Contact Modeling as Applied to the Dynamic Simulation of Legged Robots

by

Orhan Ayit

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

SABANCI UNIVERSITY

September 2015

APPROVED BY

Assoc. Prof. Dr. Kemalettin Erbatur (Thesis Supervisor)	....
Prof. Dr. Mustafa Ünel	....
Asst. Prof. Dr. Barkan Uğurlu	....

DATE OF APPROVAL: 05/08/2015

Contact Modelling as Applied to the Dynamic Simulations of Legged Robots

Orhan Ayit

ME, M.Sc. Thesis, 2015

Thesis Supervisor: Assoc. Prof. Dr. Kemalettin Erbatur

Keywords: Contact modeling, penalty based, exact based, physics based, linear complementarity problem, projected Gauss Seidel algorithm, quadruped, four legged robot.

Abstract

The recent studies in robotics tend to develop legged robots to perform highly dynamic movement on rough terrain. Before implementing on robots, the reference generation and control algorithms are preferably tested in simulation and animation environments. For simulation frameworks dedicated to the test of legged locomotion, the contact modeling is of pronounced significance. Simulation requires a correct contact model for obtaining realistic results.

Penalty based contact modeling is a popular approach that defines contact as a spring - damper combination. This approach is simple to implement. However, penetration is observed in this model. Interpenetration of simulated objects results in less than ideal realism. In contrast to penalty based method, exact contact model defines the constraints of contact forces and solves them by using analytical methods.

In this thesis, a quadruped robot is simulated with exact contact model. The motion of system is solved by the articulated body method (ABM). This algorithm has O(n) computational complexity. The ABM is employed to avoid calculation of the inverse of matrices. The contact is handled as a linear complementarity problem and solved by using the projected Gauss Seidel algorithm. Joint and contact friction terms consisting of viscous and Coulomb friction components are implemented.

Bacaklı Robotların Dinamik Simülasyonları için Temas Modeli

Orhan Ayit

ME M.Sc. Tezi, 2015

Tez Danışmanı: Doç. Dr. Kemalettin Erbatur

Anahtar kelimeler: Ceza tabanlı temas modelleme, fizik tabanlı temas modelleme, doğrusal tamamlayıcı problem, Gauss Seidel algoritması, dört bacaklı robot

Özet

Robotik alanındaki son çalışmalar engebeli arazide çok dinamik hareketler gerçekleştiren bacaklı robotların geliştirilmesine yönelmektedir. Referans sentezi ve kontrol algoritmalarının robotlara uygulamadan önce simülasyon ve animasyon ortamlarında test edilmesi tercih edilmektedir. Bacaklı robotların hareket kabiliyetine özel simülatörler için temas modellemesi çok önemlidir. Gerçekçi sonuçlar elde edebilmek için simülasyonların doğru temas modeline ihtiyacı vardır. Ceza tabanlı temas modeli, teması yay ve sönümleyici ile tanımlayan popüler bir yaklaşımdır. Bu yaklaşımı uygulanması basittir. Fakat, bu modelde iç içe geçme görülmektedir. Simule edilen objelerin birbirinin içine girmesi, ideal gerçekçilikten uzaklaşmasına neden olur. Ceza tabanlı metodun tersine, kesin temas modeli temas kuvvetlerinin kısıtlamalarını tanımlar ve bunları çözümsel metotlar kullanarak çözer.

Bu tezde, dört bacaklı bir robot için kesin temas modeli elde edilmiştir. Sistemin hareketi, döner eklemli vücut metotu ile çözülmüştür (ABM). Bu algoritma, O(n) hesaplama karmaşıklığına sahiptir. ABM, atalet matrislerinin tersini hesaplamaktan kaçınmak için kullanılmıştır. Temas, doğrusal tamamlayıcı problem olarak ele alınır ve Gauss Seidel algoritması ile çözülür. Viskoz ve Coulomb sürtünme bileşenlerinden oluşan eklem ve temas sürtünmeleri uygulanmıştır.

To my family…

Acknowledgments

First of all, I would like to express my sincere gratitude to my thesis advisor Assoc. Prof. Dr. Kemalettin Erbatur for his invaluable coaching and also sharing his experience and knowledge with me. During my graduate education, his academic advices and lectures made significant contribution on my engineering skills and background.

I also state the deepest appreciation to my M.Sc. thesis committee members; Prof. Dr. Mustafa Ünel and Assist. Prof. Dr. Barkan Uğurlu for sharing their precious time and valuable comments on my thesis.

I would like to acknowledge the financial support provided by The Scientific and Technological Research Council of Turkey (TUBITAK) through the project "Quadruped Robot Design, Construction and Control" under the grant 114E618.

I would like to thank my family for their support, encouragement and endless love not only during my graduate period but also throughout my life in general.

My sincere thanks also goes to Mert Mehmet Gülhan who collaborated in this venture.

Thanks to my friends, Ömer Kemal Adak, Beste Baheci, Koray Erkekli, Aykut Önal, Uğur Sancar, Talha Boz, Fırat Yavuz, Ahmet Eren Demirel and Sanem Evren Han for their friendship.

Lastly, I would like to express my warm thanks to İpek Köken and Gökhan Alcan for their fellowship and joyful memories throughout the time we spent together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Studies on robotics were initialized in 1950s to perform dull, dirty and dangerous work in place of humans. Industrial robots were developed with fixed base such as industrial robot of GM [86] and AMF Verstran robot [86] and these manipulators are categorized based on kinematic arrangements as articulated, spherical, SCARA, cyclindirical and cartesian [104].

In recent years, researchers' interests in mobile robotics has risen rapidly due to incenting challenges and possible employabilities in different areas such as industry, military, health, safety and environment. In contrast to fixed based industrial robots, mobile robots may require to keep self balances and generate optimal pathes for both navigation and obstacle avoidance which are valuable reasearch areas in robotics. The mobile robots are mainly categorized in terms of legged, wheeled, swimming, flying and crawler. Siegwart et al. stated that wheeled and legged robots are generally preferred categories of the mobile robots [101] on land.

Wheeled robots play a significant role in robotic studies due to inessentiality of complex algorithms for their balance issues. Besides their inherent balancing advantage, ease of setting up by using off-the-shelf components and flexibility of employment in various environments are also benefits of wheeled robots. Various wheel types are available, including standard wheel, castor wheel, swedish wheel and spherical wheel. [101].

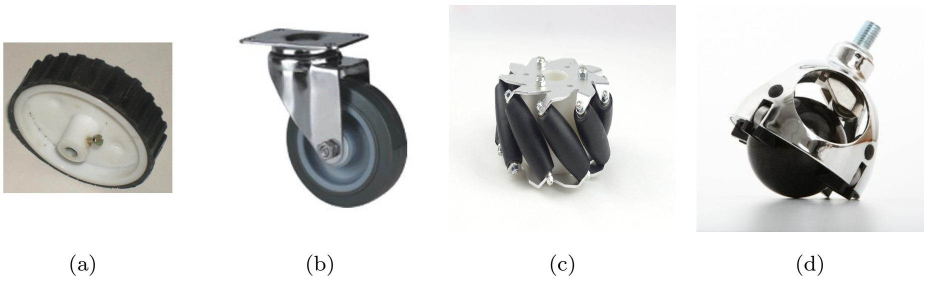


Figure 1.1: (a) Standard wheel, (b) Castor wheel, (c) Sweedish wheel, (d) Spherical wheel

Figure 1.1 shows these wheeled types. Studies on legged robots has increased in last 30 years to handle moving on rough terrain like animals [92]. Maintaining balance in harsh environment is the main research area in the field of legged robots. However, this can only be accomplished with complex control algorithms. In addition, another significant research topic is mimicing difficult movement which animal or humans can. Jumping climbing, walking, running are examples. To perform these tasks by robots, actuators must achieve fast responses and output high power. Hydraulic actuators can meet these specifications.

1.1 Motivation

With the ability to handle highly dynamic tasks, legged robots with hydrualic actuators made great impact on legged robotics research. Recent researches focused on adapting hydraulic robots to outdoor applications (BigDog [21], HyQ [99], Atlas [20]). Quadruped robots come into prominence for outdoor applications due to the advantage of keeping balance when compared with the robots which have less than four legs. They also posses manufacturing simplicity when compare with robots with more than four legs. Simulation has significant role in robotics because it provides a means to develop and verify control algorithms before implementing them on a real robot. The quality, and even more importantly, the stability of the simulation is highly dependent on the contact model. Penalty based methods, which are easy to implement from the programming point of view, are adequate for the simulation of slow motion and when the range of occuring contact forces is quite narrow. When these simplifying conditions are met, it is straightforward to apply manual and ad-hoc tuning techniques for the typically involved spring and damped parameters of the penalty based method. This is the case when only walking and slow speed quadruped gaits are considered. However, highly dynamic motion, as examplified by running, galloping and jumping over obstacles is different in this aspect. A penalty algorithm tuned for a certain force range can fail to represent realistic contacts for take-off and landing motion necessary to carry out a jump over an obstacle. More dramatically, stability of simulation can be lost. here, stability does not refer to robot balance of dynamic stability core to control sysem design. Rather, by the "loss of simulation stability," computation of very high-magnitude (and thus unrealistic) forces by the contact model is meant. These high-magnitude forces usually cause the simulated robot to spin or fly off the ground with very high speeds. This behavior of simulation can be avoided by applying the more sophisticated exact contact force computation approaches. A legged robot simulator equipped with a "stable" contact model can perform as a backbone of dynamic quadruped motion research.

1.2 Contribution of the Thesis

Ruspini and Khatib defined the constraints for contact and collision forces and offered analytical solution to solve them by using Lemke algorithm [97]. The significant deficiency of the algorithm is handling contact modeling without friction forces. In 2006, Chardonnet et al. developed an algorithm to allocate this deficiency and added Coulomb friction term [14]. In addition, the projected Gauss Seidel algorithm is used to solve contact forces with friction and simulation with this new contact model is compared with a penalty based contact simulation of the HRP humonoid robot. However, as mentioned in [14], the robot is simulated without joint friction. In this thesis, the exact contact modeling is used to simulate a quadruped robot with joint and ground friction terms. Both visocus and Coulomb friction effects are considered. The stable and realistic method will be used as an integral part in the simulation framework in the TUBITAK funded 1001

Project 114E618 Quadruped Robot Design, Construction and Control. Comparisions of penalty-based and exact computation techniques on a quadruped robot are presented with simulation results.

1.3 Outline of the thesis

This thesis is organized into the following chapters:

- Chapter 2 presents a review of contact modeling algorithms. Information on legged robots and the virtues of hydraulic actuators in legged robotic systems are briefed. The role of simulation in robotic design and control and contributions of contact modeling in the quality of simulation is stressed.
- Chapter 3 reviews articulated body method (ABM) which sits at the core of the dynamics simulation in this thesis. The application of the ABM to robotic systems is reviewed progrsssively: Derivations are carried out firstly for a serial linkage with fixed base. This is followed by the derivation for a tree-like linkage with fixed base. Finally, the family of free-fall manipulators (to which quadrupeds belong) are covered in the same context.
- Chapter 4 implements techniques for contact modeling. An ABM based method is employed for obtaining exact contact forces. Also, constraints for contact and collision are considered and modeled by using the ABM method. The mentioned model implementation is explained in detail.
- Chapter 5 presents the results of simulation. A penalty based contact model and the developed exact contact model are compared via simulations with a four legged robot.
- Chapter 6 concludes the thesis. Expected use of the results obtained in the thesis are discussed.

Chapter 2

Background

2.1 Introduction

The aim of this chapter is the review of legged robotics and the role of contact modeling in simulation studies.

2.2 Legged Robots:

Research on legged robots dates back on mid 20th century with speed up in the last thirty years. Due to capability of moving on land, mobile robots which are autonomous or controlled by remote control, come into prominence for the execution of dangerous tasks. For example, robots can detect and annihilate mines, gather information about enemies, carry heavy loads, and support battle in military applications. They can also respond to natural disasters such as fire and earthquakes. Land mobile robots can be wheeled vehicles, tracked or legged mechanisms. Based on applications, the legged robots can have advantages over tracked and wheeled robots. Legs are beneficial when robots perform activities such as climbing steps or sets, moving on rocky terrain, and crossing ditches. The wheeled and tracked robots require to contact with the ground continuously. However, legged robots can contact the grounds at points which are far from each other. That enables legged robots to avoid some obstacles easier than it is the case with wheeled and tracked robots. Also legged robots with their separate contact points, move on farms without irreversible harm to the crops. This is in contrast to tracked and wheeled robots. One legged, two legged and four legged or more than four legged robots are developed. When compared with one legged and two legged robots, four legged robots posses a more balanced structure. Moreover, manufacturing of four legged robots is simpler when it is compared to robots which have more than four leg. In many applications (for example in military operations) ability to run fast and carry heavy load become requirements. This performed by many four legged animals in nature. By these motivations, many studies about four legged robots have been carried out.

The first important legged robot project was constructed by General Electric. In this project, a vehicle with legs was designed and it was driven by a human operator (Liston and Mosher, 1968)[69]. McGhee [73] in U.S.A. and Gurfinkel [44] in S.S.C.B firstly implemented computer control on legged robots. In 1984, a computer controlled machine with pantograph type legs was designed by Hirose and this machine had the ability to climb steps [46]. These pioneer three robots have a significant common feature that the projection of the robot center of gravity on the world coordinate was in the support polygon, defined by contacting leg tips. This kind of gait is called as static walking [94]. By static walking, balance is maintained continuously. However, the robot moves with low speed. In the contrasting dynamic walking, there are some situations which make projection of the center of gravity on the world frame leaves the support polygon. First studies on dynamic walking were carried out by Kato et. al,[57] and Miura and Shimoyama [80]. Other examples of pioneer robots can be found in Raibert[94] and Raibert [93] Recently, significant legged robot projects and researh are carried out too. Some of them are Scout I [118], Scout II [8], Aibo[**?**], Kotetsu [72], Patrush[59], Tekken [40], Tekken2 [60], PAW [102], RollerWalker [26], Mrwallspect [56], Kolt [32], Cheetah-Cub [105] which had contributions on literature with successful results in 1990s and early of 2000s. Also, Raibert's studies between 1970s and today play an important role on legged robotic.

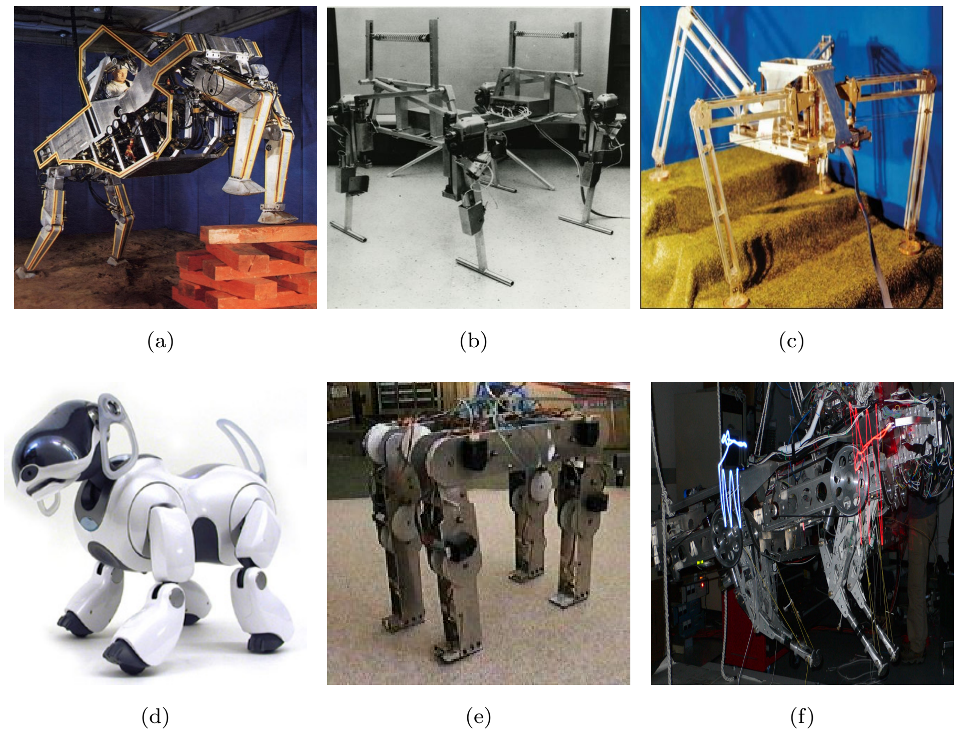


FIGURE 2.1: (a) General Electric Walking Truck [69], (b) PonyPony [73], (c) PV-II [46], (d) AIBO [?], (e) Patrush [59], (f) Kolt [32]

Raibert began his studies on legged robots by designing a jumping monopod robot with hydraulic an actuator [94, 93]. Moreover, by using the same jumping principle as with the monopod, he continued research on a two legged robot [91] and a four legged robot [93, 95]. The Boston Dynamics Company, established by Raibert, developed legged robots for military goals and these robots had a significant impact on the field of robotics. Examples of the legged robots of Boston Dynamics can be given as: BigDog [92], LS3 [24], Littledog [63], Cheetah [22], WildCat [23], Rhex [25]. In 2013, this company was sold to Google Company and this also indicates that the studies on legged robotics may create a new industrial area. The most important result of the studies in Boston Dynamics is the motivation of researchers in other institutions focus on dynamic legged robot with hydraulic actuators. During the last decade, the number studies on the four legged robots are increased substantially. For example, a four legged robot, HyQ, designed by IIT (Italian Institute of Technology) is inspired from BigDog. Due to military adressed design of the BigDog robot, only limited information about it can be gathered with the exception of video demonstrations. In contrast, the HyQ researcher group published papers which include significant information about their design of legged robots with hydraulic actuators [99, 38, 12].

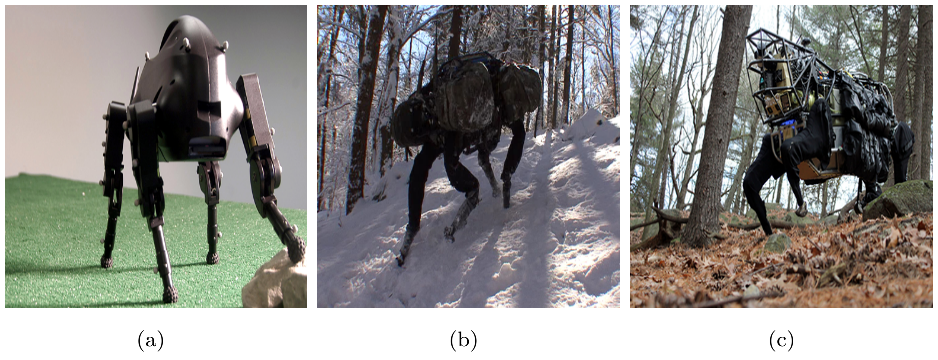


FIGURE 2.2: (a) LittleDog[63], (b) BigDog[92], (c) LS3[24]

The hydraulic actuator has a significant feature which distinguish the Boston Dynamics and Italian Institute of Technology's robots from others, mentioned above. The distinguished feature can be called as highly dynamic movement. There is no exact description of the concept of highly dynamic. However, the HyQ research group uses this term in their publications where a robot is called highly dynamic when it has capabilities to run, jump, and react fast to disturbances. These abilities play key roles on walking/runing on rough terrain. Electrical, pneumatic and hydraulic actuators are generally used in robotics. The hydraulic actuator has the highest power-to weight ratio among to these types. Therefore, when this power is controlled properly, robots can be developed to perform highly dynamic movement on rough terrain.

Studies show that many robots which are mentioned above, can walk or run on rough or smooth terrain. Kotetsu [72] has ability to move on smooth surfaces. Patrush [59] runs on smooth surfaces and walks on surfaces with 12 degrees slope. Tekken [40] can walk forward on surfaces with 10 degrees, it also walks in the lateral direction on surfaces with 5 degrees slope. In addition, Tekken can walk on pebble stones with a speed of 0.6 meter/sec. PAW [102] has the ability to walk on surfaces with 16 degree slope and it jumps over obstacles which have a height of 166 mm. Scout can climb stairs which have the height as equal to 0.45 times the leg length of the robot has [13]. Mrwallspect can move up and down surfaces which has 35 degrees and it can jump over obstacles of 1.1 meter height [92]. HyQ can run with a speed of 1.7 and jump to a height 0.2 m [12]. The dimension of the robot has a significant effect on the speed data. So do actuators, the control method and sensors. BigDog stands out with ability to move on rough terrain and being robust when disturbance forces are applied. The demonstration videos of BigDog show that the fast response hydraulic actuators have significant effect on rejected disturbances [21].

2.2.1 Kinematic Configuration:

Studies show that four legged robots can be designed with different degrees of freedom and kinematic form. Some of the robots have only shoulder and hip as revolute joints and when contact between leg tips and ground is occur, energy of collision impact is absorbed by springs which are located between joints [118]. This type robot is suitable for bounding movement. In addition, in some kinematic configurations, robot can use legs and wheels or tracks to move. These wheel and tracks are added at the middle or at the tip of the leg such legged robots can be reconfigured and change mode to move by using wheels or tracks [98, 107, 83]. Kinematic arrangements similiar to natural ones are significant for legged robotics research (BigDog, LS3, HyQ and StarlETH [51]). This leg configuration is also suitable for many walking types. In the HyQ robot, BigDog 2005 and BigDog 2006 versions, a 3 degrees of freedom kinematic arrangement with revolute joints is used. In BigDog 2010 version, one degree of freedom is added on the ankle of the robot. This development made the robot to contact the ground with a desired angle on the sagital plane.

2.2.2 Synthesis of References

For the design of a walking machine, mechanical design, synthesis of reference trajectories and control methods must be combined. In many studies the trajectory synthesis of four legged robots is inspired from nature. Animal walking types and step timing were systematically studied in 1800s. In Mulbridges studies, walking manners of mammals were investigated and recorded [82]. Walking can be performed with different types[47] that depend on the order of swing phase, timing of swing phase, duration of stance phase, and duration of swing phase. Observed walking types of four legged animals are crawl, trot, pace, canter, and gallop. During crawl, which is generally performed by turtles, at least three legs are always in contact with ground and that pattern provides stability of balance. In crawl motion, left front leg, right back leg, right front leg and left back leg move orderly. Studies show that trot movement is performed by most of the four legged animals such as horse and camel. During this type of locomotion, diagonal legs move together. Pace motion, similar to trot, is performed by salamander, lizard and similar creatures. During a pace, same sided legs move together. Canter and gallop are observed with horses. Gallop is performed for fast travels. Trot type walking was the main topic in many studies[120, 119, 64, 108, 88]. Trot is more stable than pace.

Central Pattern Generators can be used in reference gait synthesis. In this approach, for reference synthesis of joint coordinates or leg tips, fixed limit cycle dynamic equations are used. These equations are categorized in two groups as of neural oscillators and nonlinear oscillators. Parameters of oscillators can be obtained by using trial and error methods, optimization or learning algorithms. Output of an oscillator can be taken as the reference for an articulated joint of a four-legged robot. Other articulated joint references can then be obtained by adding phase differences to this main oscillators output. These phase differences determine the walking type of a robot. This approach completes reference synthesis by using only one oscillator. The addition of phase differences provides simplicity for gait transition. A gait transition means that the type of the locomotion is changed without a break between two types. The use of a stability criterion for providing the robot's balance is important. One of the stability criteria is that the projection of the robot center of gravity on ground is kept in the support polygon, defined by the tips of legs in contact with ground. This criterion is valid for slow movement however, when the robot moves fast, this projection cannot be kept in the support polygon. For this situation, the so-called Zero Moment Point criterion can be applied. Reference generation by the use of the ZMP criterion is popular in biped robotics research [113, 55, 103, 89, 29, 109]. According to this criterion, ZMP must be kept in the support polygon for a balanced gait.

2.3 Importance of Simulation in Robotics

Simulation of multiple rigid bodies has a significant place on a wide range of applications such as movies, molecular dynamics, games and robotics [114]. Many studies were the performed for improving the simulators. These improvements are in accuracy of the simulator and in computational efficiency. The requirements imposed on simulators change according to application. Some applications require a fast simulators while other one require an accurate one [10]. For example, Mirtich states that the main requirement of a simulator is accuracy and the second one is computational efficiency [79]. Bender [10] mentions that in animation, the simulator does not required to be as accurate as a simulator for robot dynamics. However, speed of the simulator is important because a real time or fast simulator can make virtual world to be perceived more realistic [10]. In robotic simulators, accuracy is more important than high speed [75], because, new theories on robotics are be tested in simulation. [14]. Via simulation, the theories can be verified without harming to the robots and their surrounding [75]. In dynamic simulators, the accurate computation of contact forces and torques between robot and environment is a significant problem [48]. For solving this problem, a considerable amount of efforts are spent. For example, David Baraf offered a new algorithm with exact contact modeling [6], also Fujimoto et al. applied a new penalty based contact modeling to biped walking robots [39]. The prominent approaches to deal with this problem are the penalty based method, analytical contact modeling, impulse based technique and time stepping methodology. In addition, to obtain satisfying solutions from these contact modeling methods, proper methods for the derivation of the motion have to be applied [62] and solvers to obtain the results of these equations must be robust and efficient [117].

Some of the popular physics engines for dynamic simulation are:

- **Bullet**: The Bullet physics engine is mostly used in robotics and computer graphics. This engine uses the maximal coordinate method to obtain equations of motion and impulse based damping [30]. The drawback of the engine is that unrealistic behaviors may be seen under some conditions [68].
- **MuJoCo**: The physic engine is developed to simulate multi-joint kinematic models rapidly [81]. That algoritm calculates the motion of the system by the reduced coordinate approach. In addition to this, this physics engine formulates the contact by a velocity-stepping approach [111].
- **PhysX**: PhysX is proposed to simulate models rapidly but not necessarily accurately. Due to this reason the engine is not preferred in robotics applications [30]. Due to its speed, the engine can convince its users of reallistic results [49].
- **ODE**: ODE is an important physics engine in the field of robotics. In this engine, the interpenetration is avoided and friction forces on joints and ground are modeled to obtain realistic results [17].
- **Havok**: Havok, which is a popular game engine, is used in Harry Potter movies, Halo game, Assassin's Creed game and so on [45]. In this engine, Coriolis forces are not calculated and due to that, the engine is not suitable for robotics applications [30].

2.3.1 Equations of Motion

Equation of motion has a significant role on contact modeling because, the applied forces and acceleration of links are found with reference to robots latest position by using the equation of motions techniques therefore, the accuracy of forces and acceleration lead to exact and effective results of contact modeling. Shabana, et al. mentioned that equation of motion techniques can be diversified according to their selection of the system coordinates [100]. For unconstrained rigid bodies motion, the opportunity of the selecting system coordinate is not much, therefore Newton - Euler is a generally used simple and effective equation of motion method for an unconstrained one [100]. However, for constrained rigid body motion, there are many opportunities to choose system coordinates differently and this means that many different methods can be used to define equations of motion for constrained rigid bodies [100]. Kenwright et al. categorize the dynamic equations of constrained rigid bodies in two groups in terms of maximal coordinate methods and reduced coordinate methods [58].

2.3.1.1 Newton Euler Formulation

Newton Euler method is generally preferred for nonconstrained system [100]. This algorithm is chosen because according to Featherstone, this method is a valuable algorithm to solve the equations of the inverse dynamic [34]. Orin et al. use the Newton Euler method recursively and the algorithm has O(n) computational time [87], also Luh et al. use O(n) recursive Newton Euler formula [70]. Non-recursive methods have slow computational time because the algorithms share large period of calculation time for repeated calculation. Featherstone provides an example about it as; the recursive Newton Euler method has lower computational time when compared with non-recursive method such as the Uicker/Kahn method [34]. These formulations are used in many robotic simulators such as, OpenHrp [54] and Open dynamic engine [54] which is based on Webot simulator [74].

2.3.1.2 Maximal Coordinate Methods

Maximal coordinate methods refer to the group of techniques to find equation of motion [62]. The methods are also referred as Cartesian methods because these methods use Cartesian coordinate for computation [110]. Maximal coordinates methods analyze each link of robots independently. Each rigid body or link have three translation and three orientation so, in total they have six degrees of freedom. For all robots, there are 6l dof where l is number of link, also, there is c number of constraints that limit the motion of the body. In maximal coordinate methods, the constraints eliminate the inessential degree of freedom. Therefore, there are 6l-c number of equations that represent the joints [58]. This group of method provide advantages, such as, this method is an expansion of rigid body so it is more simple to be learned and implemented [65]. Due to these advantages, these methods become popular for experts who study on computer graphic [62]. In contrast, the disadvantages are that, maximal coordinate methods use the Cartesian coordinates, not joint angles. For this reason, these methods cannot use joint velocities, positions and torques in the equations directly [71]. Moreover, the inexactness of integration and numerical error can result in the drifting and Bender states that drift is a significant problem to cause instability of system [10]. For this reason, maximal coordinate methods are required to post stabilization methods such as Baumgarte stabilization [15]. Studies show that Lagrange Multiplier method is one of the most popular method in maximal coordinate methods. Baraff states that, Lagrange multiplier method defines system as a set of maximal coordinate and it is mentioned that Lagrange multiplier method is simple and handle all arbitrary set of constraints together which cannot be allowed by reduced coordinate methods [7]. Also, Gleicher uses Cartesian coordinate in his paper and handles constraint problem with Lagrange multiplier method, also he mentioned that the reasons for using this technique are that it is simple and fast, also it is rewritten as different quadratic problems [42]. In addition, Surles et al. use the Lagrange multiplier to solve constraint problem [106]. Platt et al. mentioned that Lagrange multiplier transforms the problem into non-constraint problem [90]. Weyler et al. use a stabilized Lagrange multiple method to solve contact constraints which prevent interpenetration between bodies [115]. These studies show that the Lagrange method is used to solve the equation of motion by optimizing the constraints [58].

2.3.1.3 Reduced Coordinate Methods

Kenwright mentioned that, a group of methods to obtain equation of motion for constraint rigid body motion runs in O (n) time. This group of methods is reduced coordinate methods that are not popular as Maximal coordinate [58] due to its complexity [71]. The methods are also called as generalized coordinate methods because they use generalized coordinate [7]. The main advantage of these methods is, formulating motion with combining constraints implicitly. Therefore, joint angles are referred as state of system directly in contrast to Maximal coordinate methods [62]. This makes the reduced coordinate methods to be suitable for more complex bodies such as humanoid structure, quadruped robots and etc. [71] by avoiding conversion between coordinates such as Cartesian space to joint space. In addition, the group of methods solve the equation of motion by fewer DOFs and constraints [71]. It is previously mentioned, that, drift is a significant problem for maximal coordinate methods, however, reduced coordinate methods eliminate this problem and also, Baraff, mentioned that, simulator by using reduced coordinate methods, is faster due to using larger time steps for integration [7]. These are seen as reasons for preferring the reduced coordinate methods rather than maximal coordinate methods. Also, studies show the disadvantages of the system. These methodology is more difficult to implement when compared with the maximal one [71]. Non-holonomic constraints are not included to solve equation of motion and non-linear equations are solved for explicit parameterization in terms of independent coordinates [9]. According to these advantages and drawbacks of reduced coordinate methods, this method is preferred when complex rigid bodies are simulated and to obtain joint accelerations. For many complex rigid body, this group of method used by simulators, such as simulator of Hrp3 humanoid robot, OpenHrp3 [84], open source library Bullet version 2.28 [52]. Many techniques are developed for equation motion that use reduced coordinate system. In 1983, Featherstone offered a technique that is called as articulated body algorithm (ABA) [33]. The algorithm made significant effects on robotic and became a popular technique in reduced coordinate methods. This Featherstone's algorithm is handled thoroughly in Mirtich PhD thesis, "Impulse based dynamic simulation of rigid body systems" [79]. Mirtich states that articulated body algorithm is developed to be stands for O (n^3) methods where the inertia matrix is used to obtain the joint accelerations [79]. This Featherstone algorithm takes the n-joint robot as a one joint robot which has a base link. In this one joint robot, velocities of base member is known and the robot without base member is called an articulated body [34]. Featherstone mentioned that for a robot which includes n joints, link 1's motion is calculated by using base link's motion. To obtain link 2's motion, link 1 behave as base link, in addition to that, link 2 behave as link 1. This is performed until all link's motion are obtained [34]. That algorithm provides simplicity because calculation of one joint robot's acceleration is simpler than n-joint robot, also this algorithm runs in O (n) computational complexity. Moreover, this algorithm uses generalized coordinate therefore, drift problem is avoided. Also, Featherstone offered a new algorithm, named as Divide-and-Conquer Articulated-Body Algorithm (DCA) [36]. This algorithm is developed to solve equations of motion with a parallel computer. It has O(log(n)) computational complexity and it is can be implemented to any system [36]. Featherstone mentioned that this algorithm is the fastest one when has large number processors and compared this algorithm with articulated body algorithm (ABA) and the ABA become more effective than DCA when a computer with low number processors are used, in contrast, DCA becomes more effective when processor number increased [36, 35]. In addition, Yaman et al. offered a new algorithm, named as Assembly-Disassembly algorithm (ADA) to solve dynamic equation and mentioned that the new algorithm runs in O (n) for serial computation and O (log (n)) for parallel computation, therefore, author compares the new algorithm with the fastest algorithms that are ABA for serial computation and DCA for parallel computation in his paper [116]. The comparisons show that ADA comes into prominence for close kinematic chains and parallel computation; in addition to that, ABA has the lowest computational time with sufficient accuracy in open kinematic chain [116].

2.4 Contact Modeling

2.4.1 Contact Detection Algorithms

Above of this chapter, importance of contact modeling is explained and contributions on simulators to obtain more realistic behavior are mentioned. Also, contact detection or collision detection is a significant factor to obtain better contact model and more realistic simulation. For detecting collision and contact, many algorithms were developed such as Lin-Canny algorithm [67] and Gilbert-Johnson-Kerthi algorithm [41]. Previous studies are shown that, these algorithms are categorized by two general groups in terms of broad phase collision detection and narrow phase collision detection [50].

2.4.1.1 Broad Phase Collision Detection

In this group of algorithms; boxes, which contain the points of bodies or objects, are defined and when a box is overlapped with another box, this means that, points which included by boxes, are collided or in contact, therefore, most of the parts of body or objects are eliminated from consideration [76]. The algorithms make predictions whether the boxes will be overlapped or not for the next step in simulation. The advantage of bounding points with virtual boxes is, it makes detecting the collision or contact more simple, in addition to that, broad phase collision has low computational time [61]. These algorithms only control the boxes overlapping, not detecting all points in boxes. That means that the broad phase collision detection algorithms, cannot give the detailed information about detection. The broad phase collision detection algorithms are divided to three types which are exhaustive search, coordinate sorting and multi-level grids. Exhaustive search algorithms care the bounding volumes of boxes and compare them to find collision or contact [61]. These algorithms are also called as all pair test. Another type of algorithm is, coordinate sorting algorithm (also called as sweep and prune). This algorithm is developed by Baraff [5]. Tracy et al state that sweep and prune algorithms get values of maximum and minimum coordinates from each boxes and sort them and then the algorithm checks for intersection. The intersection of boxes means that there is a collision between object and bodies [112]. The third type of algorithm is multi-level grids, which is also called as hierarchical hash tables. In this algorithm, it is mapping the points with cells, therefore, many cells include points of bodies or objects. The algorithm remap for each simulation and if a cell contains points from different bodies or object, the collision is seen between these bodies [61].

2.4.1.2 Narrow Phase Collision Detection

Another group of detection algorithms, are called as narrow phase collision detection. These algorithms give accurate results and more details about detection, in contrast to broad phase algorithms [61]. Mirtich states that broad phase algorithms can be seen as a prerequirement for narrow phase algorithms [76]. Broad phase algorithms eliminate the objects or bodies which are not possible to collide, the narrow phase algorithms inspect remaining objects and give detailed information. The narrow phase collision detections do not use boxes or bounding volume that is used by broad phase algorithms and narrow phase algorithms test objects or bodies directly by complex calculation [2]. Therefore, these algorithms have high computational time. Narrow phase algorithms are separated by four different types of algorithms which are feature based, simplex based, volume based and spatial data structure [61].

Feature-based algorithms detect collision between bodies or objects by using edges, vertices, faces of them [77]. The most rapid feature-based algorithm is the Lin-Canny algorithm [66] which computes the distance between the boundaries of objects. There are two disadvantages of this algorithm. The first disadvantage is that the collision time is not calculated accurately due to interpenetration. The other drawback of the algorithm is instability in some conditions. Another popular feature based algorithm is Coronoid-clip algorithm, also called as V-clip [77]. This algorithm was developed by Mirtich. As mentioned above, the Lin-Canny algorithm is affected by degenerate configuration and this affects robustness of the algorithm. However, V-clip algorithm is not affected from this, therefore, V-clip is a robust algorithm. Also, V-clip algorithm gives good results in penetration case in contrast to Lin-Canny algorithm. Also, Mirtich mentioned that implementation of V-clip algorithm is simpler than Lin-Canny algorithm and this is also an advantage of V-clip algorithm [77].

Simplex based algorithm is another narrow phase method for detecting collision. This method takes convex envelopes of sets of vertices and finds the small distances between the convex envelopes. Therefore, collision is detected by these distance values. The most popular simplex based algorithm is the "Gilbert-Johnson-Keerthi algorithm" (also called as GJK algorithm). This algorithm was proposed by Gilbert, et al. in 1988 [41] and this algorithm searches Minkowski distances between objects to detect collision. The advantage of this algorithm, is that computational time is linearly increased with number of vertices also it calculates and gives penetrations. Also, Bergen proposed a method for robust and implementing GJK algorithm rapidly [11].

Volume based algorithm detects collision by calculating distance between images. Gudelman et al offered a volume based algorithm [43]. In this algorithm, rigid bodies are defined by triangulated surfaces and keeps the value of distance by using signed distance function, therefore, collision of nonconvex rigid bodies are determined. Also, the penetration is seen as an issue in the result of the paper [43], however Gudelman states that round off error is the reason of the penetration.

Spatial data structure algorithm detects collision by using two ways. These are the splitting spaces and bounding volume hierarchy [61]. Bounding volume hierarchy in narrow phase collision detection, has the same idea as in broad phase collision detection. However, in narrow phase, overlapping bounding box does not mean the detection of collision. In this technique, non-overlapping bounding is removed from calculation and the collision is detected by using small boxes iteratively. By splitting space technique, space where objects are located, are divided into small and equal region iteratively. As a result, when these sufficiently minimized regions include the different objects point, the algorithm estimates that these objects are collided. Jin developed a new splitting space algorithm [53] also Bandi et al, proposed an adaptive spatial subdivision [3]. Bandi states that the bounding box algorithm is effective for simple algorithms. However when objects are complex, boxes cover empty space. This may result in wrong results. Also, the solution of these problems increase the computational of time.

2.4.2 Methods of Modeling Contact Model

In the real world environment, object is affected by different disturbances and that results in difficulties to mimic the real conditions by simulation. For solving this problem, many studies are done and significant field areas are raised. One of the significant areas is, contact modeling, which plays significant roles for simulated objects behavior correctly. Contact is seen while walking, running, jumping, rolling, keeping object, touching and etc. Therefore, when correct model of the contact is obtained, these mentioned activities can be simulated properly.

Recent studies on legged robotics area tends to develop highly dynamic robots (mentioned above), for this reason many control algorithms and studies are developed. However, before implementing these algorithms to robots, the algorithms must be tested in simulation environment because inefficient algorithms can cause damages to the robot. Simulation of the highly dynamic robots, that has capabilities as jumping, walking, running and etc., requires correct contact model to test the algorithm. For this reason, humanoid platforms have their own simulators such as ASIMO, HOAP, QRIO, HRP2 and SURALP. Moreover, contact algorithms also have significant role on movies, games and animations. In movies and games, characters are interacted with others, in addition to that, these characters perform highly dynamic movements. For example, kicking a ball, tackling to a rival, jumping and similar actions are performed by characters and by correct contact model, these behaviors and also animations are shown as realistic.

Modeling contact contributes to development of many study fields, therefore, many researchers develop significant methods to model contact. However, two of them come into prominence which are the penalty based contact model and the exact contact model. Penalty based model is a simple algorithm to be understood and implemented. Also, computational complexity is less than the exact contact modeling however the method is not as accurate as exact modeling. For this reason, this algorithm is mostly used in computer graphic which requires fast computation. By penalty based model, contact is modeled as spring and damper. Stiffness of spring and damper determines the accuracy of the contact model. For this reason, the stiffness value is depended on the application. Penetration is a significant criteria to evaluate contact model accuracy. For correct contact model, penetration must be disappeared. In penalty based method, penetration is prevented by high stiffness of virtual spring and damper, however it results in problem. Generally negligible penetration can be occurred between two simulation times, however in this situation, non-realistic contact forces, also movement can be occurred due to high stiffness of spring and damper. In contrast, exact contact method is developed to obtain correct contact forces which results in real movement. This method is difficult to implement and to be understood, also, it has higher computational complexity than penalty based method. However, this algorithm is used in study areas which require correct contact model such as robotics. Generally, constraint based methods, analytical methods and impulse based methods can be referred as exact contact model.

Dumwright offered new penalty method to solve the mentioned problem of penalty based contact model [16]. He stated that, there are two significant models which are the penalty method and the analytical method. The analytical method may be unsuccessful when friction is also modeled. By penalty method, this problem can be handled, however, the drawbacks of these methods are penetrations and oscillations. His new method solved these problems with using multiple points and integral terms to obtain less oscillation and interpenetration than general usage of penalty based model.

David Baraff developed a new method to solve contact forces analytically when rigid bodies are in resting [4]. He stated that, interpenetration cannot be seen in realistic simulation, however, when law of Newtonian dynamics is held, interpenetration cannot be avoided in simulation. For this reason, exact reaction force must be calculated to solve this problem. Classic algorithms cannot be used for calculating these forces because those algorithms assume that the system is at equilibrium. However, that is not the case in simulation. Baraff offered an analytic method for preventing interpenetration and that method holds holonomic constraints.

Mirtich offered a new algorithm to modeling contact force [78]. He mentioned that this algorithm calculates the contact forces when objects are rolling, colliding, resting and sliding. His algorithm uses impulse forces sequentially to obtain realistic behaviour however the results show that impulse based contact model is not as accurate as the constraint based model. The virtue of the impulse based modeling has low computational time and implementation of this algorithm is simpler than a constraint one.

In 1994, Baraff developed a new algorithm with analytical contact modeling [6] and he mentioned that implementation of this method can be done easily and rapidly even by an inexperienced person in numerical programming. This algorithm is based on bilateral constraints and unilateral constraints. By using bilateral constraints, linear equation of system is solved and interpenetration in simulation is prevented by using unilateral constraints. Baraff states that simulator has lower computational time when computing contact forces cannot be handled as an optimization problem, with this way, it is not required to use optimization software packages. In his algorithm, contact forces is formulated as linear complementarity problem and quadratic program. As a result, he claimed that, fast, simple and reliable solution of the contact modeling is provided by mentioned constrained based algorithm.

In 2012, Drumwright et al. published his studies on linear complementary problem (LCP) [18]. Drumwright et al. mentioned that LCP has significant role on robot dynamics, optimization and simulation. In simulation, contact problem can be modeled as LCP as Baraff modeled [6]. This problem is a handicap for efficient simulation in robotics and this issue has tried to solve by non-linear optimization solvers. In theory, these solvers are defined as efficient solvers, however, in practical, they face with failures for some cases. Therefore, when interpenetration is occurred, the simulation may lose its stability and also rigid bodies slide on contact. In this paper, solvers of LCP are categorized by four group in terms of pivoting solvers, interior point solvers, PATH and iterative solvers and they are evaluated for their performances according to solubility, running time, and normal constraint violation. The experiments show that, Lemke solver is used as the pivoting solver and it has high performance according to solubility. However, the interior point solver has the worst performance with respect to solubility and running time. In spite of all these, interior point solver is not affected, when parameters are changed. Also, by PATH, same results are obtained with using different parameters, in contrast, Lemke has worse performance than PATH for the mentioned evaluation. In addition to that, Lemke has better performance than PATH when runnning time is evaluated. This study can guide the researcher to choose LCP solver.

Nakaoka et al. developed new constrained based contact model and he claimed that penalty method is not suitable for simulator which handle the advance robotic tasks [84]. Nakaoka et al. offered the constrained based method for the simulator because when forces are solved according to satisfying the contact constraints, simulation results become more accurate. Mostly, constrained based method is formulated by LCP (linear complementarity problem). However, Nakaoka et al. claimed that this method is not suitable for the simulator because in the formulation, inverse of matrix must be calculated and this results in computational complexity O (n^3) for n dof robots. Another disadvantage of the LCP formulation is that LCP is solved by using pivoting algorithm and by using the algorithm with complex constraints, obtaining robust results is not easy. Therefore, Nakaoka et al. states that this problem can be solved by using iterative algorithm. However, numerical complexity with iterative algorithm becomes O (c^2) where c is the contact point which means that simulator is slow. Moreover, this paper states that for accurate simulation, the elastic parts of robots should be modeled because this part of the robots have high effects on stability of the robot. In this paper, these elastic parts are modelled by a spring-damper combination, therefore, shock can be absorbed.

Chapter 3

Free-Fall Legged Robot Dynamics

3.1 Introduction

This chapter includes information about free-fall legged robot dynamics. In dynamics simulations, the Newton Euler algorithm and Featherstone's articulated body method are commonly used. By using these algorithms, accurate simulations can be run with low computational time. In this thesis, Featherstone's algorithm is preffered because, in contrast to the Newton Euler algorithm, joint accelerations are obtained without computing the inverse of the robot inertia matrix. The inverse operation is computationally heavy due to the size of the inertia matrix. This chapter contains two methods to compute free root dynamics of legged robots; the method proposed by Kokkevis [62] and the floating base method suggested by Mirtich [79].

3.2 The Articulated Body Method

The articulated body method, abbreviated as ABM or ABA. This algorithm is a developed version to supersede O(n^3) dynamic algorithms by Featherstone [34] in 1984. This algorithm is a developed version of the Newton Euler dynamics and it has computational complexity of O(n) for a n-link system. In this algorithm, the inverse matrix is not calculated. This is a significant advantage for contact modeling algorithms because after obtaining contact forces, the resulting contact accelerations are found by using dynamics methods (for example Newton-Euler or ABM). By not calculating the inverse matrix for each contact, increasing computational time of simulation is avoided. However, the algorithm is compex and difficult to implement. Featherstone's algorithm was explained explicitly in the Ph.D. thesis of Mirtich [79].

In the articulated body method, the system can be reduced to a link in order to solve its dynamics as shown in Fig. 3.1. In this thesis, n refers to total number of links, i refers to the link number underconsideration. The links are numbered 1 to n.

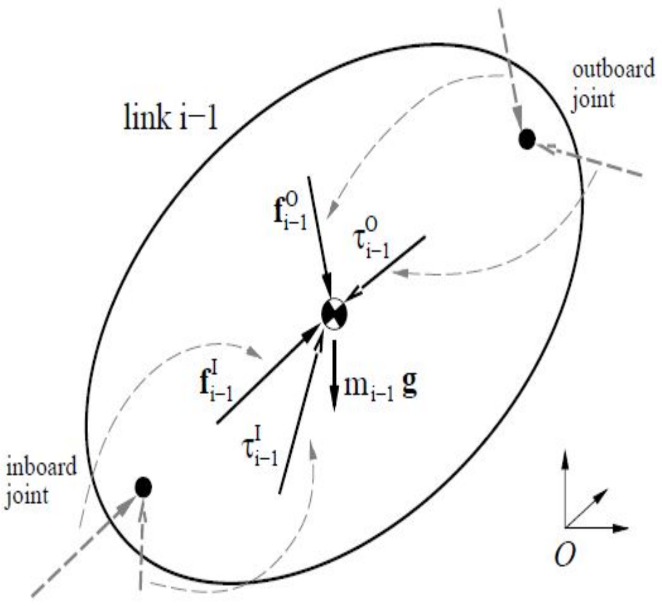


FIGURE 3.1: Free body diagrom of a link[79]

In the free body diagram in Fig. 3.1, torque and force on the link's center of gravity (called as CoG) are generated by gravity (g) and forces applied from joints. Forces and torques are labeled either as inboard (f^I, t^I) or outboard (f^O, t^O). An inboard force is defined as an applied force on the CoG of the link from the previous link while an outboard force is defined as an applied force on the CoG of the link from the next link. In this chapter, derivation of the algorithm for serial linkages and tree-like linkages will be explained seperately.

3.2.1 Articulated Body Method (ABM) for Serial Linkages

Serial link robots are mostly used for industrial applications to handle hazardous works and achieving fast production. These robots are categorized as articulated (RRR), spherical (RRP), cartesian(PPP), SCARA(RRP) and cylindrical (RPP).

FIGURE 3.2: Six axis articulated robot [96]

Forward dynamic of articulated body method is constituted by three recursion steps [14]:

- Computation of linear and angular velocity for each link like in the Newton Euler method.
- Computation of articulated inertias and spatial articulated zero acceleration force (also called bias force)
- Computation of articulated acceleration.

Spatial vector is a mathematical object for defining the three-dimensional system. It consists of two three-dimensional vectors which represent linear and angular components of a system [34]. In this thesis, spatial algebra is used to define a system to avoid complex equations.

3.2.1.1 Computation of Linear and Angular Velocity

In the articulated body method, velocity is calculated in the same way as in the Newton Euler formulation. Motion of ith link is calculated by using the joint velocity of the link and the motion of the previous link. In this thesis, linear and angular velocity of link i are labeled as ‘v_i’ and ‘w_i’ and these are defined in their own frame. In addition to that, ‘q’ stands for the generalized coordinates of system.

For primatic joints,

$$\mathrm{v}_i = R_i.\mathrm{v}_{i-1} + R_i.\mathrm{w}_{i-1} \times \mathrm{r}_i + \dot{q}_i.\mathrm{u}_i \tag{3.1}$$

$$\mathrm{w}_i = R_i.\mathrm{w}_{i-1} \tag{3.2}$$

For revolute joints,

$$\mathrm{v}_i = R_i.\mathrm{v}_{i-1} + R_i.\mathrm{w}_{i-1} \times \mathrm{r}_i + \dot{q}_i.\mathrm{u}_i \times \mathrm{d}_i \tag{3.3}$$

$$\mathrm{w}_i = R_i.\mathrm{w}_{i-1} + \dot{q}.\mathrm{u} \tag{3.4}$$

In 3.1, 3.2, 3.3, 3.4, we have:

R_i: a rotation matrix that rotates vectors on i-1th frame to ith frame
d_i: a vector that is defined from the ith joint to the center of mass of the ith link.
u_i: a unit vector that is defined as a joint axis.
r_i: a vector that is defined from the center of gravity of i-1th link to the center of gravity of ith as express on the ith frame.

In the first step of ABM, the motion of each link is calculated from the base to the tip of robot by using 3.1 to 3.4. For fixed-based robot, velocity of the base is zero. However, if the robot has a moving base, its velocity is nonzero generally.

The obtained velocities can be written in the form of a spatial vector as

$$\hat{\mathrm{v}}_i = \begin{bmatrix} \mathrm{w}_i \\ \mathrm{v}_i \end{bmatrix}_{6\times1} \tag{3.5}$$

Spatial vectors are transformed to another frame by using the spatial transformation matrix, denoted by X. This matrix is employed as:

$$\begin{bmatrix} \mathrm{w}_{i+1} \\ \mathrm{v}_{i+1} \end{bmatrix}_{6\times1} = \mathrm{X}_{6\times6}. \begin{bmatrix} \mathrm{w}_i \\ \mathrm{v}_i \end{bmatrix}_{6\times1} \tag{3.6}$$

The velocity of the ith link is expressed on the i+1th link by the use of rotation matrix,

$$\begin{bmatrix} \mathrm{w}_i^{i+1} \\ \mathrm{v}_i^{i+1} \end{bmatrix}_{6\times1} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix}_{6\times1} . \begin{bmatrix} \mathrm{w}_i^{i} \\ \mathrm{v}_i^{i} \end{bmatrix}_{6\times1} \tag{3.7}$$

v_i^i and w_i^i are notations that refer to the velocities of the ith link on the ith frame. v_i^{i+1} and w_i^{i+1} notations refer to the velocities of ith link on i+1th frame. The translation between ith link and i+1th link is carried out by 3.8:

$$\begin{bmatrix} \mathrm{w}_{i+1}^{i+1} \\ \mathrm{v}_{i+1}^{i+1} \end{bmatrix}_{6\times1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\tilde{r}_{i+1} & \mathbf{1} \end{bmatrix}_{6\times1} . \begin{bmatrix} \mathrm{w}_i^{i+1} \\ \mathrm{v}_i^{i+1} \end{bmatrix}_{6\times1} \tag{3.8}$$

By combining 3.7 and 3.8, the spatial transformation matrix is obtained:

$$\mathrm{X}_{i+1} = \begin{bmatrix} \mathbf{1} & 0 \\ -\tilde{r}_{i+1} & \mathbf{1} \end{bmatrix}_{6\times1} . \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix}_{6\times1} \tag{3.9}$$

This matrix transforms a spatial vector from frame i+1 to frame i. The matrix in 3.9 will be used in the articulated body method. The matrix for transforming a spatial vector from frame i to frame i+1 is

$$\mathrm{X}'_{i+1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ R^T.\tilde{r}_{i+1} & \mathbf{1} \end{bmatrix}_{6\times 1} . \begin{bmatrix} R^T & \mathbf{0} \\ \mathbf{0} & R^T \end{bmatrix}_{6\times 1} \tag{3.10}$$

3.2.1.2 Computation of Articulated Inertias and the Articulated Zero Acceleration Force

The term articulated body is defined by Mirtich: when a system which is composed of n links is disconnected from the ith link, the new body which is formed by the ith link to the nth link, is called an articulated body [79].The articulated inertia is defined as the inertia of the articulated body. In addition to that, the articulated zero acceleration force (also called the articulated bias force) is defined as a force that prompt the system to have zero acceleration.

The system shown in Fig. 3.1 can be modeled as:

$$\mathrm{f}^I{}_{i-1} + \mathrm{f}^O{}_{i-1} = \mathrm{M}.\mathrm{a}_{i-1} - \mathrm{m}.g \tag{3.11}$$

$$\mathrm{t}^I{}_{i-1} + \mathrm{t}^O{}_{i-1} = \mathrm{I}_{i-1}.\alpha_{i-1} + \mathrm{w}_{i-1} \times \mathrm{I}_{i-1}.(w)_{i-1} \tag{3.12}$$

In 3.11 and 3.12, m refers to mass of link and M is $\begin{bmatrix} m & \mathbf{0} \\ \mathbf{0} & m \end{bmatrix}$. g refers to gravity and I refers to inertia of the link. Also, a and α refer to linear acceleration and angular acceleration respectively. The equation 3.11 and 3.12 can be combined into the equation 3.13. In the equation 3.13, $\hat{\mathrm{f}}^I_{i-1}$ and $\hat{\mathrm{f}}^O_{i-1}$ are spatial forces which contain $\mathrm{f}^I{}_{i-1}$,$\mathrm{t}^I{}_{i-1}$ and $\mathrm{f}^O{}_{i-1}$,$\mathrm{t}^O{}_{i-1}$ respectively.

$$\hat{\mathrm{f}}^I_{i-1} + \hat{\mathrm{f}}^O_{i-1} = \begin{bmatrix} \mathbf{0} & M_{i-1} \\ \mathrm{I}_{i-1} & \mathbf{0} \end{bmatrix} . \begin{bmatrix} \alpha_{i-1} \\ \mathrm{a}_{i-1} \end{bmatrix} + \begin{bmatrix} -\mathrm{m}.g \\ \mathrm{w}_{i-1} \times \mathrm{I}_{i-1}.\mathrm{w}_{i-1} \end{bmatrix} \tag{3.13}$$

Spatial acceleration and spatial inertia refer to $\begin{bmatrix}\alpha_{i-1}\\ \mathrm{a}_{i-1}\end{bmatrix}$ and $\begin{bmatrix}\mathbf{0} & M_{i-1}\\ \mathrm{I}_{i-1} & \mathbf{0}\end{bmatrix}$ respectively. Also, spatial bias force refers to $\begin{bmatrix}-\mathrm{m}.g\\ \mathrm{w}_{i-1}\times\mathrm{I}_{i-1}.\mathrm{w}_{i-1}\end{bmatrix}$. f^{O}_{i-1} is mentioned as outboard force for i-1th link and same force is also defined as negative inboard force for ith link.

$$\hat{\mathrm{f}}^{O}_{i-1} = -\mathrm{X}'_i.\hat{\mathrm{f}}^{I}_i \tag{3.14}$$

Therefore, 3.14 is combined into 3.13. That equation is obtained as:

$$\hat{\mathrm{f}}^{I}_{i-1} - \mathrm{X}'_i.\hat{\mathrm{f}}^{I}_i = \begin{bmatrix}\mathbf{0} & M_{i-1}\\ \mathrm{I}_{i-1} & \mathbf{0}\end{bmatrix}.\begin{bmatrix}\alpha_{i-1}\\ \mathrm{a}_{i-1}\end{bmatrix} + \begin{bmatrix}-\mathrm{m}.g\\ \mathrm{w}_{i-1}\times\mathrm{I}_{i-1}.(w)_{i-1}\end{bmatrix} \tag{3.15}$$

Articulated body, articulated inertia and articulated bias force are defined above in Subsection 3.2.1.2. According to these definitions, inboard force applied on articulated body can be modeled as shown in 3.16 if there is no disturbances or external forces.

$$\hat{\mathrm{f}}^{I}_i = \hat{\mathrm{I}}^{A}_i.\hat{\mathrm{a}}_i + \hat{\mathrm{Z}}^{A}_i \tag{3.16}$$

In 3.16, $\hat{\mathrm{Z}}^{A}_i$ and $\hat{\mathrm{I}}^{A}_i$ refer to spatial articulated bias force and spatial articulated inertia respectively. Also, $\hat{\mathrm{a}}_i$ implies the spatial link′s acceleration. Moreover, spatial acceleration of ith link can be derived from previous link's acceleration. The equation is written as:

For primatic joint,

$$\mathrm{a}_i = \mathrm{a}_{i-1} + \alpha_{i-1}\times\mathrm{r}_i + \ddot{q}_i.\mathrm{u}_i + \mathrm{w}_{i-1}\times(\mathrm{w}_{i-1}\times\mathrm{r}_i) + 2.\mathrm{w}_{i-1}\times\dot{q}.\mathrm{u}_i \tag{3.17}$$

$$\alpha_i = \alpha_{i-1} \tag{3.18}$$

For revolute joint,

$$\mathrm{a}_i = \mathrm{a}_{i-1}+\alpha_{i-1}\times\mathrm{r}_i+\ddot{q}_i.\mathrm{u}_i\times\mathrm{d}_i+\mathrm{w}_{i-1}\times(\mathrm{w}_{i-1}\times\mathrm{r}_i)+2.\mathrm{w}_{i-1}\times(\dot{q}_i.\mathrm{u}_i\times\mathrm{d}_i)+\dot{q}_i.\mathrm{u}_i\times(\dot{q}_i.\mathrm{u}_i\times\mathrm{d}_i) \tag{3.19}$$

$$\alpha_i = \alpha_{i-1} + +\ddot{q}_i.\mathrm{u}_i + \mathrm{w}_{i-1} \times \dot{q}_i.\mathrm{u}_i \tag{3.20}$$

In 3.17, 3.19, 3.18 and 3.20, all variables are defined in ith frame and the mentioned equation can be written in spatial form as:

$$\hat{a}_i = \hat{a}_{i-1} + \ddot{\mathrm{q}}_i.\hat{\mathrm{s}}_i + \hat{\mathrm{c}}_i \tag{3.21}$$

For prismatic joint:

$$\hat{\mathrm{c}}_i = \begin{bmatrix} \mathbf{0} \\ \mathrm{w}_{i-1} \times (\mathrm{w}_{i-1} \times \mathrm{r}_i) + 2.\mathrm{w}_{i-1} \times \dot{q}_i.\mathrm{u}_i \end{bmatrix} \tag{3.22}$$

$$\hat{\mathrm{s}}_i = \begin{bmatrix} \mathbf{0} \\ \mathrm{u}_i \end{bmatrix} \tag{3.23}$$

For revolute joint:

$$\hat{\mathrm{c}}_i = \begin{bmatrix} \mathrm{w}_{i-1} \times \dot{q}_i.\mathrm{u}_i \\ \mathrm{w}_{i-1} \times (\mathrm{w}_{i-1} \times \mathrm{r}_i) + 2.\mathrm{w}_{i-1} \times (\dot{q}_i.\mathrm{u}_i \times \mathrm{d}_i) + \dot{q}_i.\mathrm{u}_i \times (\dot{q}_i.\mathrm{u}_i \times \mathrm{d}_i) \end{bmatrix} \tag{3.24}$$

$$\hat{\mathrm{s}}_i = \begin{bmatrix} \mathbf{0} \\ \mathrm{u}_i \times \mathrm{d}_i \end{bmatrix} \tag{3.25}$$

3.16 and 3.15 are combined into 3.26 and in 3.26, $\hat{\mathrm{a}}_i$ is defined as $\hat{\mathrm{a}}_{i-1}$ by using equation 3.21. Therefore, all variables depend on previous one. If the spatial vectors are written in their own frames, vectors should be tranformed by mentioned spatial transformation matrix as shown in 3.27. It can be modified as 3.28.

$$\hat{\mathrm{f}}^I_{i-1} = \hat{\mathrm{I}}_{i-1}.\hat{\mathrm{a}}_{i-1} + \hat{\mathrm{Z}}_{i-1} + \mathrm{X}'_i.(\hat{\mathrm{I}}^A_i.\hat{\mathrm{a}}_i + \hat{\mathrm{Z}}^A_i) \tag{3.26}$$

$$\hat{\mathrm{f}}^I_{i-1} = \hat{\mathrm{I}}_{i-1}.\hat{\mathrm{a}}_{i-1} + \hat{\mathrm{Z}}_{i-1} + \mathrm{X}'_i.(\hat{\mathrm{I}}^A_i.(\mathrm{X}_i.\hat{a}_{i-1} + \ddot{\mathrm{q}}_i.\hat{\mathrm{s}}_i + \hat{\mathrm{c}}_i) + \hat{\mathrm{Z}}^A_i) \tag{3.27}$$

$$\hat{\mathrm{f}}^I_{i-1} = (\hat{\mathrm{I}}_{i-1} + \mathrm{X}'_i.\hat{\mathrm{I}}^A_i.\mathrm{X}_i).\hat{\mathrm{a}}_{i-1} + \hat{\mathrm{Z}}_{i-1} + \mathrm{X}'_i.(\hat{\mathrm{Z}}^A_i + \hat{\mathrm{I}}^A_i.\hat{\mathrm{c}}_i + (\hat{\mathrm{I}}^A_i.\hat{s}_i).\ddot{q}_i) \tag{3.28}$$

3.28 has same meaning as 3.16. Therefore, Z^A_{i-1} and I^A_{i-1} can be determined from the 3.28.

$$\hat{\mathrm{I}}^A_{i-1} = \hat{\mathrm{I}}_{i-1} + \mathrm{X}'_i.\hat{\mathrm{I}}^A_i.\mathrm{X} \tag{3.29}$$

$$\hat{\mathrm{Z}}^A_{i-1} = \hat{\mathrm{Z}}_{i-1} + \mathrm{X}'_i.(\hat{\mathrm{Z}}^A_i + \hat{\mathrm{I}}^A_i.\hat{\mathrm{c}}_i + (\hat{\mathrm{I}}^A_i.\hat{s}_i).\ddot{q}_i) \tag{3.30}$$

The variables are known except $\ddot{q}_i$ in 3.29 and 3.30. 3.28 multiplied by $\hat{\mathrm{s}}'_i$ due to obtaining $\hat{\mathrm{s}}'_i\hat{\mathrm{f}}^I_{i-1}$ that gives the exerted force for prismatic joint and exerted torque for revolute joint. Also, it is labeled as Q_i.

For prismatic joint,

$$\hat{\mathrm{s}}'_i = \begin{bmatrix} \mathrm{u}^T_i & \mathbf{0} \end{bmatrix} \tag{3.31}$$

For revolute joint,

$$\hat{\mathrm{s}}'_i = \begin{bmatrix} (\mathrm{u}^T_i \times \mathrm{d})^T & \mathrm{u}^T_i \end{bmatrix} \tag{3.32}$$

Exerted force,

$$\mathrm{Q}_i = \hat{\mathrm{s}}'_i.\hat{\mathrm{f}}^I_i \tag{3.33}$$

After multiplying 3.28 by $\hat{\mathrm{s}}'_i$, $\ddot{q}_i$ is obtained as:

$$\ddot{q}_i = \frac{\mathrm{Q}_i - \hat{\mathrm{s}}'_i.\hat{\mathrm{I}}^{\mathrm{A}}_i.\mathrm{X}_i.\hat{\mathrm{a}}_{i-1} - \hat{\mathrm{s}}'_i(\hat{\mathrm{Z}}^{\mathrm{A}}_i + \hat{\mathrm{I}}^{\mathrm{A}}_i.\hat{\mathrm{c}}_i)}{\hat{\mathrm{s}}'_i.\hat{\mathrm{I}}^{\mathrm{A}}_i.\hat{\mathrm{s}}_i} \tag{3.34}$$

3.34 is combined into 3.29 and 3.30. Spatial articulated inertia matrix $\hat{\mathrm{I}}^A_{i-1}$ and spatial articulated bias force $\hat{\mathrm{Z}}^A_{i-1}$ are found as:

$$\hat{\mathrm{I}}^A_{i-1} = \hat{\mathrm{I}}_{i-1} + \mathrm{X}'_i.(\hat{\mathrm{I}}^A_i - \frac{\hat{\mathrm{I}}^A_i.\hat{\mathrm{s}}_i.\hat{\mathrm{s}}'_i.\hat{\mathrm{I}}^A_i}{\hat{\mathrm{s}}'_i.\hat{\mathrm{I}}^{\mathrm{A}}_i.\hat{\mathrm{s}}_i}).\mathrm{X}_i \tag{3.35}$$

$$\hat{Z}^A_{i-1} = \hat{Z}_{i-1} + X'_i . \left[\hat{Z}^A_i + \hat{I}^A_i . \hat{c}_i + \frac{\hat{I}^A_i . \hat{s}_i . \left[Q_i - \hat{s}'_i . (\hat{Z}^A_i + \hat{I}^A_i . \hat{c}_i) \right]}{\hat{s}'_i . \hat{I}^A_i} \right] \tag{3.36}$$

3.2.2 Articulated Body Method (ABM) for Tree Like Linkage

Tree topologies are used in kinematic of biped, quadruped and so on. In tree topologies, there is a base link (also called as mother link) which connected with one or more than one link (also called as child link). In Fig. 3.3, A link is a mother link and its children are B and C links, also B and C links are mother and their children are D and E links respectively.

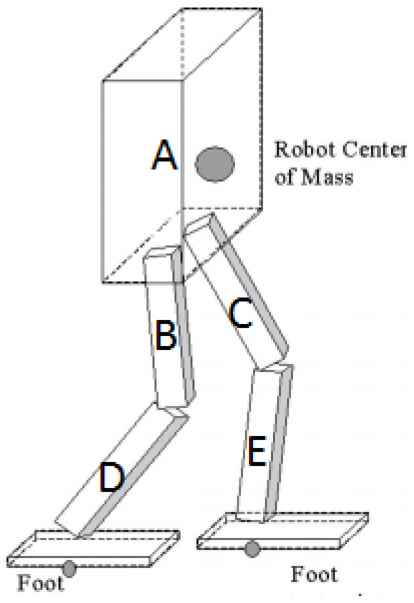


Figure 3.3: Typical kinematic arrangement of a biped robot [28]

Articulated body method for tree like linkage has same logic as ABM for serial linkage. In serial one, most calculation for ith link depend on dynamic parameter of i-1th link, however, in tree like linkage, ith link is called as mother and its calculation uses its children's dynamic parameters vice versa.

Computation of linear and angular velocity of tree like linkage:

For primatic joint,

$$\mathrm{v}_i = R_i.\mathrm{v}_m + R_i.\mathrm{w}_m \times \mathrm{r}_i + \dot{q}_i.\mathrm{u}_i \tag{3.37}$$

$$\mathrm{w}_i = R_i.\mathrm{w}_m \tag{3.38}$$

For revolute joint,

$$\mathrm{v}_i = R_i.\mathrm{v}_m + R_i.\mathrm{w}_m \times \mathrm{r}_i + \dot{q}_i.\mathrm{u}_i \times \mathrm{d}_i \tag{3.39}$$

$$\mathrm{w}_i = R_i.\mathrm{w}_m + \dot{q}.\mathrm{u} \tag{3.40}$$

where 'm' is used as index that reflects to mother of ith link.

Computation of spatial articulated inertia matrix of tree like linkage:

$$\hat{\mathrm{I}}_m^A = \hat{\mathrm{I}}_m + \sum \left[\mathrm{X}_i'.(\hat{\mathrm{I}}_i^A - \frac{\hat{\mathrm{I}}_i^A.\hat{\mathrm{s}}_i.\hat{\mathrm{s}}_i'.\hat{\mathrm{I}}_i^A}{\hat{\mathrm{s}}_i'.\hat{\mathrm{I}}_i^{\mathrm{A}}.\hat{\mathrm{s}}_i}).\mathrm{X}_i\right] \tag{3.41}$$

Computation of spatial articulated bias force of tree like linkage:

$$\hat{\mathrm{Z}}_m^A = \hat{\mathrm{Z}}_m + \mathrm{X}_i'.\left[\hat{\mathrm{Z}}_i^A + \sum \left[\hat{\mathrm{I}}_i^A.\hat{\mathrm{c}}_i + \frac{\hat{\mathrm{I}}_i^A.\hat{\mathrm{s}}_i.\left[\mathrm{Q}_i - \hat{\mathrm{s}}_i'.(\hat{\mathrm{Z}}_i^A + \hat{\mathrm{I}}_i^A.\hat{\mathrm{c}}_i)\right]}{\hat{\mathrm{s}}_i'.\hat{\mathrm{I}}_i^{\mathrm{A}}}\right]\right] \tag{3.42}$$

In 3.41 and 3.42, $\sum$ is used to calculate total effect of children links on mother link. In addition, ith link refers to children links of mother link.

Computation of acceleration of tree like linkage:

$$\ddot{q}_i = \frac{\mathrm{Q}_i - \hat{\mathrm{s}}_i'.\hat{\mathrm{I}}_i^{\mathrm{A}}.\mathrm{X}_i.\hat{\mathrm{a}}_m - \hat{\mathrm{s}}_i'(\hat{\mathrm{Z}}_i^{\mathrm{A}} + \hat{\mathrm{I}}_i^{\mathrm{A}}.\hat{\mathrm{c}}_i)}{\hat{\mathrm{s}}_i'.\hat{\mathrm{I}}_i^{\mathrm{A}}.\hat{\mathrm{s}}_i} \tag{3.43}$$

$$\hat{a}_i = \mathrm{X}_i.\hat{a}_m + \ddot{\mathrm{q}}_i.\hat{\mathrm{s}}_i + \hat{\mathrm{c}}_i \tag{3.44}$$

3.3 Modeling Free Root Dynamic of Legged Robot

Modeling free root dynamic has significant role on dynamic of free fall legged robot. To simulate free movement of base correctly, testing the control algorithm in simulation environment gives true hints to researchers. Because of this, the section is involved in this thesis. Two methods will be mentioned which are mimicing free base link with 6 dof link, offered by Koskevis [62] and floating base model, offered by Mirtich [79].

Mathematically, free link is modeled by six parameters in terms of orientation parameters (α, β, γ) and translational parameters (X,Y,Z). The translational parameters are mimiced by prismatic joints as shown in Fig. 3.4(a) which first link is connected to fixed base. In addition, orientation paremeters are mimiced by rotational joints in Fig. 3.4(b) which are connected to prismatic link serially.

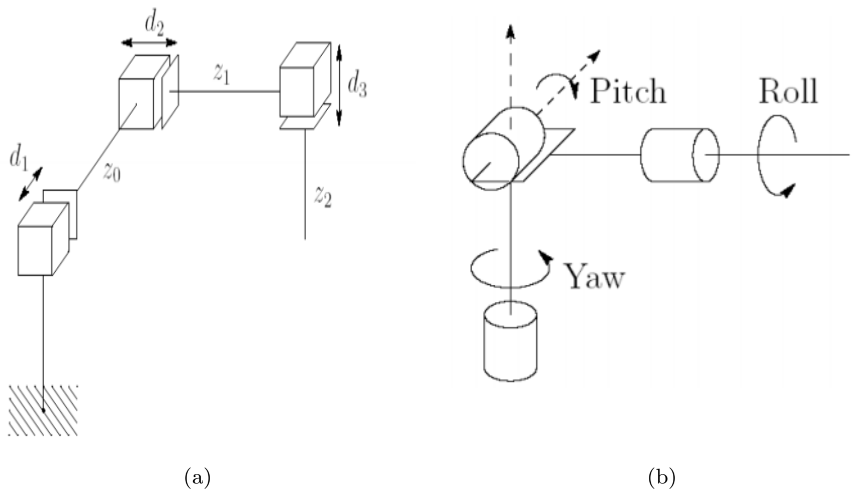


FIGURE 3.4: (a) PPP kinematic configuration [85], (b) RRR kinematic configuration [85]

This method is simple to understand and implement however, it increases the computational complexity due to addition of six links. Accumulation of calculation error of these links increases risk to trigger unrealistic movement in simulation environment.

Another method is offered by Mirtich [79] and this method use ABM to model free link directly. In ABM, system is modeled as 3.16 however, there is no inboard force that affects to free link. Therefore, 3.16 can be modified as shown in 3.45.

$$\mathbf{0} = \hat{\mathrm{I}}_i^A . \hat{\mathrm{a}}_i + \hat{\mathrm{Z}}_i^A \tag{3.45}$$

From this equation, free link acceleration which contains rotational and translational dynamics, is found easily. By using 3.46, inverse of matrix 6×6 is computed in each iteration and, effect of this on computational time is negligible. Therefore, this method is preferred in the simulations.

$$\hat{\mathrm{a}}_i = -(\hat{\mathrm{I}}_i^A)^{-1} . \hat{\mathrm{Z}}_i^A \tag{3.46}$$

where $(\hat{\mathrm{I}}_i^A)^{-1}$ means the inverse of $\hat{\mathrm{I}}_i^A$.

Chapter 4

Application of Exact Contact Modeling for Legged Robot Dynamics

4.1 Introduction

Recent studies on legged robot focus on complex activities such as carrying, walking, running, climbing and jumping which require permanent or temporary contact with the environment. For simulating these activities, contact modeling plays a significant role. Among the contact models, penalty based method and exact contact method come into prominence. With penalty based contact model, highly dynamic motions such as jumping and running are simulated fast but not correctly. In this chapter, an exact contact method, developed by Chardonnet et al.[14], will be explained in detail and difficulties of implementing the contact model will be clarified.

4.2 Derivation of Constraints

The contact model calculates exact contact forces, but there is no option to solve the forces directly and certain contact constraints should be determined. Ruspini and Khatib defined the constraints for collision and contact to solve the forces without contact friction [97]. There are seperate constraints for collision and contact which are explained in Subsections 4.2.1 and 4.2.2 respectively. Mathematically, motion of the system is defined as shown in 4.1 where the generalized coordinate is defined by q. In addition, acceleration of generalized coordinate is calculated in 4.2.

$$\mathrm{T} = \mathrm{A}(q).\ddot{q} + C(q,\dot{q}).\dot{q} + g(q) \tag{4.1}$$

$$\ddot{q} = \mathrm{A}(q)^{-1}.(\mathrm{T} - C(q,\dot{q}).\dot{q} - g(q)) \tag{4.2}$$

where $\mathrm{A}(q)$ is inertia matrix, $C(q,\dot{q})$ is coriolis and centrifugal term and $g(q)$ is gravitational term.

In this thesis, motion of system is calculated by using articulated body method (ABM) as explained in Section 3.2. In Subsections 4.2.1 and 4.2.2, constraints are derived by using 4.1.

4.2.1 Derivation of Collision Constraint

Collision occurs between two contact points when their relative distance is decreased to zero from a positive value. Collision process is assumed instantenous therefore friction can be neglected. The emprical law for frictionless collision mentions that magnitude of relative velocity of colliding objects will decrease or stays constant after collision [4] based on the objects elasticity as shown in 4.3. The relative velocity term expresses the velocity of an object with respect to another object in collision. In this thesis, relative velocity is denoted by v_k and k refers number of contact. ϵ refers to elesticity constant, t + dt and t show the time after and before collision respectively.

$$v_k(t + dt) \geq -\epsilon.v_k(t) \tag{4.3}$$

In 4.3, ϵ is 0 when collision is an inelastic, ϵ is 1 when collision is a fully elastic.

For the 4.3 to be satisfied, there needs to be a force applied to the system. This is an impulse force since the collision process is instantenous. The impulse force only has a value on normal direction it can only be used to push object apart from each other, not other way around. This is shown by a constraint:

$$p \geq 0 \tag{4.4}$$

where p refers to impulse force.

Based on these constraints, collision is simulated correctly. For the analytical calculation, the relative velocity is required to be derived by using generalized coordinate as equation of motion. This is done by a jacobian matrix which transforms joint space into contact space.

$$v_k = \mathrm{J}_c.\dot{q} \tag{4.5}$$

Impulse is defined as the change of the system's momentum therefore, the difference of momentum between $t + dt$ and t gives the impulse force. By using 4.3, impulse is found as

$$v_k(t + dt) - v_k(t) \geq -(1 + \epsilon).v_k(t) \tag{4.6}$$

$$p_k = \lambda.(v_k(t + dt) - v_k(t)) \tag{4.7}$$

where λ is an inertia matrix in contact space.

By combining 4.6 and 4.7,

$$p_k \geq \lambda.\big[-(1 + \epsilon).v_k(t)\big] \tag{4.8}$$

By using the jacobian matrix, impulse force is written in joint space and it is shown as 'f^I' in this space.

$$f^I_k \geq \mathrm{A}.\big[-(1+\epsilon).\ddot{q}_k(t)\big] \tag{4.9}$$

By using jacobian matrix, 4.8 can be written as

$$\mathrm{J}_c.f^I_k \geq \lambda.\big[-(1+\epsilon).\mathrm{J}_c.\ddot{q}_k(t)\big] \tag{4.10}$$

$$f^I_k \geq \mathrm{J}_c^{\mathrm{T}}.\lambda.\mathrm{J}_c.\big[-(1+\epsilon).\ddot{q}_k(t)\big] \tag{4.11}$$

From 4.9 and 4.11, relationship of inertia matrices is found as:

$$\begin{aligned}\lambda &= \mathrm{J}_c.\mathrm{A}.\mathrm{J}_c^{\mathrm{T}}\\ \lambda^{-1} &= \mathrm{J}_c.\mathrm{A}^{-1}.\mathrm{J}_c^{\mathrm{T}}\end{aligned} \tag{4.12}$$

Contraints of collision are derived in 4.8 and 4.4. In 4.13, the constraints are written as a quadratic problem.

$$\begin{aligned}p_k.(\lambda^{-1}.p_k + \big[(1+\epsilon).v_k(t)\big]) &= 0\\ \lambda^{-1}.p_k + \big[(1+\epsilon).v_k(t)\big] &\geq 0\\ p_k &\geq 0\end{aligned} \tag{4.13}$$

These constraints can also be written as a linear complementarity problem (LCP):

$$p^{\mathrm{T}}.\Big[\lambda^{-1}.p + \big[(1+\epsilon).v(t)\big]\Big] = 0 \tag{4.14}$$

4.2.2 Derivation of Contact Constraints

Contact is determined when distance between objects is zero at time t and $t + dt$. In some conditions such as falling, running and walking, collision is preliminary condition of contact. Collision constraints are built to keep relative velocity equal or less than previous velocity as shown in 4.3. The aim of a contact model is to keep bodies in contact without interpenetration. Penetration is undesirable and to prevent it, contact constraints are used to keep the relative velocity zero.

$$v_k = 0 \tag{4.15}$$

4.15 is the definition of contact and it is not a constraint of contact model. After contact, relative velocity reaches zero. Negative relative acceleration causes penetration while positive relative acceleration causes seperation from contact. There is no movement when relative acceleration is equal to zero. Contact force is equal or greater than zero since contacts can only push each other. This shows in 4.16, where F refers to contact force and a refers to relative accelerations.

$$a \geq 0 \qquad\qquad F \geq 0 \tag{4.16}$$

Relative acceleration is mathematically modeled as:

$$a_k = a_c + a_j \tag{4.17}$$

where a_k is relative acceleration in k^{th} contact, a_c is contact acceleration and a_j is collide object acceleration which is defined in contact space.

a_c is defined explicitly in 4.18 and a_j is defined by using jacobian matrix in 4.19.

$$a_c = \lambda^{-1}.F \tag{4.18}$$

$$a_j = \dot{\mathrm{J}}_c.\dot{q} + \mathrm{J}_c.\ddot{q} \tag{4.19}$$

where $\mathrm{J}_c.\ddot{q} = \mathrm{J}_c.\mathrm{A}(q)^{-1}.(\mathrm{T} - C(q,\dot{q}).\dot{q} - g(q))$.

Contact constraint is written as quadratic problem:

$$\begin{aligned} F_k.(\lambda^{-1}.F_k + a_j) &= 0 \\ F_k &\geq 0 \\ (\lambda^{-1}.F_k + a_j) &\geq 0 \end{aligned} \tag{4.20}$$

As linear complementarity problem:

$$F^T.(\lambda^{-1}.F + a_j) \geq 0 \tag{4.21}$$

In addition to the normal force, the contact model includes friction forces. The friction is modeled as coloumb friction and viscous friction. It is also used in a constraint to obtain exact contact force.

$$F_t \leq \mu_c.F_n + \mu_v.\mathrm{V}_t \tag{4.22}$$

where F_t is tangential components of contact force and F_n is normal of contact force. μ_c is coulomb friction constant and μ_v is viscous friction constant. V_t is tangential components of linear velocity.

4.3 Solution of Collision and Contact Forces

The aim of contact models is to obtain collision or contact forces correctly. There is not enough information to solve forces directly, therefore the constraints are derived and equations are formulated as LCP and quadratic problem as shown in Section 4.2. As mentioned by Baraff in [6], solvers of quadratic problem require higher computational times compared to LCP solvers. Due to this, contact forces are modeled as a LCP in this thesis. When solving LCPs, mainly two solvers are used which are Lemke's algorithm and projected Gauss Seidel algorithm. These are explained in Subsections 4.3.1 and 4.3.2 respectively.

The constraints of the contact forces are shown in 4.14 and 4.21. In these constraints, the unknown parameters (λ^{-1}, v_k and a_k) must be obtained to solve the forces. These parameters can be derived from $\dot{q}$ and A^{-1} which are known from solving ABM equations.

$$\begin{aligned} v_k &= \mathrm{J}_c.\dot{q} \\ a_k &= \dot{v}_k \\ \lambda^{-1} &= \mathrm{J}_c.\mathrm{A}^{-1}.\mathrm{J}_c \end{aligned} \tag{4.23}$$

In 4.23, all unknown parameters depend on jacobian matrix which converts joint space to contact space. Chandler et al. offer a method to solve the unknown parameters without obtaining the jacobian matrix [14]. In this method, the contact or collision forces are seen as external forces and torque of system is written as:

$$\mathrm{T} = \mathrm{T}_j + T_{cont} \tag{4.24}$$

where T_j vector is calculated joint torques without involving contact forces. T_{cont} refers to joint torques resulting from contact/collision forces and it is defined in joint space. It is solved by using 4.25 and 4.26.

For collision,

$$T_{cont} = \mathrm{J}_c^T.p \tag{4.25}$$

For contact,

$$T_{cont} = \mathrm{J}_c^T.F \tag{4.26}$$

Joint accelerations are mathematically modeled with contact forces in 4.27.

$$\ddot{q} = \ddot{q}_j + \ddot{q}_{cont} \tag{4.27}$$

where $\ddot{q}_j$ is calculated joint accelerations without considering contact forces. $\ddot{q}_{cont}$ refers to joint accelerations resulting from contact/collision forces and it is defined in joint space. In contact space, acceleration is defined in 4.18 and the calculation of $\ddot{q}_{cont}$ is shown in 4.28.

$$\ddot{q}_{cont} = \mathrm{J}_c.\lambda^{-1}.F \tag{4.28}$$

Combining 4.12, 4.27 and 4.28, joint accelerations of system are rewritten as:

$$\ddot{q} = \ddot{q}_j + \mathrm{A}^{-1}.\mathrm{J}_c^{\mathrm{T}}.F \tag{4.29}$$

where $\ddot{q}_j$ is known from 4.2. According to this equation, if gravitational force, joint torques and joint velocities are zero, $\ddot{q}_j$ is also zero. This way, joint accelerations are equal to $\ddot{q}_{cont}$.

$$\ddot{q} = \mathrm{A}^{-1}.\mathrm{J}_c^{\mathrm{T}}.F \tag{4.30}$$

When a unit force is given into the system in 4.30, joint accelerations are equal to $\mathrm{A}^{-1}\mathrm{J}_c^{\mathrm{T}}$ and by ABM, the $\mathrm{A}^{-1}\mathrm{J}_c^{\mathrm{T}}$ is solved. Same idea can be used for finding λ^{-1}. For this, gravitational force, joint velocities and joint torque are set to zero so that a_j term in the 4.17 vanishes. The resulting equation is:

$$a_k = \lambda^{-1}.F \tag{4.31}$$

When a unit force is given into the system in 4.31, the relative acceleration is equal to λ^{-1} (also refered as $\mathrm{J}_c\mathrm{A}^{-1}\mathrm{J}_c^{\mathrm{T}}$). Another unknown parameter is relative acceleration (a_k) as shown in 4.17. a_j is projection of linear acceleration of bodies on contact space without considering external forces and it can be solved by ABM. Therefore, a_k can be solved by using known paremeters a_j and $\lambda^{-1}F$ according to 4.17. By taking integrate of relative acceleration, relative velocity is obtained as shown in 4.32.

$$v_k(t + dt) = a_k.dt + v_k(t) \tag{4.32}$$

where ‘dt’ refers to difference between two consecutive simulation time frames.

The constraints are obtained as linear complementarity problem therefore, the exact contact force can be solved by using solvers such as Lemke's method and projected Gauss-Seidel method.

4.3.1 Lemke's Algorithm

Lemke's algorithm is a pivoting solver which is used in linear complementarity problem with semimonotone matrices [18]. By Lemke algorithm, LCP is solved in two steps which are initialization and solving by pivot algorithm. To explain this algorithm, a LCP is written in 4.33 where u is unknown parameter.

$$\mathrm{u}^{\mathrm{T}}.(\mathrm{A}.\mathrm{u} + b) = 0 \tag{4.33}$$

First step;

- if b ≥ 0, u $= 0$ therefore, solution is obtained
- if b < 0, the system is modified by using Jordan Exchange method [37] and to obtain a solution, second step is performed.

Second step;

- According to nonbasic variable, the pivoting column is chosen.
- By ratio test, the pivoting row is chosen.
- Then, Jordan Exchange method is applied.
- When u and $(\mathrm{Au} + b)$ are complementarity, the algorithm finds the solution. Otherwise, the second step is performed again.

4.3.2 Projected Gauss-Seidel Method

Projected Gauss-Seidel algorithm is a significant algorithm to solve LCP iteratively. Chardonnet et al. stated that projected Gauss Seidel method is faster than Lemke's algorithm when there are more than 10 contact, also it gives more accurate results than Lemke's algorithm. Moreover, the pivoting algorithm is not robust when constraints include friction [14].

For explaining algorithm, a LCP is written in 4.33 where u is unknown parameter. By splitting method, A matrix is split into two matrices as explained in [31].

$$\mathrm{A} = \mathrm{M} - \mathrm{N} \tag{4.34}$$

$$\mathrm{u}_{l+1}^{\mathrm{T}}.(\mathrm{M}.\mathrm{u}_{l+1} - \mathrm{N}.\mathrm{u}_{l} + b) = 0 \tag{4.35}$$

where l refers to iteration number. According to constraints, the inequalities of u_{l+1} are written as:

$$\begin{aligned} \mathrm{u}_{l+1} &> 0 \qquad & (\mathrm{M}.\mathrm{u}_{l+1} - \mathrm{N}.\mathrm{u}_{l} + b) = 0 \\ \mathrm{u}_{l+1} &= 0 \qquad & (\mathrm{M}.\mathrm{u}_{l+1} - \mathrm{N}.\mathrm{u}_{l} + b) > 0 \end{aligned} \tag{4.36}$$

From 4.36, u_{l+1} obtained as

$$\mathrm{u}_{l+1} = 0 \tag{4.37}$$

$$\mathrm{u}_{l+1} = \mathrm{M}^{-1}.(\mathrm{N}.\mathrm{u}_{l} - b) \tag{4.38}$$

In our contact and collision models, when contact or collision occurs, the forces are zero, otherwise forces get positive value as shown 4.37 and 4.38. The contact or collision is detected easily by using algorithms as mentioned in Chapter 2. By using 4.38 iteratively, u_{l+1} are solved. The iteration is terminated when the condition is satisfied as in [19].

$$\epsilon \geq \frac{\|\mathrm{u}_{l+1} - \mathrm{u}_{l}\|}{\|\mathrm{u}_{l+1}\|} \tag{4.39}$$

4.4 Implementation of Exact Contact Modeling

The contact model is defined in Section 4.2 and Section 4.3. As shown in 4.24, contact forces are applied as external forces. Therefore, system dynamic can be calculated before adding contact forces

1. Linear and angular velocity of links' center of gravity is obtained by using 3.37 to 3.40.
2. Spatial articulated bias force 'Z^{A}', spatial articulated inertia matrix 'I^{A}' are initialized for each link. The initial values of them are same as spatial bias force and spatial inertia matrix which are shown in 3.13.
3. Coriolis vectors are calculated for each link as 3.22 and 3.24.
4. Spatial articulated bias force and spatial articulated inertia matrix are calculated iteratively by using 3.41 and 3.42.
5. Joint accelerations and acceleration of links' center of gravity is calculated by 3.43 and 3.44.

The implementation of ABM algorithm to obtain dynamic of system without contact forces is performed in five steps. This implementation can be modified based on linkage type (serial linkage or tree like linkage) and base type (fixed base or float base). The following part is the calculation of system dynamic which results only contact forces. First step of this is the calculation of '$\mathrm{A}^{-1}.\mathrm{J}_c^{\mathrm{T}}$' and '$\mathrm{J}_c.\mathrm{A}^{-1}.\mathrm{J}_c^{\mathrm{T}}$' to obtain contact forces.

1. Linear and angular velocity of links' center of gravity, joint velocities, applied torques and gravity are given as zero.
2. Spatial articulated bias force 'Z^{A}' and spatial articulated inertia matrix 'I^{A}' are initialized for each link.
3. Coriolis vectors are calculated for each link.

4. A unit force in x direction of contact space is defined for i^{th} contact.

5. The unit force is transformed to CoG of link which is in i^{th} contact.

6. The obtained force in (5) is added to spatial articulated bias force of link in i^{th} contact.

7. Spatial articulated bias force and spatial articulated inertia matrix are calculated iteratively.

8. Joint accelerations and linear acceleration of links' center of gravity is calculated.

9. The linear acceleration of links' center of gravity in contact is transformed to contact space. Therefore, the effect of contact forces on each contact is calculated.

10. (a) A unit force in y direction of contact space is defined for i^{th} contact and return (5)

    (b) A unit force in z direction of contact space is defined for i^{th} contact and return (5)

11. (1) to (9) is performed for each contact.

In above algorithm, 'i' and 'm' refer to contact number index and total contact number respectively where $1 \leq i \leq m$.

The obtained acceleration in the second part of algorithm shows how system's accelerations are changed when a unit force is applied in x direction or y direction or z direction of contact space. These obtained joint accelerations are used to build the '$A^{-1}J_c^T$' matrix which is a $n\times$ 3m matrix where n is a contact number. Similarly, the linear acceleration of link center of gravity, defined in contact space, are used to built '$J_c.A^{-1}J_c^T$' matrix which is a $3m\times$ 3m.

Contacts are detected by collision detection algorithms and then contact forces are solved by projected Gauss Seidel algorithm. The algorithm is preferred because of its advantages, explained in Subsection 4.3.2. The equation from 4.32 is written as:

$$v_k(t + dt) = \lambda^{-1}.F.dt + a_j.dt + v_k(t) \tag{4.40}$$

In contact or collision, the normal component of $v_k(t + dt)$ must be equal to zero. Projected Gauss Seidel is implemented as:

1. Take $\mathrm{A}^{-1}\mathrm{J}_c^{\mathrm{T}}$, a_j and $v_k(t)$ as inputs.
2. Multiply $\mathrm{A}^{-1}\mathrm{J}_c^{\mathrm{T}}$ and a_j by 'dt' to obtain $\lambda^{-1}F.dt$, $a_j.dt$ respectively.
3. Initialize the forces as zero.
4. Check the contact condition, if there is contact, go to (5), else go to (9).
5. With splitting method which is explained in Subsection 4.3.2;
   (a) Velocity is obtained by contact forces of 1th to i-1th contact from current iteration
   (b) Velocity is obtained by contact forces of i-1th to mth contact from previous iteration
6. From 4.40, normal force is obtained as,

$$F_i^n = F_i - \frac{V}{(\lambda_i^{-1})_n} \tag{4.41}$$

7. If F_i^n is equal or less than zero, go to (8), else go to (9)
8. $F_i^n = 0$, stop
9. From 4.40, normal force is obtained as,

$$F_i^t = F_i - \frac{V}{\lambda_t} \tag{4.42}$$

10. Check 4.22. If it is satisfied, go to 12, else go to 11

11. Recompute F_i^t as:
$$F_i^t = \frac{F_i^t}{\|F_i^t\|}.(\mu.F_n + \mathrm{V}_t) \tag{4.43}$$

12. Go to (4), do calculation for i+1th contact force.

13. Check the termination condition as shown in 4.39. If it is satisfied, stop algorithm. Else go to (14)

14. Go to (4), increase iteration number by 1 and start from first contact again

where $(\lambda_i^{-1})_n$ refers to normal part of λ^{-1}, related to ith contact force and 1×1. λ_t refers to average eigenvalue of tangential part of λ^{-1}, related to ith contact force and also 1×1.

After obtaining contact forces by projected Gauss Seidel algorithm, acceleration of system is obtained from 4.29.

Chapter 5

Simulation of the Quadruped Dynamics

5.1 Introduction

In this chapter, a quadruped robot is simulated by combining the articulated body method with the exact contact method which are explained in Chapter 3 and Chapter 4, respectively. The main purpose of this simulation is to find accurate contact forces. This was not always possible with the previously used penalty based contact model [27]. Simulations with exact and penalty based contact models are compared. This chapter presents information about the kinematic arrangement of our quadruped robot, a brief explanation of penalty based contact model, and simulation results.

5.2 The Quadruped Model

The quadruped model is formed by four legs and a torso. Each leg has four DOF with two axes at the hip, one at the knee and one at the ankle. All joints are revolute. The quadruped kinematic arrangement is shown in Fig. 5.1.

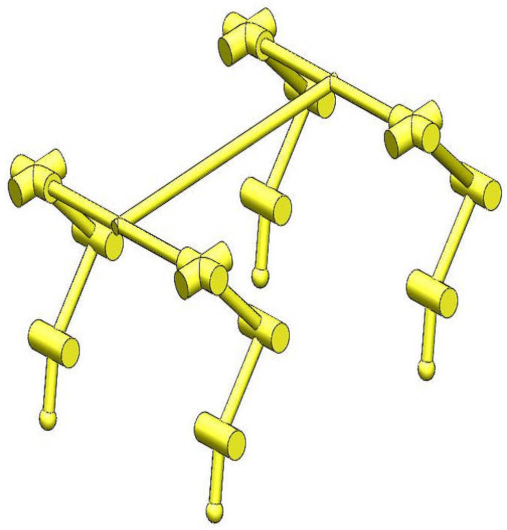


FIGURE 5.1: The quadruped kinematic arrangement

The dynamics parameters of the quadruped robot are presented in Table 5.1.

TABLE 5.1: Properties of the Quadruped Model

Links	**Dimensions [m] (LxWxH)**	**Mass [kg]**
Torso	1.2x0.6x0.15	50
Thigh	0.28x0.05x0.1	4.4
Shank	0.27x0.05x0.1	3.65
Feet	0.22x0.05x0.5	3.65

5.3 The Penalty Based Algorithm

The penalty based approach is a popular in contact modeling. It has low computational complexity and is simple to implement. Due to these advantages, researchers in computer graphics and robotics use this model very often in simulations. Penalty based methods model contact as a spring - damper system. Contact forces are generated based on penetration, as penalties against penetration. For the simulation to be realistic, penetrations have to be minimal. This means that large spring stiffness values are required. Also, simulation cycle times have to be kept reasonably large so that admissible simulation durations can be acheived. As a result, interpenterations between simulated bodies can be deeper then in the real world and the too deep interpenetration is penalized by huge spring forces. Such a contact model is handicapped by unrealistic contact forces that can be encountered in high-impact and high-speed situations such as jumping. This makes penalty-based methods undesirable for highly dynamic robotics applications.

5.4 Simulation Results

Simulations are carried out on Matlab and visual basic with OpenGl library is used for animation. Same dynamics parameters shown in Table 5.2 are used for both contact methods. PID position controller is used to control joint positions. Table 5.3 shows the control parameters which are the same for each leg and across both contact models.

Table 5.2: Parameters of the Simulation

Parameters	
Step time	0.0005 s
Stop time	10.6 s
Coefficient of Joint Viscous Friction	0.3 Ns/m^2
Coefficient of Joint Coulomb Friction	0.55
Coefficient of Ground Viscous Friction	0.4 Ns/m^2
Coefficient of Ground Coulomb Friction	0.3
Iteration limit for projected Gauss Seidel algorithm	100
Epsilon value of Terminated Condition of projected Gauss Seidel algorithm	10^{-20}

Table 5.3: Controller Parameters

Controller Parameters				
	Hip (Lateral Plane)	Hip (Ventral Plane)	Knee	Ankle
Kp	6000	20000	70000	30000
Kd	1	1	1	1
Ki	120	120	120	120

By using parameters in Table 5.2, the quadruped robot is simulated for different scenarios which are:

1. Simulation of quadruped trot
2. Simulation of falling of the quadruped robot from 0.1 meter height on the ground
3. Simulation of falling of the quadruped robot from 0.5 meter height on the ground
4. Simulation of falling of the quadruped robot from 1.5 meter height on the ground
5. Simulation of the jumping motion of the quadruped robot

These simulations are performed for both the penalty-based algorithm and the exact contact model.

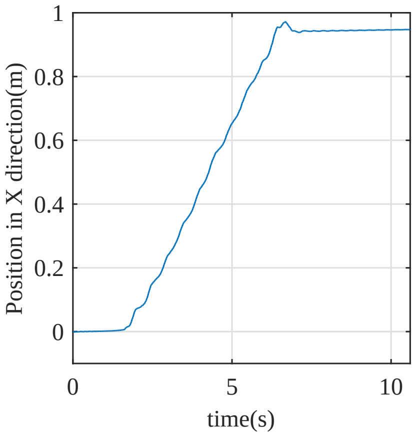
5.4.1 Simulation of Quadruped Trot

Trot is a running gait in four-legged locomotion where diagonal legs lifted off the ground at the same time. In these simulations, the trot reference is generated by a zero moment point (ZMP) based method [1]. Two contact models (and also their associated simulation integration techniques) are evaluated based on results of simulation, namely, recorded curves of body position, body orientation and contact forces. The robot has four contact points, each located at a leg tip.

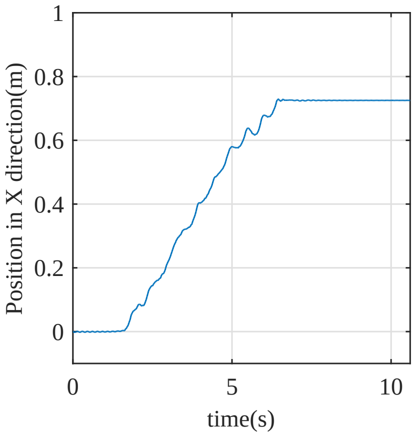
In this simulation, the robot runs along the x direction and the position components are seen in Figures 5.2a and 5.2b. The total traveled distance in x direction is around 0.9 meter with the exact contact model and around 0.7 meter with the penalty-based model. The difference can be explained by the slip of the robot feet. The slip is related to the by the friction model. The friction is modeled as viscous and Coulomb friction in both simulations. However, Coulomb friction forces are modeled via a special ad hoc spring system [27] in the penalty-based method while being formulated as a LCP in exact contact model. Figures 5.2c and 5.2d show that the body traveled in the y direction around 0.1 meter with the exact contact model and around 0.18 meter with the penalty-based model. This results from the asymetric right and left side foot references in the beginning of the trot. It should be noted that the trot gate controller does not have yaw directional orientational feedback. The sole control action being executed is that of PID joint position controller. The trot position references are provided in terms of joint positions. There is no remarkable difference between body position in the z direction in Figures 5.2e and 5.2f.

The roll, pitch and yaw angles of the body are shown in Fig. 5.3. In penalty-based and exact contact simulations, orientation of the robot is almost identical except for the yaw axis. The differences in yaw angle result from different realizations of Coulomb friction at the contacts. The forces are shown in Figures 5.4, 5.5, 5.6 and 5.7. The figures refer to the contact forces in leg for right back leg, left back leg, right front leg and left front leg, respectively. The contact forces in the x direction and the y direction are friction forces and the forces in the z direction keep the robot up to ground. Moreover, the forces in the z direction are equal to zero when there is no contact. When contact occurs, there is a positive force in the z direction.

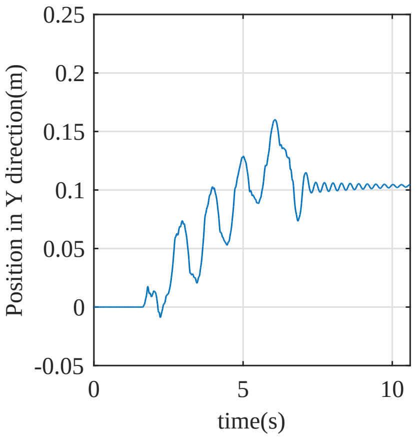
The significant point of the results, according to the observed forces and positions is that the contact forces do not increase the energy of system. In contrary, they conserve the energy or absorb the energy based on the elasticity of system. That results in a realistic behaviour in the simulation environment.



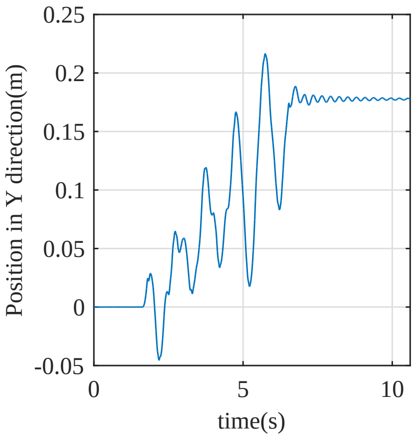
(a)



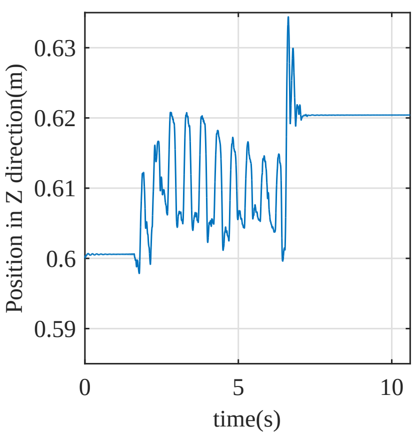
(b)



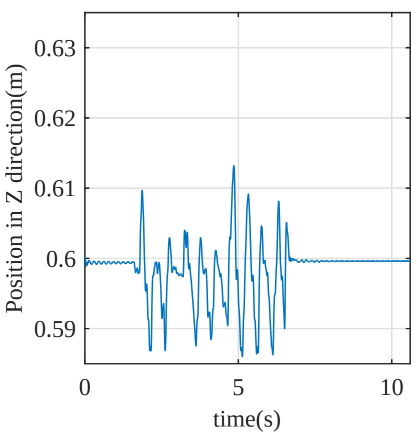
(c)



(d)



(e)



(f)

FIGURE 5.2: (a) position of body in x direction for exact contact model, (b) Position of body in x direction for penalty based model, (c) Position of body in y direction for exact contact model, (d) Position of body in y direction for penalty based model, (e) Position of body in z direction for exact contact model, (f) Position of body in z direction for penalty based model

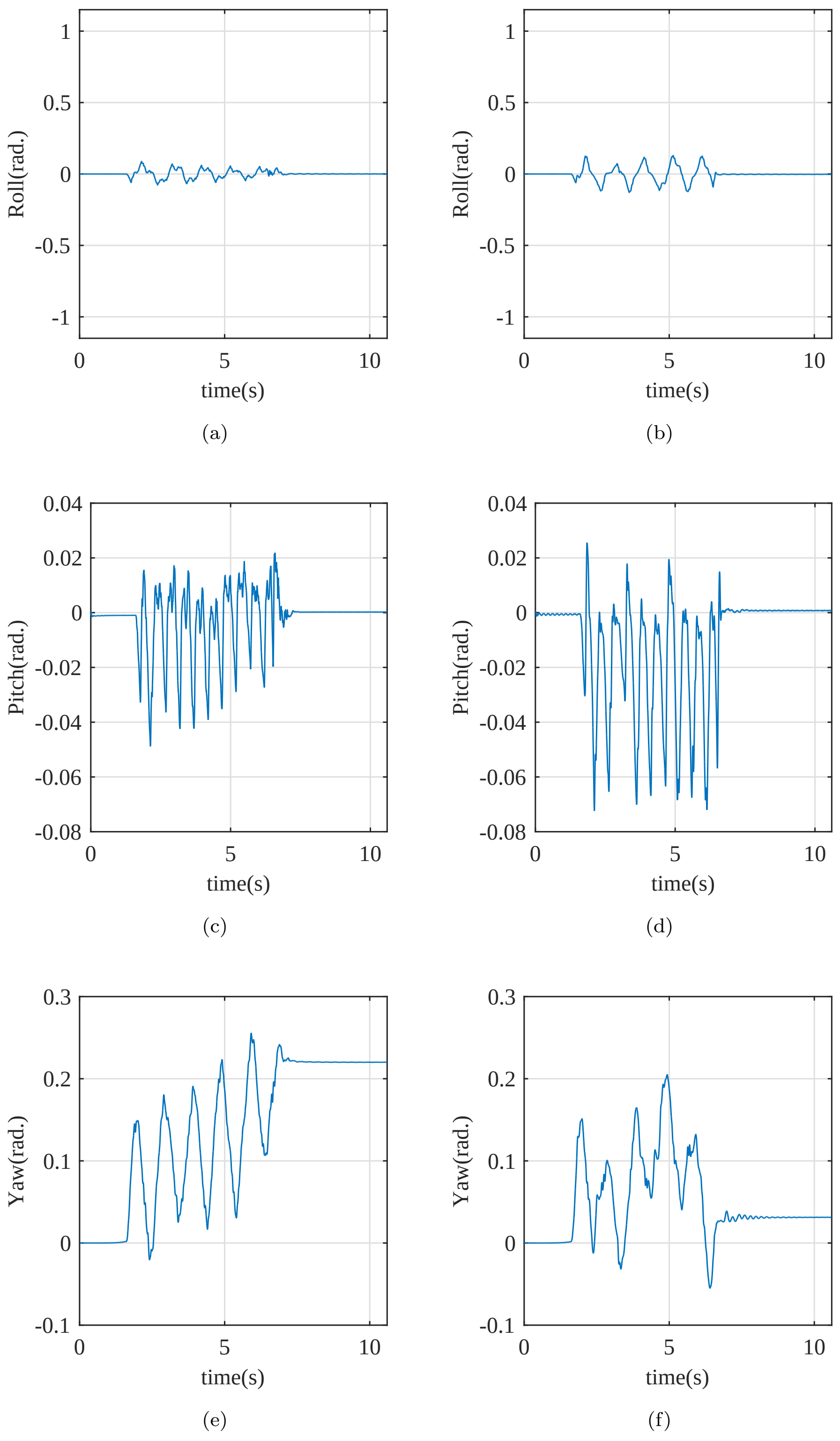


FIGURE 5.3: (a) Roll of body for exact contact model, (b) Roll of body for penalty based model, (c) Pitch of body for exact contact model, (d) Pitch of body for penalty based model, (e) Yaw of body for exact contact model, (f) Yaw of body for penalty based model

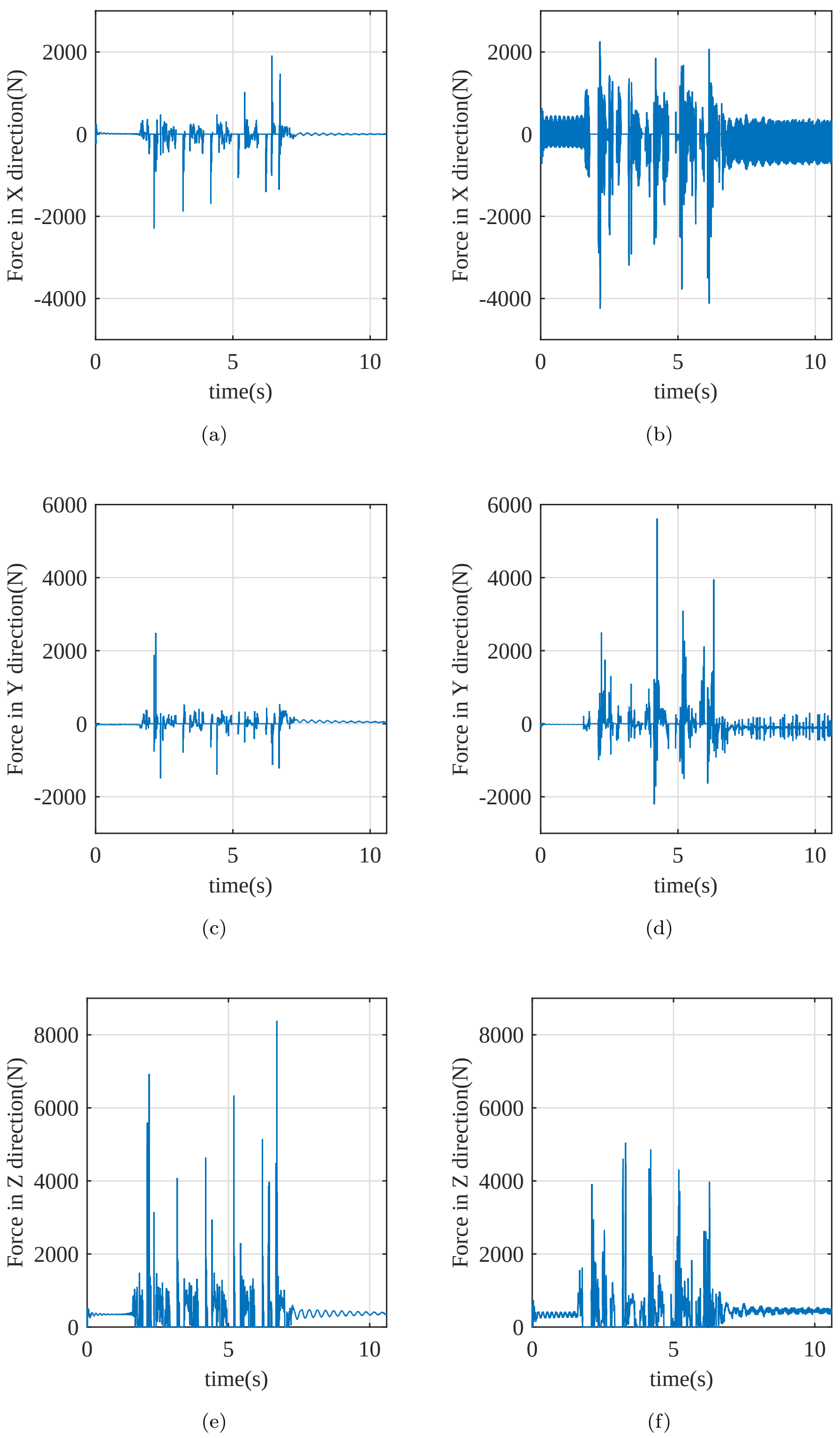


FIGURE 5.4: (a) Contact Force of right back leg in x direction for exact contact model, (b) Contact Force of right back leg in x direction for penalty based model, (c) Contact Force of right back leg in y direction for exact contact model, (d) Contact Force of right back leg in y direction for penalty based model, (e) Contact Force of right back leg in z direction for exact contact model, (f) Contact Force of right back leg in z direction for penalty based model

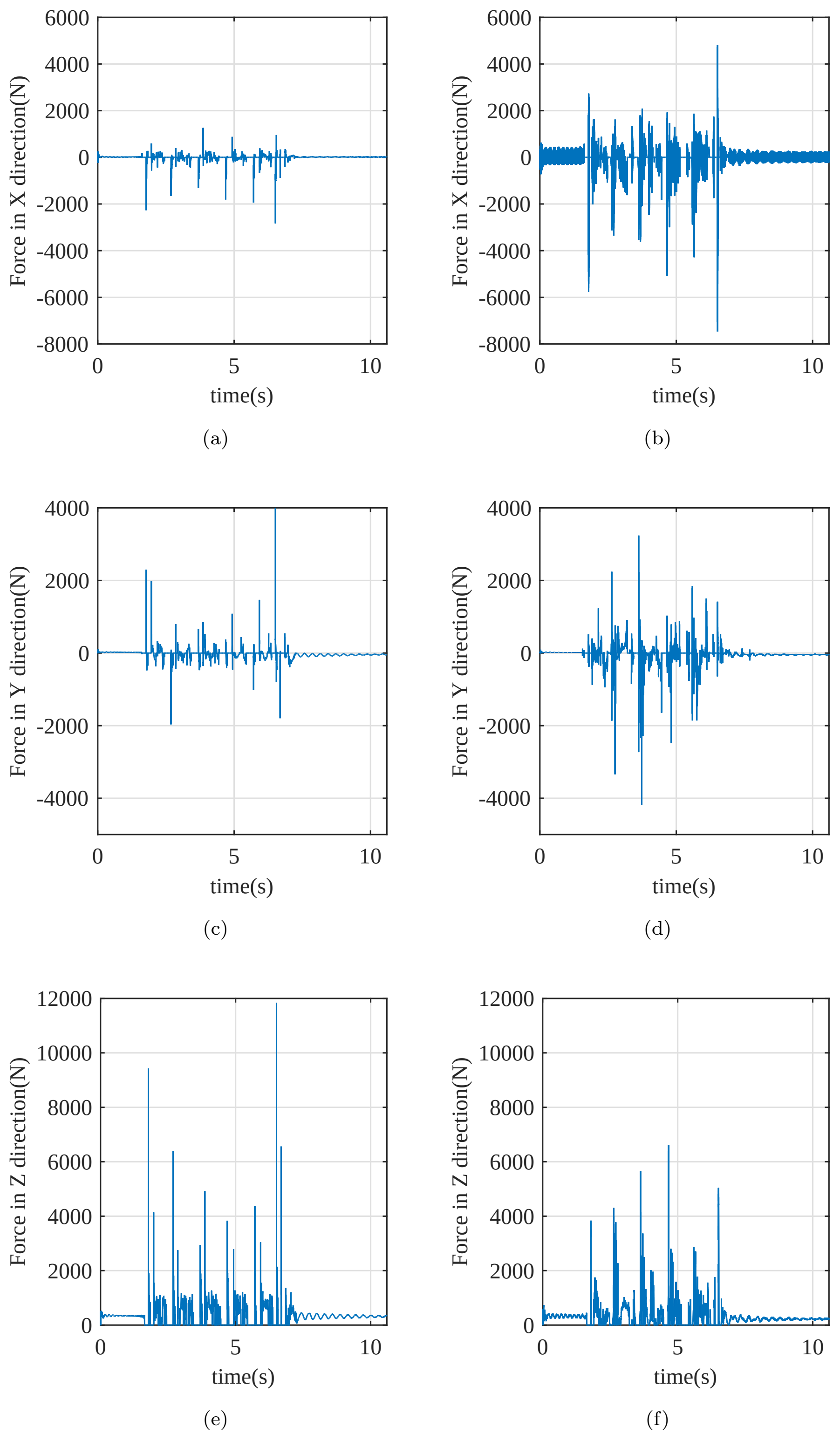


FIGURE 5.5: (a) Contact Force of left back leg in x direction for exact contact model, (b) Contact Force of left back leg in x direction for penalty based model, (c) Contact Force of left back leg in y direction for exact contact model, (d) Contact Force of left back leg in y direction for penalty based model, (e) Contact Force of left back leg in z direction for exact contact model, (f) Contact Force of left back leg in z direction for penalty based model

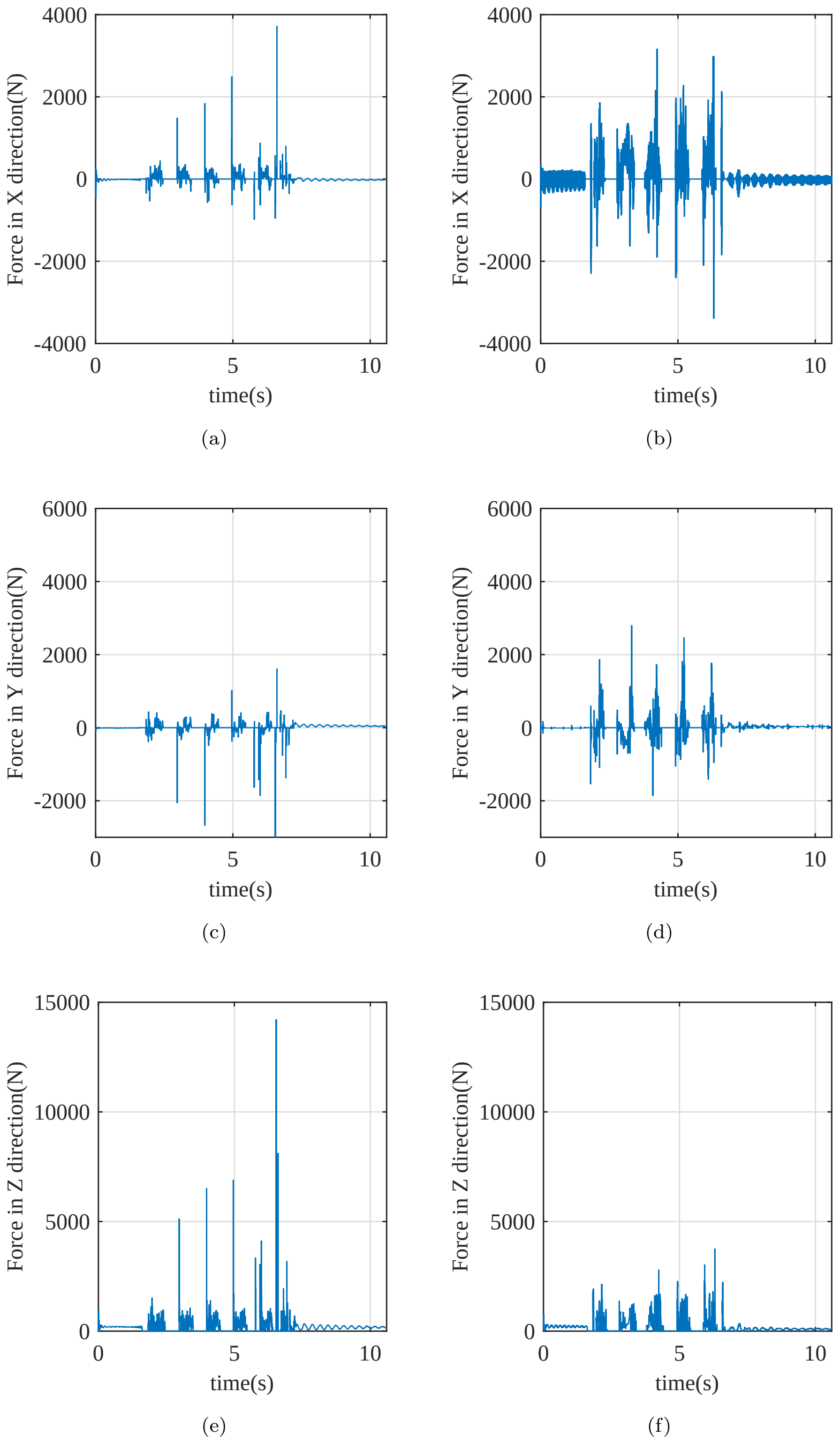
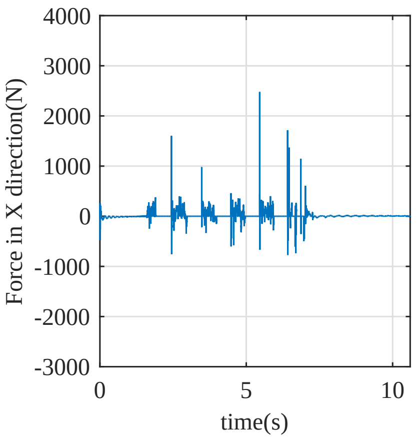
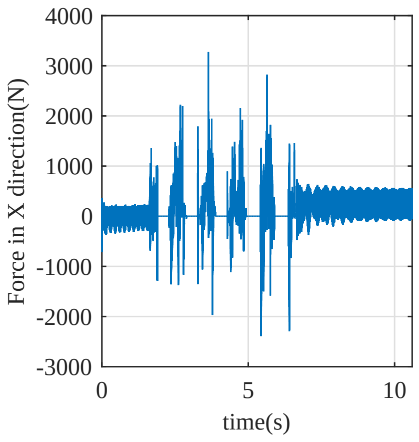


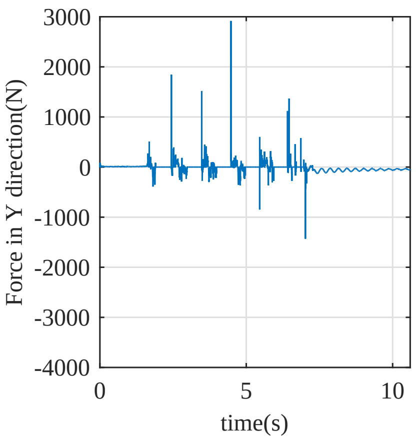
FIGURE 5.6: (a) Contact Force of right front leg in x direction for exact contact model, (b) Contact Force of right front leg in x direction for penalty based model, (c) Contact Force of right front leg in y direction for exact contact model, (d) Contact Force of right front leg in y direction for penalty based model, (e) Contact Force of right front leg in z direction for exact contact model, (f) Contact Force of right front leg in z direction for penalty based model



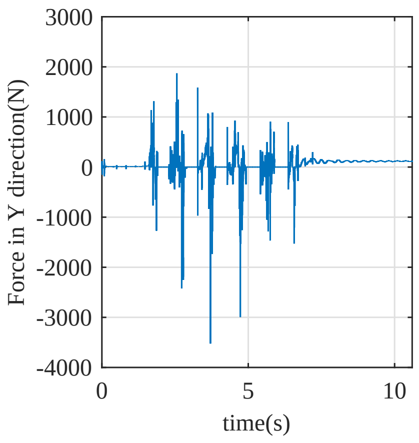
(a)



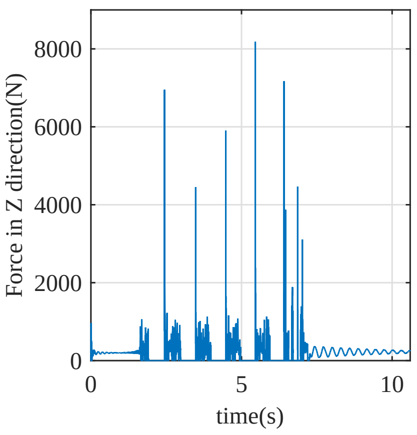
(b)



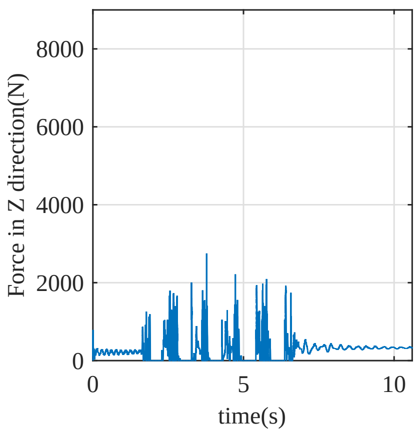
(c)



(d)



(e)



(f)

FIGURE 5.7: (a) Contact Force of left front leg in x direction for exact contact model, (b) Contact Force of left front leg in x direction for penalty based model, (c) Contact Force of left front leg in y direction for exact contact model, (d) Contact Force of left front leg in y direction for penalty based model, (e) Contact Force of left front leg in z direction for exact contact model, (f) Contact Force of left front leg in z direction for penalty based model

5.4.2 Simulation of Fall of the Quadruped Robot from a 0.1 meter Height

In this scenario, the quadruped is initially stationary at 0.1 meter altitude and it falls due to gravity. The aim of this simulation is evaluating the performance of the penalty-based model and the exact contact model when the quadruped falls from a low altitude. The joint position controller keeps joint positions fixed in their initial values.

The position of the robot CoG is shown in Fig. 5.8. According to the results, the robot falls from 0.1 meter and stays at ground level after the establisment of contact is occured. Both simulations are succeeded with this moderately low altitude fall.

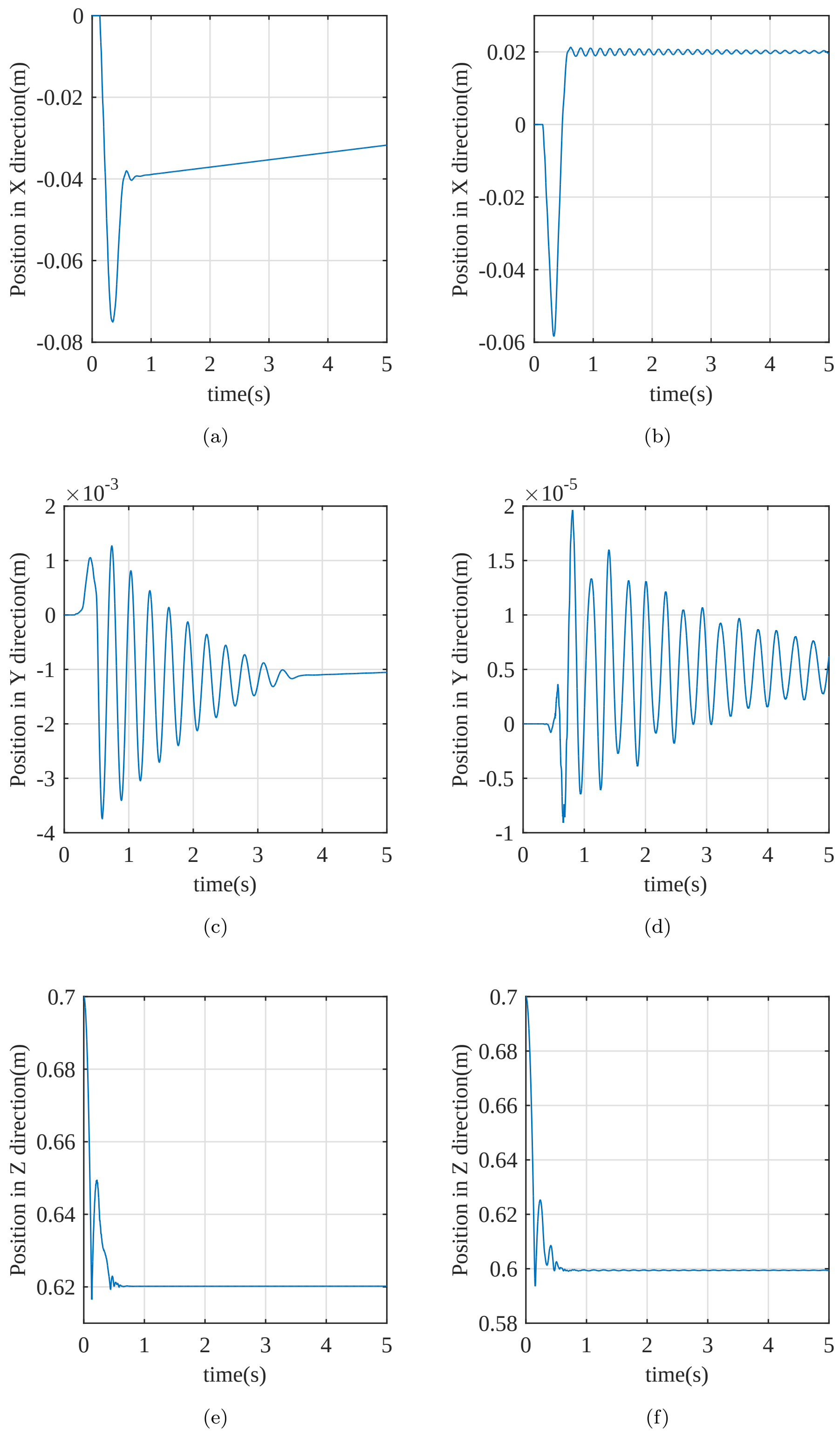
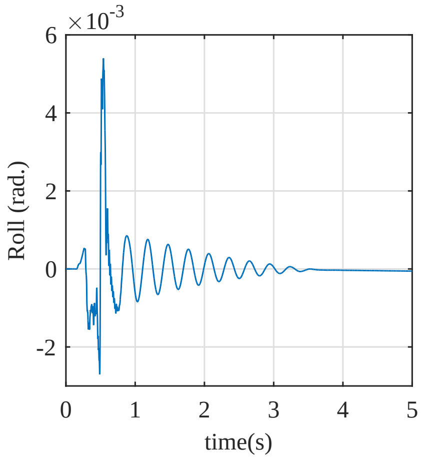
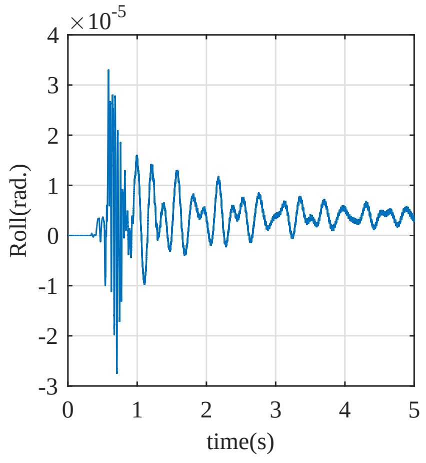


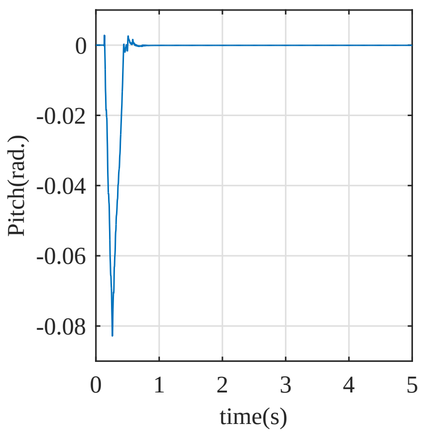
FIGURE 5.8: (a) position of body in x direction for exact contact model, (b) Position of body in x direction for penalty based model, (c) Position of body in y direction for exact contact model, (d) Position of body in y direction for penalty based model, (e) Position of body in z direction for exact contact model, (f) Position of body in z direction for penalty based model



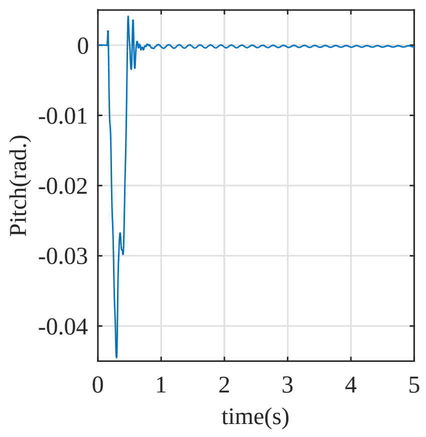
(a)



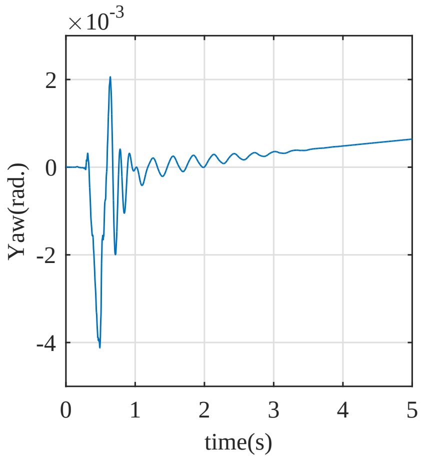
(b)



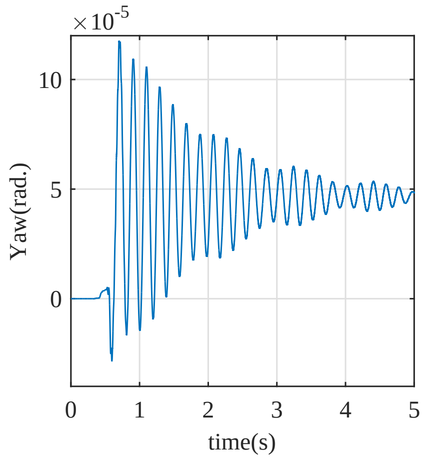
(c)



(d)



(e)



(f)

FIGURE 5.9: (a) Roll of body for exact contact model, (b) Roll of body for penalty based model, (c) Pitch of body for exact contact model, (d) Pitch of body for penalty based model, (e) Yaw of body for exact contact model, (f) Yaw of body for penalty based model

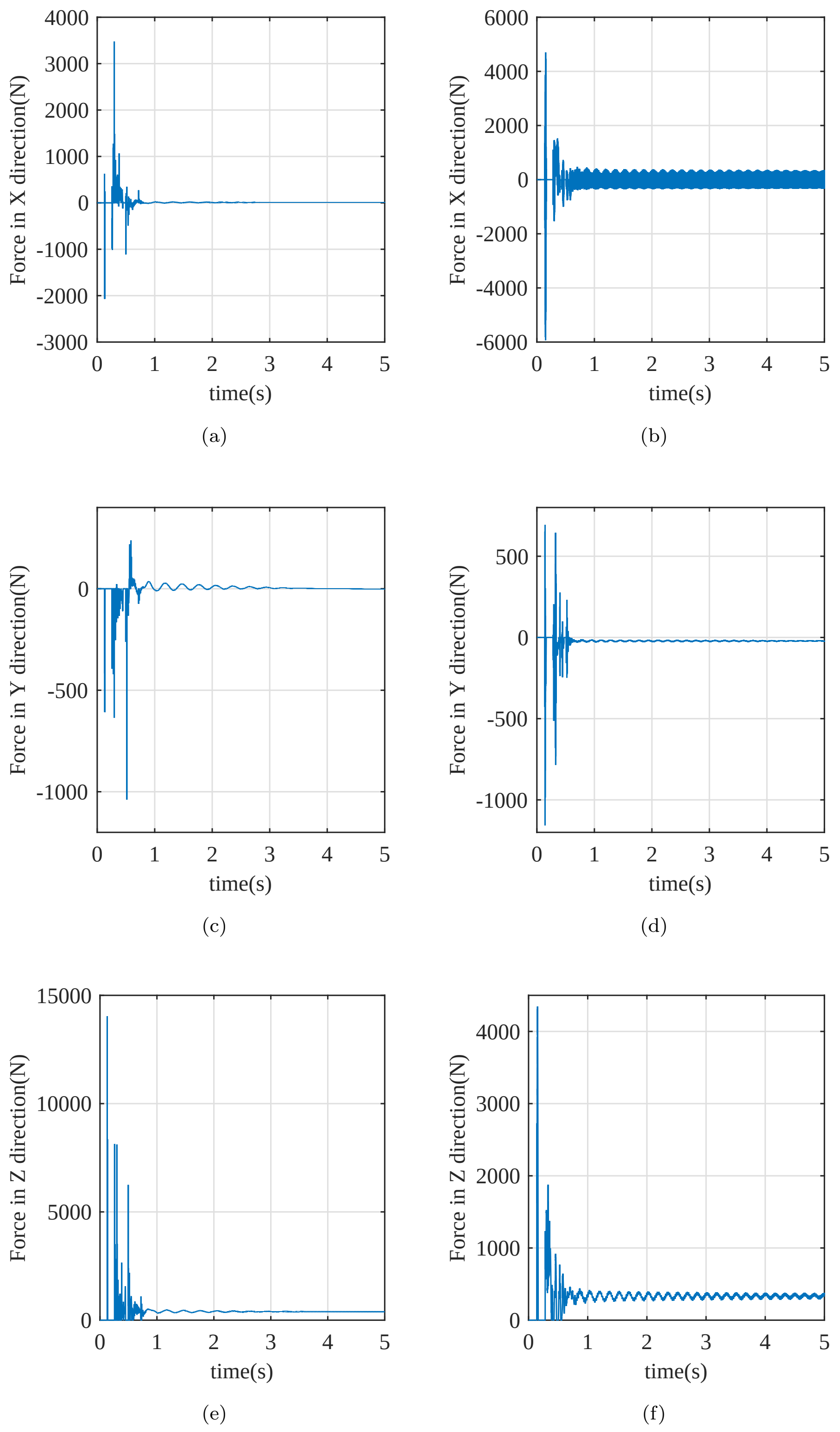
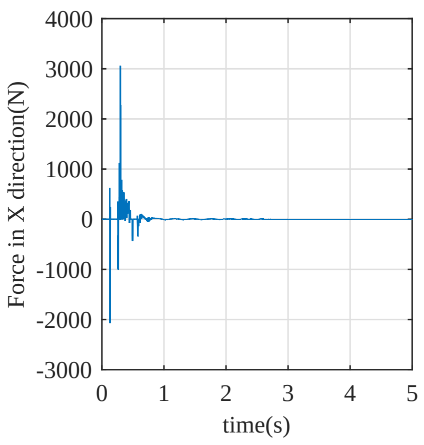
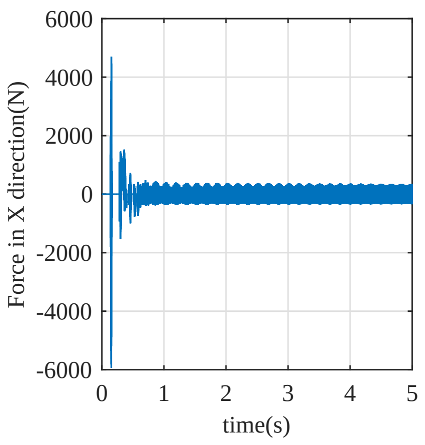


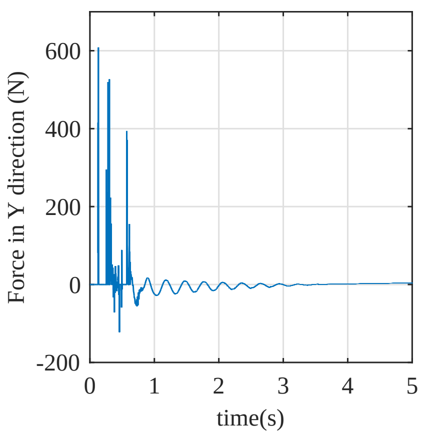
FIGURE 5.10: (a) Contact Force of right back leg in x direction for exact contact model, (b) Contact Force of right back leg in x direction for penalty based model, (c) Contact Force of right back leg in y direction for exact contact model, (d) Contact Force of right back leg in y direction for penalty based model, (e) Contact Force of right back leg in z direction for exact contact model, (f) Contact Force of right back leg in z direction for penalty based model



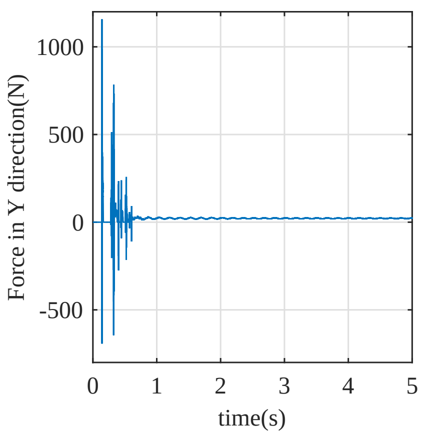
(a)



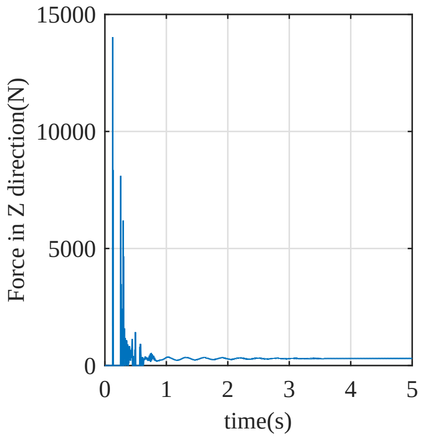
(b)



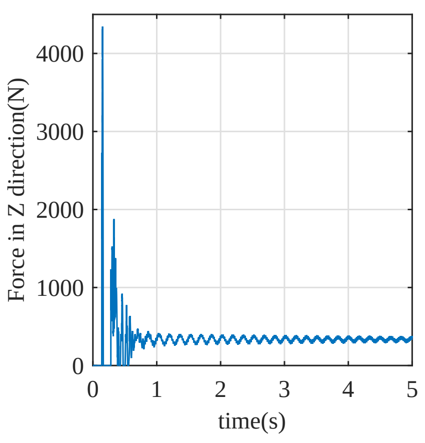
(c)



(d)

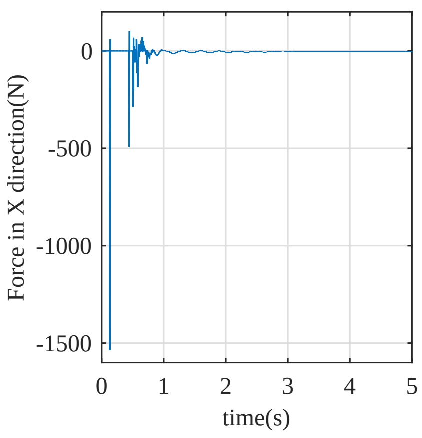


(e)

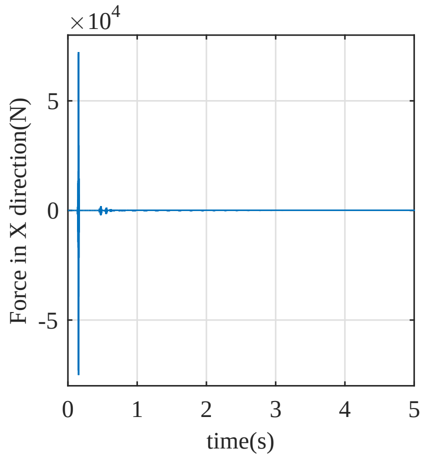


(f)

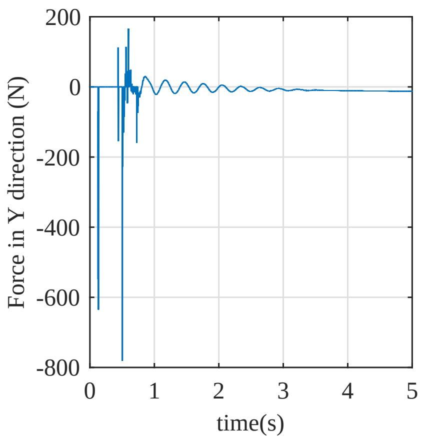
FIGURE 5.11: (a) Contact Force of left back leg in x direction for exact contact model, (b) Contact Force of left back leg in x direction for penalty based model, (c) Contact Force of left back leg in y direction for exact contact model, (d) Contact Force of left back leg in y direction for penalty based model, (e) Contact Force of left back leg in z direction for exact contact model, (f) Contact Force of left back leg in z direction for penalty based model



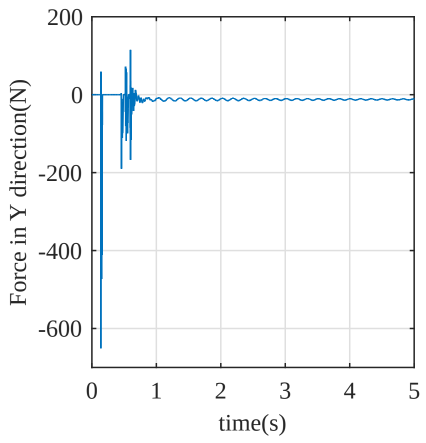
(a)



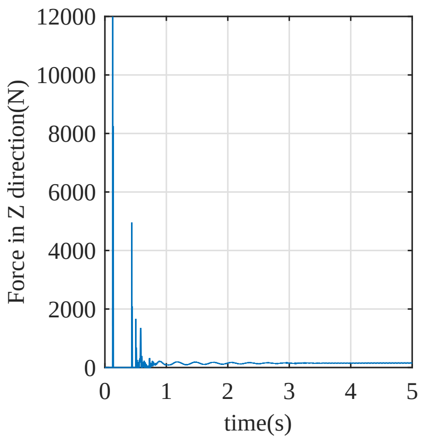
(b)



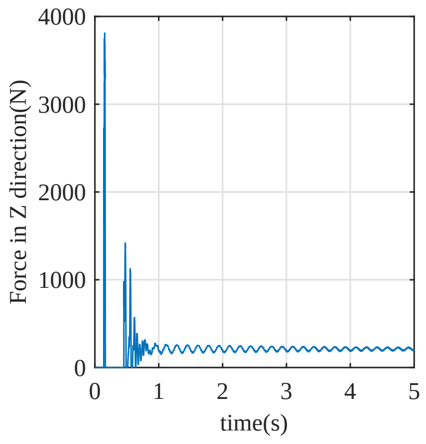
(c)



(d)

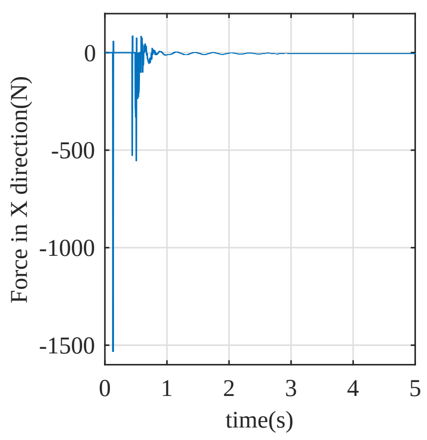


(e)

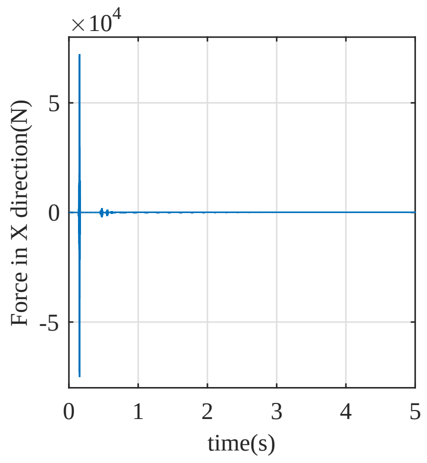


(f)

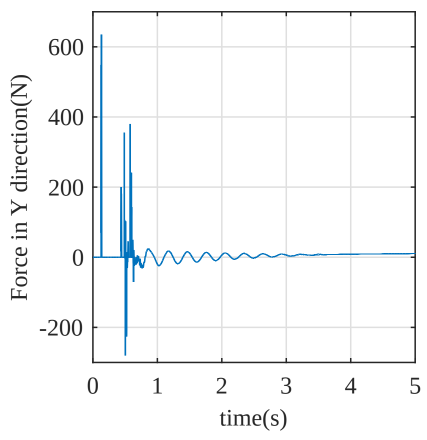
FIGURE 5.12: (a) Contact Force of right front leg in x direction for exact contact model, (b) Contact Force of right front leg in x direction for penalty based model, (c) Contact Force of right front leg in y direction for exact contact model, (d) Contact Force of right front leg in y direction for penalty based model, (e) Contact Force of right front leg in z direction for exact contact model, (f) Contact Force of right front leg in z direction for penalty based model



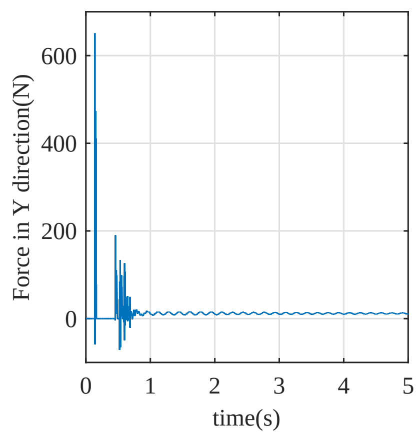
(a)



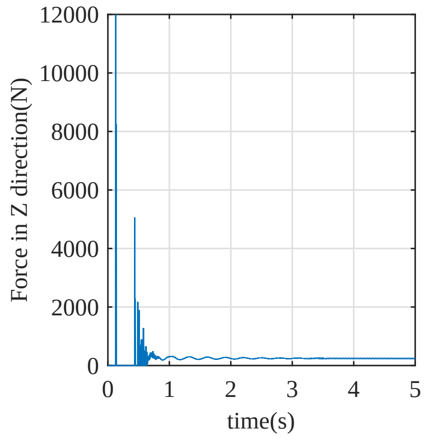
(b)



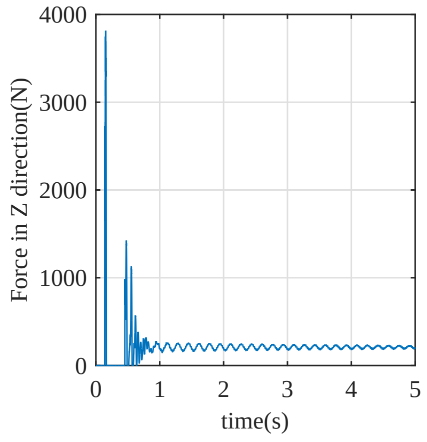
(c)



(d)



(e)



(f)

FIGURE 5.13: (a) Contact Force of left front leg in x direction for exact contact model, (b) Contact Force of left front leg in x direction for penalty based model, (c) Contact Force of left front leg in y direction for exact contact model, (d) Contact Force of left front leg in y direction for penalty based model, (e) Contact Force of left front leg in z direction for exact contact model, (f) Contact Force of left front leg in z direction for penalty based model

5.4.3 Simulation of the Fall of the Quadruped Robot From a 0.5 meter Height

In the previous simulation, robot falls from 0.1 meter and the simulations are successful in both penalty-based and exact contact models. In this simulation, the models are tested by increasing the height to 0.5 meter.

In the penalty-based simulation, collision and contact forces reach unrealistic values as shown in Figures 5.16, 5.17, 5.18 and 5.19. When dropped from this height, there are high amounts of ground penetration, which, as a result cause these high forces. The robot flies to very high altitude as shown in Fig. 5.14. This is the main drawback of penalty-based model and it is observed in this simulation.

In contrast to the penalty-based method, the exact contact model is successful in the simulation. As shown in Fig. 5.14e, when the robot falls from 0.5 meter and when it collides with the ground, the contact forces in the z direction reach high but reasonable values. After the fall, the robot bounces of the ground, turns upside down and falls again. This is observed in Fig. 5.15. As mentioned above, the robot model has contact points only at the tips of its legs. For this reason, the body of the robot does not contact the ground and goes to -0.6 meter in the z direction. Energy of the fall is absorbed by the collisions between the tip of the legs and ground. Magnitudes of collision forces decline steadily as shown in Figures 5.16, 5.17,5.18 and 5.19. Moreover, the body position components in x and y directions converge to constant values due to friction forces.

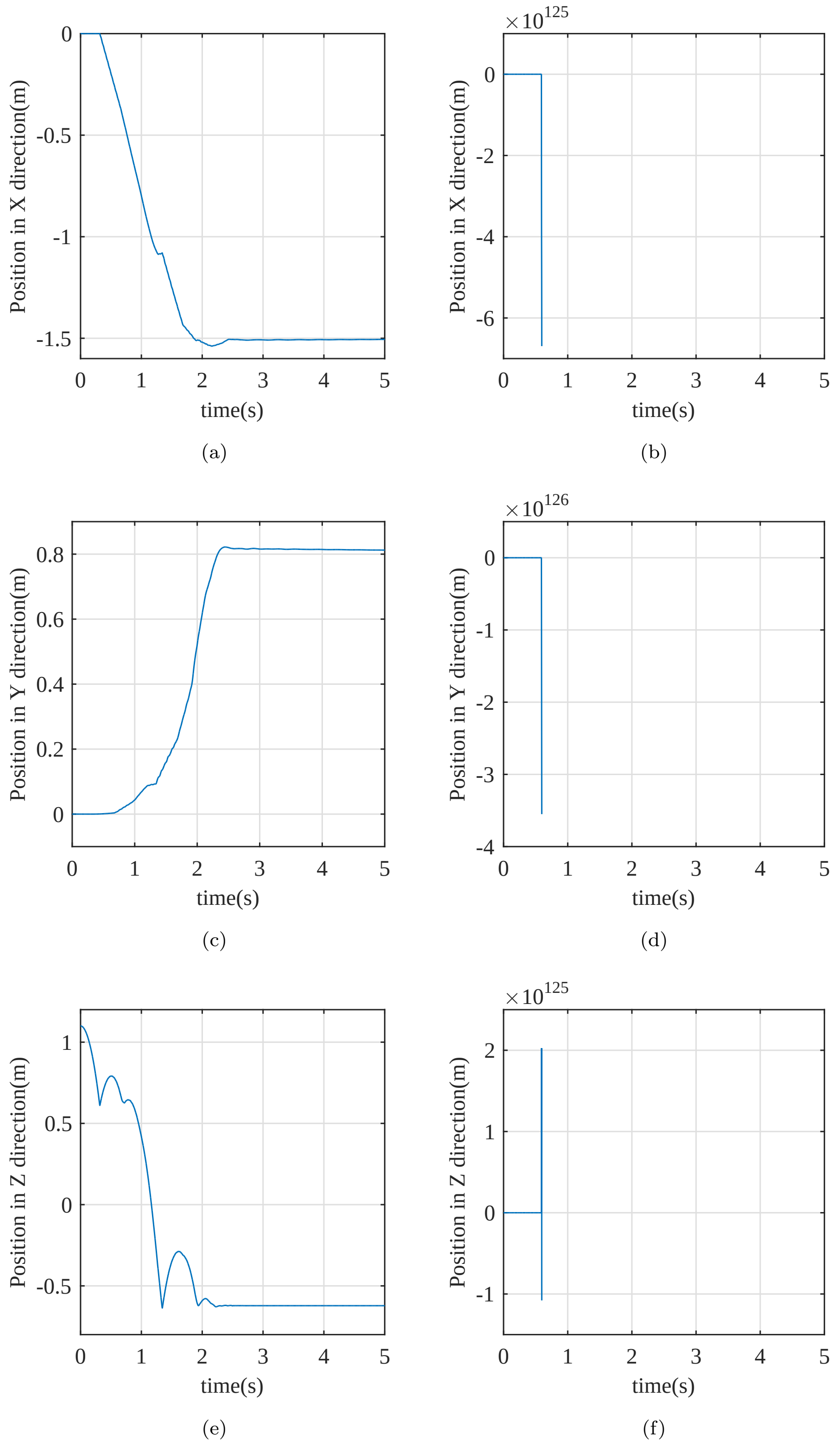


FIGURE 5.14: (a) position of body in x direction for exact contact model, (b) Position of body in x direction for penalty based model, (c) Position of body in y direction for exact contact model, (d) Position of body in y direction for penalty based model, (e) Position of body in z direction for exact contact model, (f) Position of body in z direction for penalty based model

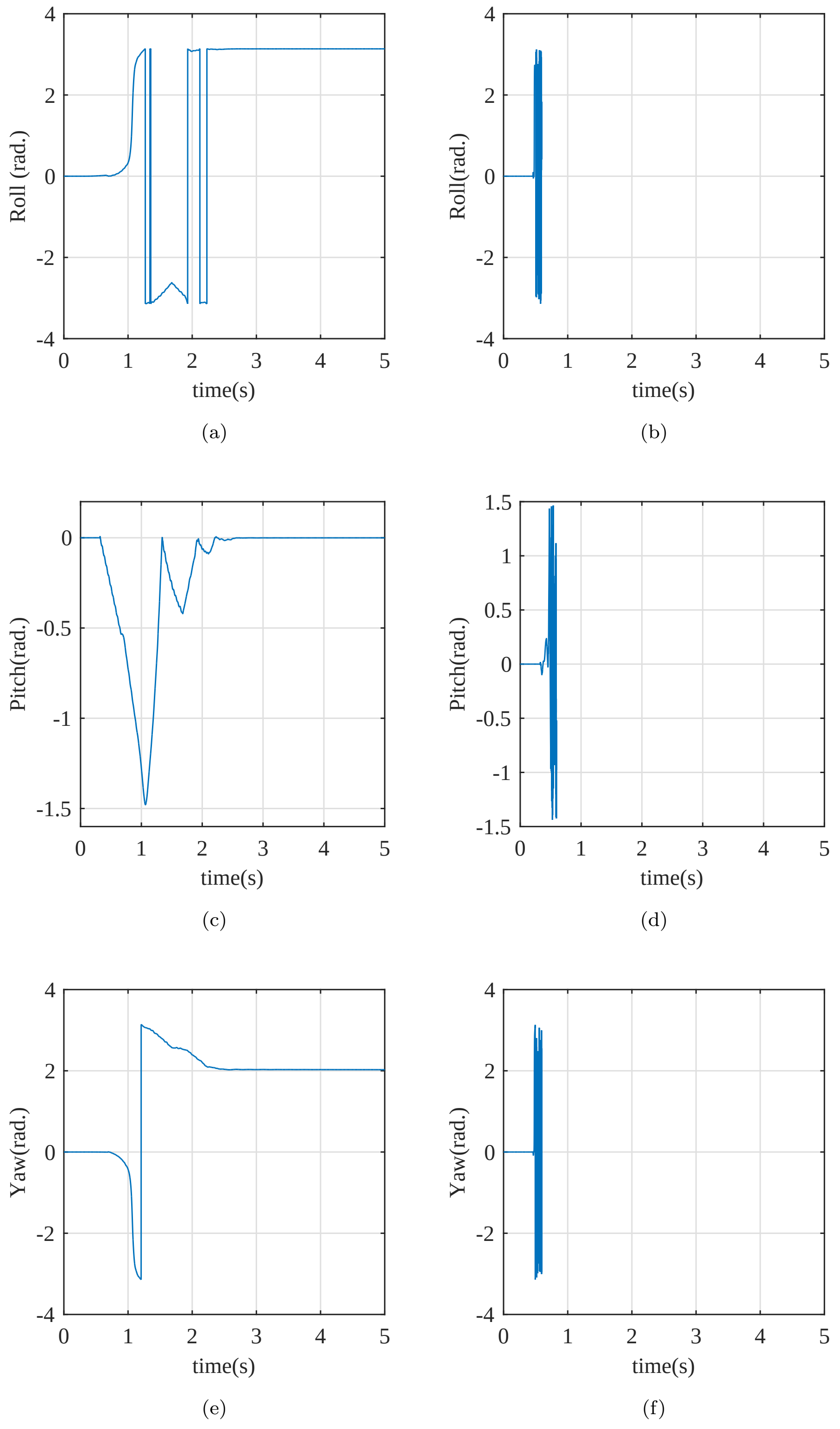


FIGURE 5.15: (a) Roll of body for exact contact model, (b) Roll of body for penalty based model, (c) Pitch of body for exact contact model, (d) Pitch of body for penalty based model, (e) Yaw of body for exact contact model, (f) Yaw of body for penalty based model

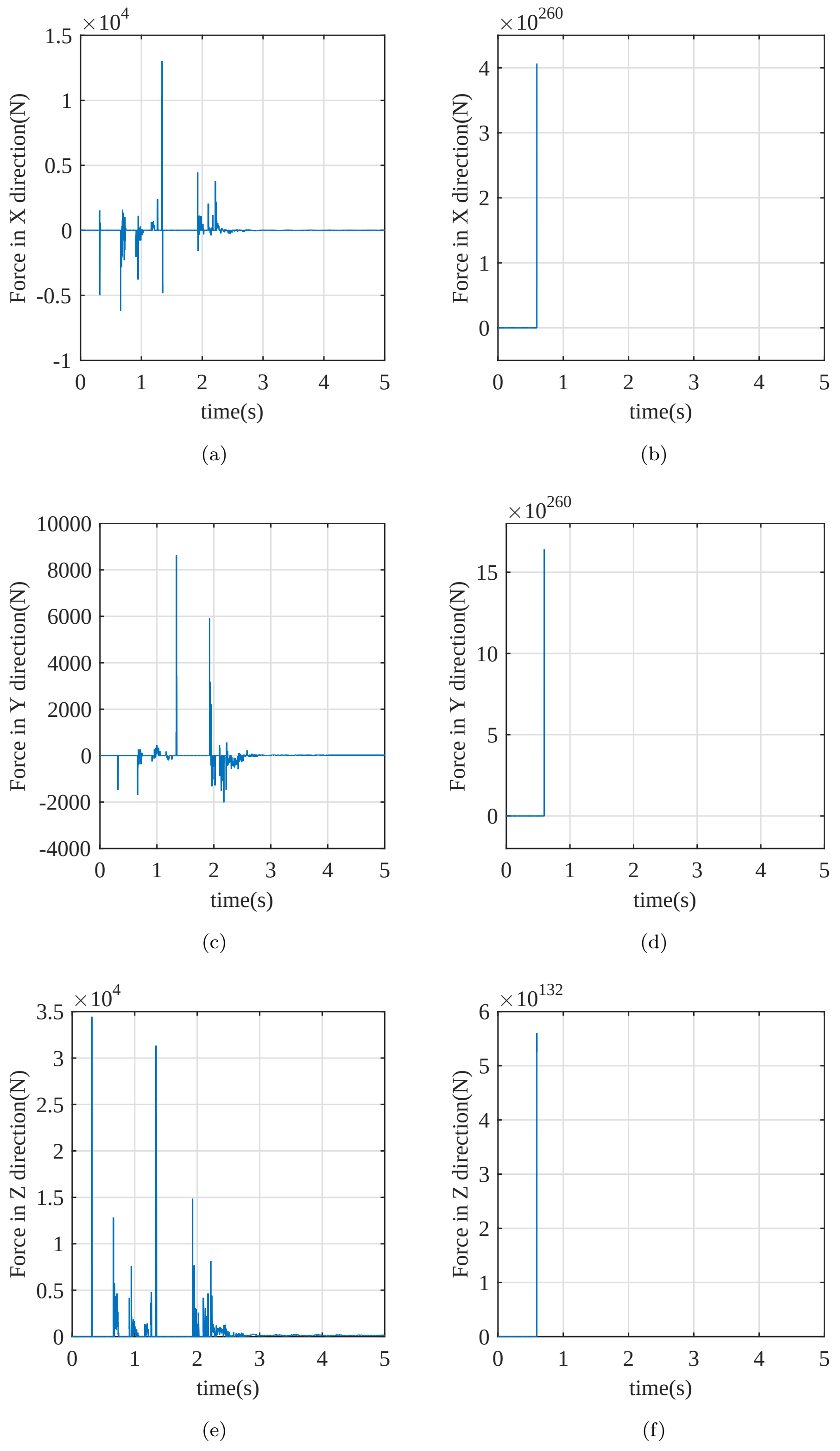
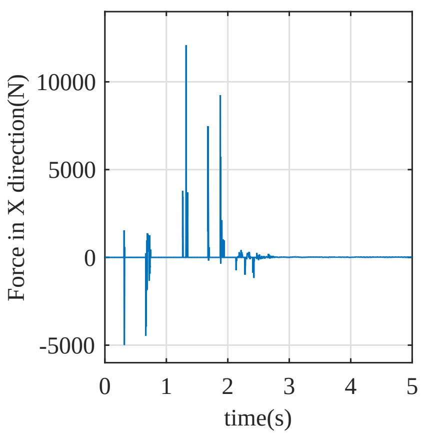
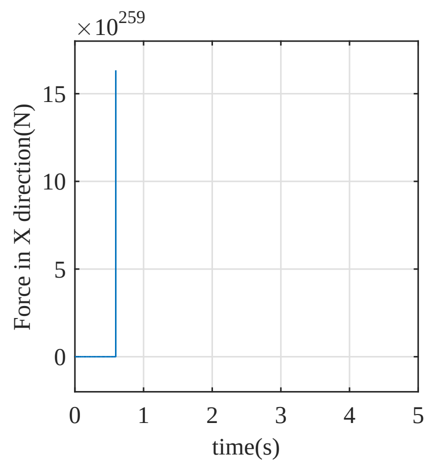


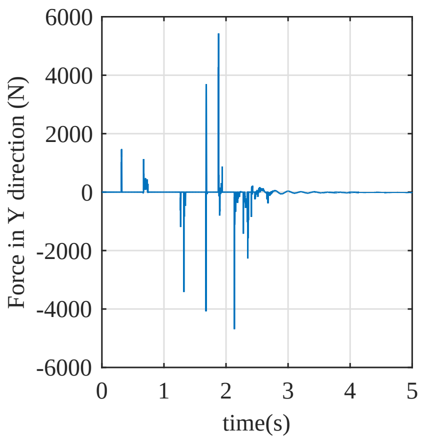
FIGURE 5.16: (a) Contact Force of right back leg in x direction for exact contact model, (b) Contact Force of right back leg in x direction for penalty based model, (c) Contact Force of right back leg in y direction for exact contact model, (d) Contact Force of right back leg in y direction for penalty based model, (e) Contact Force of right back leg in z direction for exact contact model, (f) Contact Force of right back leg in z direction for penalty based model



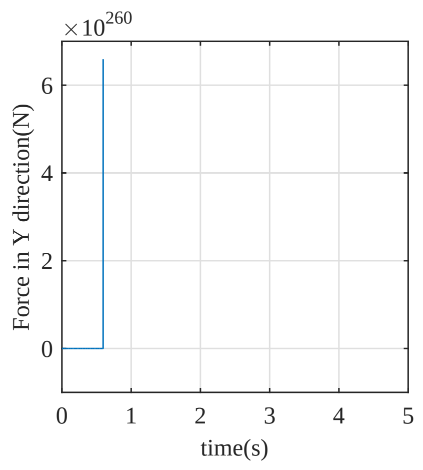
(a)



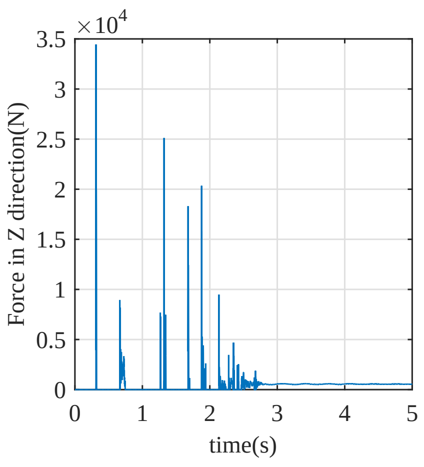
(b)



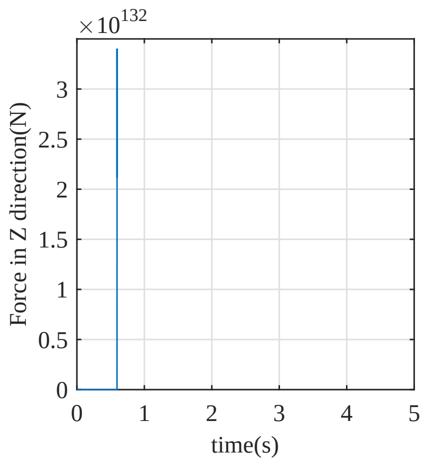
(c)



(d)

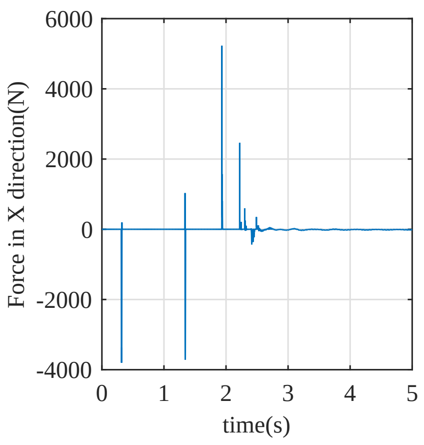


(e)

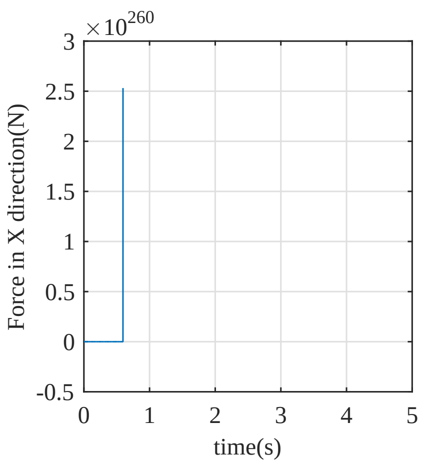


(f)

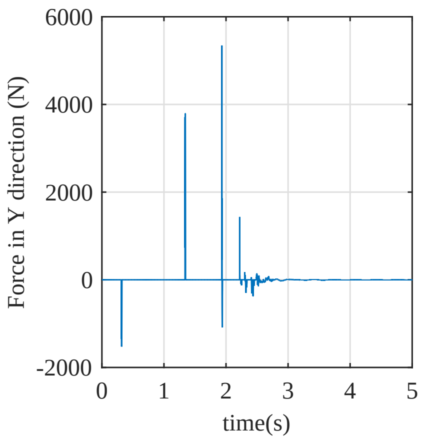
FIGURE 5.17: (a) Contact Force of left back leg in x direction for exact contact model, (b) Contact Force of left back leg in x direction for penalty based model, (c) Contact Force of left back leg in y direction for exact contact model, (d) Contact Force of left back leg in y direction for penalty based model, (e) Contact Force of left back leg in z direction for exact contact model, (f) Contact Force of left back leg in z direction for penalty based model



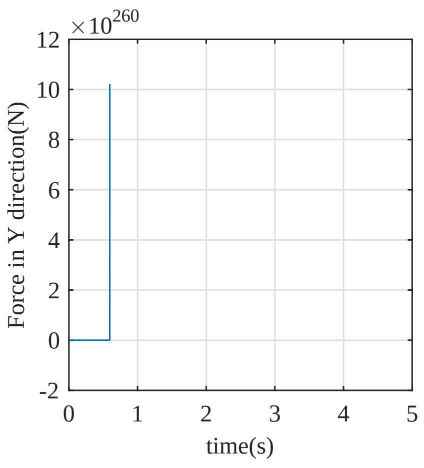
(a)



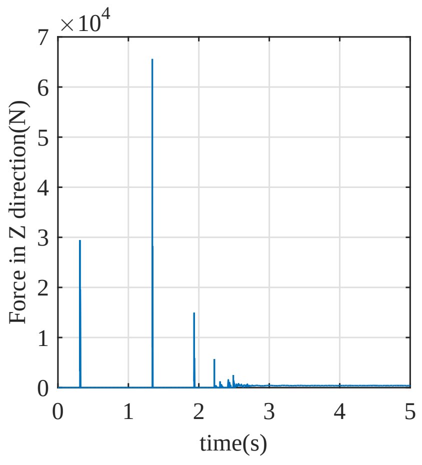
(b)



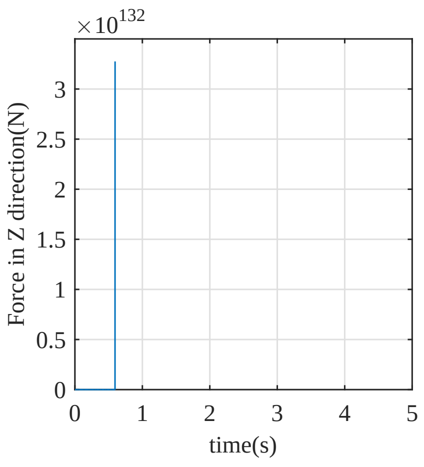
(c)



(d)

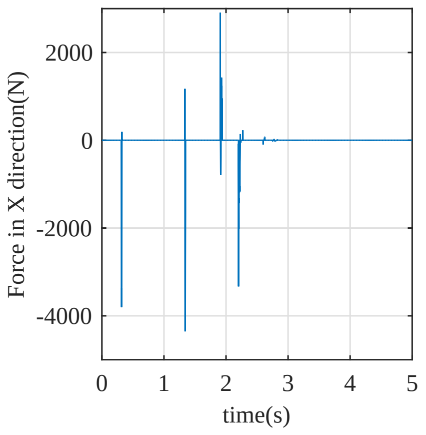


(e)

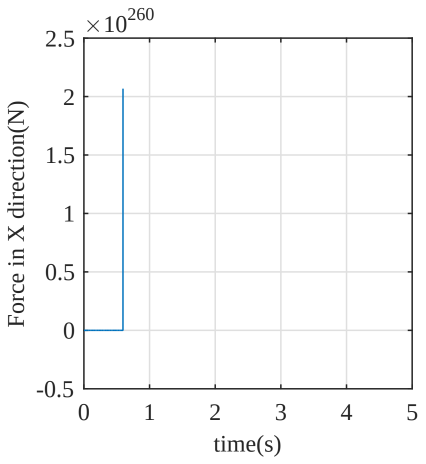


(f)

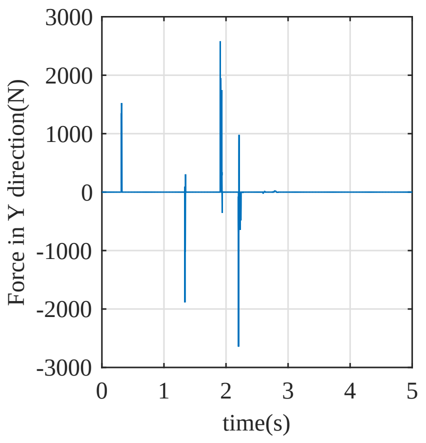
FIGURE 5.18: (a) Contact Force of right front leg in x direction for exact contact model, (b) Contact Force of right front leg in x direction for penalty based model, (c) Contact Force of right front leg in y direction for exact contact model, (d) Contact Force of right front leg in y direction for penalty based model, (e) Contact Force of right front leg in z direction for exact contact model, (f) Contact Force of right front leg in z direction for penalty based model



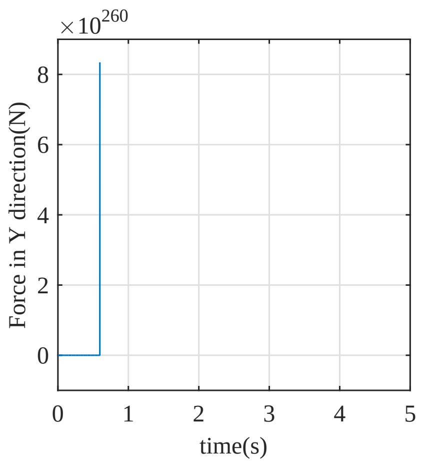
(a)



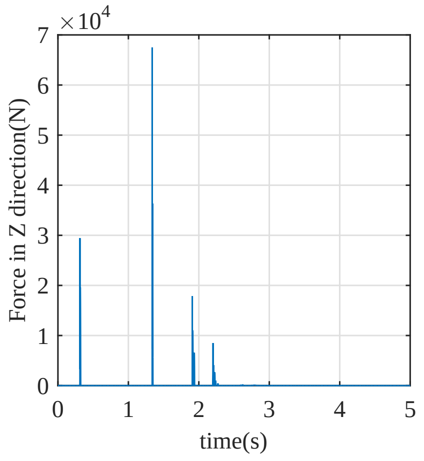
(b)



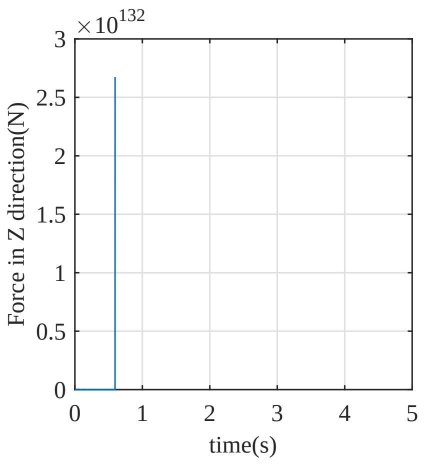
(c)



(d)



(e)



(f)

FIGURE 5.19: (a) Contact Force of left front leg in x direction for exact contact model, (b) Contact Force of left front leg in x direction for penalty based model, (c) Contact Force of left front leg in y direction for exact contact model, (d) Contact Force of left front leg in y direction for penalty based model, (e) Contact Force of left front leg in z direction for exact contact model, (f) Contact Force of left front leg in z direction for penalty based model

5.4.4 Simulation of the Fall of the Quadruped Robot From a 1.5 meter Height

The falling simulation is carried out again with an increasing height of 1.5 meters. The penalty-based method failed in the previous simulation. Therefore, it fails as expected in this simulation too. According to the results, the ground forces go to extremely high values with the penalty-based method. The exact contact model was successful in this case too. The same behaviour as in the previous simulation is observed. The robot falls from 1.5 meter, collides with the ground, bounces of the ground, and falls upside down. By consecutive collisions, the potential energy is absorbed and the robot settles on the ground. The results of this simulation with the two models are presented in Figures 5.20 - 5.25

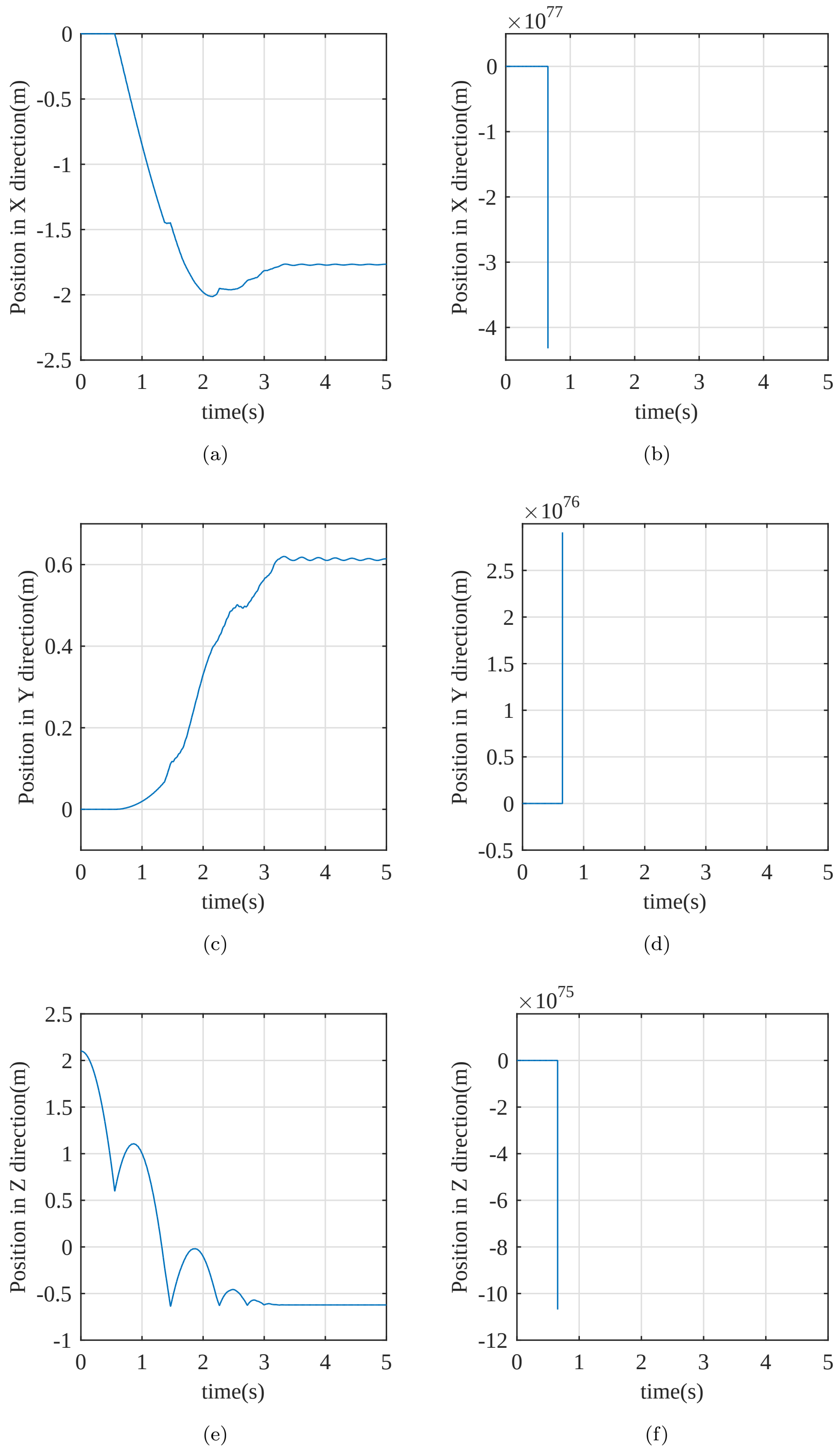


FIGURE 5.20: (a) position of body in x direction for exact contact model, (b) Position of body in x direction for penalty based model, (c) Position of body in y direction for exact contact model, (d) Position of body in y direction for penalty based model, (e) Position of body in z direction for exact contact model, (f) Position of body in z direction for penalty based model

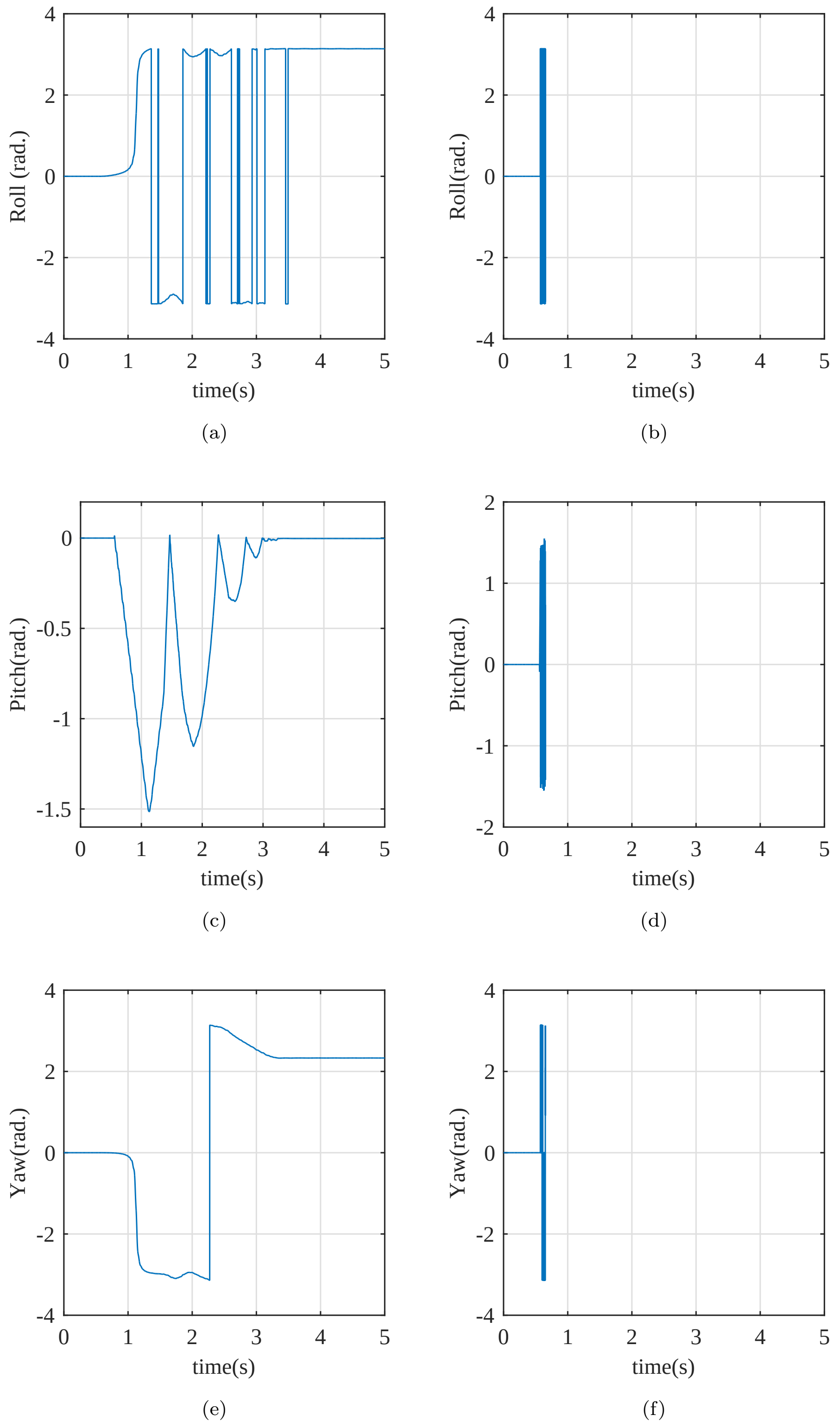


FIGURE 5.21: (a) Roll of body for exact contact model, (b) Roll of body for penalty based model, (c) Pitch of body for exact contact model, (d) Pitch of body for penalty based model, (e) Yaw of body for exact contact model, (f) Yaw of body for penalty based model

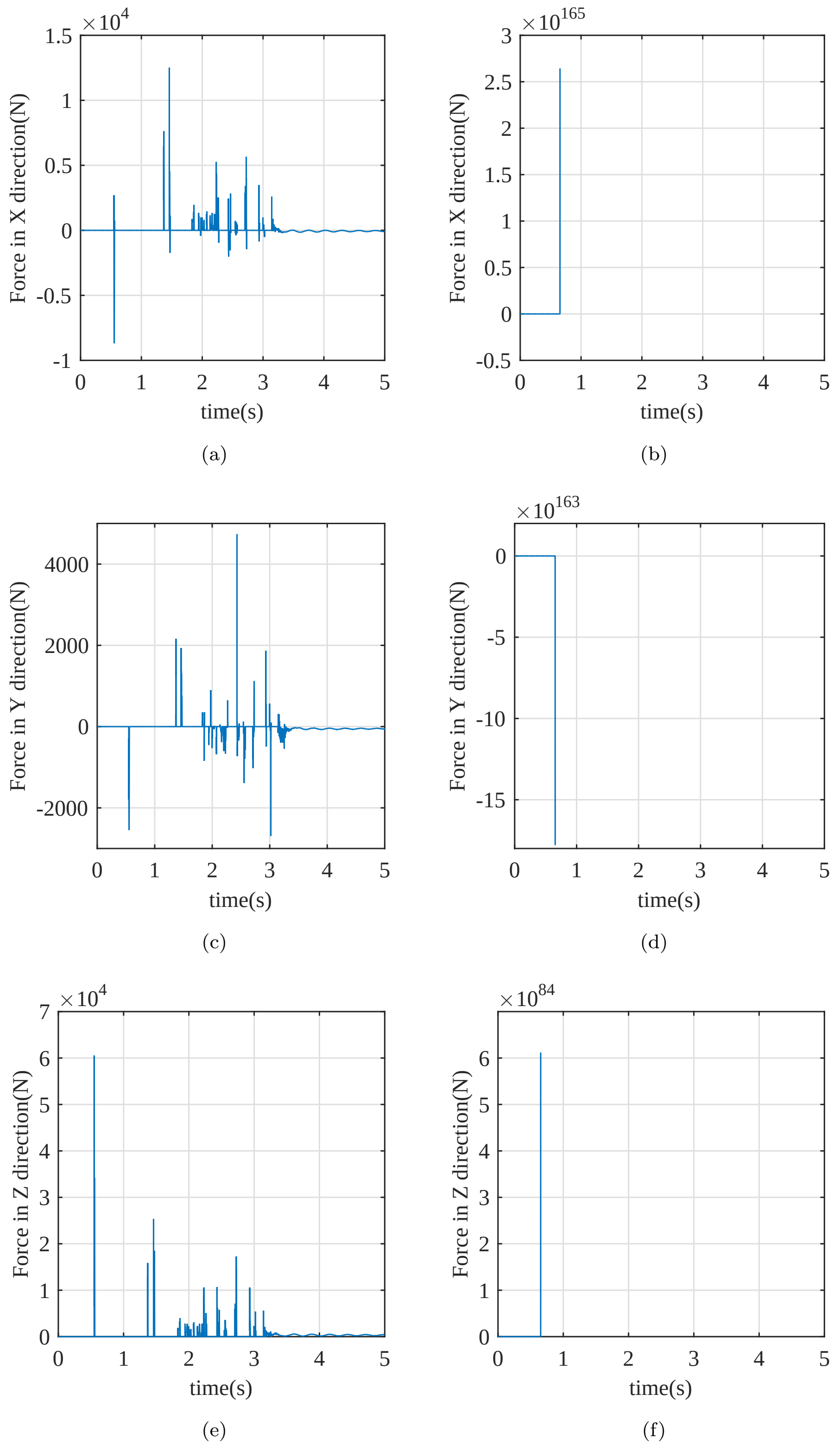


FIGURE 5.22: (a) Contact Force of right back leg in x direction for exact contact model, (b) Contact Force of right back leg in x direction for penalty based model, (c) Contact Force of right back leg in y direction for exact contact model, (d) Contact Force of right back leg in y direction for penalty based model, (e) Contact Force of right back leg in z direction for exact contact model, (f) Contact Force of right back leg in z direction for penalty based model

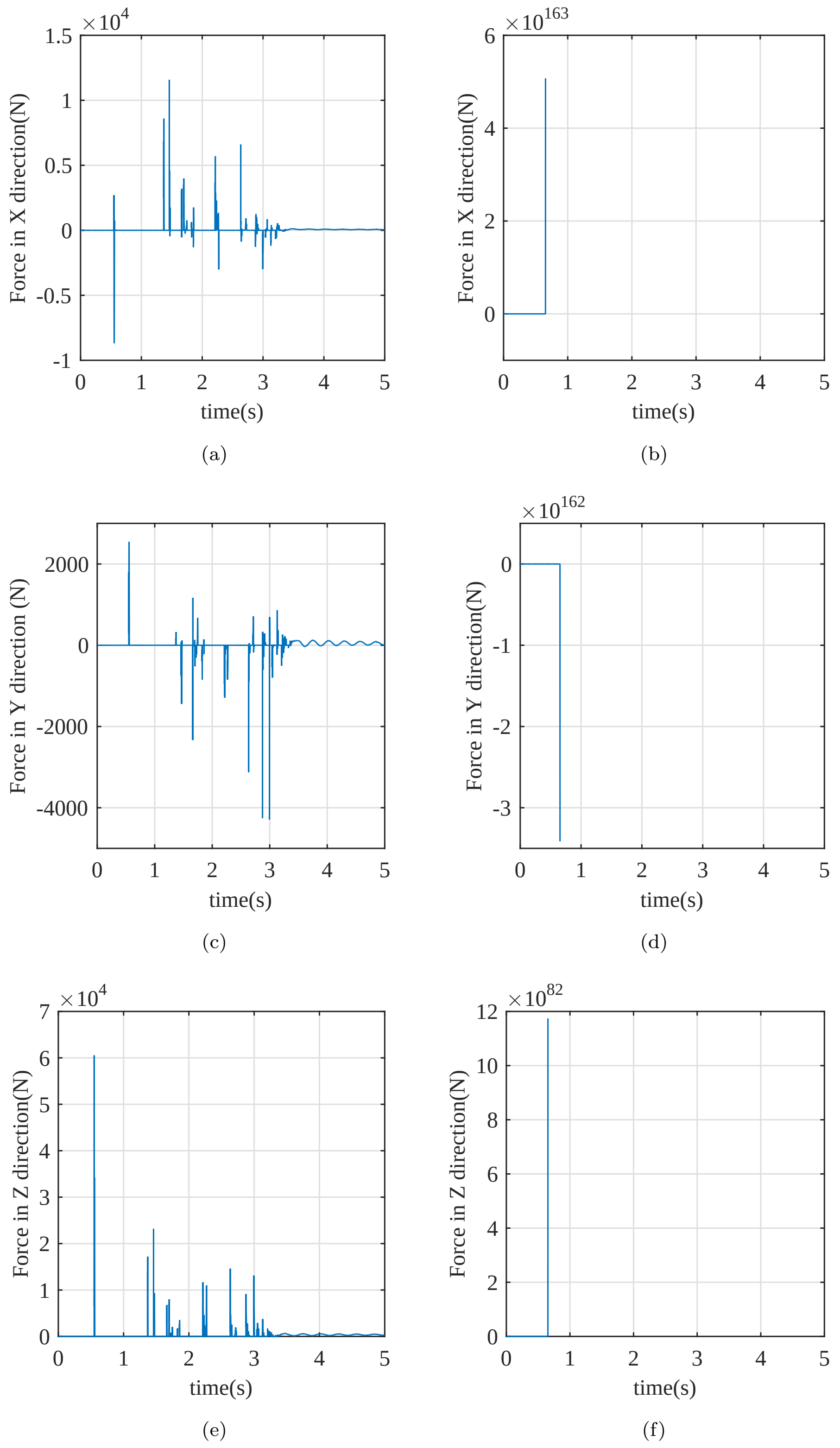


FIGURE 5.23: (a) Contact Force of left back leg in x direction for exact contact model, (b) Contact Force of left back leg in x direction for penalty based model, (c) Contact Force of left back leg in y direction for exact contact model, (d) Contact Force of left back leg in y direction for penalty based model, (e) Contact Force of left back leg in z direction for exact contact model, (f) Contact Force of left back leg in z direction for penalty based model

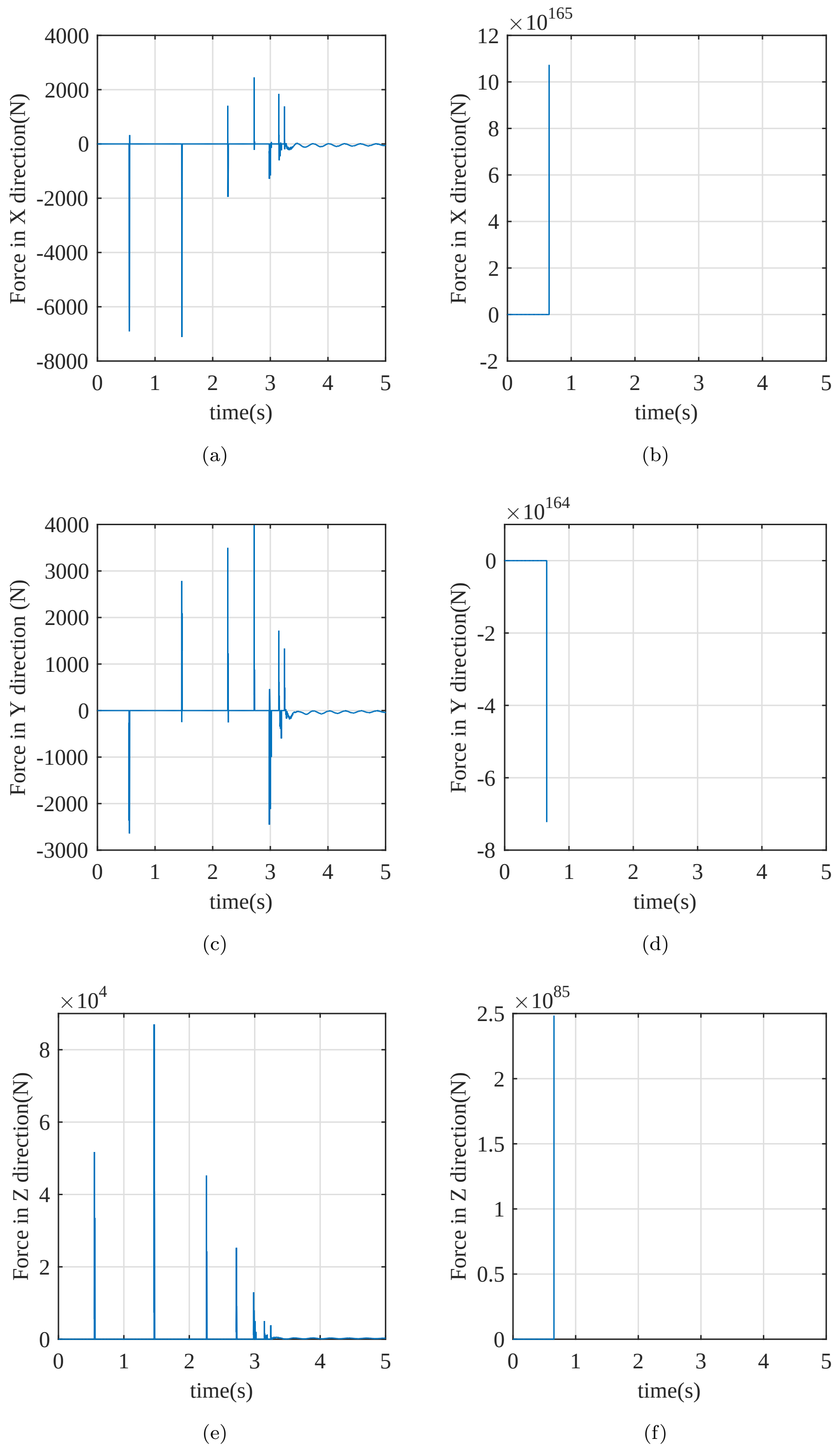


FIGURE 5.24: (a) Contact Force of right front leg in x direction for exact contact model, (b) Contact Force of right front leg in x direction for penalty based model, (c) Contact Force of right front leg in y direction for exact contact model, (d) Contact Force of right front leg in y direction for penalty based model, (e) Contact Force of right front leg in z direction for exact contact model, (f) Contact Force of right front leg in z direction for penalty based model

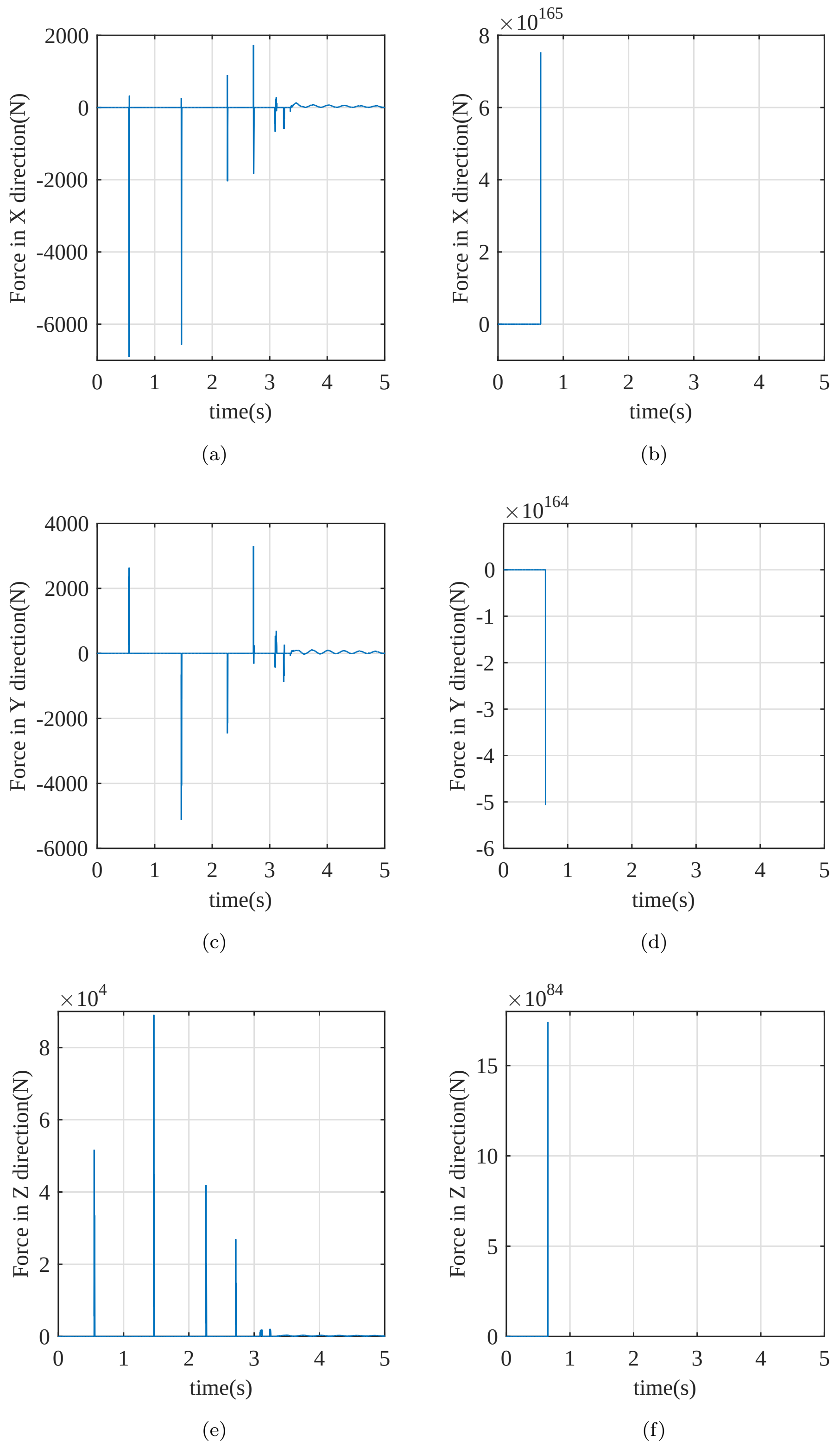


FIGURE 5.25: (a) Contact Force of left front leg in x direction for exact contact model, (b) Contact Force of left front leg in x direction for penalty based model, (c) Contact Force of left front leg in y direction for exact contact model, (d) Contact Force of left front leg in y direction for penalty based model, (e) Contact Force of left front leg in z direction for exact contact model, (f) Contact Force of left front leg in z direction for penalty based model

5.4.5 Simulation of the Jumping Motion of the Quadruped Robot

In our research, we are interested in a quadruped robot that is capable of jumping to certain heights. The two contact methods are used in simulations to see if they handle jumping differently than they did with falling. The jumping reference generated first prompt the quadruped to crouch and then to jump upwards by extending its legs. The change of positions and forces are tracked just like in the previous simulations.

Results for the penalty-based contact model are similiar to its high altitude fall counterparts. As the quadruped is extending its legs to jump, there is a sudden increase in the ground penetration which results in extremely high contact forces. This can be observed in Figures 5.28f, 5.29f, 5.30f and 5.31f. In the corresponding figures for the exact contact model, realistic forces are observed. As shown in Fig. 5.26e, with the exact model, the robot jumps and reaches to 2.2 meters.

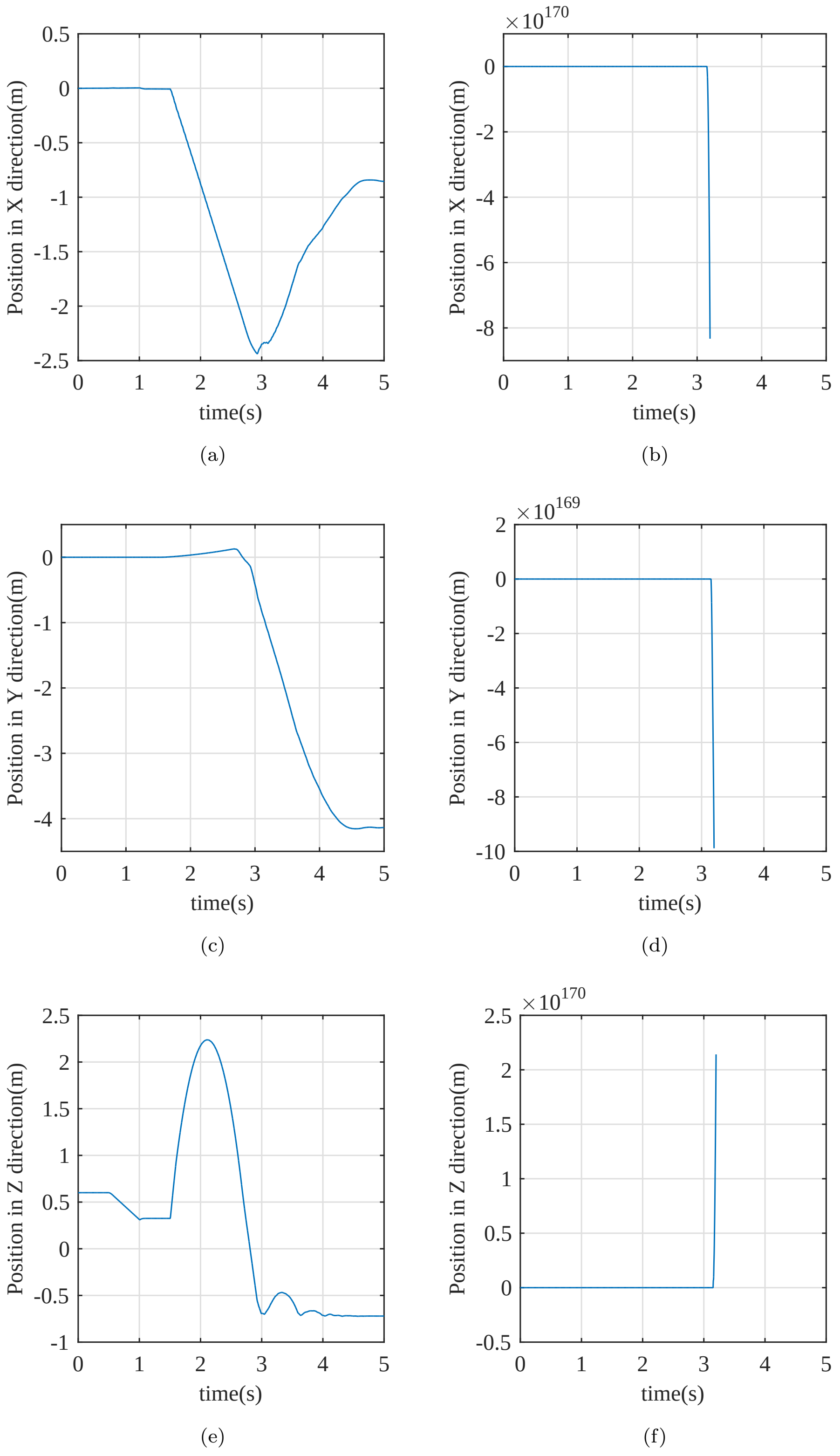


FIGURE 5.26: (a) position of body in x direction for exact contact model, (b) Position of body in x direction for penalty based model, (c) Position of body in y direction for exact contact model, (d) Position of body in y direction for penalty based model, (e) Position of body in z direction for exact contact model, (f) Position of body in z direction for penalty based model

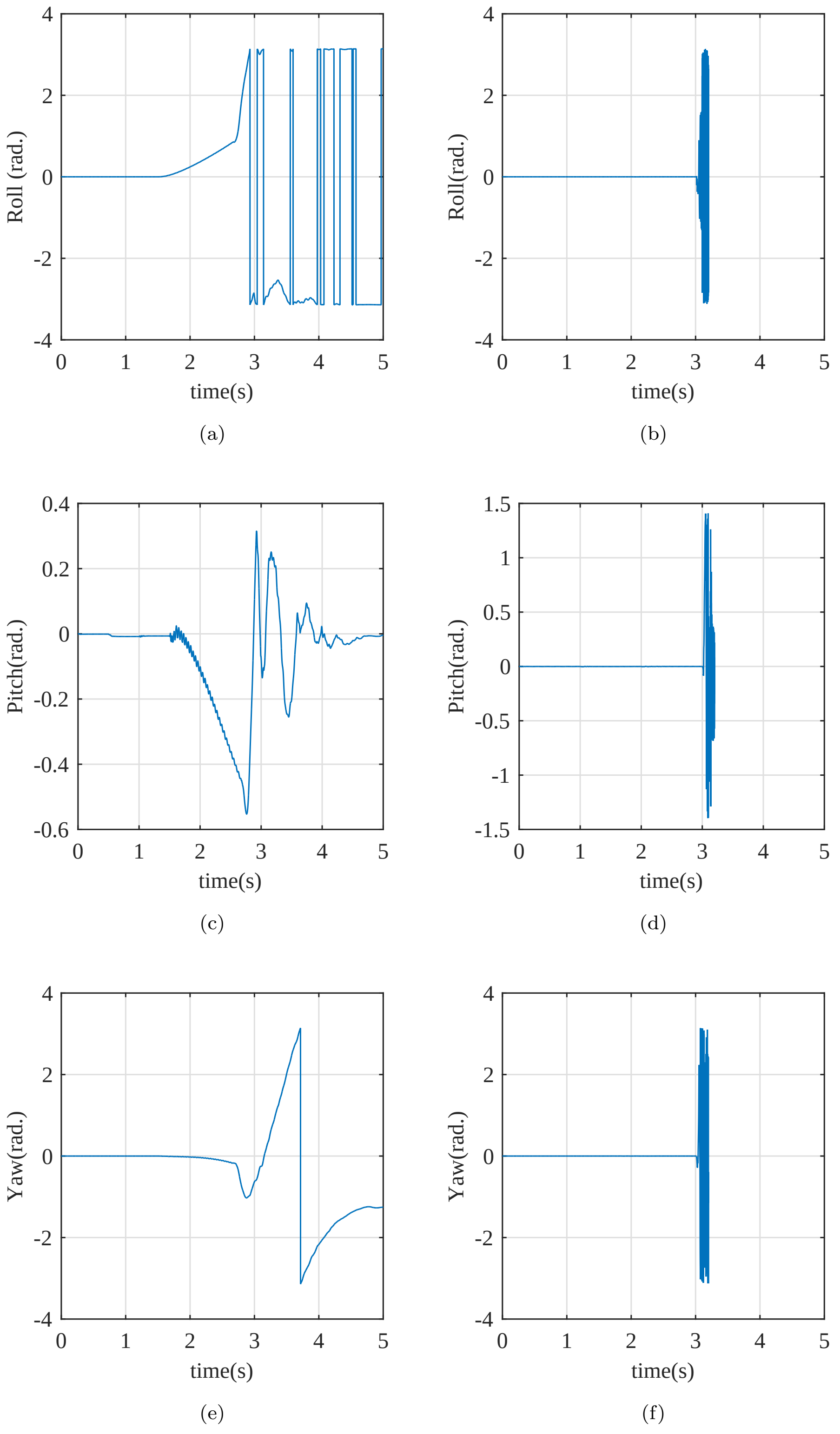


FIGURE 5.27: (a) Roll of body for exact contact model, (b) Roll of body for penalty based model, (c) Pitch of body for exact contact model, (d) Pitch of body for penalty based model, (e) Yaw of body for exact contact model, (f) Yaw of body for penalty based model

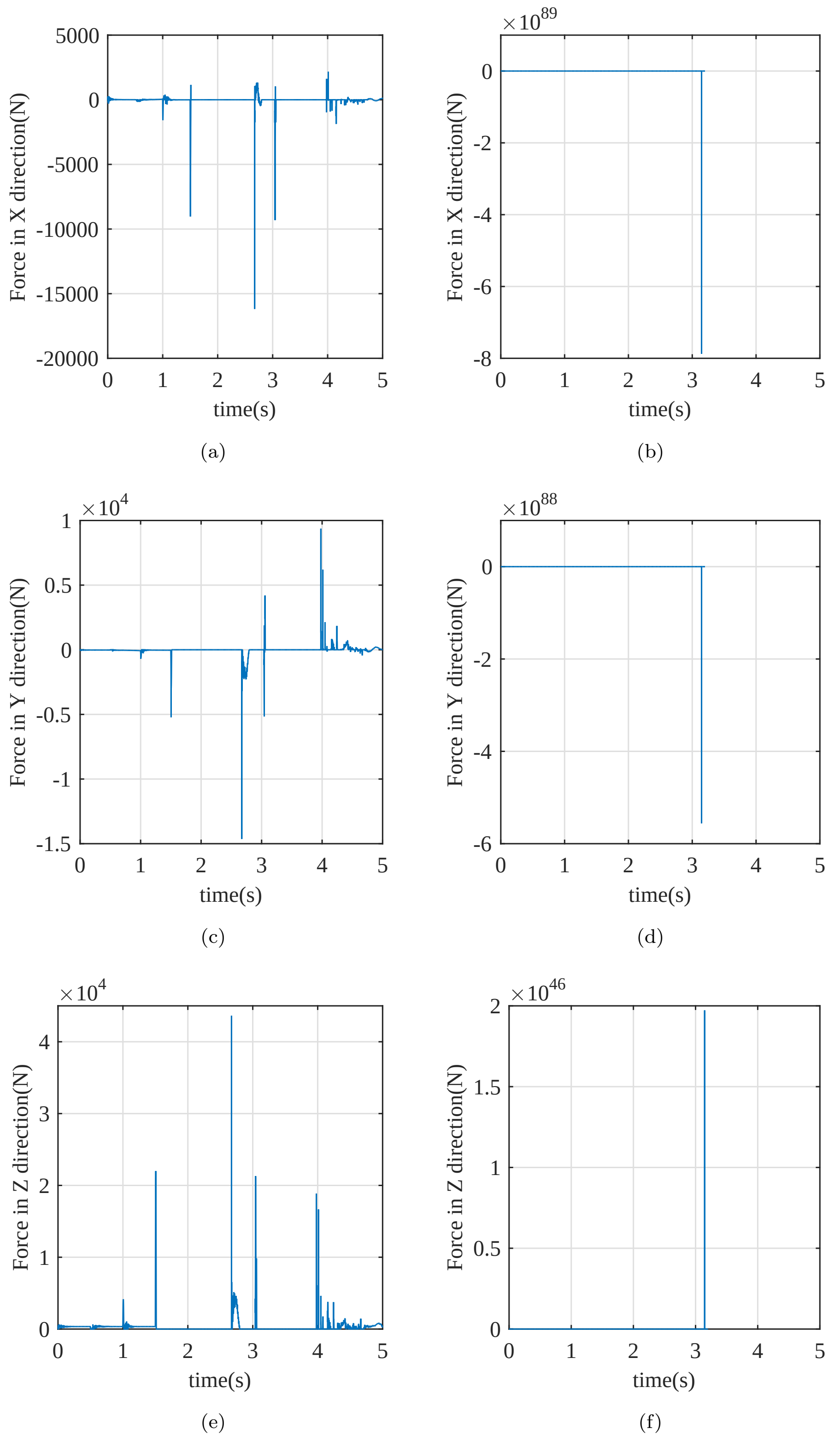


FIGURE 5.28: (a) Contact Force of right back leg in x direction for exact contact model, (b) Contact Force of right back leg in x direction for penalty based model, (c) Contact Force of right back leg in y direction for exact contact model, (d) Contact Force of right back leg in y direction for penalty based model, (e) Contact Force of right back leg in z direction for exact contact model, (f) Contact Force of right back leg in z direction for penalty based model

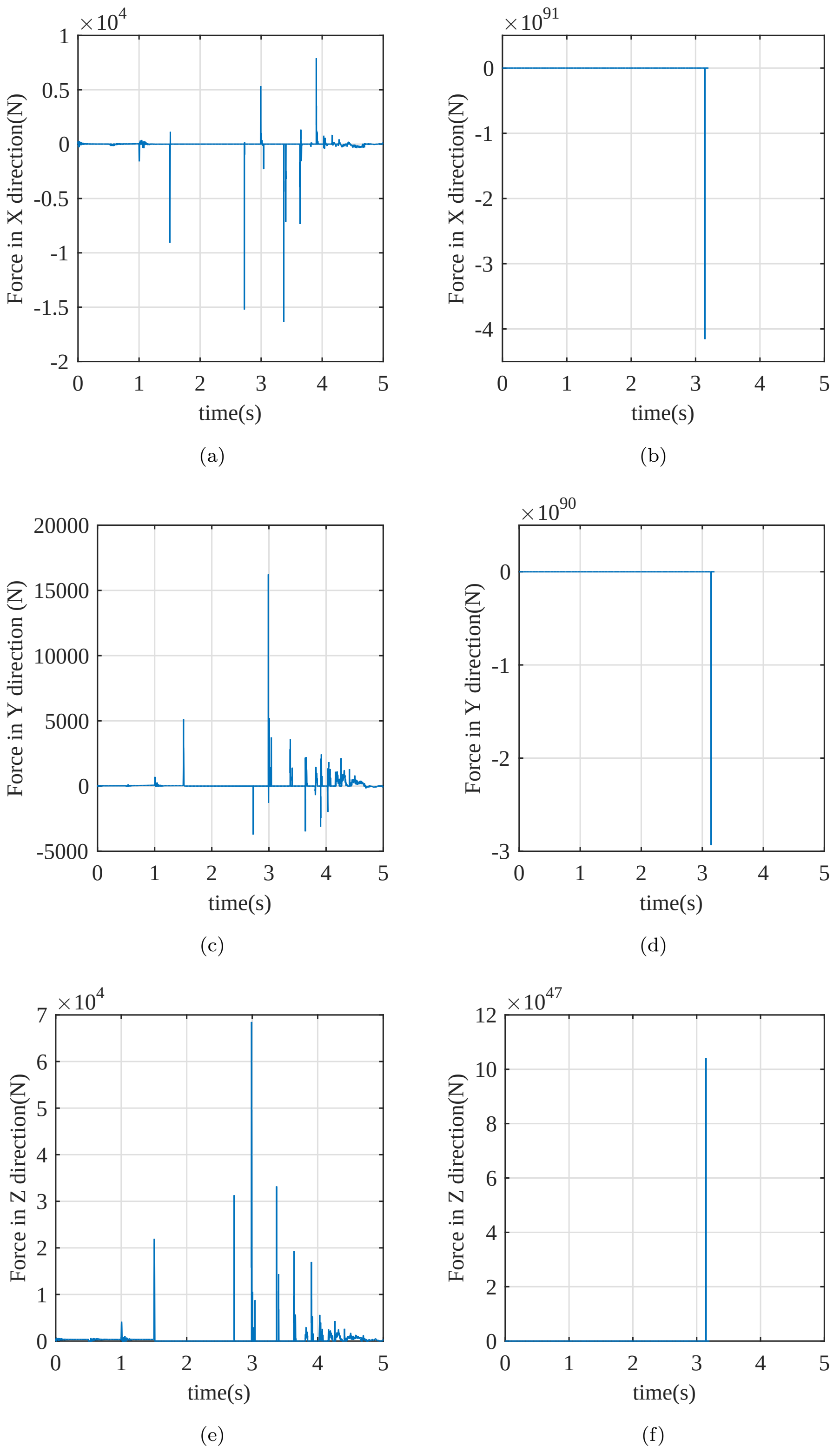
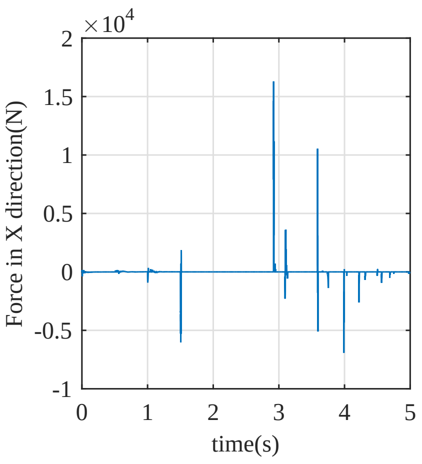
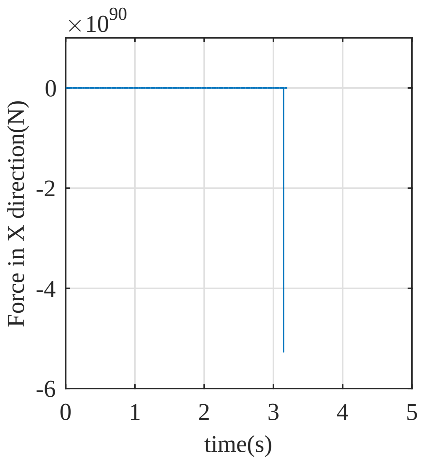


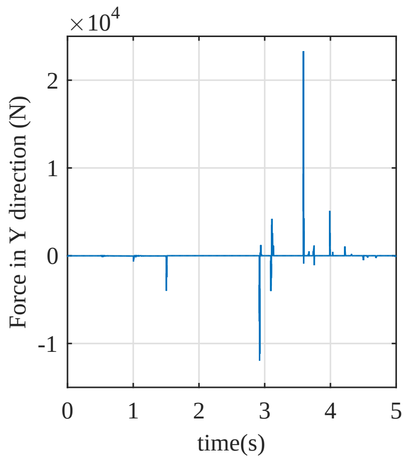
FIGURE 5.29: (a) Contact Force of left back leg in x direction for exact contact model, (b) Contact Force of left back leg in x direction for penalty based model, (c) Contact Force of left back leg in y direction for exact contact model, (d) Contact Force of left back leg in y direction for penalty based model, (e) Contact Force of left back leg in z direction for exact contact model, (f) Contact Force of left back leg in z direction for penalty based model



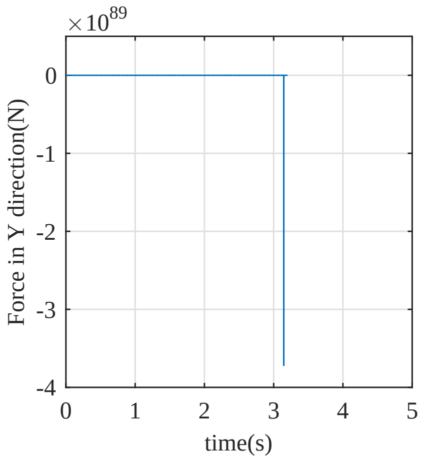
(a)



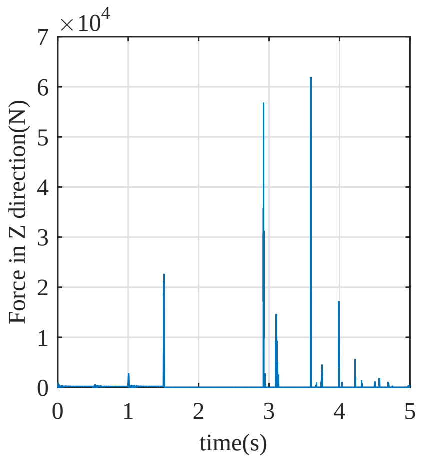
(b)



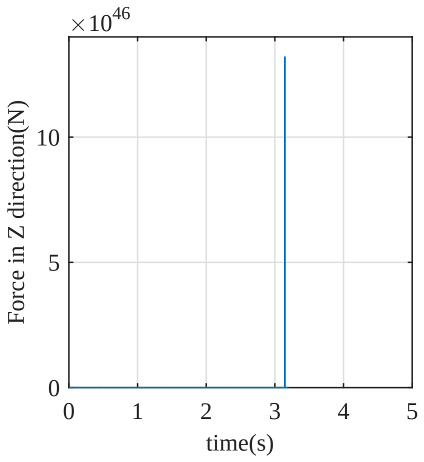
(c)



(d)



(e)



(f)

FIGURE 5.30: (a) Contact Force of right front leg in x direction for exact contact model, (b) Contact Force of right front leg in x direction for penalty based model, (c) Contact Force of right front leg in y direction for exact contact model, (d) Contact Force of right front leg in y direction for penalty based model, (e) Contact Force of right front leg in z direction for exact contact model, (f) Contact Force of right front leg in z direction for penalty based model

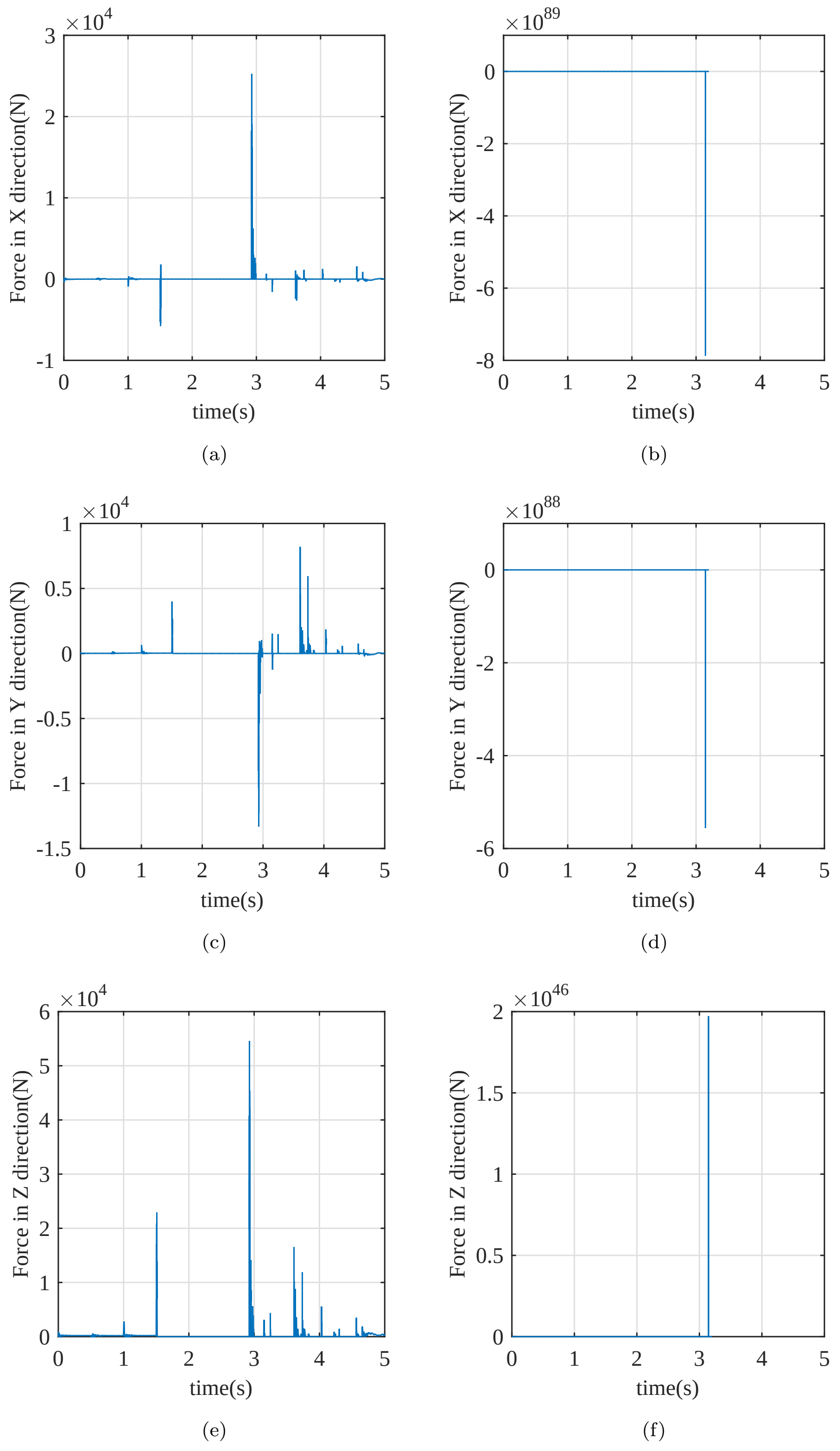


FIGURE 5.31: (a) Contact Force of left front leg in x direction for exact contact model, (b) Contact Force of left front leg in x direction for penalty based model, (c) Contact Force of left front leg in y direction for exact contact model, (d) Contact Force of left front leg in y direction for penalty based model, (e) Contact Force of left front leg in z direction for exact contact model, (f) Contact Force of left front leg in z direction for penalty based model

Chapter 6

Conclusion

In this thesis, an exact contact model for legged robot dynamics simulations is implemented.

Widely-used dynamics model derivation techniques for simulation are the Newton Euler algorithm and the Euler Lagrange formulation. These approaches require computational steps involving the inversion of the inertia matrix during the simulation. In contrast, the ABM does not require this inversion. The ABM is employed in this thesis in order to avoid computational complexity.

Constraints which are used by the exact contact model to obtain contact forces between the robot and ground were employed. Additional constraints related to collision dynamics are used too. These constraints are defined as a LCP. To solve the LCP, Lemke's algorithm and the projected Gauss Seidel algorithm are used. Advantages and disadvantages of the algorithms are discussed. A comparison is made between exact contact method and penalty based contact modeling algorithms.

Simulation studies are carried out for a quadruped robot. Two different simulation models with identical dynamics parameters are employed for the purpose of comparison. One of them uses the exact contact modeling algorithm equipped with the ABM, and the other one employs spring-damper based penalty contact with the Newton-Euler algorithm. In the simulation scenarios, the robot trots, falls from varius heights and jumps. These acts are defined as highly dynamic movements.

Simulation results show that exact contact model satisfies the expectations. In other words, the model gives realistic results for all conditions. However, the penalty-based model failed for most of the fall down and jumping scenarios. The main reason of the failure is that, in penalty based algorithm, the resulting contact force becomes massive when the penetration into the ground is deep. Landing and take-off are dynamic motion phases in which unrealistically deep interpenetrations can occur due to the finite simulation integration frequency.

The exact contact model has significant advantages in comparison to the penalty based model and it has superior performance for high dynamic motion in simulation environment.

The resulting contact model and ABM based integration technique will be used as a building block of the simulation environment in the TUBITAK 114E618 Project Quadruped Robot Design, Construction and Control.

Bibliography

[1] Tunc Akbas, S Emre Eskimez, Selim Ozel, O Kemal Adak, Kaan C Fidan, and Kemalettin Erbatur. Zero moment point based pace reference generation for quadruped robots via preview control. In *Advanced Motion Control (AMC), 2012 12th IEEE International Workshop on*, pages 1–7. IEEE, 2012.

[2] Hussein Aly, Tamer Abd Elmouty Elawady, et al. A new narrow phase collision detection algorithm using height projection. In *Education and Research Conference (EDERC), 2010 4th European*, pages 111–115. IEEE, 2010.

[3] Srikanth Bandi and Daniel Thalmann. An adaptive spatial subdivision of the object space for fast collision detection of animated rigid bodies. In *Computer Graphics Forum*, volume 14, pages 259–270. Wiley Online Library, 1995.

[4] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *ACM SIGGRAPH Computer Graphics*, volume 23, pages 223–232. ACM, 1989.

[5] David Baraff. Dynamic simulation of non-penetrating rigid bodies. Technical report, Cornell University, 1992.

[6] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34. ACM, 1994.

[7] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM, 1996.

[8] Robert F Battaglia. *Design of the SCOUT II Quadruped with Preliminary Stair-climbing*. PhD thesis, McGill University, 1999.

[9] Jan Bender. Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds*, 18(4-5):225–233, 2007.

[10] Jan Bender and Alfred A Schmitt. Fast dynamic simulation of multi-body systems using impulses. In *VRIPHYS*, pages 81–90, 2006.

[11] Gino van den Bergen. A fast and robust gjk implementation for collision detection of convex objects. *Journal of graphics tools*, 4(2):7–25, 1999.

[12] Thiago Boaventura. *Hydraulic Compliance Control of the Quadruped Robot HyQ*. PhD thesis, PhD Thesis, University of Genoa, Italy and Istituto Italiano di Tecnologia (IIT), 2013.

[13] Martin Buehler, R Battaglia, A Cocosco, Geoff Hawker, J Sarkis, and Kinya Yamazaki. Scout: A simple quadruped that walks, climbs, and runs. In *ICRA*, pages 1707–1712, 1998.

[14] J-R Chardonnet, Sylvain Miossec, Abderrahmane Kheddar, Hitoshi Arisumi, Hirohisa Hirukawa, François Pierrot, and Kazuhito Yokoi. Dynamic simulator for humanoids using constraint-based method with static friction. In *Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on*, pages 1366–1371. IEEE, 2006.

[15] Michael B Cline and Dinesh K Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3744–3751. IEEE, 2003.

[16] Evan Drumwright. A fast and stable penalty method for rigid body simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):231–240, 2008.

[17] Evan Drumwright, John Hsu, Nathan Koenig, and Dylan Shell. Extending open dynamics engine for robotics simulation. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 38–50. Springer, 2010.

[18] Evan Drumwright, Dylan Shell, et al. Extensive analysis of linear complementarity problem (lcp) solver performance on randomly generated rigid body contact problems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5034–5039. IEEE, 2012.

[19] Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):36–47, 2006.

[20] Boston Dynamics. Atlas - the agile anthropomorphic robot. `http://www.bostondynamics.com/robot_Atlas.html`, 2013.

[21] Boston Dynamics. Bigdog. `http://www.youtube.com/watch?v=cHJJQ0zNNOM`, 2014.

[22] Boston Dynamics. Cheetah - fastest legged robot. `http://www.bostondynamics.com/robot_cheetah.html`, 2014.

[23] Boston Dynamics. Introducing wildcat. `http://www.youtube.com/watch?v=wE3fmFTtP9g`, 2014.

[24] Boston Dynamics. Ls3 legged squad support system. `http://www.bostondynamics.com/robot_cheetah.html`, 2014.

[25] Boston Dynamics. Rhex - devours rough terrain. `http://www.bostondynamics.com/robot_rhex.html`, 2014.

[26] Gen Endo and Shigeo Hirose. Study on roller-walker-adaptation of characteristics of the propulsion by a leg trajectory. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1532–1537. IEEE, 2008.

[27] K Erbatur and A Kawamura. A new penalty based contact modeling and dynamics simulation method as applied to biped walking robots. In *Proc. 2003 FIRA World Congress, Vienna, Austria*, 2003.

[28] Kemalettin Erbatur and Okan Kurt. Natural zmp trajectories for biped robot reference generation. *Industrial Electronics, IEEE Transactions on*, 56(3):835–845, 2009.

[29] Kemalettin Erbatur, Utku Seven, Evrim Taşkiran, Özer Koca, M Ylmaz, Mustafa Ünel, Güllü Kiziltaş, Asif Sabanovic, and Ahmet Onat. Suralp: a new full-body humanoid robot platform. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4949–4954. IEEE, 2009.

[30] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx.

[31] Kenny Erleben. Kenny erleben. `http://image.diku.dk/kenny/download/vriphys10_course/lcp.pdf`.

[32] Joaquin Estremera and Kenneth J Waldron. Thrust control, stabilization and energetics of a quadruped running robot. *The International Journal of Robotics Research*, 27(10):1135–1151, 2008.

[33] Roy Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983.

[34] Roy Featherstone. *Robot dynamics algorithms*. Kluwer Academic Publisher, 1984.

[35] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 2: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9):876–892, 1999.

[36] Roy Featherstone. 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, 1999.

[37] Michael C Ferris, Olvi L Mangasarian, and Stephen J Wright. *Linear programming with MATLAB*, volume 7. SIAM, 2007.

[38] Michele Focchi. *Strategies to Improve the Impedance Control Performance of a Quadruped Robot*. PhD thesis, Ph. D thesis, Istituto Italiano di Tecnologia, Genoa, Italy, 2013.

[39] Yasutaka Fujimoto and Atsuo Kawamura. Simulation of an autonomous biped walking robot including environmental force interaction. *Robotics & Automation Magazine, IEEE*, 5(2):33–42, 1998.

[40] Yasuhiro Fukuoka, Hiroshi Kimura, and Avis H Cohen. 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, 2003.

[41] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of*, 4(2):193–203, 1988.

[42] Michael L Gleicher. A differential approach to graphical interaction. Technical report, DTIC Document, 1994.

[43] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 871–878. ACM, 2003.

[44] VS Gurfinkel, EV Gurfinkel, A Yu Shneider, EA Devjanin, AV Lensky, and LG Shtilman. Walking robot with supervisory control. *Mechanism and Machine Theory*, 16(1):31–36, 1981.

[45] Havok. About havok. `http://www.havok.com/`, 2015.

[46] Shigeo Hirose. A study of design and control of a quadruped walking vehicle. *The International Journal of Robotics Research*, 3(2):113–133, 1984.

[47] Shigeo Hirose, Yasushi Fukuda, Kan Yoneda, Akihiko Nagakubo, Hideyuki Tsukagoshi, Keisuke Arikawa, Gen Endo, Takahiro Doi, and Ryuichi Hodoshima. Quadruped walking robots at tokyo institute of technology: design, analysis, and gait control methods. *IEEE Robotics and Automation Magazine*, 16(2):104–114, 2009.

[48] Hirohisa Hirukawa, Fumio Kanehiro, Shuji Kajita, Kiyoshi Fujiwara, Kazuhito Yokoi, Kenji Kaneko, and Kensuke Harada. Experimental evaluation of the dynamic simulation of biped walking of humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Citeseer, 2003.

[49] Niu Hongpan, Gao Yong, and Hou Zhongming. Application research of physx engine in virtual environment. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, pages 587–591. IEEE, 2010.

[50] Philip M Hubbard. Interactive collision detection. In *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*, pages 24–31. IEEE, 1993.

[51] Marco Hutter, Michael Gehring, Michael Bloesch, A Hoepflinger Mark, C David Remy, Roland Yves Siegwart, A Hoepflinger Mark, A Hoepflinger Mark, Roland Yves Siegwart, and Roland Yves Siegwart. *StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion*. World Scientific, 2012.

[52] Serena Ivaldi, Vincent Padois, and Francesco Nori. Tools for dynamics simulation of robots: a survey based on user feedback. *arXiv preprint arXiv:1402.7050*, 2014.

[53] Hanjun Jin, Zhiliang Liu, Tianzhen Wu, and Yanxia Wang. The research of collision detection algorithm based on spatial subdivision. In *Computer*

Engineering and Technology, 2009. ICCET'09. International Conference on, volume 2, pages 452–455. IEEE, 2009.

[54] Elliot R Johnson and Todd D Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *Robotics, IEEE Transactions on*, 25(6):1249–1261, 2009.

[55] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1620–1626. IEEE, 2003.

[56] Taehun Kang, Hyungseok Kim, Taeyoung Son, and Hyoukryeol Choi. Design of quadruped walking and climbing robot. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 619–624. IEEE, 2003.

[57] T Kato, A Takanishi, H Jishikawa, and I Kato. The realization of the quasi-dynamic walking by the biped walking machine. In *Fourth Symposium on Theory and Practice of Walking Robots*, pages 341–351, 1981.

[58] Ben Kenwright and Graham Morgan. Practical introduction to rigid body linear complementary problem (lcp) constraint solvers. *Algorithmic and Architectural Gaming Design*, pages 159–205, 2012.

[59] Hiroshi Kimura, Seiichi Akiyama, and Kazuaki Sakurama. Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous robots*, 7(3):247–258, 1999.

[60] Hiroshi Kimura and Yasuhiroo Fukuoka. Biologically inspired adaptive dynamic walking in outdoor environment using a self-contained quadruped robot:'tekken2'. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 986–991. IEEE, 2004.

[61] Sinan Kockara, Tansel Halic, K Iqbal, Coskun Bayrak, and Richard Rowe. Collision detection: A survey. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 4046–4051. IEEE, 2007.

[62] Evangelos Kokkevis. Practical physics for articulated characters. In *Game Developers Conference*, volume 2004, 2004.

[63] J Zico Kolter and Andrew Y Ng. The stanford littledog: A learning and rapid replanning approach to quadruped locomotion. *The International Journal of Robotics Research*, 30(2):150–174, 2011.

[64] Ryo Kurazume, Kan Yoneda, and Shigeo Hirose. Feedforward and feedback dynamic trot gait control for quadruped walking vehicle. *Autonomous Robots*, 12(2):157–172, 2002.

[65] Karen L. Rigid body simulation[powerpoint slides]. urlhttp://www-scf.usc.edu/ csci520/slides/RigidSim.pdf, 2014.

[66] Ming C Lin. *Efficient collision detection for animation and robotics*. PhD thesis, Citeseer, 1993.

[67] Ming C Lin and John F Canny. A fast algorithm for incremental distance calculation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1008–1014. IEEE, 1991.

[68] Morten Lind and Amund Skavhaug. Using the blender game engine for real-time emulation of production devices. *International Journal of Production Research*, 50(22):6219–6235, 2012.

[69] RA Liston and RS Mosher. A versatile walking truck. In *Transportation Engineering Conference*, 1968.

[70] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980.

[71] Kallmann M. Articulated bodies[powerpoint slides]. `http://graphics.ucmerced.edu/~mkallmann/courses/cse287-07s/lectures/L12-articulated.pdf`, 2007.

[72] Christophe Maufroy, Tomohiro Nishikawa, and Hiroshi Kimura. Stable dynamic walking of a quadruped robot kotetsu using phase modulations based on leg loading/unloading. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5225–5230. IEEE, 2010.

[73] Robert B McGhee. Vehicular legged locomotion. *Advances in Automation and Robotics*, 1:259–284, 1985.

[74] Olivier Michel. Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*, 2004.

[75] Andrew T Miller, Henrik Christensen, et al. Implementation of multi-rigid-body dynamics within a robotic grasping simulator. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2262–2268. IEEE, 2003.

[76] Brian Mirtich. Efficient algorithms for two-phase collision detection. *Practical motion planning in robotics: current approaches and future directions*, pages 203–223, 1997.

[77] Brian Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions On Graphics (TOG)*, 17(3):177–208, 1998.

[78] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff. ACM, 1995.

[79] Brian Vincent Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California at Berkeley, 1996.

[80] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3(2):60–74, 1984.

[81] MuJoCo. Mujoco advanced physics simulation. `http://www.mujoco.org/`, 2015.

[82] Eadweard Muybridge. *Animals in motion.* Courier Corporation, 1957.

[83] Keiji Nagatani, Hiroaki Kinoshita, Kazuya Yoshida, Kenjiro Tadakuma, and Eiji Koyanagi. Development of leg-track hybrid locomotion to traverse loose slopes and irregular terrain. *Journal of Field Robotics*, 28(6):950–960, 2011.

[84] Shinichiro Nakaoka, Shizuko Hattori, Fumio Kanehiro, Shuuji Kajita, and Hirohisa Hirukawa. Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3641–3647. IEEE, 2007.

[85] University of California Santa Barbara. Cs 209 lecture slide. `http://excelsior.cs.ucsb.edu/courses/cs290n_cg_modeling/notes/ucsb_slides/kinematics_position.pdf`.

[86] International Federation of Robotic. History of industrial robot. `http://www.ifr.org/history/`.

[87] DE Orin, RB McGhee, M Vukobratović, and G Hartoch. Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods. *Mathematical Biosciences*, 43(1):107–130, 1979.

[88] Hisashi Osumi, Shogo Kamiya, Hirokazu Kato, Kazunori Umeda, Ryuichi Ueda, and Tamio Arai. Time optimal control for quadruped walking robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1102–1108. IEEE, 2006.

[89] Jong H Park and Yong K Rhee. Zmp trajectory generation for reduced trunk motions of biped robots. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 90–95. IEEE, 1998.

[90] John C Platt and Alan H Barr. Constraints methods for flexible models. In *ACM SIGGRAPH Computer Graphics*, volume 22, pages 279–288. ACM, 1988.

[91] Robert R Playter and Marc H Raibert. Control of a biped somersault in 3d. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, volume 1, pages 582–589. IEEE, 1992.

[92] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, et al. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress*, volume 17, pages 10822–10825, 2008.

[93] Marc H Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[94] Marc H Raibert. *Legged robots that balance.* MIT press, 1986.

[95] Marc H Raibert. Trotting, pacing and bounding by a quadruped robot. *Journal of biomechanics*, 23:79–98, 1990.

[96] Kawasaki Robotics. 6 axis articulated robot. `http://www.directindustry.com/prod/kawasaki-robotics/product-18836-585512.html`, 2015.

[97] Diego Ruspini and Oussama Khatib. Collision/contact models for the dynamic simulation of complex environments. In *IEEE/RSJ IROS*, volume 97. Citeseer, 1997.

[98] Paul S Schenker, Paolo Pirjanian, J Balaram, KS Ali, Ashitey Trebi-Ollennu, Terrance L Huntsberger, Hrand Aghazarian, Brett A Kennedy, Eric T Baumgartner, Karl D Iagnemma, et al. Reconfigurable robots for all-terrain exploration. In *Intelligent Systems and Smart Manufacturing*, pages 454–468. International Society for Optics and Photonics, 2000.

[99] Claudio Semini. Hyqdesign and development of a hydraulically actuated quadruped robot. *PD Thesis, University of Genoa, Italy*, 2010.

[100] Ahmed A Shabana. *Computational dynamics.* John Wiley & Sons, 2009.

[101] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[102] James Andrew Smith. *Galloping, bounding and wheeled-leg modes of locomotion on underactuated quadrupedal robots*. PhD thesis, McGill University, 2006.

[103] Kenji Sorao, Toshiyuki Murakami, and Kohei Ohnishi. A unified approach to zmp and gravity center control in biped dynamic stable walking. In *Advanced Intelligent Mechatronics' 97. Final Program and Abstracts., IEEE/ASME International Conference on*, page 112. IEEE, 1997.

[104] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[105] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950, 2013.

[106] Mark C Surles. An algorithm with linear complexity for interactive, physically-based modeling of large proteins. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 221–230. ACM, 1992.

[107] Masaharu Takahashi, Kan Yoneda, and Shigeo Hirose. Rough terrain locomotion of a leg-wheel hybrid quadruped robot. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1090–1095. IEEE, 2006.

[108] Hiroki Takeuchi. Real time optimization for robot control using receding horizon control with equal constraint. *Journal of Robotic Systems*, 20(1):3–13, 2003.

[109] Evrim Taskiran, Metin Yilmaz, Ozer Koca, Utku Seven, and Kemalettin Erbatur. Trajectory generation with natural zmp references for the biped

walking robot suralp. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4237–4242. IEEE, 2010.

[110] A Tasora. An optimized lagrangian multiplier approach for interactive multibody simulation in kinematic and dynamical digital prototyping. *Proceedings of VIII ISCSB. CLUP, Milano*, pages 1–12, 2001.

[111] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[112] Daniel J Tracy, Samuel R Buss, and Bryan M Woods. Efficient large-scale sweep and prune methods with aabb insertion and removal. In *Virtual Reality Conference, 2009. VR 2009. IEEE*, pages 191–198. IEEE, 2009.

[113] Miomir Vukobratovic, Branislav Borovac, Dusan Surla, and Dragan Stokic. *Biped locomotion: dynamics, stability, control and application*, volume 7. Springer Science & Business Media, 2012.

[114] Rachel Weinstein, Joseph Teran, and Ronald Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *Visualization and Computer Graphics, IEEE Transactions on*, 12(3):365–374, 2006.

[115] R Weyler, J Oliver, T Sain, and JC Cante. On the contact domain method: A comparison of penalty and lagrange multiplier implementations. *Computer Methods in Applied Mechanics and Engineering*, 205:68–82, 2012.

[116] Katsu Yamane and Yoshihiko Nakamura. Parallel o (log n) algorithm for dynamics simulation of humanoid robots. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 554–559. IEEE, 2006.

[117] Katsu Yamane and Yoshihiko Nakamura. A numerically robust lcp solver for simulating articulated rigid bodies in contact. *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland*, 19:20, 2008.

[118] Kenneth S Yamazaki. *The Design and Control of Scout I, a Simple Quadruped Robot.* PhD thesis, McGill University, 1999.

[119] Kan Yoneda and Shigeo Hirose. Dynamic and static fusion gait of a quadruped walking vehicle on a winding path. *Advanced Robotics*, 9(2):125–136, 1994.

[120] Kan Yoneda, Hiroyuki Iiyama, and Shigeo Hirose. Intermittent trot gait of a quadruped walking machine dynamic stability control of an omnidirectional walk. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3002–3007. IEEE, 1996.