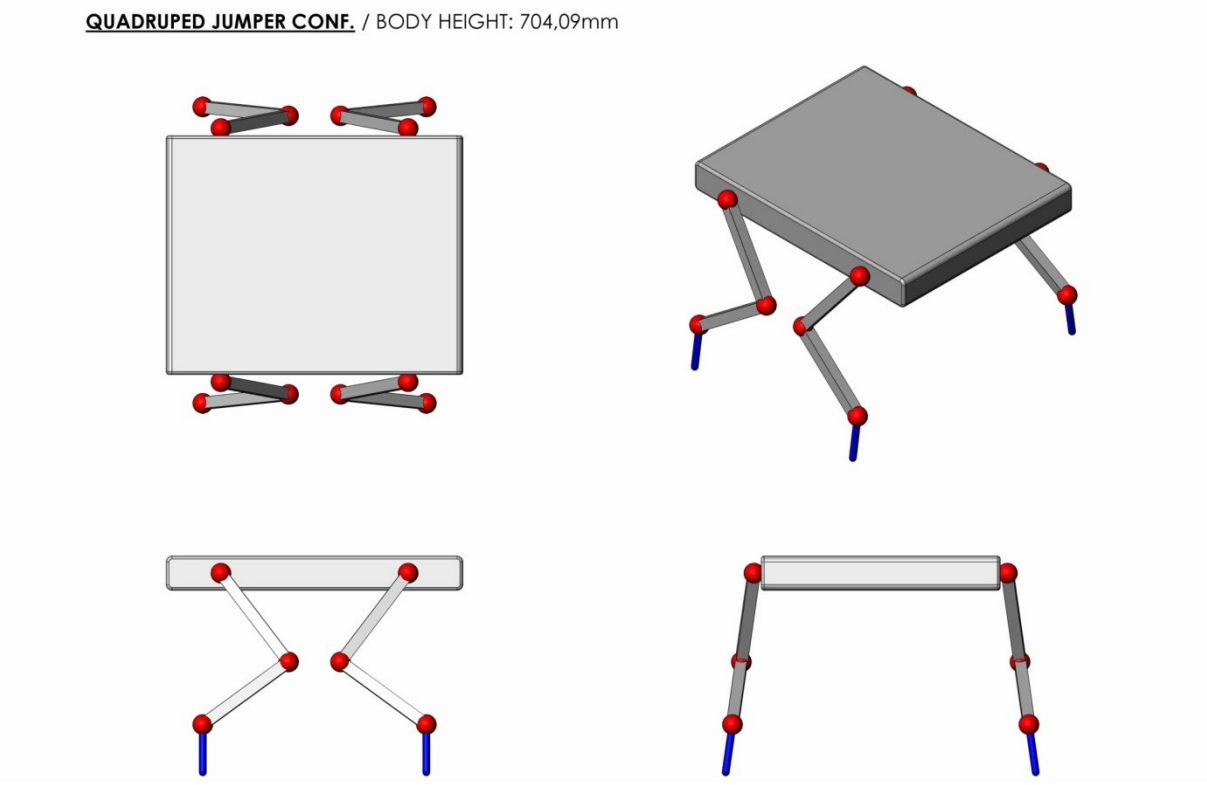


Figure 6.3. Fittest jumper design views

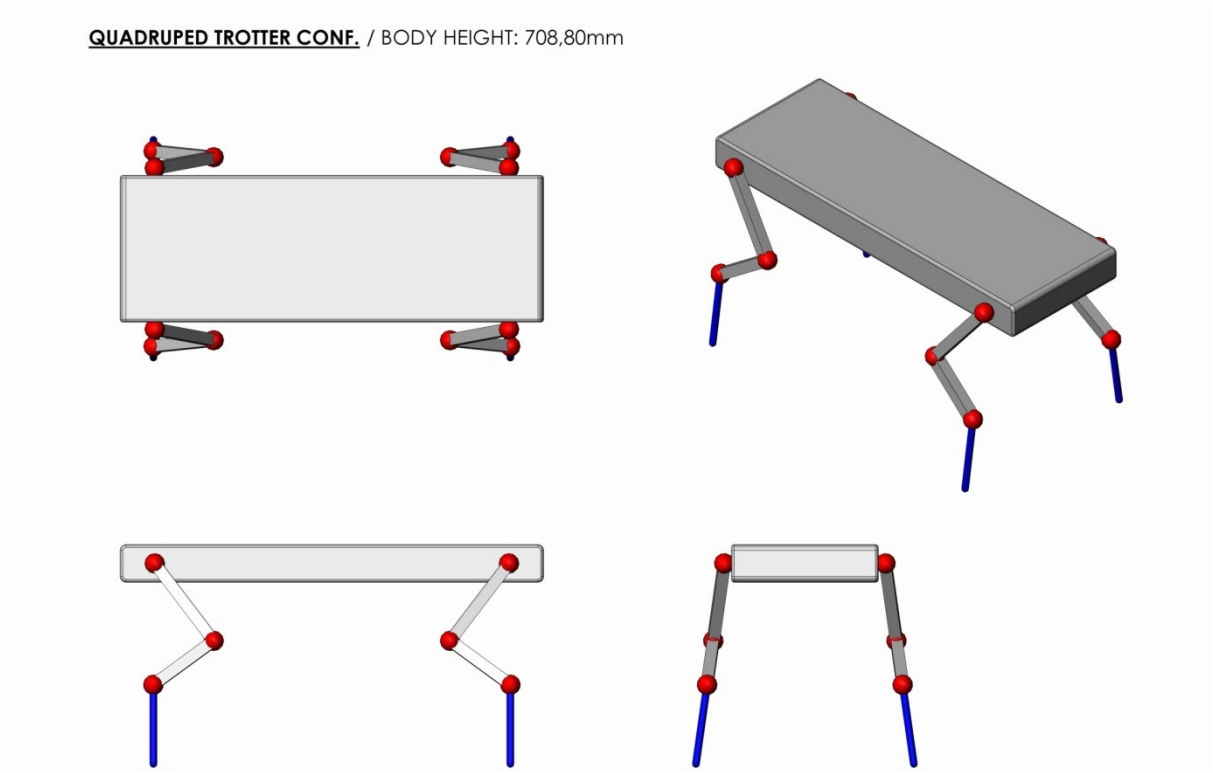


Figure 6.4. Fittest trotter design views

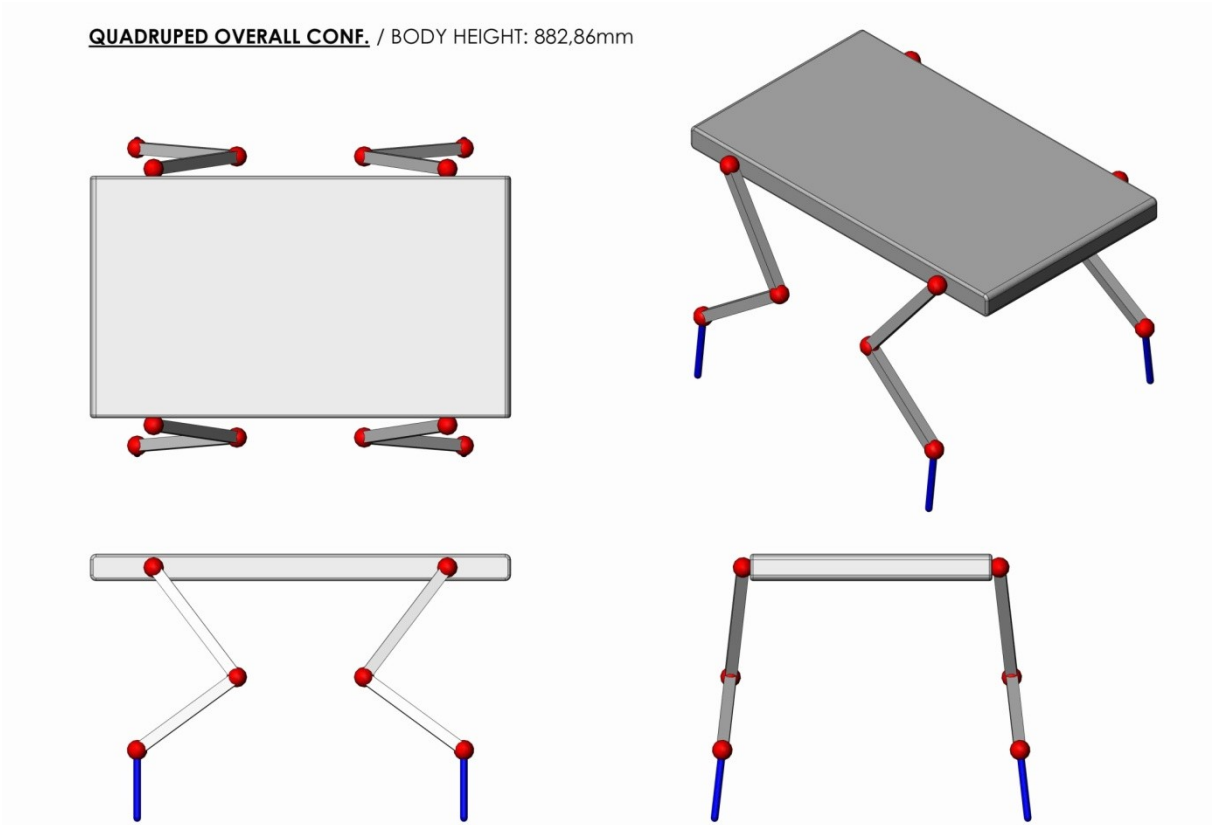


Figure 6.5. Fittest overall design views

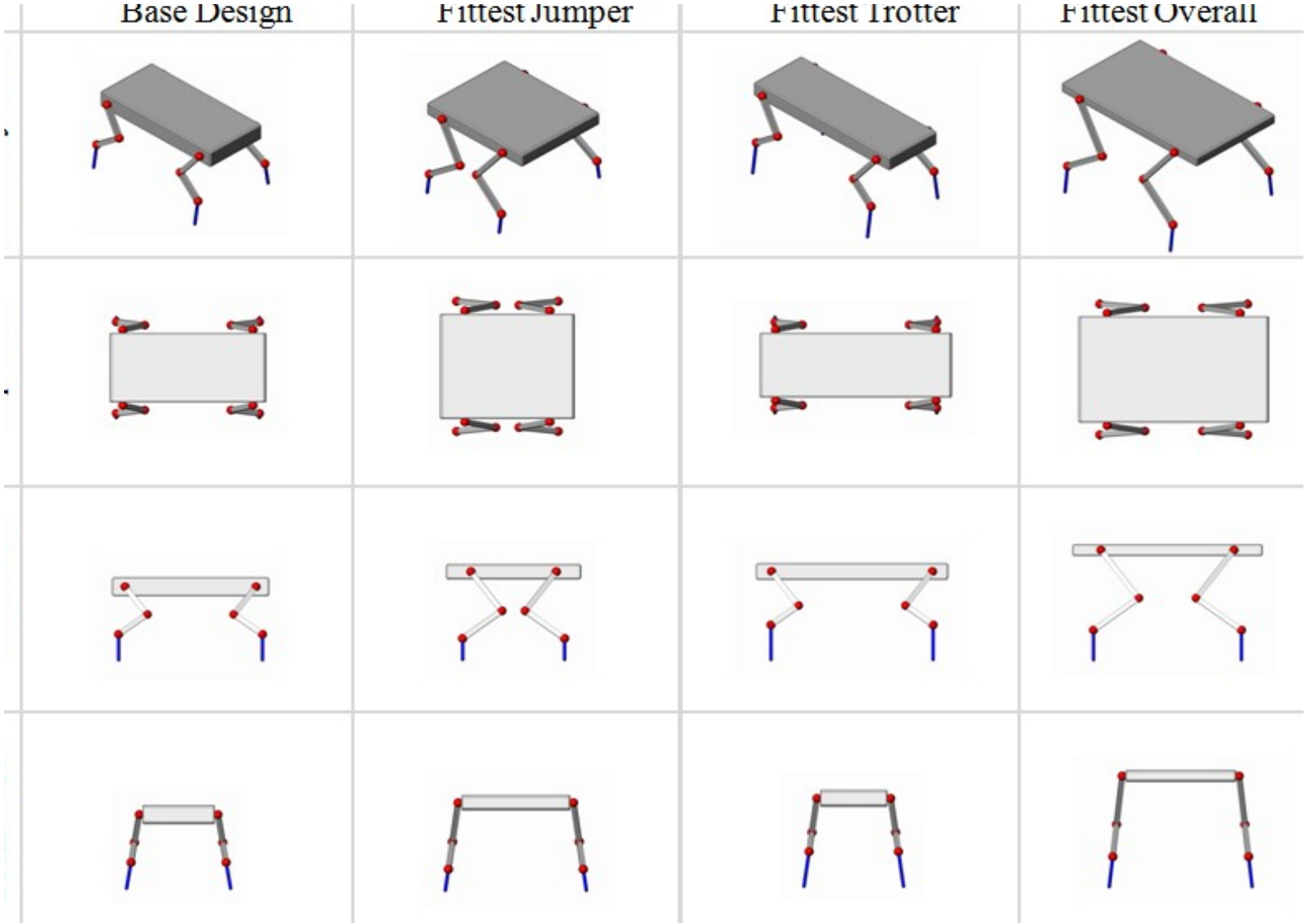


Figure 6.6. Comparison of the base design and the designs generated by GA optimization. The drawings are to scale.

The fittest jumper is the design with the most successful vertical jumping kinematic structure. The fittest trotter is successful at locomotion via trotting. The fittest overall design seeks success in both tasks. This chapter discusses these three designs one by one. It ends with a section that analyzes the performance of the genetic algorithm itself.

6.1. Fittest Jumper

The fittest jumper is the result of running the genetic algorithm multiple times with a fitness function that evaluates the first task only. The first task is a pure vertical jump. The higher the peak jump height gets for an individual design, the fitter it is. It is intuitive to assume that the fittest quadruped for this task will have long legs. The jumping animals' kinematic arrangement (antelope, goat) supports this idea. Also long link lengths can provide a long vertical workspace for upwards acceleration. As it can be seen in Figure 3.2, a quadruped with longer legs would allow for bigger body pull lengths, resulting in longer distance available for the robot to be thrust upwards by the hydraulic actuators. However, the maximum power that is available for the robot is fixed as a result of power budget considerations. Also, design cannot have infinitely long legs since as they get longer they also get heavier and the limited power lowers the peak jump height. Considering the electronic and hydraulic equipment that is required to fit the robot, the volume of the body is also fixed, adding another constraint. A shorter body has to be thicker or wider, potentially causing other problems. These conditions combined with the highly nonlinear and complex dynamics of a 16 DoF quadruped turns this into a multivariable optimization task.

The resulting kinematic structure for the fittest jumper is the second design displayed on Figure 6.1. It is also shown in Figures 6.3 and 6.6 from various angles and compared with other designs. At first look, it has long legs as expected. This can be confirmed by comparing the resulting kinematic parameter values with the base design. The obtained values are displayed in Table 6.1.

Table 6.1. Parameter values for base and fittest jumper designs

Parameter/Result	Base design (m)	Fittest jumper (m)
L_2	0.28	0.39
L_3	0.27	0.37
L_4	0.22	0.18
L_b	1.2	1.03
W_b	0.6	0.88
Shoulder pos	0.1	0.19
Jump height	0.46	1.076
Trot distance	1.56	1.162

The results show that the fittest jumper reaches heights more than double what the base design can reach, although it performs worse on the trotting task.

Looking at the parameter values in Table 6.1, the two leg links L_2 and L_3 ended up significantly longer in the jumper design, while the length of L_4 remains similar. This can also be observed in quadruped animals in nature, where the upper two links are longer compared to the feet. The contribution of the links closer to the body seems to be higher when jumping strength and balance are considered.

As described in Section 3, the parameter L_b is the length of the robot body from tail to head. The jumper design resulted in having a shorter body than the base design. The shoulder position is the value that defines where the legs are attached to the body and is measure from front and rear sides. A larger shoulder position value means the shoulders are closer to each

other. The jumper design has this value higher than with the base design. Combining this with the shorter body length, the jumper design has its legs significantly closer to each other in the walking direction. As a result it can be said that longer and closely positioned legs are advantageous in achieving a strong and stable vertical jumping motion for a quadruped.

6.2. Fittest Trotter

The fittest trotter is the result of running the genetic algorithm multiple times with a fitness function that evaluates the second task only. The resulting kinematic structure for the fittest trotter is the third design displayed on Figure 6.1. The design is also illustrated in Figures 6.4 and 6.6. The differences that can be observed from these figures when compared to the base and jumper designs are the long feet links and the wide distance between the front and hind legs. A closer look at this is again by comparing the kinematic parameter values of the trotter design to the base design. These are displayed along with the jumper values in Table 6.2.

Similar to the jumper design, the results for the trotting robot design optimization show significant progress. The fittest trotter moves about four times the velocity of the base design, and the tradeoff is a 35% drop in jumping height. This again proves that the base design has room for optimization, since a design which performs trots faster and jumps higher looks to be very attainable from the results so far.

Comparing the parameter values of the base design and the fittest trotter in Table 6.2, the two leg links L_2 and L_4 are longer on the trotter design, while the length of L_3 remains similar. For the trotting task, the optimal parameter values show that the quadruped is more efficient with legs with all similar sized links to each other. This resembles the leg structure of fast trotters in nature like a horse which have three long links above their feet. From the landing impact handling point of view, it is logical to assume that adding an extra ankle joint with a small link just to adjust the angle the feet lands on the ground would improve the

design. However, the L_4 parameter results show that having a third long leg link would outperform this style for the trotting motion. Having all three links with similar lengths would make the contribution of each actuator on the distance the leg covers similar as well, which makes the design suitable for the trotting motion according to the optimization results.

Table 6.2. Parameter values for base, fittest jumper and fittest trotter designs

Parameter/Result	Base design (m)	Fittest jumper (m)	Fittest trotter (m)
L_2	0.28	0.39	0.34
L_3	0.27	0.37	0.26
L_4	0.22	0.18	0.29
L_b	1.2	1.03	1.47
W_b	0.6	0.88	0.56
Shoulder pos	0.1	0.19	0.12
Jump height	0.46	1.076	0.3
Trot distance	1.56	1.162	6.5

When compared with the base design, the robot body length L_b is longer for the trotter design, without significant change in shoulder positions with respect to body edges. This results in overall greater distances between quadruped legs. The assumption is that having more distance between the front and hind legs results in a more balanced trotting motion. The longer dimension in the x direction translates into a higher moment of inertia about the pitch axis. This is the horizontal axis through the body center of mass perpendicular to the

locomotion direction. Large inertia about this axis creates a resisting effect against body pitch angle fluctuations. This enhances the balance of the motion in the locomotion direction and allows higher forward speeds. Off-center mass distribution in the top plane also results from the longer body shape. This creates an effect similar to the one with the pitch angle. The larger inertia about the vertical axis through the robot body center of mass resists yaw motion and this effect also plays a stabilizing role during the forward motion to assist higher locomotion speeds. The body dimension comparison is also interesting when the comparison is made with the fittest jumper. The jumper design has larger W_b and smaller L_b; wider but shorter body. A square shaped body with similar length and width is more suited to a vertical jump where the motion is quite symmetric with respect to horizontal axis compared to the trotting motion which takes place on the axis parallel to body length. The result is to have a wider body for the jumping task and a longer body for the trotting task. These selections would respectively add stability to these motions. It is interesting to note that jumping quadrupeds from nature, like antelope and goat have shorter bodies while trotters with less vertical jumping capabilities as in the example of a horse and mule have longer body aspect ratios (Ellenberger et al., 1949).

From Table 6.2, a comparison between the performance results for the fittest jumper design and the fittest trotter design can be made. The tradeoff between the jumping performance and the trotting performance of a quadruped design is clearly visible since the fittest jumper has very little trotting speed while the fittest trotter has very little jumping capability. This is important because it shows how far a task oriented optimization can change a quadruped design. A kinematic structure formation picked up from an animal does not necessarily make a design successful in a particular task. Even the animal that has the most efficient trot in nature has to be successful in other tasks in its daily life. This means that a design optimized for the sole purpose of achieving great trotting velocity will have different kinematic proportions. This is a strong result to back up the claim that a quadruped robot with a defined task priority will benefit greatly from the optimization process presented in this thesis.

6.3. Fittest Overall Design

The fittest overall design is achieved by running the genetic algorithm multiple times with a fitness function that combines the evaluation of both the vertical jumping and trotting tasks. This final challenge is planned to reach a design that in the overall, that is, in jumping and trotting outperforms the base design and justifies the merit of the optimization route. Fitter individual designs in this section are expected to jump higher and move faster in a balanced fashion compared to our base design. Results from the previous optimizations show that the fittest overall design is expected to have longer links than the base design, though it is not clear which particular links will be longer. Longer legs benefit both the jumping task and the trotting task and it is already concluded that the available torque for this particular project can support heavier legs. The optimal body dimensions for the two tasks are in conflict. The jumping task favors a shorter, wider body resembling a square while the trotting task favors a long, thin body. It is not straightforward to estimate which motion will dominate in determining the body dimension. The height of the robot body seems to play a relatively small role compared to the other two dimensions and it would be efficient to get it as small as possible, which is limited to the height of the equipment planned to be placed in the robot body. It is also required to state here that the robot design will change based on the importance assigned to each task. A quadruped that has the priority of trotting fast is expected to have a longer body length as a result of the optimization process compared to one with jumping high as its priority.

Table 6.3. Parameter values and results for different jump weights.

Parameter/Result	Jump Weight 8	Jump Weight 8.5	Jump Weight 9
L_2	0.49	0.48	0.48
L_3	0.38	0.43	0.43
L_4	0.16	0.25	0.29

L_b	1.27	1.46	1.39
W_b	0.63	0.89	0.85
Shoulder pos	0.12	0.22	0.27
Jump height	0.58	0.69	0.79
Trot distance	5.62	5.26	3.13

Table 6.3 shows data on three different jump weight values introduced in (4.3) to show how it affects the optimal design. The three jump weights used in the table are 8, 8.5, and 9. Looking at the rows for resulting jump height and trot distance, it can be observed that jump height goes higher as jump weight increases. This is because the fitness function weighs the objective of the jumping task more heavily when the jump weight W is higher, resulting in quadruped designs more successful at that task. This also means that the trot distance gets lower however, resulting in slower locomotion capabilities. Jump weight of 8 is favored in this work since with this value, the robot is capable of jumping above 60cms while having a significantly higher trot distance compared to the base design. Depending on the tasks planned for the quadruped to accomplish, the jump weight would change. Additional tasks can be included in the optimization process too. Table 6.3 shows that every kinematic parameter the quadruped has is subject to change when the jump weight which correlates to the importance of one task over another changes. Even a small change has an effect on the parameters, indicating once again that a design is bound to benefit from an optimization process.

The resulting kinematic structure for the fittest overall design is the fourth and last robot displayed on Figure 6.1. Figures 6.5 and 6.6 also display the resulting robot structure. This quadruped design has a long body similar to the fittest trotter design, while also having its front and hind shoulders closer to each other like the fittest jumper. To make a more detailed assessment, kinematic parameter values for all designs are displayed in Table 6.4.

When compared to the base design, most of the robot dimensions are significantly larger for the fittest overall kinematic design. This was expected since previous tests showed that both tasks favor longer dimensions. Longer leg links improved quadruped's performance in

both the jumping and the trotting only optimizations. The final design agrees with this by having longer L_2, L_3, and L_4 dimensions. L_2 and L_3 are significantly longer than L_4, which is more in line with the fittest trotter design than the fittest jumper design which had all three links at comparable lengths. This might be due to the selection of the jump weight parameter in a range where trotting success is weighted more than jumping success. Still, there are multiple interactions involved in the complex dynamics of a quadruped. This was one of the reasons why an optimization method using search algorithms was utilized.

Table 6.4. Parameter values and results for all four designs.

Parameter	Base design (m)	Fittest jumper (m)	Fittest trotter (m)	Fittest overall (m)
L_2	0.28	0.39	0.34	0.48
L_3	0.27	0.37	0.26	0.43
L_4	0.22	0.18	0.29	0.25
L_b	1.2	1.03	1.47	1.46
W_b	0.6	0.88	0.56	0.89
Shoulder pos	0.1	0.19	0.12	0.22
Jump height	0.46	1.076	0.3	0.69
Trot distance	1.56	1.162	6.5	5.26

Robot body length and robot body width (L_b and W_b) for the fittest overall design are almost at their allowable limits. As previously stated, judging by the previous test results, it is expected to have the robot body height as small as possible. The length of the robot helps stability for the trotting motion, while having a width comparable to its length helps stability for the jumping motion. The length of the body turned out to be larger than the body width. In this aspect as well, the combined objective design resembles the trotter design more, again influenced by jump weight.

Comparing the fittest overall design parameters with the other two optimized results, one can see from Table 6.4 that the fittest overall design has a longer body than the fittest jumper and a wider body than the fittest trotter. One clear result is the position of the leg connection points (shoulders). The fittest overall design has similar body length to the fittest trotter, but it has its legs much closer to each other. Combining this with the results from Table 6.3, where an increase in jump weight is accompanied by a shortening of the distance between the front and hind leg attachment points, it can be stated that a quadruped's ability to perform a vertical jumping task benefits from having its leg attachment points close to each other in the longitudinal axis of the robot.

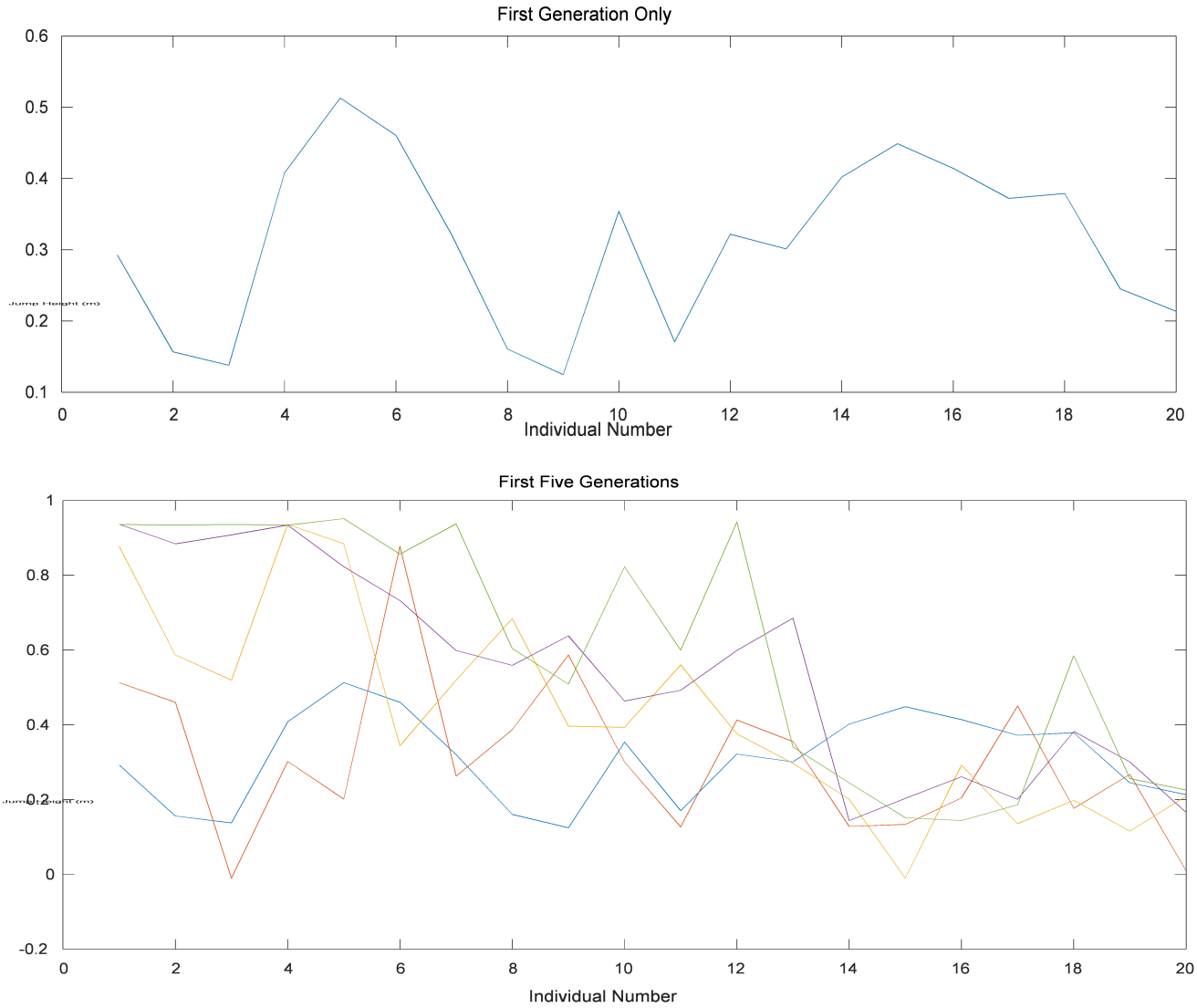
The fittest overall design is the final kinematic structure studied for the quadruped. Table 6.4 indicates an immense increase in the trotting capabilities of this design compared to the base design, while also achieving an increase in peak vertical jumping height. This shows how a robot kinematic design can benefit from optimization via GA. Further sections in this chapter display the plots for all three optimization tasks to make more detailed data accessible.

6.4. Plots for the Jumping Optimization

The vertical jumping results discussed in the previous sections of this chapter are reached through optimization with genetic algorithms. As explained in Chapter 4, these algorithms work by generating populations of individuals (in our case, quadruped designs) and carrying on fitter individual properties to future generations. At the end of this process,

the result is a population which includes the optimal individual. This section explains this process further by giving data from the optimization of the vertical jumper quadruped.

The process starts with a population of individual designs with random parameter values. In this algorithm, any population but the first generation is formed up of two elite, eight cross-over, two mutation and eight random individuals. As we move up in generations, we expect to get the selected half of the populations to get fitter. Here, selected half refers to the two elite individuals (best two of the previous generation) and the eight cross-over individuals (via the process explained in Chapter 4). A fitter individual for this task is a quadruped kinematic design with the trajectory parameters that result in higher peak vertical jump heights. In Figure 6.7, the data for the jumping height values for all individuals for each population is displayed in multiple plots.



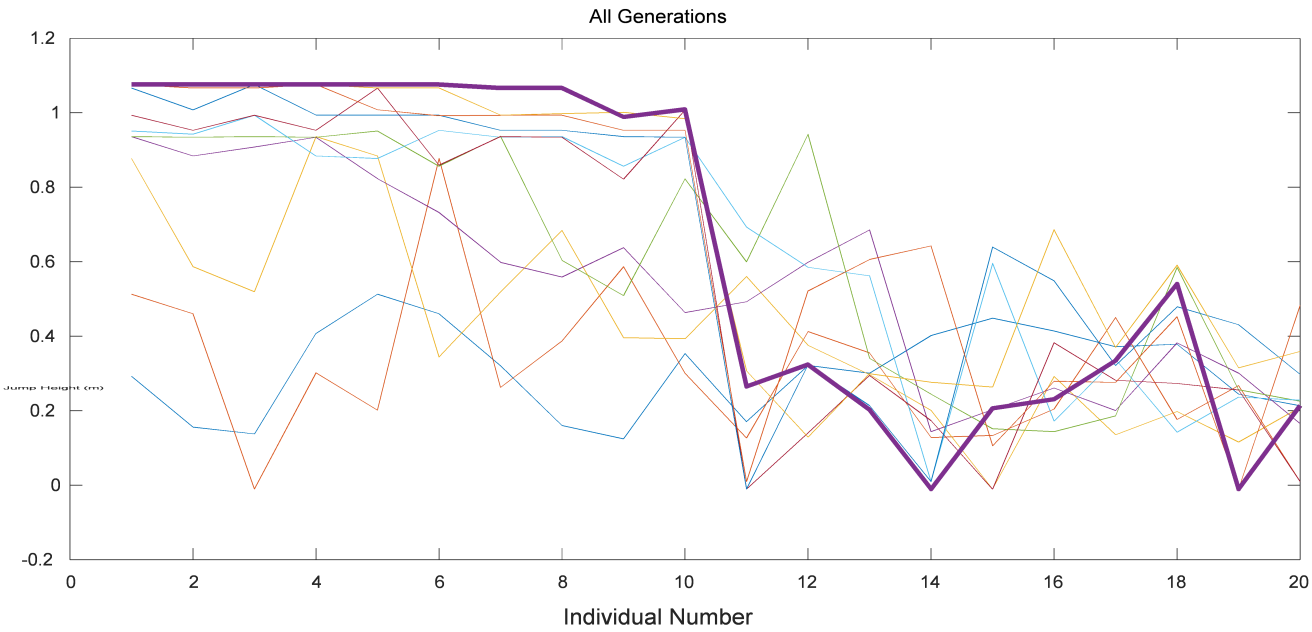


Figure 6.7. From top to bottom; vertical jumping height results for first generation of individuals, first five generation of individuals, and individuals from all generations.

The first plot in Figure 6.7 shows the initial random population's vertical jumping height simulation values. The horizontal axis indicates the individual number and for this population the individuals are all randomly generated. The vertical axis shows the peak jumping height each individual reaches. As the algorithm iterates and more populations are formed the results look like the ones in the second plot in the figure. Here, each curve represents a population. For the population curves excluding the first generation, the horizontal axis numbers represent the elite, cross-over, mutation, and random individuals, respectively, from the left to the right on the plot. Therefore, the individuals displayed on the left side of the plot are reached through the selection process. They steadily (from generation to generation) get better jumping results compared to the left half of the curves that belong to the previous populations. On the other hand, the right half of the curves remain randomly generated and their jumping results vary. The last plot on the figure includes all populations formed in this algorithm run. The bold curve on this plot belongs to the final generation. Most of the individuals on the left half of this curve have the same jumping height results, which points to convergence. It is not expected at this point that further iterations can bring better results.

The plots displaying data for all individuals in all generations are hard to read even though they have a lot of information in them. A clearer way to observe population fitness is

plotting the mean vertical jumping height of each generation. For this, it is logical to remove the data of the second half of populations that are generated randomly and use the first half composed of selected individuals. Figure 6.8 displays the plot for first ten individual design vertical jumping height mean values for each generation. The horizontal axis shows the generation number which goes higher as the data moves to future populations. The vertical axis is for the mean vertical jumping height of the first ten individuals in meters.

Similar to the Figure 6.7 plots; it is possible to see the constant increase in population fitness as the generations are iterated on the plot of Figure 6.8. The increase slows down as the algorithm converges. Even after the convergence on a fittest individual, there will be increase on the mean fitness as the other individuals in the population become a copy of the fittest individual at further generations.

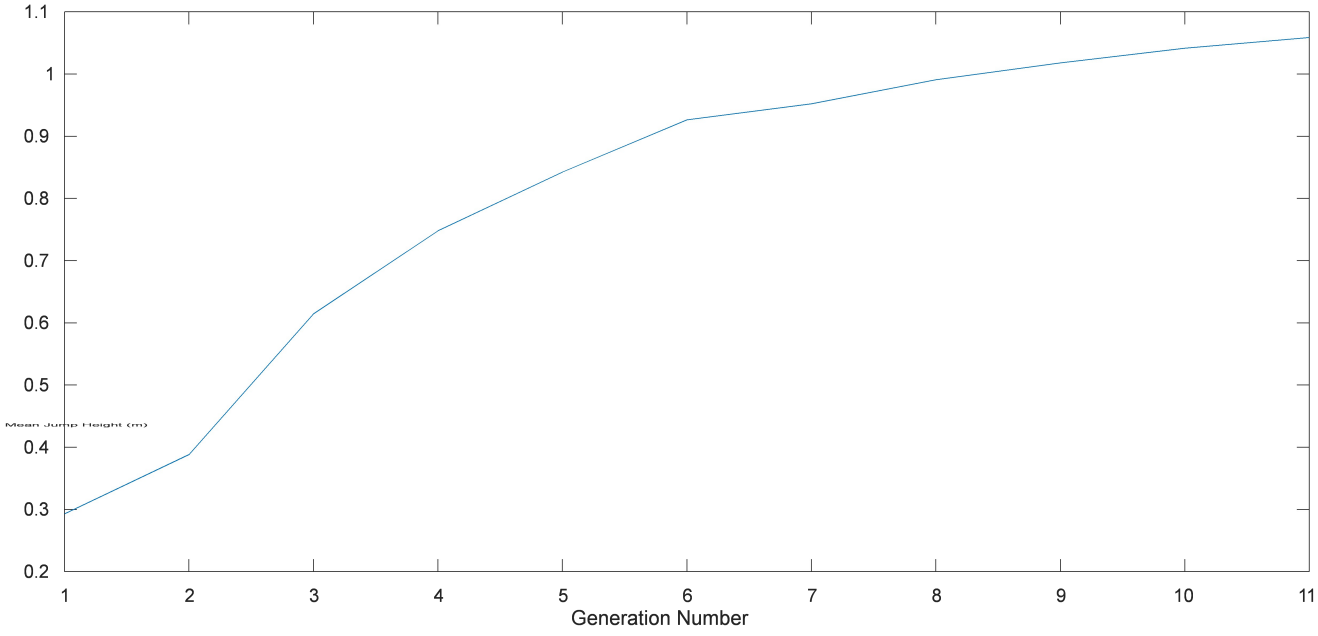
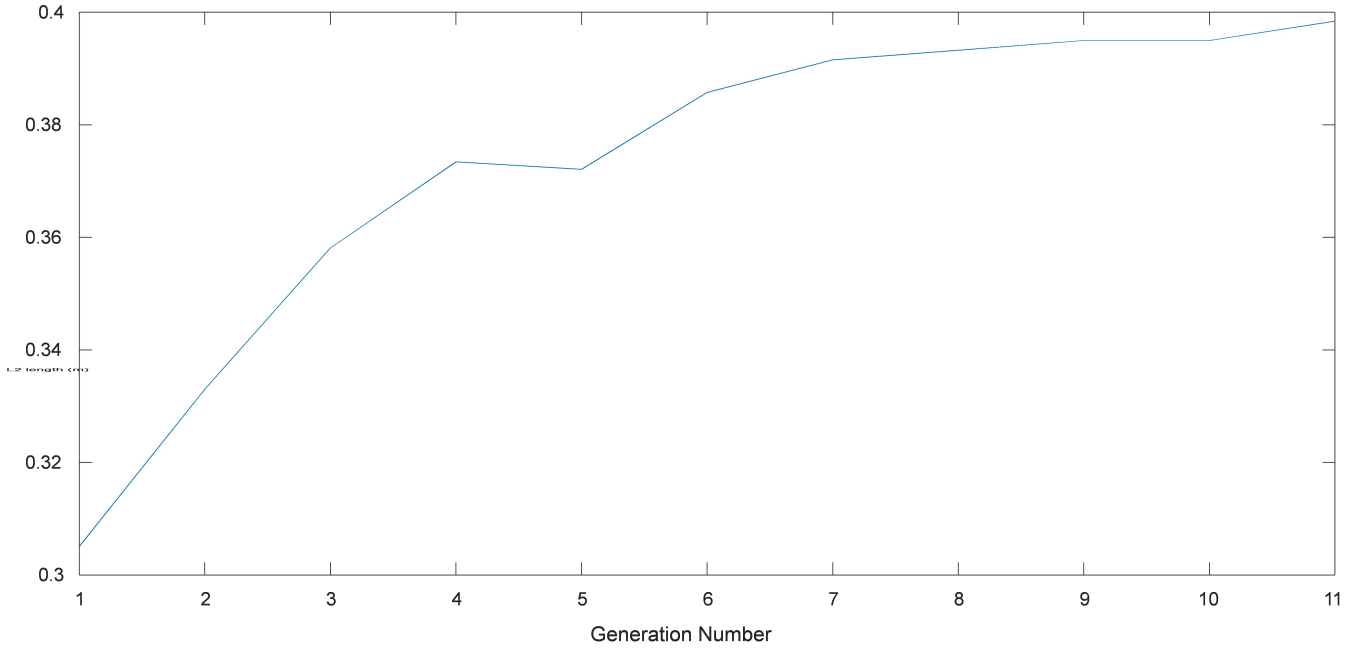


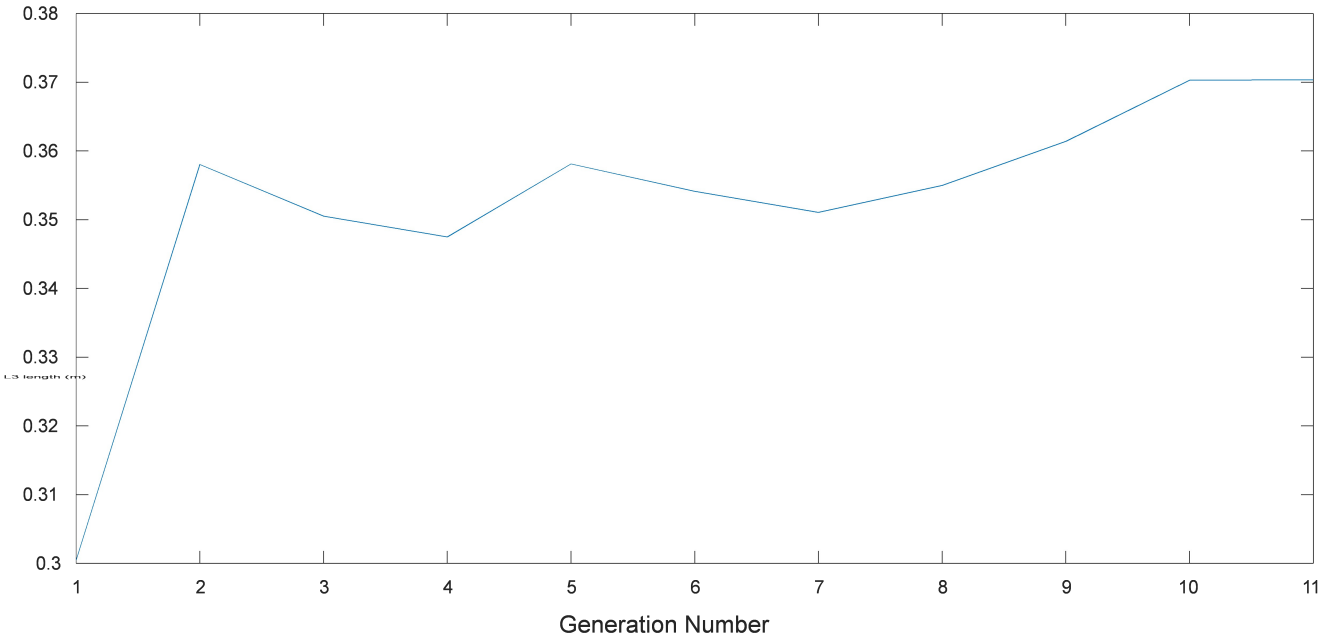
Figure 6.8. Mean vertical jump height of first ten individuals in each generation

The vertical jumping height is the result of the kinematic structure parameters of each individual design. Mean plots for these can also be drawn similar to the fitness plots. These could provide additional insight on the analysis made in Section 6.1.



L_2

Figure 6.9. Mean shoulder to knee link ( ) length of first ten individuals in all generations



L_3

Figure 6.10. Mean knee to ankle link ( ) length of first ten individuals in all generations

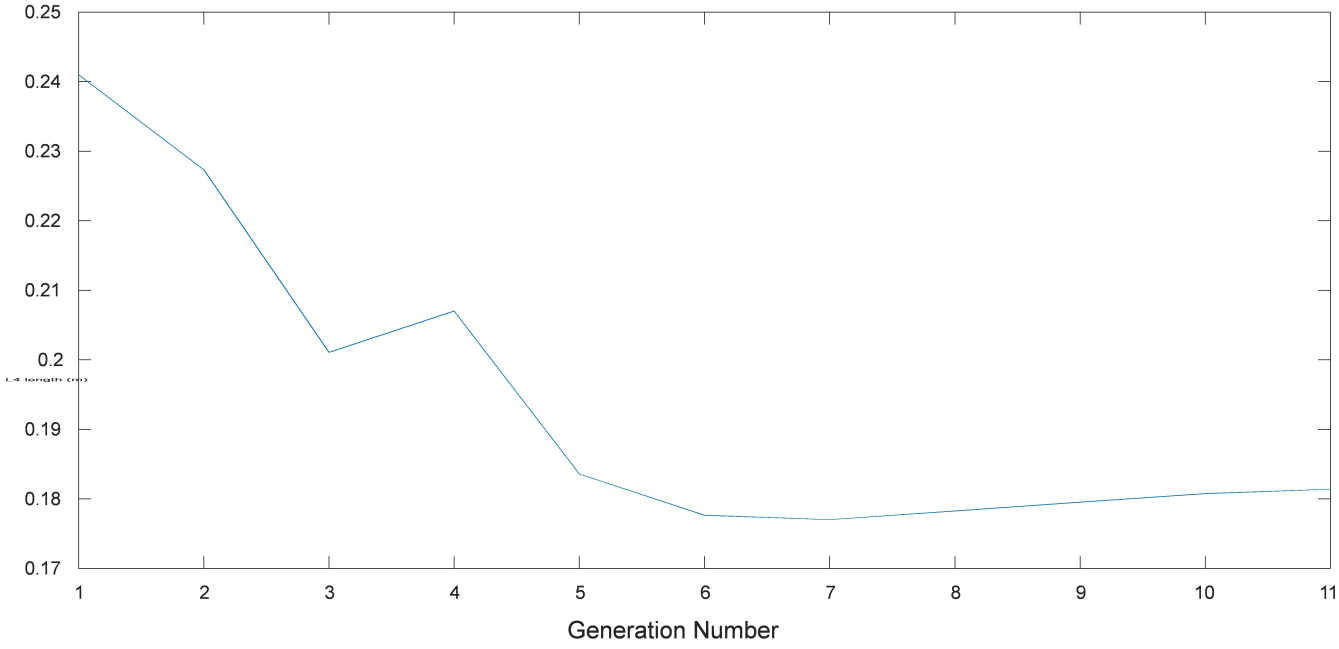


Figure 6.11. Mean foot link (L_4) length of first ten individuals in all generations

Figures 6.9 to 6.11 are the plots for the mean length of leg links. In Section 6.1 it was discussed that the optimal quadruped design for the vertical jumping task is composed of long L_2 and L_3 links and a relatively shorter L_4 link. The above plots are in parallel with this notion since the L_2 and L_3 graphs display an increasing mean length while the L_4 graph shows a decreasing one. The vertical jumping motion seems to demand these particular link proportions where the upper links outsize the lower one for a quadruped design to show jumping qualities. The fittest individual design moves to these values steadily through the iterations of the algorithm.

Another observation that can be made from the plots in Figures 6.9 to 6.11 is that in comparison to Figure 6.8, the curves change direction multiple times on the vertical axis. The mean fitness plot is expected to be smooth overall, mostly going upwards unless the cross-over individuals turn out to be weak. The kinematic parameters however, are related to one another and a change in on has an effect in the optimal value of the other. For example, in

Figure 6.10 the plot for L_3 length curves up and down for some generations. This is a mean plot, so multiple individuals need to start having shorter L_3 lengths to take it down. The best explanation for this is to consider the other design parameter values and how there is a different L_3 value that maximizes the vertical jumping height for each set of values the rest of the parameters can take. For the quadruped to have a better jumping result with a higher L_3 value, it might need a lower L_4 value.

Figure 6.12 displays the mean body length and width plots similar to the link length ones. The plots are in line with the previous claim that having a shorter, wider body makes a quadruped more successful in the vertical jumping task. The body length steadily decreases with every iteration of the algorithm while the width shows an overall increasing trend as the parameters settle on their optimal values. Since the initial population is formed of individuals with randomized design parameters, the mean of every parameter plot starts somewhere around the middle of their search range. The search ranges for each parameter are formed around initial design values with consideration of physical limitations. Displaying a steady increase from this point means the desired parameter value is bigger than average while a steady decrease means it is smaller than average. For all the parameter plots investigated so far, all the curves were overall following an increasing or decreasing trajectory, making it easy to state that a particular parameter needs to be big or small to be optimal for this specific task at hand.

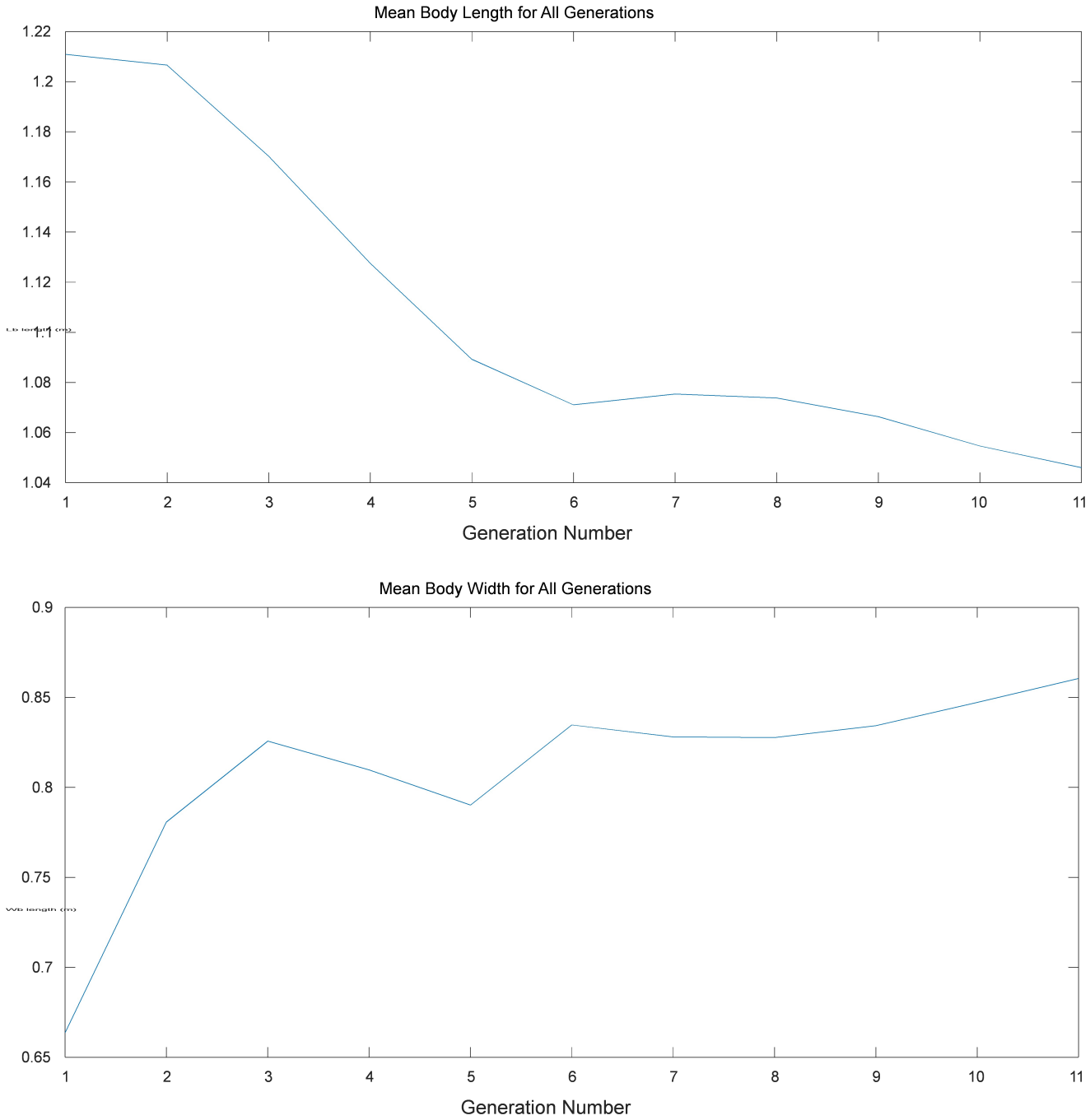


Figure 6.12. Mean body length and width plots of first ten individuals in all generations

The next parameter does not follow this trend, making it harder to analyze. Figure 6.13 displays the plot of mean shoulder distance of first ten individuals for all generations. Shoulder distances are calculated from the edges of the body and are symmetrical; for the front legs it is the distance from the front end of the quadruped while for the hind legs it is the distance from the back end of it. A bigger shoulder distance means the front and hind legs are closer to each other and a smaller one means they are farther apart.

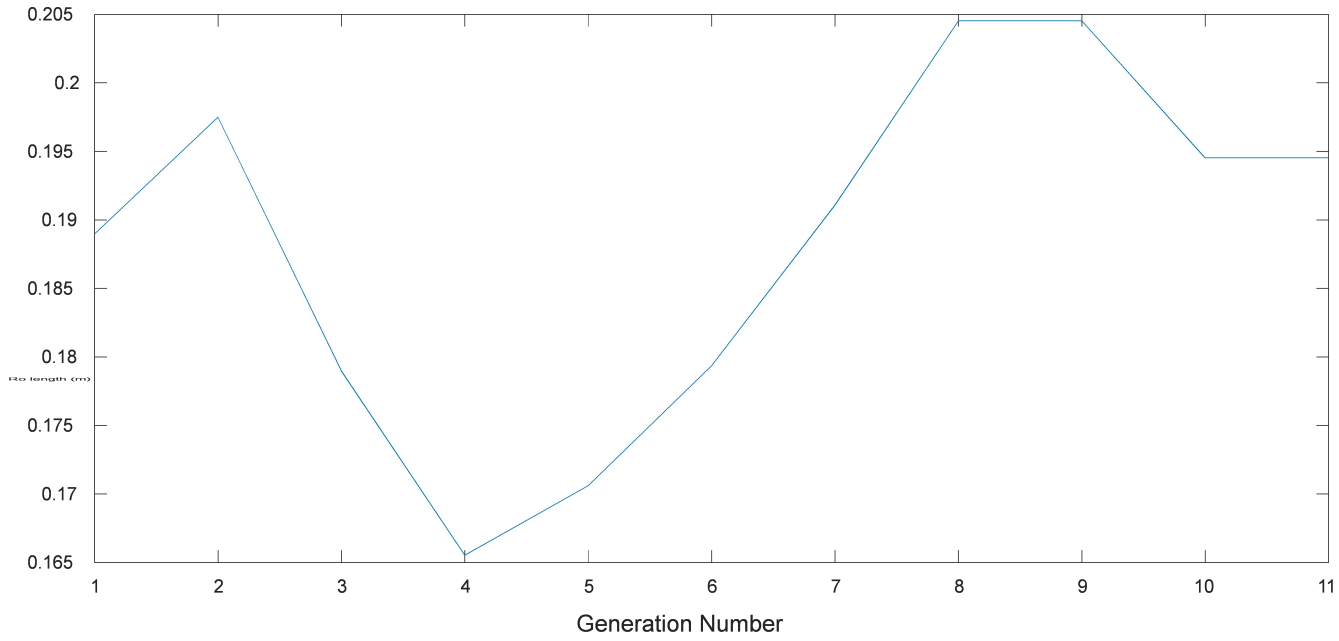


Figure 6.13. Mean shoulder positions of first ten individuals for all generations

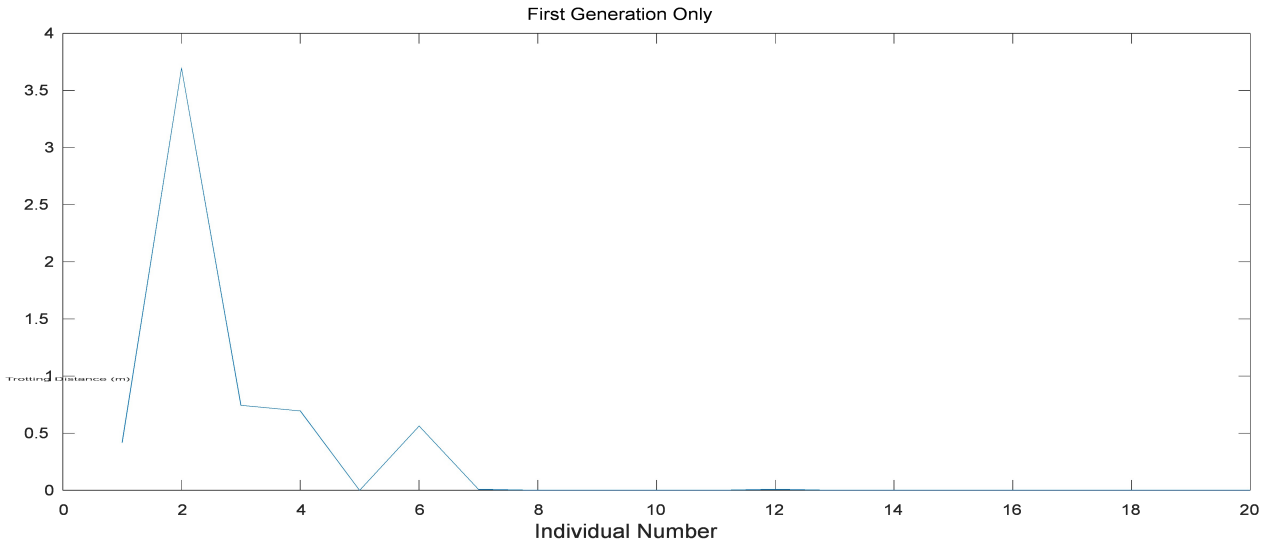
The shoulder position curve moves up and down multiple times and settles in a value close to its original one. This can be explained by this parameter being linked with other parameters in a way that their effects on it are balanced. For example, a longer L_2 might require a bigger shoulder position while a longer L_3 might require a smaller one. When considering the physics of a quadruped structure, a shorter body with longer legs (the kinematic structure the other plots are headed to) should need wider space between the shoulders to be more stable. The success of the robot at executing the vertical jumping task itself should be improved by having the legs closer the each other. Adding in the fact that the resulting robot is wider and has shorter feet, this might be the balancing factor that keeps the shoulder positions mostly the same.

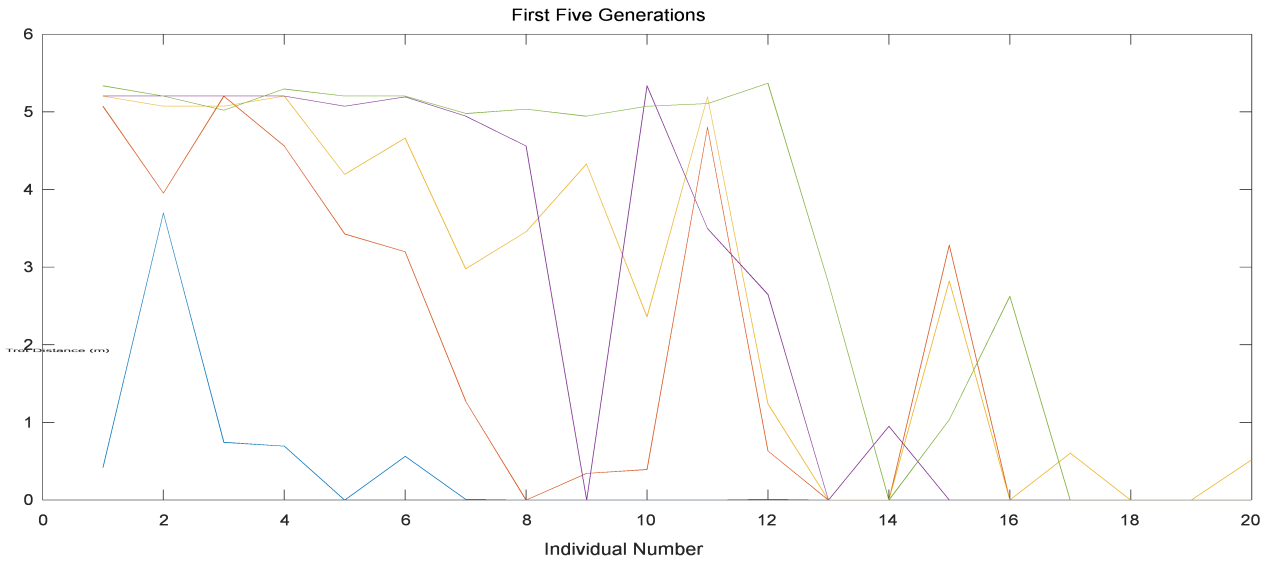
The following sections present data similar to the above for the trotting simulations and the overall design simulations.

6.5. Plots for the Trotting Optimization

The trotting optimization is carried out using the same genetic algorithm method as the one used in the jumping optimization. The differences come from the fitness function, the task the individual robots undertake during their simulations, and the additional task specific parameters added to the algorithm. The jump height plots are replaced by trot distance plots, which display the distance the robot covers in meters in five seconds. These are shown in Figure 6.14.

Similar to Section 6.4, Figure 6.14 plots show the individual performance results. Unlike the vertical jumping task where the robot pulls itself down on its four feet and then launches itself upwards, the trotting task is a repetitive motion on the ground where a significant portion of the time is spent balancing on two feet. In addition to the kinematic parameters, the trotting motion has task specific trajectory parameters like double support period and step height. Testing an individual design with short legs but high step heights will end up in failure. Considering about half of the individuals across all generations have randomly assigned design parameters, it is expected for some quadruped designs to be paired up with infeasible trajectory parameters. In these cases, the quadruped will lose its balance and fall during the simulation, effectively failing the task completely. These are evaluated as covering zero distance on the performance plots shown below. Seen on the first plot of Figure 6.14, the first generation has plenty of individuals that fall into this category, mostly because all the individuals in this population have random design parameters. The results quickly pick up after this however, and as more iterations pass the performance plot gets healthier as displayed in the second plot of the figure. The general shapes of the curves look similar to the jumping performance plots in Figure 6.7. As we reach the final generation and acquire all populations, we obtain the converged curves similar to the jumping plots. The plot showing all generations is the last one in Figure 6.14. The bolded curve again represents the results of the final population of the algorithm and involves the fittest individual. Its left half is horizontal similar to the final population of the jumping height plot and this can be considered a sign of convergence.


First Generation Only
Trolling Distance (m)
Individual Number


First Five Generations
Trol Distance (m)
Individual Number

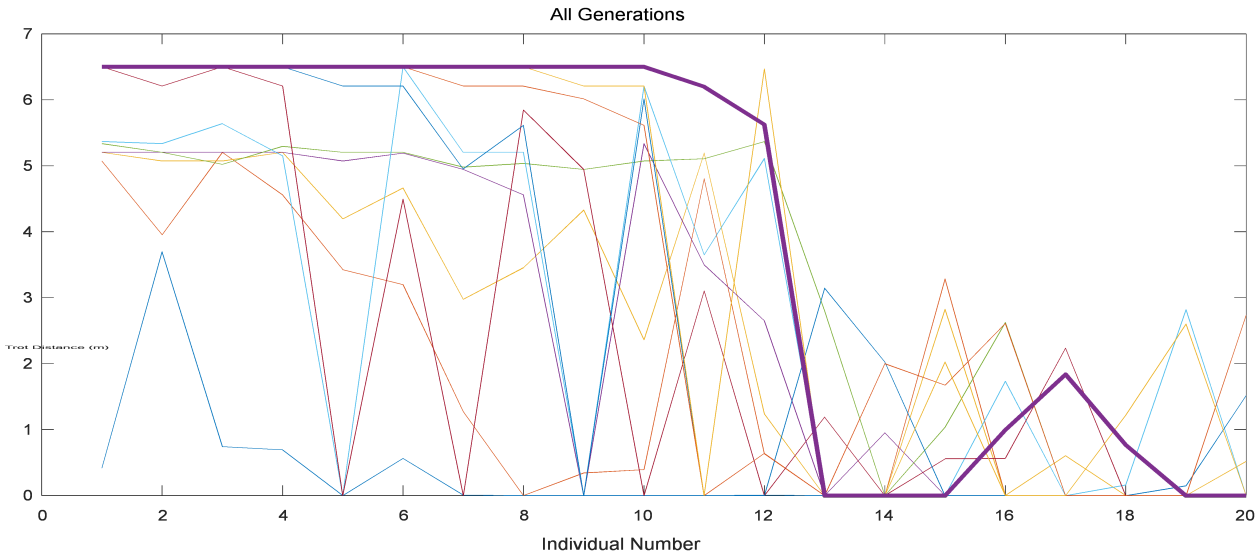

All Generations
Trol Distance (m)
Individual Number

Figure 6.14 From top to bottom; trotting distance results for first generation of individuals, first five generation of individuals, and individuals from all generations.

When all populations are included in the plots, the readability suffers from having multiple curves. In order to get an alternative look at the data, like it was done in the previous section, the mean plot for trotting distance of first ten individuals in every population is drawn again. This is displayed in Figure 6.15.

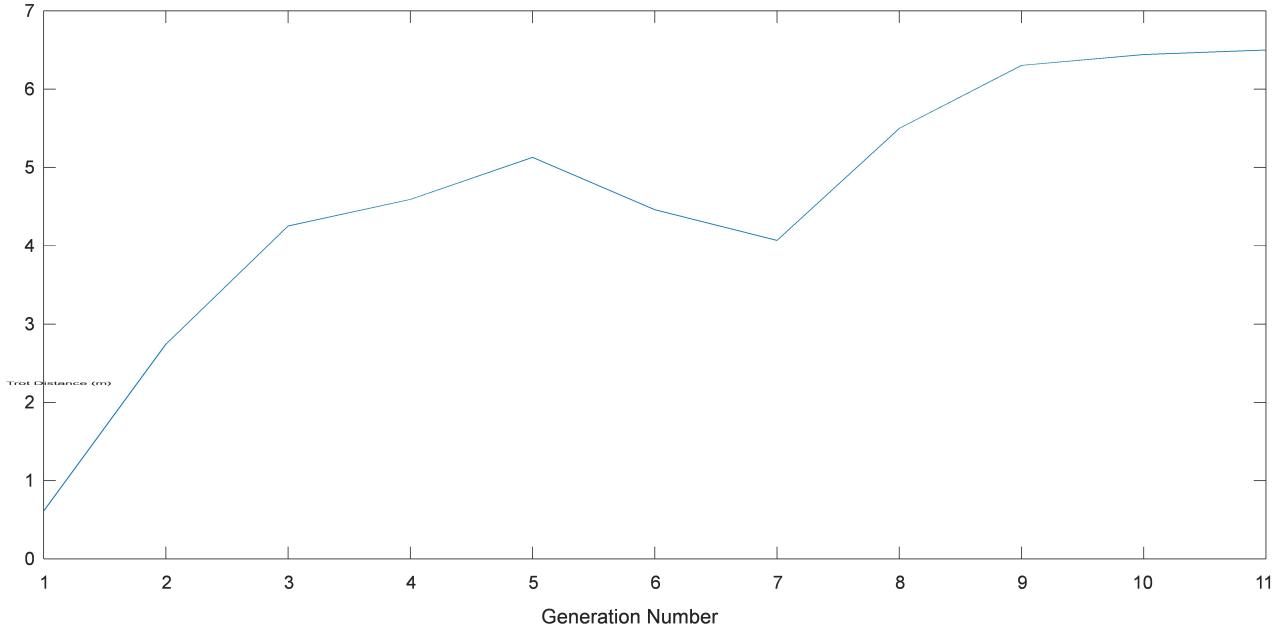


Figure 6.15. Mean trot distance of first ten individuals in each generation

It is expected to see an increase in the mean performance of populations as the generation number goes up. This was the case for the jumping performance plot in Figure 6.7. The overall increase is visible on the above plot for the trot performance as well, with the exception of a dip at generations six and seven. This is due to one or more individuals formed by the cross-over process failing to perform the trotting task and thus pulling the mean by a considerable amount. The cross-over step of genetic algorithms has random elements in it and can explain big performance changes in an individual formed this way.

From the data so far it is noticeable that the quadruped losing its balance is a big factor for the trotting task. Even if an individual design is successful and can trot without falling, it does not mean that it displays good balance during its motion. Balance is an important aspect of this task and only having a binary metric of successful and unsuccessful is not sophisticated

enough for our needs. For this reason, the orientation change around the axis of the trot direction (roll axis) is employed in the fitness function (4.2.). This addition aims to minimize the total angle change in this axis over the course of the entire trotting motion.

The mean of total roll angle change is plotted similar to the mean trot distance in Figure 6.16. A higher total roll angle through a complete trotting task is interpreted as the quadruped being less balanced. The robot body having less shaking/rocking around its center should result in a more steady motion which is desirable for this work. Since we want to keep this value low the fitness function is constructed in a way that having a lower roll angle change contributes more to the fitness of an individual. As seen in the plot, as the curve moves to further generations the roll angle change decreases, making the robot more balanced. This, in combination with the increasing trot distance displayed in Figure 6.15, results in a fitter population with every new generation.

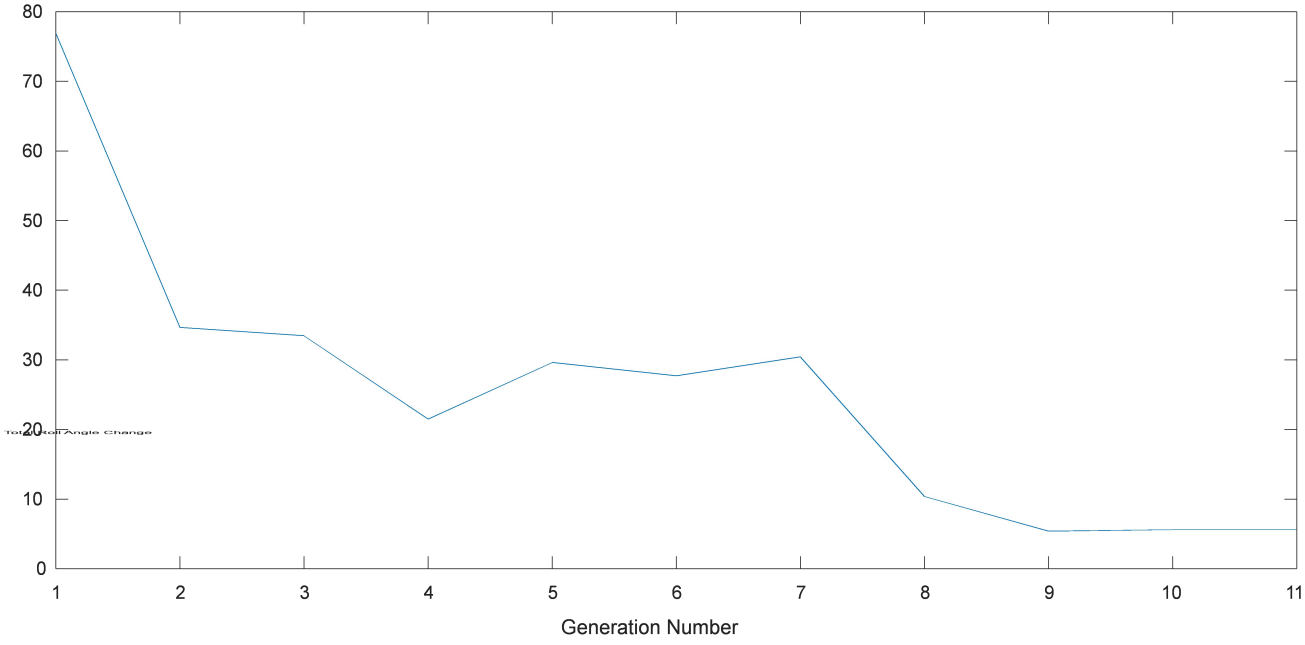
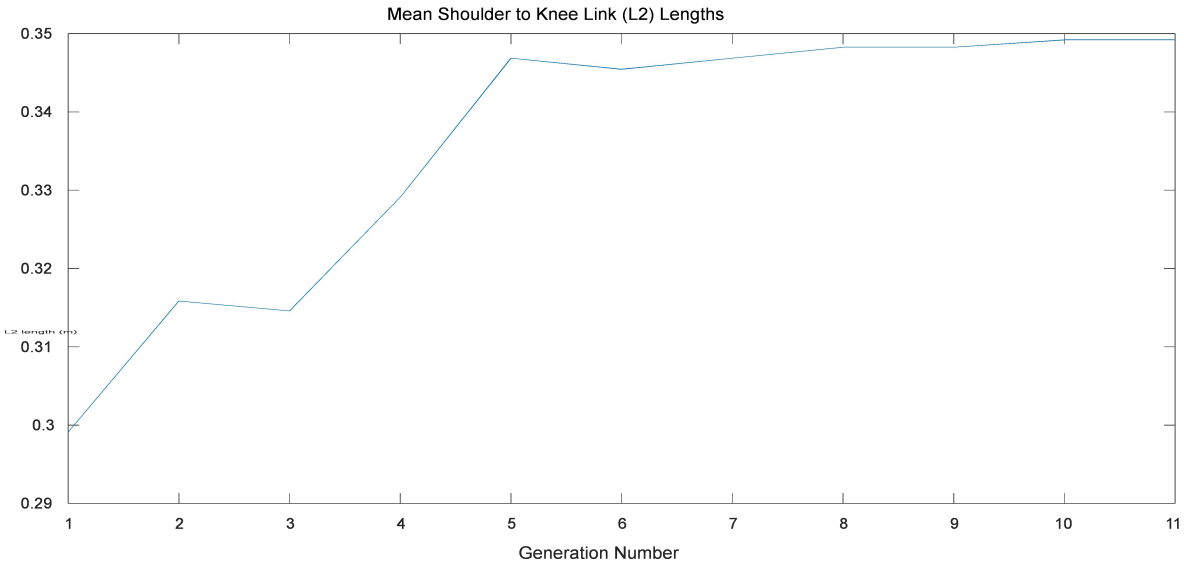
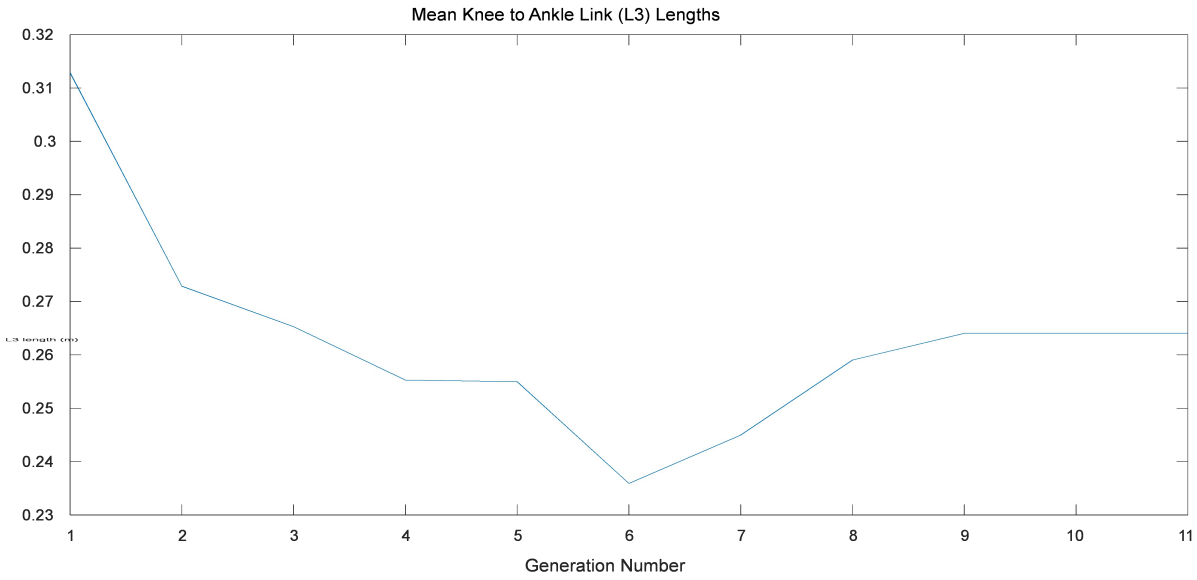


Figure 6.16. Mean angle change of first ten individuals for all generations

In Section 6.2, kinematic parameter values of the individual design for the optimal trotting results were discussed. Below are the mean plots of the parameter value data for all generations.


Mean Shoulder to Knee Link (L2) Lengths
0.35
0.34
0.33
0.32
0.31
0.3
0.29
1
2
3
4
5
6
7
8
9
10
11
Generation Number


Mean Knee to Ankle Link (L3) Lengths
0.32
0.31
0.3
0.29
0.28
0.27
0.26
0.25
0.24
0.23
1
2
3
4
5
6
7
8
9
10
11
Generation Number

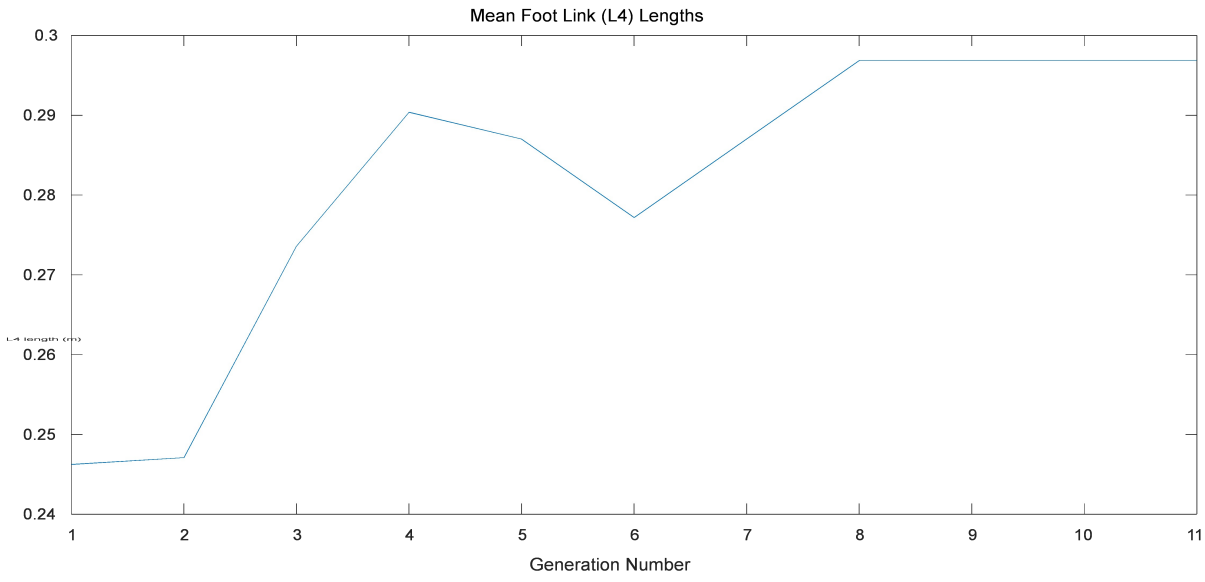
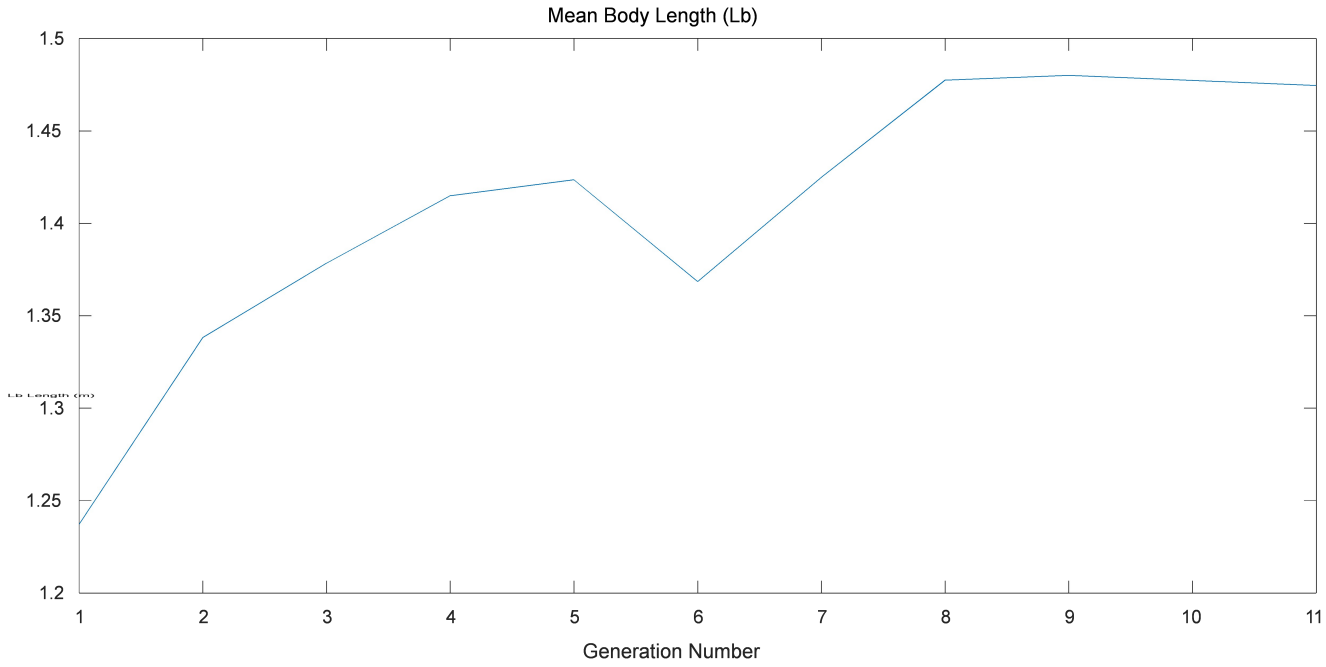

Mean Foot Link (L4) Lengths
0.3
0.29
0.28
0.27
0.26
0.25
0.24
1
2
3
4
5
6
7
8
9
10
11
Generation Number

Figure 6.17. From top to bottom; mean length plots for links L_2, L_3 and L_4 of top ten individuals for all generations.

All link length plots of the algorithm run for the trotting task are shown on Figure 6.17. When compared to the equivalent plots for the jumping tasks displayed in Figures 6.9 to 6.11, the curves for L_3 and L_4 links follow different directions. The trotting task favors quadrupeds with shorter L_3 links and longer L_4 links in contrast to the jumping task. The curves for these two links both display a dip from their path at generation six. This change in the link lengths is the probable reason for the failed attempt in generation six and explains the similar dip that was present for the mean trot distance plot in Figure 6.15. The mean data for the individuals converge back to their optimal values after this as the weaker individuals are replaced by the fitter ones.

Figure 6.18 has the plots for the mean body length and width plots for the trotting task. Generation six is the anomaly in these plots as well, matching the performance and link length plots. Looking at the rest of the generations the body length is getting longer while the width of the body is slightly getting slimmer. Especially the body length increase is significant, showing pretty direct relation between the length of a quadruped on the axis of motion and its efficiency in locomotion by trotting. A similar relation can be made between the shoulder positions and trot performance when the related plot in Figure 6.19 is observed.



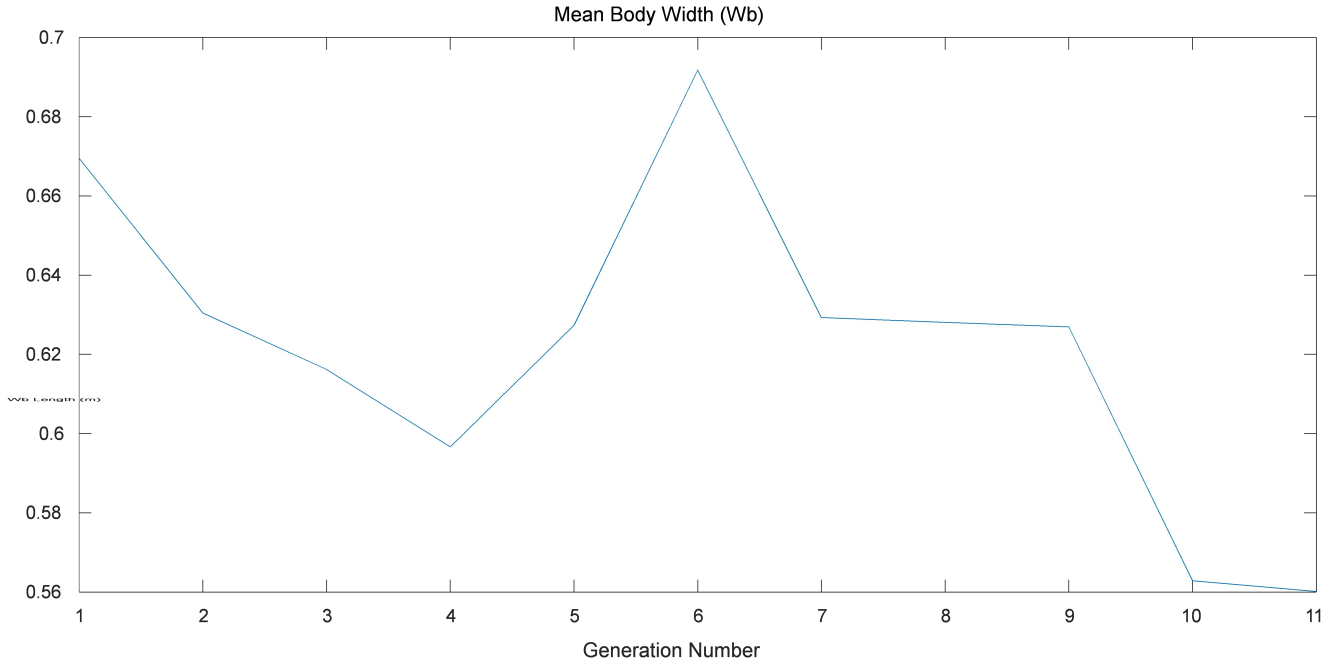


Figure 6.18. Mean body length and width plots of first ten individuals in all generations

The curve for shoulder position shows a steady decrease from the initial value. A decrease in this value corresponds to an increase in the distance between the front and hind legs of the quadruped. This seems to help the trotting motion capabilities of a quadruped by resisting pitch angle fluctuations, as mentioned earlier in this chapter.

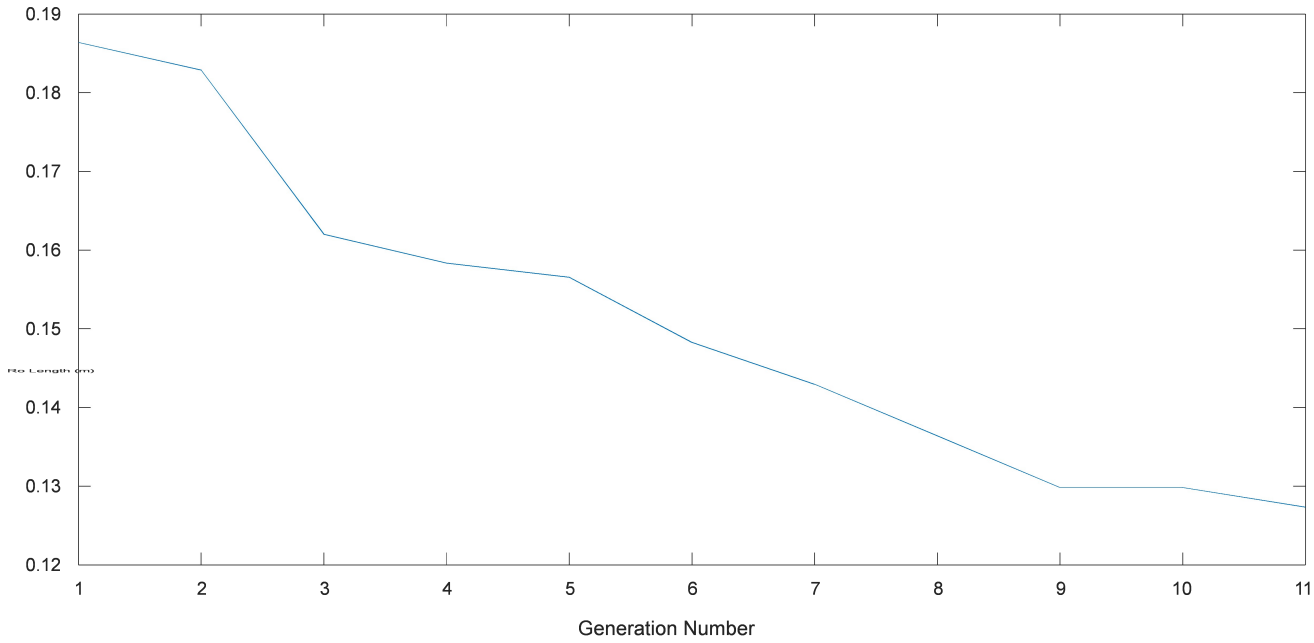


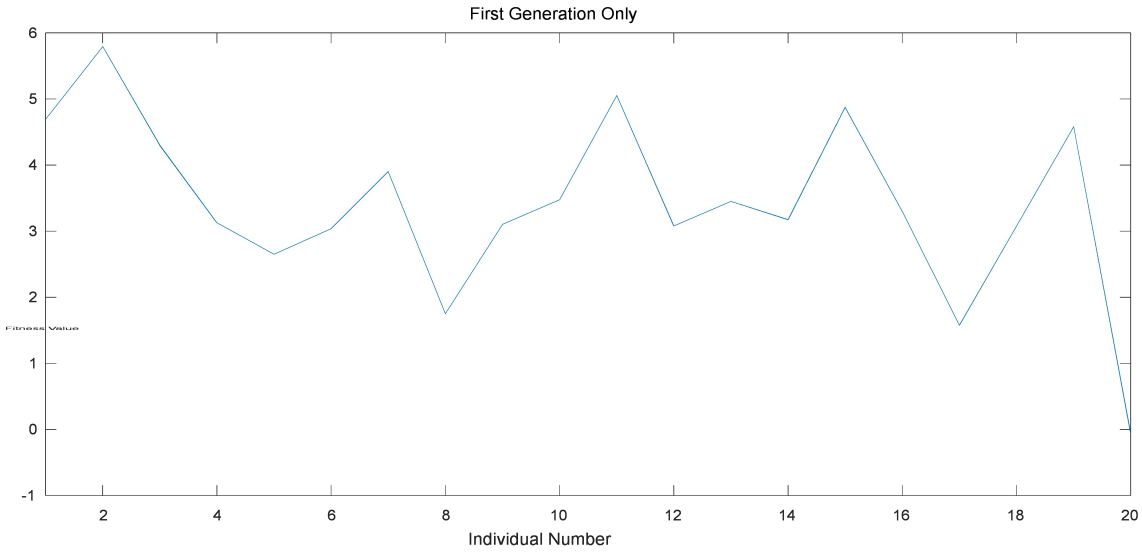
Figure 6.19. Mean shoulder positions of first ten individuals for all generations

6.6. Plots for the Overall Optimization

The genetic algorithm for the combined optimization process takes each individual design through both the vertical jumping and the trotting simulations. The fitness function is modified to combine the evaluations from the two tasks and as expressed in (4.3). The task specific trajectory parameters from both tasks are included in the tuning process. The main metric of success of an individual is its fitness value, and the values belonging to all individuals for all generations are displayed in Figure 6.20.

The plots in Figure 6.20 are structured identical to the plots in Figure 6.7 and 6.14. Having each design go through two tasks in which they perform and get evaluated differently makes this optimization more complex compared to the two sections above where only one task based evaluation was taking place. The search space is also bigger for this work since the task specific trajectory parameters for both the vertical jumping and trotting tasks are included in the algorithm. As a result of these, it might not be expected to reach a similar level of convergence that was able to be reached during the optimization described in the earlier sections within the same amount of generations. Looking at the fitness plots, the results are observed to converge slower compared to previous two optimizations. In the second plot

where only the first five generations are plotted, the curves are more separated and the individuals that lie in a single curve have bigger fitness value gaps between each other. When evaluating the third plot where all the individuals from all the generations are plotted though, the curves have the horizontal shapes reached in the case of convergence. In fact, these algorithm runs were performed with excess number of iterations reaching to 16 populations in anticipation of delayed convergence. However, the convergence reached by the 11th generation was maintained, so the results are cut down to 11 populations for readability purposes. Adding more iterations and thus more populations to a genetic algorithm reduces the chance of getting stuck on a local maximum in the expense of computational burden. However, the left half of a curve converging to a horizontal point in these graphs means that the only possible new solution can come from the mutation or random individuals which essentially turns the algorithm into a random search algorithm and the chances of getting a fitter individual this way is significantly slim.



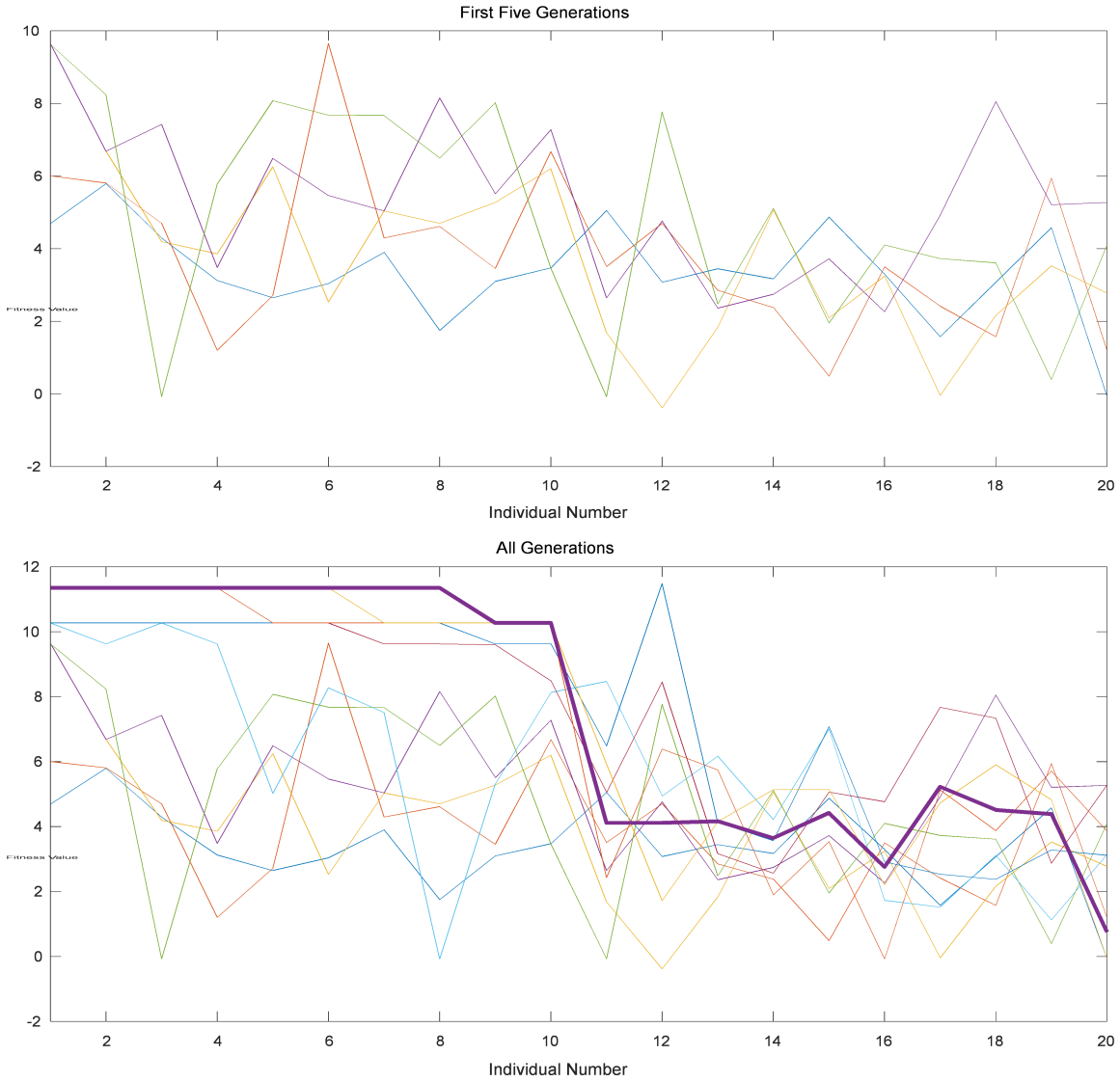


Figure 6.20. From top to bottom; fitness value results for first generation of individuals, first five generation of individuals, and individuals from all generations.

Following the presentation in the previous sections, the next step is to display the data for mean performance. The performance gauge for this section is the fitness value which includes both the jump height and the trot distance data. It is beneficial to the analysis that the data for jump height, trot distance and fitness value of the populations are displayed separately.

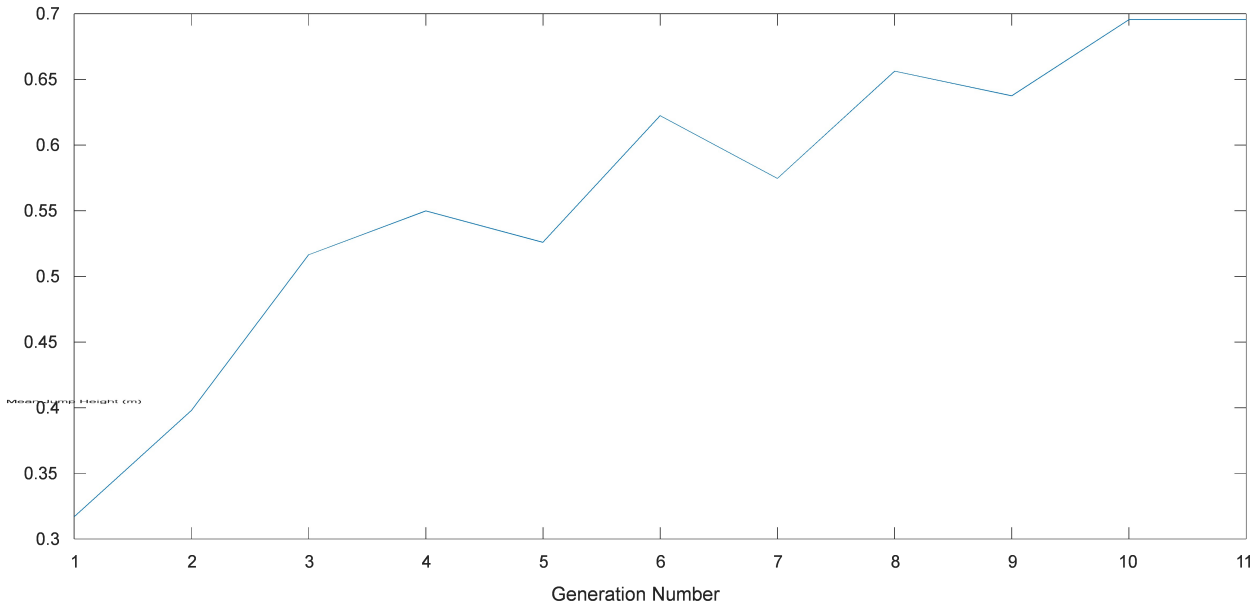


Figure 6.21. Mean jump height of first ten individuals for all generations

Plot of the mean jump height data is displayed in Figure 6.21. This plot reveals us a part of the whole picture. A quadruped design getting more efficient in executing the vertical jumping task does not necessarily improve its trotting performance. As a result the data curve has multiple points where the mean jumping performance of populations degrades, but overall there is an increase. The next plot shows the trotting perspective of the data.

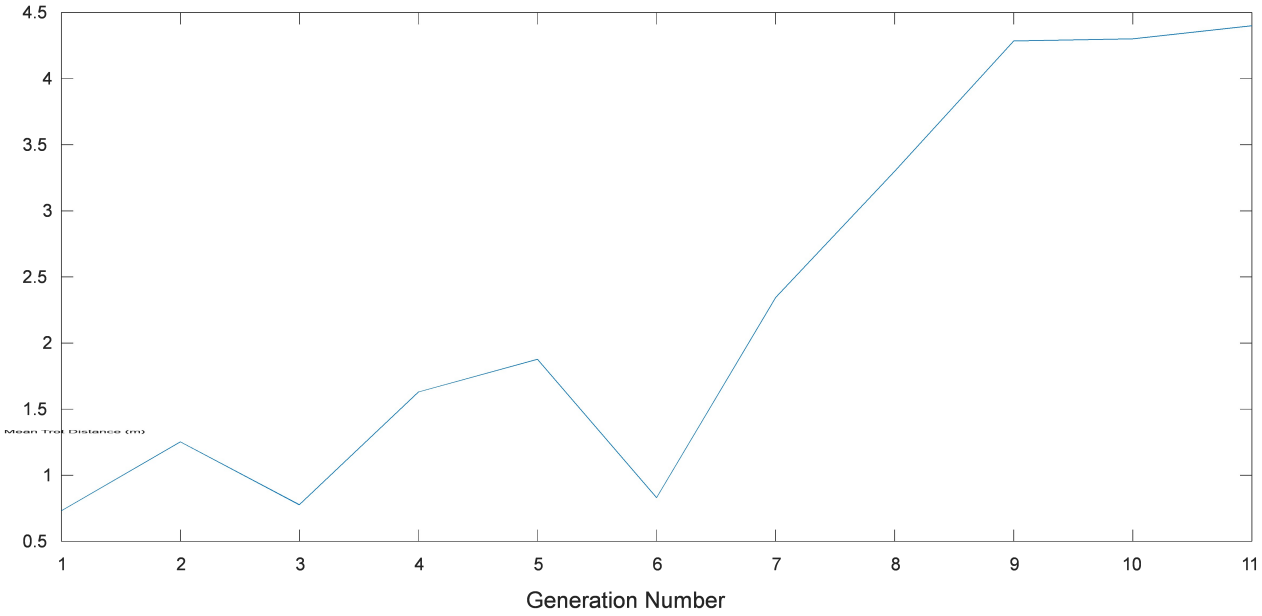


Figure 6.22. Mean trot distance of first ten individuals for all generations

Figure 6.22 shows the trot performance data for the overall optimization run. Looking at it, there are two points where a population gets worse at executing the trotting task, generations three and six. As expected, these populations were more successful than their previous generation at performing the trotting task as shown in Figure 6.21. This indicates that the jumping task and the trotting task are not compatible in the sense that they each require certain design properties in a quadruped that conflict with one another. This is important because it backs up the premise that the optimization algorithm needs to find a compromise between designs to satisfy our performance goals defined by our task weights.

The mean fitness value plot is displayed in Figure 6.23. This plot exhibits a more steady increase compared to the two plots above. The behavior is similar to the performance plots from the previous sections. The rate of increase goes down and the gradient of the curve lowers as we reach the final populations. At this point, the fittest individual is reached and the ongoing mean performance increase is a result of the rest of the individuals converging to the fittest one.

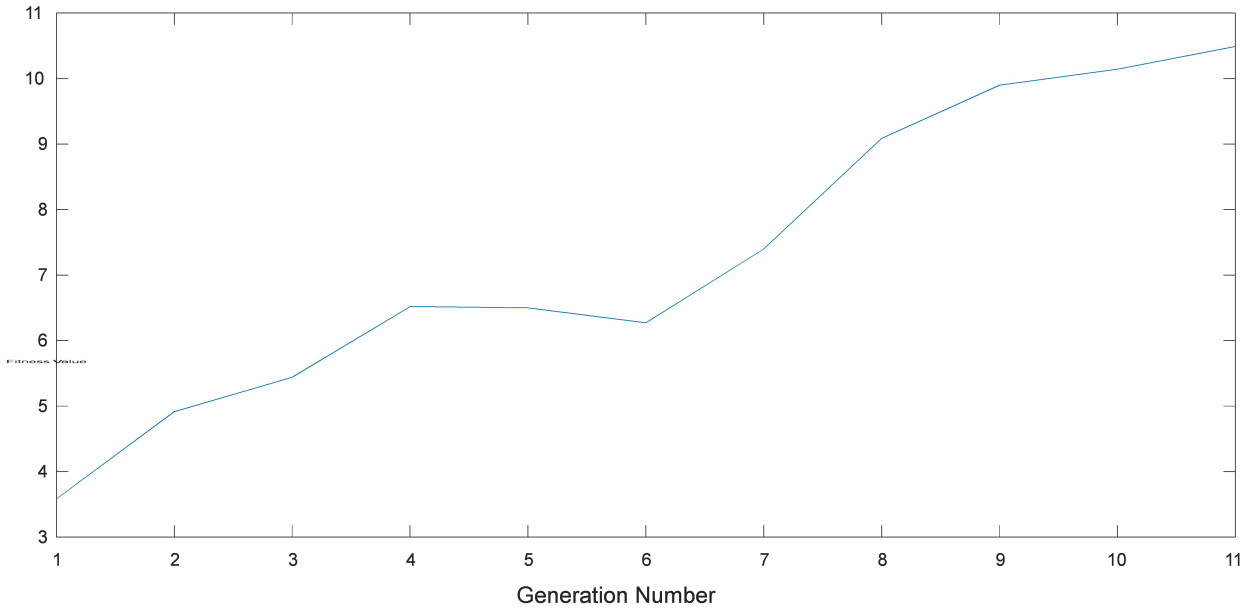


Figure 6.23. Mean fitness value of first ten individuals for all generations

The values kinematic parameters converge towards give us an idea on what traits of a quadruped are beneficial to the task at hand. A comparison of these to the ones in the previous sections should also be informative. The data for all the kinematic parameters are plotted below in Figure 6.24.

Comparing the parameter plots for the overall optimization run in Figure 6.24 to the ones belonging to the jumping and trotting optimization runs, one can see that the final design is a compromise of the fittest jumper and the fittest trotter designs. Both the vertical jumping task and the trotting task benefits from a long L_2, which is the link that connects the shoulder to the knee joint. Same can be observed in L_2 plot below as expected. For L_3 and L_4 links however, jumping and trotting tasks had different results. The jumping task seems to favor a long L_3 and a short L_4, while the trotting task prefers a short L_3 and a long L_4. Looking at their plots below, the jumping requirements are more dominant for the L_3 length since it converges to a longer than average value. The L_4 length is a compromise and stays relatively close to the starting point, which means that it is equally important for both tasks.

In case of the body length dimensions, the fittest jumper design is a wider, square shaped robot while the fittest trotter design is longer and slimmer. The plots for the overall fittest design have both the L_b and the W_b increase steadily. Body length is much more relevant for the trotting task while body width is more important for the jumping task.

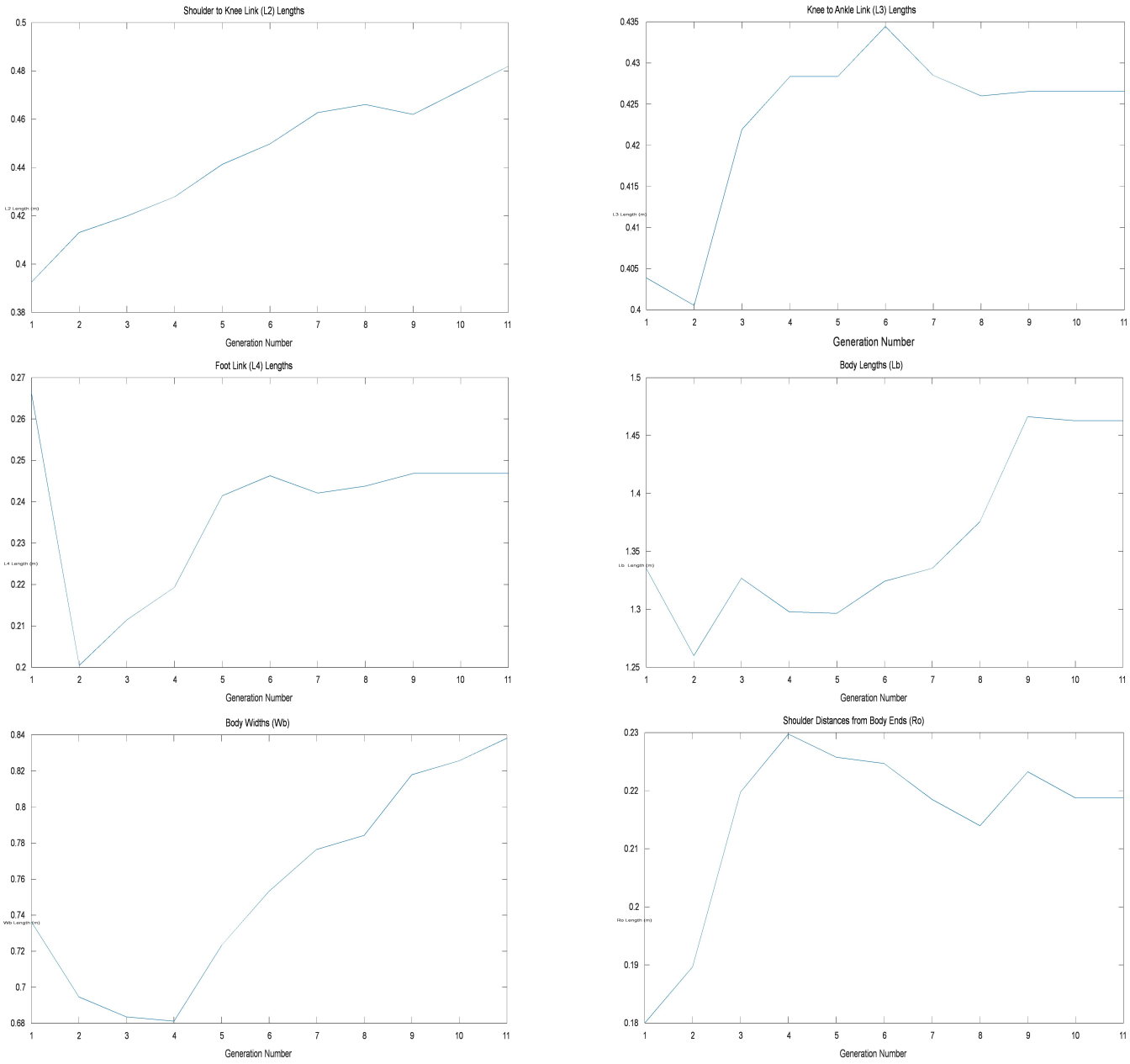


Figure 6.24. Mean parameter value plots of first ten individuals for all generations

The last kinematic parameter is the position of quadruped shoulders where the legs are attached. The fittest trotter design has its front and hind legs as separated from each other significantly. It had a long body length and a short shoulder distance. The fittest jumper on the other hand has its legs close to each other by having a short body length and a long shoulder distance. The overall fittest design has a long body length but its fitness value benefits from having a smaller gap between the legs as it can be seen from the increasing shoulder distance plot.

6.7. Performance of the Genetic Algorithms

Genetic algorithms are tools used to solve an optimization problem by exploring the search space and finding the global maximum. All the performance plots displayed so far belonging to different design problems show that the algorithm does result in each population getting better than the previous one. A typical plot for fitness values of all individuals in all generations for the overall design optimization is shown below in Figure 6.25.

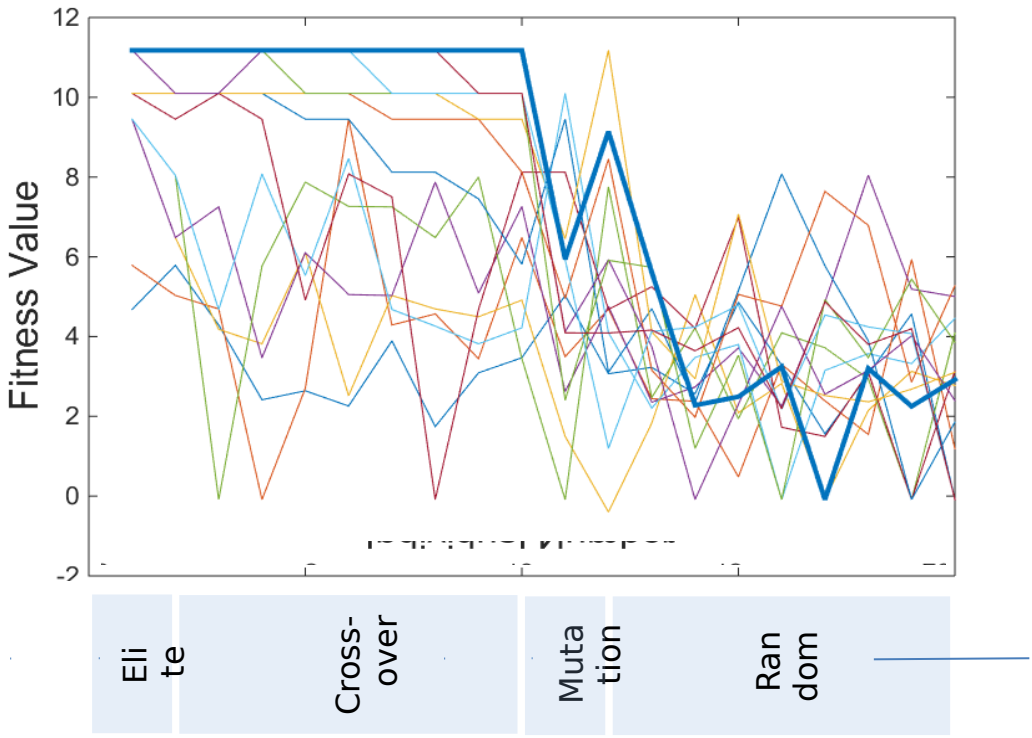


Figure 6.25. Fitness value plot of a 16 generation run

On this figure the band under the horizontal axis shows how each individual is generated (except the first population which is generated randomly). The right halves of the curves are similar throughout the populations since all the individuals are generated randomly. The left halves of the curves involve data formed by the exploitation of information from previous populations. This half keeps increasing for each generation until it starts making a horizontal line. This happens when the fittest individual is reached and other selected individuals are slowly turning into identical copies of it. The last bold line represents the final population in which all the left half of is occupied by the same individual design.

In genetic algorithms, every parameter that forms up an individual is referred to as a gene in a chromosome. A fit individual's genes will naturally spread to the rest of the population in the next generation through the cross-over process. If none of the cross-over individuals replace the fittest individual for several generations the selected individuals of a population will slowly end up looking entirely the same.

Other than keeping a certain part of each population composed of randomly generated individuals, there are two more practices in place to counteract the assimilation of a population by the fittest genes. This is done in order to minimize the risk of converging on a local maximum. The first of these is the mutation step. This is where a random bit from a random gene is changed to form the new individual.

The second practice takes place in the cross-over step. Cross-over is done by cutting two chromosomes into two and combining the first half of the first one with the second half of the second and vice versa to form two new chromosomes. The randomness is introduced at the selection of the cutoff gene. The cutoff happens at a completely random bit, meaning that most of the time it lands through a gene and breaks it rather than landing on a border between two genes and preserving both. This causes the information for that gene to be lost and the gene form after the combination step to be completely different.

An additional measure is to run the algorithm multiple times with different initial populations and going through all the random processes again. This multiple run approach is employed in this work. In Figure 6.26, six different plots belonging to six different runs for the vertical jumping task is displayed. The data is for the fittest individual of each generation. The individual design with the highest reached peak height is plotted for each population. Even though the results are close, the algorithm does not always converge at the same peak height value. Separate tests with different initial conditions, which are the randomized starting population individuals, yield slightly different results. Number of tests run should increase the certainty of the optimum result.

With the inclusion of multiple measures to increase the confidence, genetic algorithms proved to be suitable for the optimization problem presented in this work.

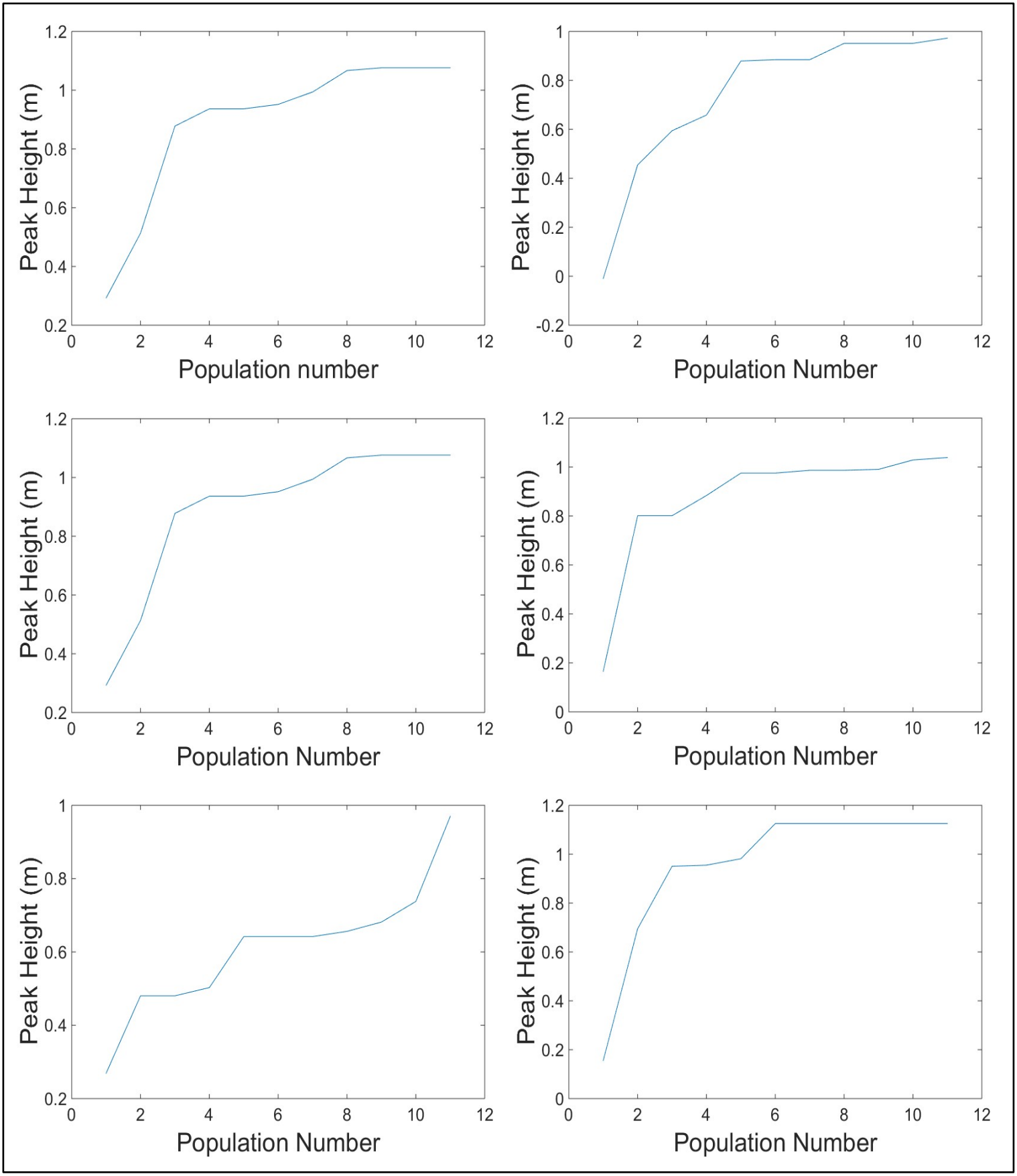


Figure 6.26. Fittest individual jump height plots for six separate test

Chapter 7

7. CONCLUSION

In recent years, there is a trend in legged robotics to devise machines which traverse rugged terrain with dynamic motion capabilities. Quadruped robot studies outnumber other main architectures in these studies, from nature inspiration reasons and since they make a compromise between well balanced structures and overly complicated and costly multi-legged ones. Still, no mathematically systematic approach in designing their kinematic structure is reported. The field is promising for search and rescue, surveying and military applications. The task environment, the locomotion surface and locomotion style can vary between extremes. How would the selection of a specific kinematic parameter set, mainly the link and body dimensions influence the success of the robot in a particular task? This thesis fills the gap of systematic design and answers the above question by employing genetic optimization techniques. Also it makes, to our knowledge, the unique kinematic parameter GA optimization example with dynamic task objectives, not only for legged robots, but also in the robotics field in general.

The proposed procedure requires the preliminary definition for the task in terms of the main trajectory formulation. The trajectory generation parameters are tuned further together with the kinematic parameters in a co-design approach via the optimization algorithm. Another requirement for the optimization system is a full dynamics simulation environment which possesses the following three key properties:

i) It should be computationally efficient. The optimization procedure relies on evaluation of many designs based on simulations. From the practicality point of view, computational efficiency is indispensible.

ii) The ground contact model has to allow dynamic interactions like falling from a height or sudden powerful pushes of the foot against a contact surface. This model should be efficient in computation times too.

iii) The simulation model should be embedded in the computation environment where the GA routines run. This requirement is achieved by writing the dynamic simulation code from scratch. The ABM in conjunction with the Gauss-Seidel search algorithm which was used to obtain exact contact forces for robot motion involving impacts is implemented to address the demands mentioned above.

The first motion investigated in the framework of kinematic optimization was the vertical jump. A reference trajectory featuring a sudden rise from a squat pose was employed for the quadruped model which had 4 DoF legs. Initial and final shoulder heights from the ground level were the trajectory parameters and these parameters were optimized together with the various leg and body length features.

The outcome of the GA optimization is compared to our base design. Our base design is a nature inspired quadruped structure. A number of gaits are generated and tested on this model in the last decade in Sabancı University studies. It presents a natural and practical means of comparison with the GA optimized designs. What the base model and the quadrupeds subject to optimization share in common are the torque capacities of the actuators positioned at the joints.

Our first result in the thesis work was that the base model, which was inspired with its kinematic arrangement from the nature and which had hand tuned parameters for trotting gait, was drastically outperformed by the fittest jumper resulting from genetic tuning in jumping. The huge performance difference, although notable, is not the main observation of the jumping quadruped optimization. The main result was that the GA tuning introduced a complete new set of body and leg proportions. The body proportions created featured an almost square top view – very different from our “quite standard” (among contemporary robots) long rectangular body shaped base model. This result supports the motivation of tuning the kinematic parameters of robots by task dependant systematic optimization means, by considerably deviating from sole nature inspiration.

This thesis stresses the following standpoint. The kinematic proportions found in nature fulfill a multitude of tasks and excel at the combination thereof. Being a living being a tiger,

to sustain its life, needs to run fast, jump and hunt. However, it should be able to lie down to a comfortable sleeping/resting position too. Taking a tiger shape as is as a model for a quadruped robot for fast motion capabilities would be then too conservative. It is a better approach to optimize a robot just on the basis of the tasks it is made to perform. A running robot can be different from the best running animal, although similarities will be natural too. The fact that cost and technology limitations force a robot designer to deviate the mechanical quadruped a lot from a biological one also supports that the kinematic parameter design should be handled via techniques beyond the biological inspiration.

The next optimization problem after jumping was the trotting motion in the thesis. A longer body and long leg positions resulted. The length of the robot body in the walking direction is revealed as the distinctive feature of the fittest trotter.

The isolated tasks and the associated resulting kinematic parameter sets are quite informative for the kinematic features favorable in these tasks. They justify the motivation and success of the GA optimization for kinematic parameters. Still, it should be remembered that most quadruped robot operation scenarios involve more than one task. With this in mind, the GA optimization strategy is put to work with the combination of jumping and trotting tasks. A weighted fitness function is employed to tune the contribution of the two tasks. A mixture of the features obtained from the two isolated optimization problems resulted from the combined optimization. It is important to note that by the use of the weight parameter in the fitness function the design can be tuned closer to the trotter or jumper sides.

The optimization examples presented indicate valuable kinematic design features for the tested task scenarios which are rooted in real world demands from quadruped robots. They also indicate that the GA optimization technique prepared is successful in creating designs suitable for these applications. Worth noting is that the parameters resulting from the GA optimization are reasonable when the balancing and stability enhancement aspects are considered. Still the optimization task is far from straightforward. It represents a very complex optimization problem. It is not difficult to project the level of complexity which would arise by the inclusion of additional tasks to the portfolio of the quadruped. The dual objective optimization presented indicates that the proposed algorithm can be considered as a tool of kinematic design in cases with a multitude of weighted task objectives.

It is interesting to note that changes in link lengths can result in dramatic changes in performance. This also justifies the need for a systematic design approach as presented in this work.

The internal workings and convergence properties of the GA algorithm is investigated in detail. It is observed that the algorithms converge within the employed number of generations and with the various GA settings. These settings can be used as a starting point for further kinematic optimization studies with dynamic task objectives too.

The technique presented and the optimization outputs motivate a number of future research directions.

One such direction is towards a configurable robot concept. A robot with on-board resources to change its link lengths or leg attachment points to adapt to different tasks. The kinematic parameter adjustment method of this thesis can serve as a guide to determine the various configurations suitable to isolated or weighted tasks.

Another idea is the application of the same optimization route for robots which have passive links in addition to the actuated ones. Parameters of spring and damper structures, typically located at the leg tips for shock absorption or energy storage purposes, can also be optimized together with the kinematic and reference trajectory features presented in this thesis.

The work itself can be improved as well. While optimizing the kinematic structure, different assessments other than the success of executing the given tasks, such as minimizing the energy usage of the system, can be utilized. As it stands now the simulations for the entire genetic algorithm process take about 15 hours for individual task optimization, and twice of that for the overall optimization. Finding a way to implement tools like neural networks might improve this. In the dynamic simulations, proposing other solutions to problems like contact forces should also speed up the simulation times.

REFERENCES

Adak, Ö., (2013). *Quadruped locomotion reference synthesis with central pattern generators tuned by evolutionary algorithms.* Sabancı University.

Adak, Ö, Ayit, O. Gülhan, M & Erbatur, K. (2015). Genetic tuning of a central pattern generator for quadruped locomotion.

Akbas, T., (2012). *Bipedal humanoid robot walking reference tuning by the use of evolutionary algorithms.* Sabancı University.

Akbas, T., Eskimez, S. E., Ozel, S., Adak, O. K., Fidan, K. C., &Erbatur, K. (2012, March). Zero Moment Point based pace reference generation for quadruped robots via preview control. In *Advanced Motion Control (AMC), 2012 12th IEEE International Workshop on* (pp. 1-7). IEEE.

Bakdi, A., Hentout, A., Boutami, H., Maoudj, A., Hachour, O., & Bouzouia, B. (2017). Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robotics and Autonomous Systems*, *89*, 95-109.

Beasley, D., Bull, D. R., & Martin, R. R. (1993). An overview of genetic algorithms: Part 1, fundamentals. *University computing*, *15*(2), 56-69.

Beasley, D., Bull, D. R., & Martin, R. R. (1993). An overview of genetic algorithms: Part 2, research topics. *University computing*, *15*(4), 170-181.

Boaventura, T. (2013).*Hydraulic Compliance Control of the Quadruped Robot HyQ*. Diss. PhD Thesis, University of Genoa, Italy and Istituto Italiano di Tecnologia (IIT).

Boston Dynamics,(2014). "Big Dog." http://www.youtube.com/watch?v=cHJJQ0zNNOM

Boston Dynamics,(2014). ―CHEETAH - Fastest Legged Robot.‖ http://www.bostondynamics.com/robot_cheetah.html

Boston Dynamics,(2014). ―Introducing Wildcat.‖ http://www.youtube.com/watch?v=wE3fmFTtP9g

Boston Dynamics,(2014). ―LS3 – Legged Squad Support System.‖ http://www.bostondynamics.com/robot_ls3.html

Boston Dynamics,(2014). Boston Dynamics. ―RHex - Devours Rough Terrain.‖ http://www.bostondynamics.com/robot_rhex.html

Chardonnet, J. R., Miossec, S., Kheddar, A., Arisumi, H., Hirukawa, H., Pierrot, F., & Yokoi, K. (2006, December). Dynamic simulator for humanoids using constraint-based method with static friction. In *Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on* (pp. 1366-1371). IEEE.

Chen, I, and Burdick, J. (1995). "Determining task optimal modular robot assembly configurations." *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. Vol. 1. IEEE.

Chung, W. K., Han, J., Youm, Y., & Kim, S. H. (1997, April). Task based design of modular robot manipulator using efficient genetic algorithm. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on* (Vol. 1, pp. 507-512). IEEE.

Deb, K, and Gulati, S. (2001). "Design of truss-structures for minimum weight using genetic algorithms." *Finite elements in analysisand design* 37.5. 447-465.

Denavit, J., & Hartenberg, R. S. (1955). Kinematic modelling for robot *calibration. *Trans. ASME Journal of Applyed Mechanics*, *22*, 215-221.

Diolaiti, N., Melchiorri, C., & Stramigioli, S. (2005). Contact impedance estimation for robotic systems. *IEEE Transactions on Robotics*, *21*(5), 925-935.

Duan, J., Gan, Y., Chen, M., & Dai, X. (2018). Adaptive variable impedance control for dynamic contact force tracking in uncertain environment. *Robotics and Autonomous Systems*, *102*, 54-65.

Elhoseny, M., Shehab, A., & Yuan, X. (2017). Optimizing robot path in dynamic environments using genetic algorithm and Bezier curve. *Journal of Intelligent & Fuzzy Systems*, *33*(4), 2305-2316

Ellenberger, W., Baum, H., & Dittrich, H. (1949). *An atlas of animals.* New York, NY: Dover Publications.

Erbatur, F., Hasançebi, O., Tütüncü, I., & Kılıç, H. (2000). Optimal design of planar and space structures with genetic algorithms. *Computers & Structures*, *75*(2), 209-224.

Erbatur, K., & Kawamura, A. (2003, October). A new penalty based contact modeling and dynamics simulation method as applied to biped walking robots. In *Proc. 2003 FIRA World Congress* (pp. 1-3).

Estremera, J., and Waldron, K. (2008) "Thrust control, stabilization and energetics of a quadruped running robot." *The International Journal of Robotics Research* 27.10: 1135-1151.

Featherstone, R. (1999). A divide-and-conquer articulated-body algorithm for parallel O (log (n)) calculation of rigid-body dynamics. Part 1: Basic algorithm. *The International Journal of Robotics Research*, *18*(9), 867-875.

Fidan, K. C., Akbaş, T., Eskimez, Ş. E., Özel, S., Adak, Ö. K., Drama, Ö., ... & Erbatur, K. (2011). Dört bacaklı robotlar için önizlemeli kontrol ile sıfır moment noktası tabanlı yürüme yörüngesi sentezi.

Focchi, M. (2013).*Strategies to Improve the Impedance Control Performance of a Quadruped Robot*. Diss. Ph. D thesis, Istituto Italiano di Tecnologia, Genoa, Italy.

Fukuoka et al. (2003). "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts."*The International Journal of Robotics Research* 22.3-4: 187-202.

Grierson, D. E., and W. H. Pak. (1993) "Optimal sizing, geometrical and topological design using a genetic algorithm." *StructuralOptimization* 6.3: 151-159.

Gurfinkel, V. S., et al. (1981). "Walking robot with supervisory control." *Mechanism and Machine Theory* 16.1: 31-36.

Herzog, A., Rotella, N., Schaal, S., & Righetti, L. (2015). Trajectory generation for multi-contact momentum-control. *arXiv preprint arXiv:1507.04380*.

Herzog, A., Schaal, S., & Righetti, L. (2016, October). Structured contact force optimization for kino-dynamic motion generation. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on* (pp. 2703-2710). IEEE

Hirose, S. (1984) "A study of design and control of a quadruped walking vehicle."*The International Journal of Robotics Research* 3.2: 113-133.

Hutter, et al. (2012)*StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion*. Autonomous Systems Lab, ETH Zurich.

Jakiela, M. J., Chapman, C., Duda, J., Adewuya, A., & Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 339-356..

Kane, C., & Schoenauer, M. (1996). Topological optimum design using genetic algorithms. *Control and Cybernetics*, *25*, 1059-1088.

Kato, T., Takanishi, A., Jishikawa, H., & Kato, I. (1981, September). The realization of the quasi-dynamic walking by the biped walking machine. In *Fourth Symposium on Theory and Practice of Robots and Manipulators* (pp. 341-351). Warsaw.

Khan, H., Featherstone, R., Caldwell, D. G., & Semini, C. (2015, February). Bio-inspired knee joint mechanism for a hydraulic quadruped robot. In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*(pp. 325-331). IEEE.

Khatami, S. & Sassani, F. (2002) "Isotropic design optimization of robotic manipulators using a genetic algorithm method," in Proc. IEEE Int. Symp. Intell. Contro, Vancouver, BC, Canada, pp. 562–567.

Kim, H., Kang, T., Loc, V. G., & Choi, H. R. (2005, April). Gait planning of quadruped walking and climbing robot for locomotion in 3D environment. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* (pp. 2733-2738). IEEE..

Kim, J. O., & Khosla, P. (1992). A multi-population genetic algorithm and its application to design of manipulators.

Kimura et al. (1999). "Realization of dynamic walking and running of the quadruped using neural oscillator."*Autonomous robots* 7.3: 247-258.

Kolter, J. Zico, and Andrew Y. Ng. (2011). "The stanford littledog: A learning and rapid replanning approach to quadruped locomotion." *The International Journal of RoboticsResearch* 30.2: 150-174.

Lee, J., & Kim, D. W. (2016). An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. *Information Sciences*, *332*, 1-18.

Liu, X. J., Li, J., & Zhou, Y. (2015). Kinematic optimal design of a 2-degree-of-freedom 3-parallelogram planar parallel manipulator. *Mechanism and Machine Theory*, *87*, 1-17.

Luh, J. Y. S., Walker, M. W., & Paul, R. P. (1982). Resolved motion force control of robot manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, *102*, 126-133.

Mason, S., Rotella, N., Schaal, S., & Righetti, L. (2017). A MPC Walking Framework With External Contact Forces. *arXiv preprint arXiv:1712.09308*

Maufroy, et al. (2010). "Stable dynamic walking of a quadruped robot ―Kotetsu‖ using phase modulations based on leg loading/unloading." *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE.

McGhee, Robert B. (1985). "Vehicular legged locomotion." *Advances in Automation and Robotics* 1: 259-284.

Merlet, J. P. (2003). "Determination of the optimal geometry of modular parallel robots," in Proc. IEEE Int. Conf. Robot. Autom., Taipei, Taiwan, Sep. 14–19, pp. 1197–1202.

Mirtich, B. V. (1996). *Impulse-based dynamic simulation of rigid body systems*. University of California, Berkeley.

Miura, Hirofumi, and Isao Shimoyama. "Dynamic walk of a biped." *The International Journal of Robotics Research* 3.2 (1984): 60-74.

Momani, S., Abo-Hammour, Z. S., & Alsmadi, O. M. (2016). Solution of inverse kinematics problem using genetic algorithms. *Applied Mathematics & Information Sciences*, *10*(1), 225.

Mosher, R. S. (1968). "Test and evaluation of a versatile walking truck." *Proceedings of off-road mobility research symposium*.

Nagatani, K., Kinoshita, H., Yoshida, K., Tadakuma, K., & Koyanagi, E. (2011). Development of leg-track hybrid locomotion to traverse loose slopes and irregular terrain. *Journal of Field Robotics*, *28*(6), 950-960.

Playter, et al. (1992). "Control of a Biped Somersault in 3D."*Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*. Vol. 1. IEEE.

Raibert, Marc H. (1986).*Legged robots that balance*. MIT press.

Raibert, Marc H. (1986). "Legged robots." *Communications of the ACM* 29.6: 499-514.

Raibert, Marc H. (1990) "Trotting, pacing and bounding by a quadruped robot." *Journal of biomechanics* 23: 79-98.

Raibert, Marc, et al. (2008) "Bigdog, the rough-terrain quadruped robot." *Proceedings of the 17th World Congress*. Vol. 17. No. 1.

Raibert M., Chepponis M., and Brown Jr.(1986). "Running on four legs as though they were one." *Robotics and Automation, IEEE Journal of*2.2: 70-82.

RaibertM, Blankespoor K, Nelson G, et al. (2008) Bigdog, therough-terrainquadrupedrobot.*Proceedings of the 17th World Congress*. 17(1)

Rao, A. C. (2003). "A genetic algorithm for epicyclic gear trains." *Mechanism and machine theory* 38.2: 135-147.

Roston, G. (1994). *A genetic methodology for configuration design* Doctoral dissertation, Carnegie Mellon University.

Schenker, P. S., Pirjanian, P., Balaram, J., Ali, K. S., Trebi-Ollennu, A., Huntsberger, T. L., ... & Rzepniewski, A. (2000, October). Reconfigurable robots for all-terrain exploration. In *Sensor Fusion and Decentralized Control in Robotic Systems III* (Vol. 4196, pp. 454-469). International Society for Optics and Photonics.

Seok, S., Wang, A., Chuah, M. Y., Otten, D., Lang, J., & Kim, S. (2013, May). Design principles for highly efficient quadrupeds and implementation on the MIT Cheetah robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (pp. 3307-3312).

Semini, C. (2010). "HyQ—Design and development of a hydraulically actuated quadruped robot." *PD Thesis, University of Genoa, Italy*.

Smith, J. (2006).*Galloping, bounding and wheeled-leg modes of locomotion on underactuated quadrupedal robots*. Diss. McGill University.

Song, B., Wang, Z., & Sheng, L. (2016). A new genetic algorithm approach to smooth path planning for mobile robots. *Assembly Automation*, *36*(2), 138-145.

Takahaashi, M., Yoneda, K., & Hirose, S. (2006, May). Rough terrain locomotion of a leg-wheel hybrid quadruped robot. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (pp. 1090-1095). IEEE.

Yamazaki, Kenneth S. (1999).*The Design and Control of Scout I, a Simple Quadruped Robot*. Diss. McGill University.