A STUDY ON AUTOMATIC GAIT PARAMETER TUNING FOR BIPED WALKING ROBOTS

by

ÖZKAN BEBEK

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

Sabancı University

July 2003

A STUDY ON AUTOMATIC GAIT PARAMETER TUNING FOR BIPED WALKING ROBOTS

APPROVED BY:

Assist. Prof. Dr. Kemalettin Erbatur .....
(Thesis Advisor)

Prof. Dr. Asif Şabanoviç .....
(Thesis Co-advisor)

Assist. Prof. Dr. Gökhan Göktuğ .....

Assist. Prof. Dr. Serhat Yeşilyurt .....

Assist. Prof. Dr. Selçuk Öğrenci .....

DATE OF APPROVAL: 27/06/2003

ABSTRACT

Automatic gait parameter tuning for biped walking robots is the subject of this thesis.

The biped structure is one of the most versatile ones for the employment of mobile robots in the human environment. Their control is challenging because of their many DOFs and nonlinearities in their dynamics.

Open loop walking with offline walk pattern generation is one of the methods for walking control. In this method the reference positions of the foot centers with respect to the body center are generated as functionals. Commonly, the tuning process for the trajectory generation is based on numerous trial and error steps. Obviously, this is a time consuming and elaborate process. In this work, online adaptation schemes for one of the trajectory parameters, "x-reference asymmetry", which is used for the compensation of uneven weight distribution of the robot in the sagittal plane, is proposed. In one of the approaches presented, this parameter is tuned online. As an alternative to parameter tuning, a functional learning scheme employing fuzzy identifiers is tested too. Fuzzy identifiers are universal function approximators. Fuzzy system parameters are adapted via back-propagation.

An on-line tuning scheme for biped walk parameters however can only be successful if there is sufficient time for training without falling. The training might last hundreds of reference cycles. This implies that a mechanism for keeping the robot in continuous walk, even when the parameter settings are totally wrong, is necessary during training. In this work, virtual torsional springs which resist against deviations of the robot trunk angles from zero, are attached to the trunk center of the biped. The torques generated by the springs serve as the criteria for the tuning and help in maintaining a stable and a longer walk. The springs are removed after training. This novel approach can be applied to a wide range of control systems that involve parameter tuning.

3-D simulation techniques using C++ are employed for the model of a 12-DOF biped robot to test the proposed adaptive method. In order to visualize the walking, simulation results are animated using an OpenGL based animation environment. As a result of the simulations, a functional for the desired parameter, keeping the system in balance while walking, is generated.

ÖZET

Bu tezin konusu iki ayaklı yürüyen robotların adım parametrelerinin otomatik ayarlanmasıdır.

İki ayaklı yapı, mobil robotların insan ortamlarında kullanılmaları açısından en uygun tasarımlardan biridir. Ancak, bu yapının denetimi zordur çünkü dinamikleri pek çok serbestlik derecesine sahiptir ve çoğunlukla doğrusal değildir.

Yürüme hareketinin denetim yöntemlerinden biri acik cevrim bir yürüme seyri belirleyerek geri beslemesiz yürüyüş yaptırmaktır. Bu yöntemde ayak merkezlerinin gövde merkezindeki referans noktalarına göre fonksiyonlar oluşturulur.Genellikle, izlenecek yol üzerinde yapılan ayarlama süreci birçok deneme yanılma adımlarından oluşur. Bu da kesinlikle zaman kaybettirici ve dikkat isteyen bir süreçtir. Bu tezde izlenecek yola ait parametrelerden biri olan ve robotun yürüme doğrultusundaki düzlem üzerinde eşit olmayan ağırlık dağılımını dengelemek için kullanılan "x-referans asimetrisi" için adaptasyon düzenleri önerilmektedir. Bir yaklaşımda, parametre çevrim içi ayarlanmaktadır. Parametre ayarına alternatif olarak, bulanık mantık kullanan fonksiyonel parametre öğrenme düzeni denenmiştir. Bulanık mantık tanımlıyıcıları evrensel tahmin edicilerdir. Bulanık sistem parametreleri geri-yayma yöntemiyle yeni durumlara uyarlanır.

Diğer yandan çevrim içi ayarlama düzeninin başarılı olabilmesi için robotun düşmeden uzun süre yürümesi gereklidir. Bu da yüzlerce referans döngüsü kadar süre demektir. Ayrıca bunun için parametre ayarlarının tamamen yanlış olması durumunda bile robotun düşmeden yürümesini sağlayacak bir mekanizmaya ihtiyaç vardır. Bu çalışmada robot gövde açılarının sıfır noktasından sapmasını engelleyen sanal bükmeli yaylar gövde merkezine tutturulmuştur. Yayların ürettiği torklar parametre ayarlama için birer kriter oluştururlar. Bunlar, ayrıca, dengeli ve uzun bir yürüyüşe de yardımcı olurlar. Yaylar ayarlama tamamlandıktan sonra kaldırılırlar. Bu yaklaşım literatürde yenidir ve parametre ayarı içeren diğer sistemler için de kullanılabilir.

Bu çalışmada 12 serbestlik derecesine sahip iki ayaklı bir robotun modeli kullanılmıştır. Önerilen uyarlama yönteminin ölçülmesi için C++ kullanılarak üç boyutlu benzetim tekniği uygulanmıştır. Benzetim sonuçları OpenGL tabanlı bir animasyon ortamı kullanılarak görselleştirilmiştir. Benzetimlerin sonucunda, istenen parametre için sistemi yürüyüş sırasında dengede tutan bir fonksiyon oluşturulmuştur.

To my love, Ebru…

ACKNOWLEDGMENTS

First of all, I would like to extend my appreciation for the guidance, motivation, and support provided to me during the conduct of my thesis, with special thanks to my thesis advisor, Assistant Professor Kemalettin Erbatur.

I also would like to thank to Professor Asif Şabanoviç, for his help and encouragement.

Also thanks to all Mechatronics graduate students for sharing nights with me in the research lab, supporting me, and especially for their friendship.

Finally I would like to thank Ebru Kılıç for her support,

My parents, Meryem and Veli Bebek for their encouragement to do this work,

And my brother Gürkan Bebek for his full-time helps during my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

${}^{x}\boldsymbol{A}_{y}$	:	Rotation matrix
$\boldsymbol{A}_{B}$	:	Base-link attitude matrix
a	:	Numerator of $f(\underline{x})$
a_i^l	:	Height of membership function
b	:	Denominator of $f(\underline{x})$
$\boldsymbol{C}(\boldsymbol{x},\boldsymbol{v})$	:	Centrifugal and Corioli's force matrix
$\boldsymbol{c}$	:	Contact point position vector
D	:	Viscous damping coefficient
d	:	Desired value
E	:	Error of the fuzzy model
$\boldsymbol{F}$	:	Total force
$\boldsymbol{f}_{E}$	:	External force vector
$f(x)$	:	Fuzzy model
$\boldsymbol{g}(\boldsymbol{x})$	:	Gravity vector
$\boldsymbol{H}(\boldsymbol{x})$	:	Inertia matrix
i	:	Index numbers
j	:	Index numbers
K_d	:	PID controller derivative gain
$\boldsymbol{K}_{E}$	:	External force to generalized force transformation matrix
K_i	:	PID controller integral gain
K_p	:	PID controller proportional gain
k	:	Sampling period index
l	:	Number of input variables

M	:	Number of rules
M	:	Number of time-variant active contact points
M_A	:	Index numbers of active contact points
$\boldsymbol{M}$	:	Index numbers of external forces
N	:	Number of joints of the robot
$\boldsymbol{N}$	:	Total moment
n	:	Number of input variables
$\boldsymbol{p}$	:	Axes origin position vector
$\dot{\boldsymbol{p}}$	:	Axes origin velocity vector
$\ddot{\boldsymbol{p}}$	:	Axes origin acceleration vector
$\boldsymbol{p}_B$	:	Base-link position vector
$\boldsymbol{r}$	:	Center of mass position
$\dot{\boldsymbol{r}}$	:	Center of mass velocity
$\ddot{\boldsymbol{r}}$	:	Center of mass acceleration
$\boldsymbol{s}$	:	Center of mass position vector with respect to coordinate system
$\boldsymbol{u}$	:	Generalized force vector
u	:	Control output
$\boldsymbol{u}_E$	:	Generalized force vector generated by external forces
$\boldsymbol{v}$	:	Generalized velocities vector
$\dot{\boldsymbol{v}}$	:	Generalized accelerations vector
$\boldsymbol{v}_B$	:	Base-link velocity vector
$\dot{\boldsymbol{v}}_B$	:	Base-link acceleration vector
$\boldsymbol{w}$	:	Joint angular velocity vector
$\dot{\boldsymbol{w}}$	:	Joint angular acceleration vector
$\boldsymbol{w}_B$	:	Base-link angular velocity vector
$\dot{\boldsymbol{w}}_B$	:	Base-link angular acceleration vector
$\boldsymbol{x}$	:	Generalized coordinate vector
x_i	:	i^{th} input variable
$\bar{x}_i^l$	:	Center of the membership function for x_i for rule l
xra	:	x-reference asymmetry

$\bar{y}^l$	:	Output constant of rule
$\boldsymbol{z}$	:	Direction vector of the joint
α	:	Roll angles
β	:	Pitch angle
γ	:	Yaw angle
$\boldsymbol{\theta}$	:	Joint angle vector
μ	:	Membership function
η	:	Constant step size
σ_i^l	:	Width of membership function
$\boldsymbol{\tau}$	:	Actuator torque vector

LIST OF ABBREVIATIONS

c_θ	:	$\cos(\theta)$
DOF	:	Degree of freedom
D-H	:	Denavit – Hartenberg
GUI	:	Graphical user interface
kg	:	Kilogram
LxWxH	:	Length - Width – Height
m	:	Meter
s	:	Seconds
s_θ	:	$\sin(\theta)$
ZMP	:	Zero moment point

1. INTRODUCTION

Past three decades witnessed a growing interest in biped walking robots because of their possible advantageous use in the human environments. Many research labs and universities started to work on biped and humanoid projects, in order to develop their own robots. Nowadays we can see some companies' robots that are commercialized and put on to market, like Honda, Sony and Fujitsu [1] (Fig 1.1). Probably the ultimate goal when developing these machines is to replace the humans with robots, that are able to work with or instead of humans having some of the human skills, in the factories, malls, restaurants, our homes and even in the soccer fields. There are numerous places that humanoids might be used to ease our life. This will be one of the biggest revolutions after the step on moon that technology will accomplish in the human history. Biped robot research is one of the bricks to realize the humanoid technology.

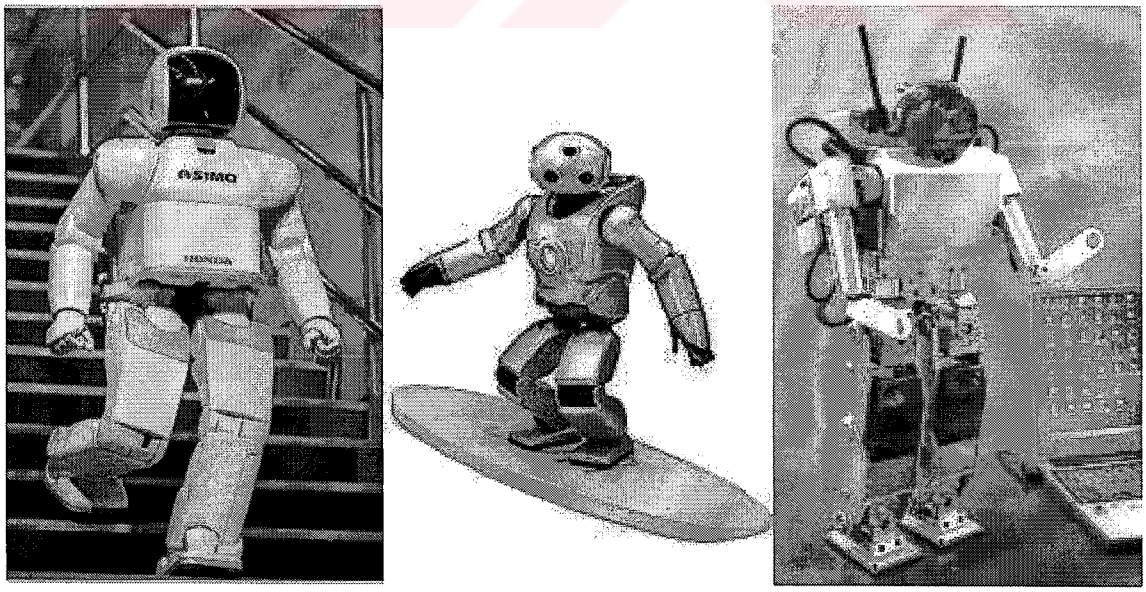

Figure 1.1. Most popular commercial humanoid robots. From left to right: Honda's Asimo, Sony's SDR-4X and Fujitsu's HOAR-1.

Despite their functionality, their control is challenging because of their many DOFs and nonlinearities in their dynamics. Offline trajectory generation and the so-called open loop walking is one of the control approaches in the literature for bipeds. There are various problems involved in this approach, the most pronounced one being the difficulty in tuning the gait parameters.

In this work, online tuning schemes for some of the trajectory parameters in the offline generated walking pattern is proposed. A scalar parameter tuning scheme and a fuzzy logic system, represented as a three-layer feed-forward neural network is employed to compute the parameters as a function of time, as alternatives. Fuzzy system parameters are adapted via back-propagation. The adapted fuzzy systems are termed as fuzzy identifiers which are universal approximators.

In order to keep the biped in balance during parameter tuning, virtual torsional springs are attached to the trunk center of the biped. The torques generated by the springs serve as the criteria for the tuning and they help maintaining a stable and a longer walk which is necessary for the on-line tuning process. 3D simulation techniques are employed for a 12-DOF biped robot to test the proposed adaptive method.

The organization of the thesis is as follows.

The next chapter contains an overview of the literature on the topics of the history of biped robots and biped gait adaptation techniques. It also mentions briefly from the proposed problem and the possible solutions.

Chapter 3 describes the biped model used in this work explaining the dynamics equations of the test bed and calculations made in the simulation. It also introduces an open loop walking scheme for the biped robot.

The idea of attaching virtual torsional springs to the biped robot is introduced in Chapter 4. It explains the necessity of this method and possible applications of the virtual springs on the biped robot.

Application of adaptive systems to the tuning of walking trajectory parameters of the biped robot is outlined in Chapter 5.

Chapter 6 presents the simulation results achieved during this work. Plots are also given for the corresponding work done in this section.

The last chapter summarizes the work carried out and results obtained. Comments on the results, conclusion remarks and directions for the future work are also given in this part.

2. A SURVEY ON BIPED GAIT ADAPTATION TECHNIQUES

2.1. History of Biped Robotics

Biped robots have been in the research and development phase since late 1960s and the field of biped robotics attracted a rapidly growing number of researchers during the last decade. In 1968, the first fully computer-controlled walking machine was built by Frank and McGhee at the University of Southern California [2,3]. But the first statically stable walking biped at all times was build by Kato and his colleagues; it first walked in 1973 [4]. It is, relying on keeping its center-of-gravity above at least one of its large feet. Then in the 1970s and early 1980s new and more sophisticated walking robots are built in Japan and USA.

In the 1990s, humanoid robot research became very popular, especially in Japan. One of the most surprising developments of the technology was Honda announcing that a humanoid research project had been held since many years and the robot, which is called Prototype-1 (P1), is able to realize dynamic walk on transverse plane. The development phase of Honda's biped robots is shown in Fig. 2.1. After P1, Honda

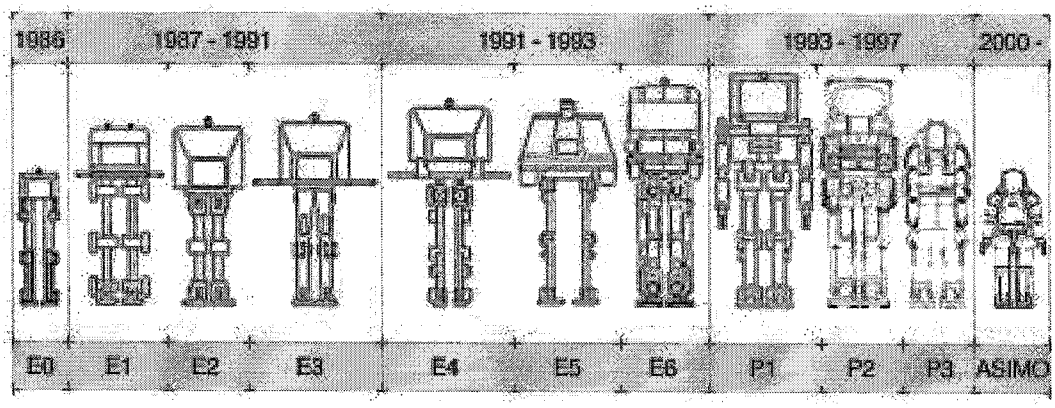

Figure 2.1. Development history of Honda's walking robots.

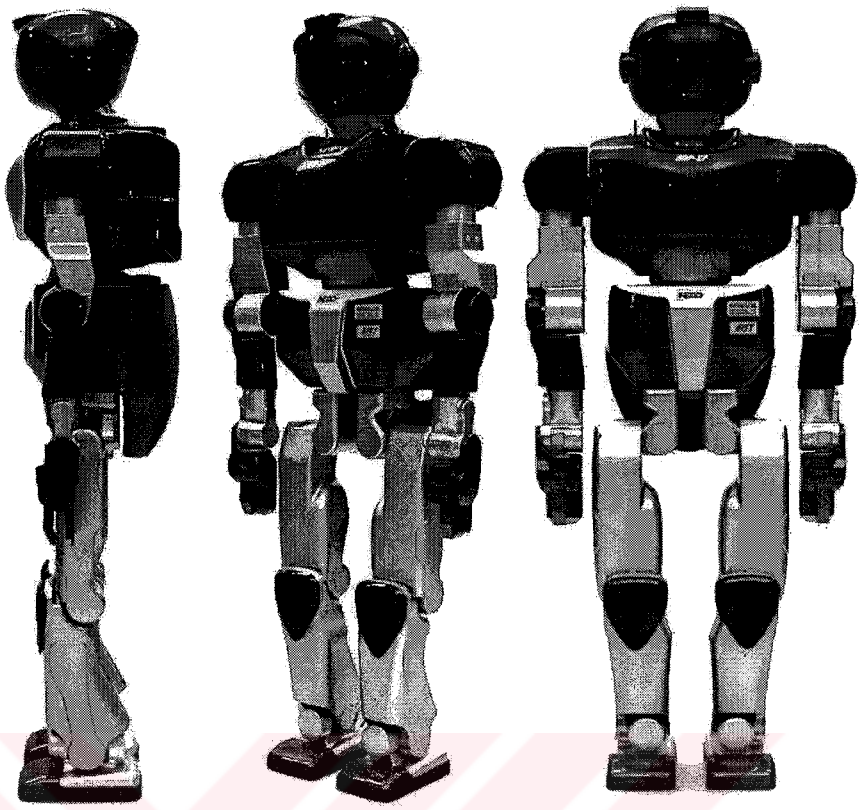

Figure 2.2. The last prototype of Humanoid Research Project: HRP-2.

announced other humanoid prototypes that are autonomous, Prototype-2 (P2) in 1996 and Prototype-3 (P3) in 1997 [5]. In year 2000, Asimo was demonstrated with decreased weight and size, and increased speed and functionality, with respect to its primitives.

Humanoid Research Project (HRP) [6] is another striking project which is launched by the Ministry of Economy, Trade and Industry of Japan. Last developed prototype of this project, HRP-2, is also a promising work, which is shown in Fig. 2.2.

Another important humanoid project that is in progress is called PINO (Fig 2.3), which is an open source humanoid project on the internet [7]. PINO's design criteria arise from the following four requirements: Behavior generation, recognition with sensors, relatively cheap components and adequate size for interaction with the environment.

There are many humanoid projects that continue throughout the world. Every developed country has a famous humanoid project that is going on. It is obvious that in

a few decades we will start to see humanoids everywhere. Therefore it is an important issue to start a humanoid project in Turkey.

2.2. Literature Survey on Biped Gait Adaptation Techniques

The biped structure is one of the most versatile ones for the employment of mobile robots in the human environment. It has compatibility in tasks as human substitutes and supreme characteristics in obstacle avoidance, when compared with wheeled and multi-legged robots. However, the biped robot dynamics are highly nonlinear, complex and unstable. This makes biped walking control a highly challenging task.

Keeping balance of the biped robot is a complicated task that requires every DOF of the robot to be used. It is highly desirable for the robot to adapt to the ground conditions with a foot motion, and maintain its stability with a torso motion [8]. By varying the values of the constraint parameters, we can produce different types of foot motion to adapt to the environment can be produced.

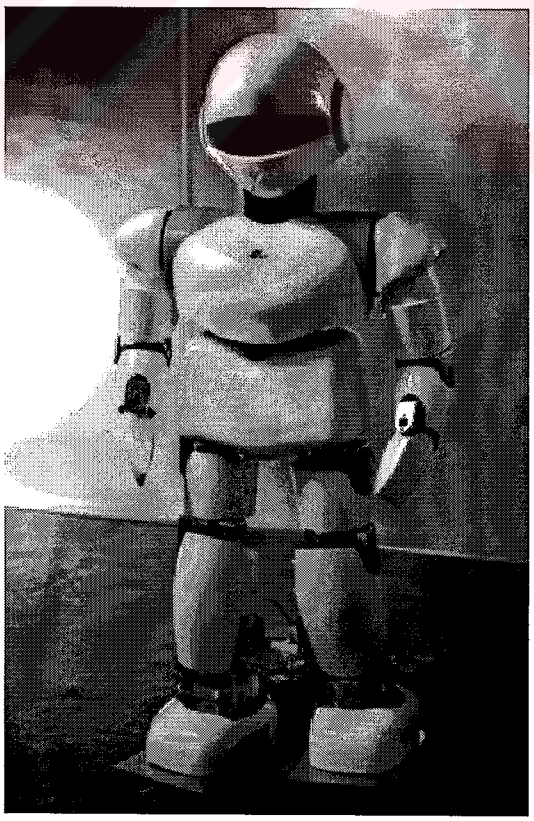

Figure 2.3. One of the most popular humanoid robots: PINO

In [9] an adaptation scheme for walking on an unknown surface is proposed for an anthropomorphic biped robot. An adaptive mechanism for walking control over a step of unknown height on a flat floor was achieved. The authors of [9] consider that the adaptability is an integral key for the biped robots which will support human living in the near future.

In order to compensate for the disturbances the robot, should be robust, and requires on-line adaptation ability. In [10] an adaptive trajectory generation strategy of using online ZMP information with an impedance control method is applied to overcome disturbances and keep the balance of the biped. Adaptation of the generated pattern by online trajectory planning during walk is the proposed solution for stable biped walking. Similarly, adaptation of the base link trajectory in both single and double support phases, using the magnitude of ZMP is done in [11] by the same authors, such that appropriate angular momentum is generated to maintain stable walk. The modified trajectory is recovered after stability is maintained.

In [12] the proposed system is adaptive in that a high-level parameter controller can adjust the parameters of the pattern generator to change the frequency of the desired joint trajectories. By adjusting the parameters of gait pattern generator, stable gaits are realized.

A biped robot tends to tip over very easily. Stable and reliable biped walking is a very important issue to overcome the mentioned problem. A balance control method based on offline planned walking pattern with real-time modification is proposed in [13]. Real-time modifications consist of body posture control, actual ZMP control and landing time control. Smooth walk and adaptation to unknown environments is achieved on a biped robot dynamic simulator.

Online pattern generation using motion parameters of the robot is another gait adaptation technique.

In [14] online generation of humanoid walking patterns is proposed. This is realized by dynamically stable mixture and connection of pre-designed motions and characteristics of ZMP, in order to maintain the overall dynamic stability of the mixed motions. Validity of the method was tested on the experimental humanoid robot using the mixture of 21 pre-designed stable motions.

An online method generating walking patterns for biped walking robots having a trunk is proposed in [15]. For stability the trunk and waist motion is generated by a

walking stabilization control that is based on ZMP trajectory and the motion of lower limbs. Validity of the method is tested with experiments on a versatile biped walking on a plane surface.

An online interactive locomotion method for a biped robot is described in [16]. The motion of the lower limbs is generated by the online pattern generator based on the locomotion parameters. Continues locomotion experiments while changing the walk parameters are carried in real time using a biped robot.

A method of generating a high stability smooth walking pattern is presented in [17]. The stability is achieved by setting a series of defined walking pattern parameters to the actuators of the biped. By simulation results, correlation found between actuator specifications and walking patterns is described.

Different control techniques are applied in the literature for the adaptation of generated walking pattern or tuning of the walking parameters.

In [18], on an experimental biped, pre-planned but adaptive motion sequences were implemented. Neural networks were responsible for the adaptive control of side-to-side and front-to-back balance, as well as for maintaining good foot contact. Qualitative and quantitative test results show that the biped performance improved with neural network training in that work.

Another control scheme used in biped robotics for stabilization of walking thorough learning is Reinforcement Learning. In [19], in order to determine leg swing parameters, which are key determinants for stabilization of walking speed and lateral motion, the learning scheme is used.

In [20], a way to apply Genetic Algorithms to Fuzzy Logic Controllers is proposed and an application designed to control the synthesis of biped walk of a 2-D biped robot is presented.

A multi layer Neural Network learning architecture with back propagation algorithm is used to control the dynamic stability of a simulated planer biped in [21].

The next chapter will describe the biped model used in this work. The walking scheme and simulation techniques are also discussed.

3. THE BIPED MODEL EMPLOYED AS THE SIMULATION TEST BED

The biped model used in this work as a simulation test bed is called "Mari-2", one of the biped robots of Yokohama National University, Japan [22] (Fig 3.1).

This model is selected since it is an experimentally tested model and suitable for our simulations. The test bed consists of two 6-DOF legs and a trunk connecting them. Approximate link sizes and the masses of the biped are given in Table 3.1.

Human beings can execute both voluntary and reflexive (involuntary) motions.

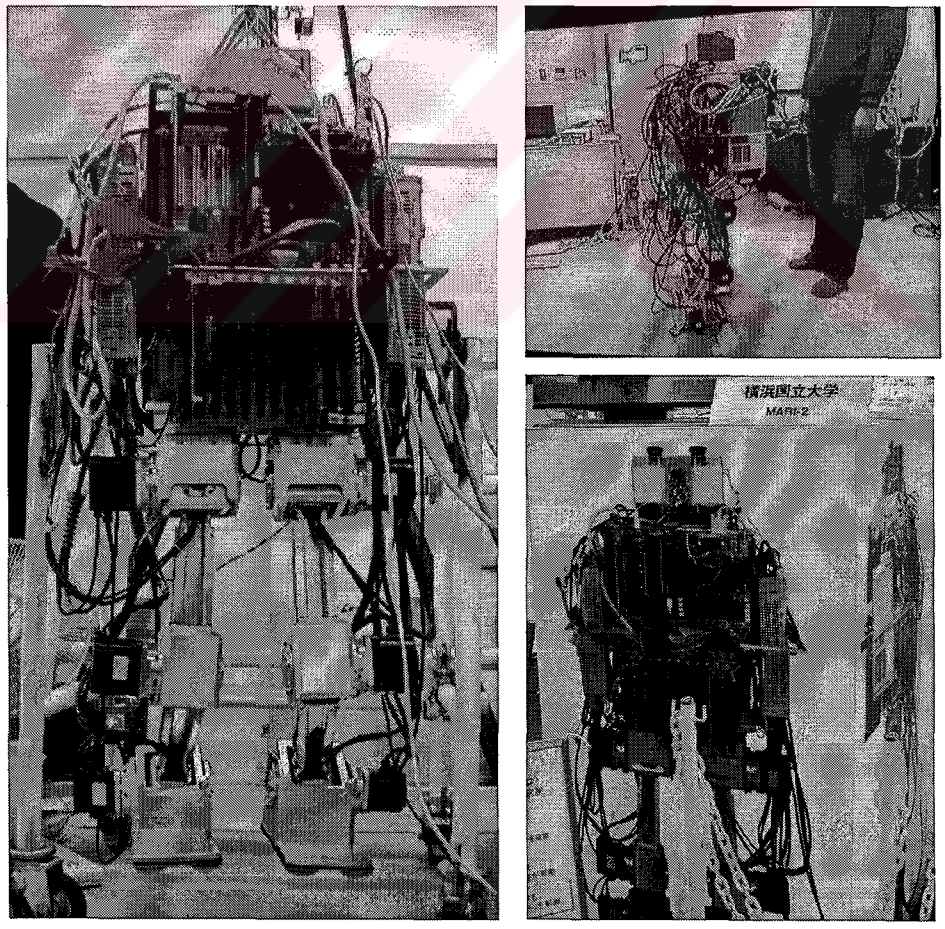

Figure 3.1. Some pictures of the used test bed robot, Mari-2

Table 3.1. Masses and dimensions of the biped robot links.

Link	Dimensions (LxWxH) [m]	Mass [kg]
Trunk	0.2 x 0.4 x 0.5	50
Thigh	0.1 x 0.1 x 0.1	12
Calf	0.22 x 0.05 x 0.1	0.5
Foot	0.1 x 0.12 x 0.25	5.5

While walking on a familiar terrain, locomotion is voluntary. Locomotion becomes reflexive when terrain conditions become difficult. Furthermore, a human is able to learn and memorize new gait patterns while executing reflexive motions. As a result, reflexive motions become voluntary after the same terrain has been traversed for a number of times. Because of this learning capability, humans use voluntary motions for walking most of the time [23]. Therefore open loop trajectory generation and open loop walking are essential in the biped robots' literature.

Open loop walking with offline walk pattern generation is one of the methods for the walking control [24,25]. In this method the reference x, y and z coordinates of

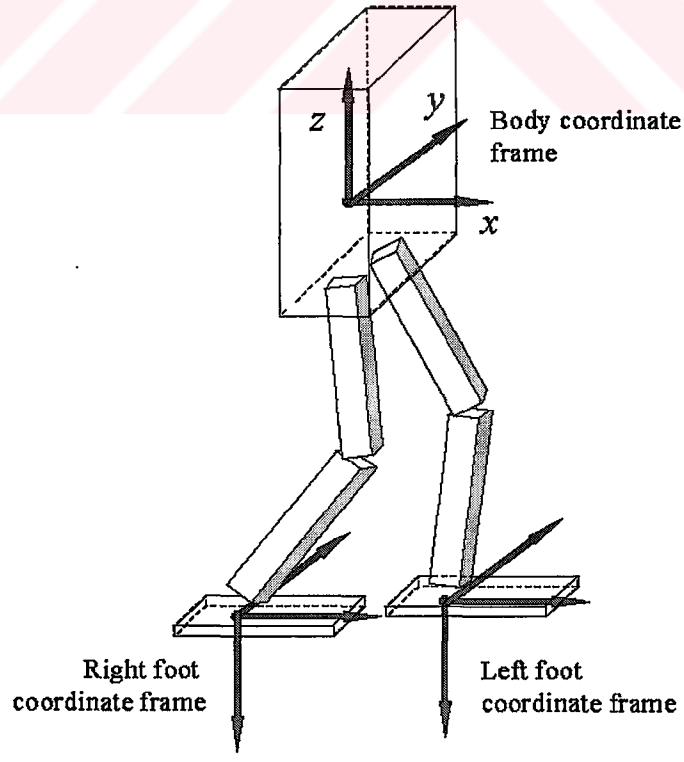

Figure 3.2. The body and the foot coordinate frames

the foot centers (Fig. 3.2) with respect to the body center are offline generated as periodic functions of time on one complete gait cycle and these reference positions are repeated periodically for the following steps. It was shown by experiments that biological systems have a central pattern generator which generates rhythmic patterns for voluntary motions [26], that was the inspiration point for using periodic functions. According to these reference positions, necessary joint positions and velocities are calculated using inverse kinematics, and independent joint controllers are employed to control the joint angles or displacements. In the offline generated reference trajectory used in this work, the periodic functions are the x, y and z position references for coordinate frame centers attached to the two feet, with respect to a coordinate frame attached to the trunk of the biped robot. The feet orientations are kept constantly parallel to ground. The x, y and z foot trajectories are parameterized by a number of constants. One of these constants is termed the "x-reference asymmetry" which is used for the

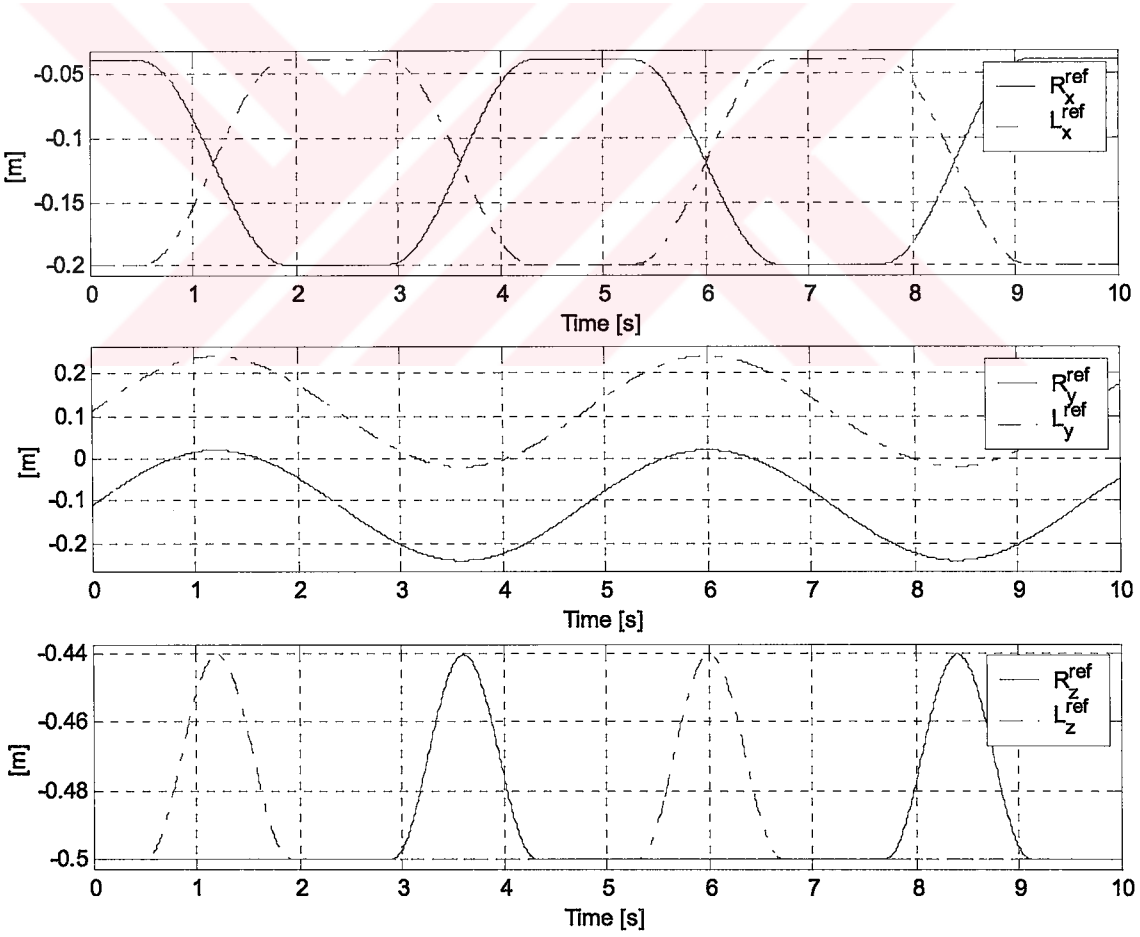

Figure 3.3. Reference x, y, and z positions of the foot centers of the biped robot with respect to the body center.

compensation of uneven weight distribution of the robot in the x direction, that is, in the direction of the walk. Commonly, the tuning process for the trajectory generation is based on numerous trial and error steps. The trial and error search concludes when the robot walks without falling. Obviously, this is a time consuming and elaborate process. Because of the high degree of the coupling between the degrees of freedom of the robot, numerous iterations are necessary in order to obtain a set of parameters, which perform in harmony. In order to maintain a stable and human like walk each of these parameters should be tuned precisely.

Fig. 3.3 shows the functions used in this work for the two feet. These functions are obtained via trial and error, and generated differently for the single support and double support periods. They are continuous and symmetric for the left and right feet.

According to these reference positions of the feet, the desired joint trajectories are computed using the inverse kinematics of the robot [27]. Overall joint placement of the biped can be seen in Fig. 3.4.

According to D-H convention driven at [27], the joint axes can be shown in Fig. 3.5 and accordingly, D-H parameters are listed at Table 3.2.

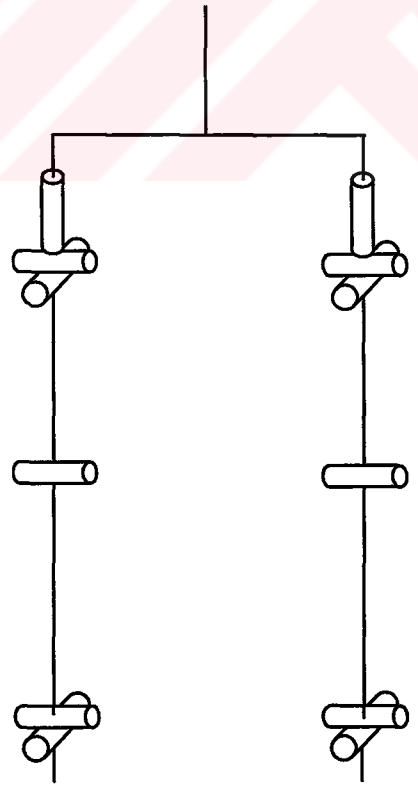

Figure 3.4. Joint placements of the biped robot.

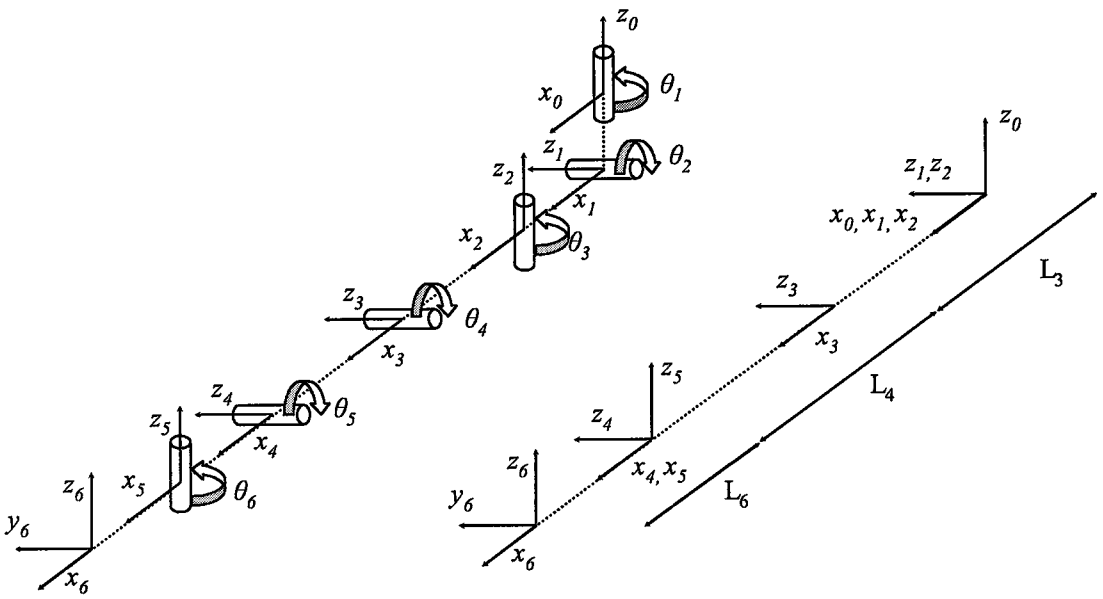

Figure 3.5. (a) Exploded view of the joints and their axes; (b) Joint axes and their placements.

The biped robot is modeled as a free-fall manipulator which is not fixed to the ground but has interaction with. In order to formulate the dynamics of a free-fall manipulator, position and attitude variables of the base-link should be introduced. Let generalized coordinates $\boldsymbol{x}$, generalized velocities $\boldsymbol{v}$, and generalized forces $\boldsymbol{u}$ be:

Table 3.2. D-H Parameters of the biped leg

Link	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	0	θ_1^*
2	0	$-\frac{\pi}{2}$	0	θ_2^*
3	L_3	$\frac{\pi}{2}$	0	θ_3^*
4	L_4	0	0	θ_4^*
5	0	$-\frac{\pi}{2}$	0	θ_5^*
6	L_6	0	0	θ_6^*

$$\boldsymbol{x}^T = [\boldsymbol{p}_B^T, \boldsymbol{A}_B^T, \boldsymbol{\theta}^T] \in R^3 \times SO(3) \times R^N \tag{3.1}$$

$$\boldsymbol{v}^T = [\boldsymbol{v}_B^T, \boldsymbol{w}_B^T, \boldsymbol{w}^T] \in R^3 \times R^3 \times R^N \tag{3.2}$$

$$\boldsymbol{u}^T = [\boldsymbol{f}_B^T, \boldsymbol{n}_B^T, \boldsymbol{\tau}^T] \in R^3 \times R^3 \times R^N \tag{3.3}$$

where

$\boldsymbol{p}_B$: 3×1 vector specifying base-link position

$\boldsymbol{A}_B$: 3×3 matrix specifying base-link attitude

$\boldsymbol{\theta}$: N×1 vector specifying joint angle

$\boldsymbol{v}_B$: 3×1 vector specifying base-link velocity

$\boldsymbol{w}_B$: 3×1 vector specifying angular velocity of base-link

$\boldsymbol{w}$: N×1 vector specifying joint angular velocity

$\boldsymbol{f}_B$: 3×1 force vector generated in base-link

$\boldsymbol{n}_B$: 3×1 torque vector generated in base-link

$\boldsymbol{\tau}$: N×1 torque vector generated by actuator

N : Number of joints of the robot

$\boldsymbol{A}_B$ is the transformation matrix giving the position of link axes relative to the world axes. The equations of motion of the robot become:

$$\dot{\boldsymbol{p}}_B = \boldsymbol{v}_B \tag{3.4}$$

$$\dot{\boldsymbol{A}}_B = \boldsymbol{w}_B \times \boldsymbol{A}_B \tag{3.5}$$

$$\dot{\boldsymbol{\theta}}_B = \boldsymbol{w} \tag{3.6}$$

and

$$\boldsymbol{H}(\boldsymbol{x})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{u} + \boldsymbol{u}_E \tag{3.7}$$

where

$\boldsymbol{H}(\boldsymbol{x})$: $(N+6)\times(N+6)$ inertia matrix

$\boldsymbol{C}(\boldsymbol{x},\boldsymbol{v})$: $(N+6)\times(N+6)$ matrix specifying centrifugal and Corioli's effects

$\boldsymbol{g}(\boldsymbol{x})$: $(N+6)\times 1$ vector specifying gravity effect

$\boldsymbol{u}_E$: $(N+6)\times 1$ vector specifying generalized forces generated by external forces

Equation (3.7) represents a general form of the dynamic equation.

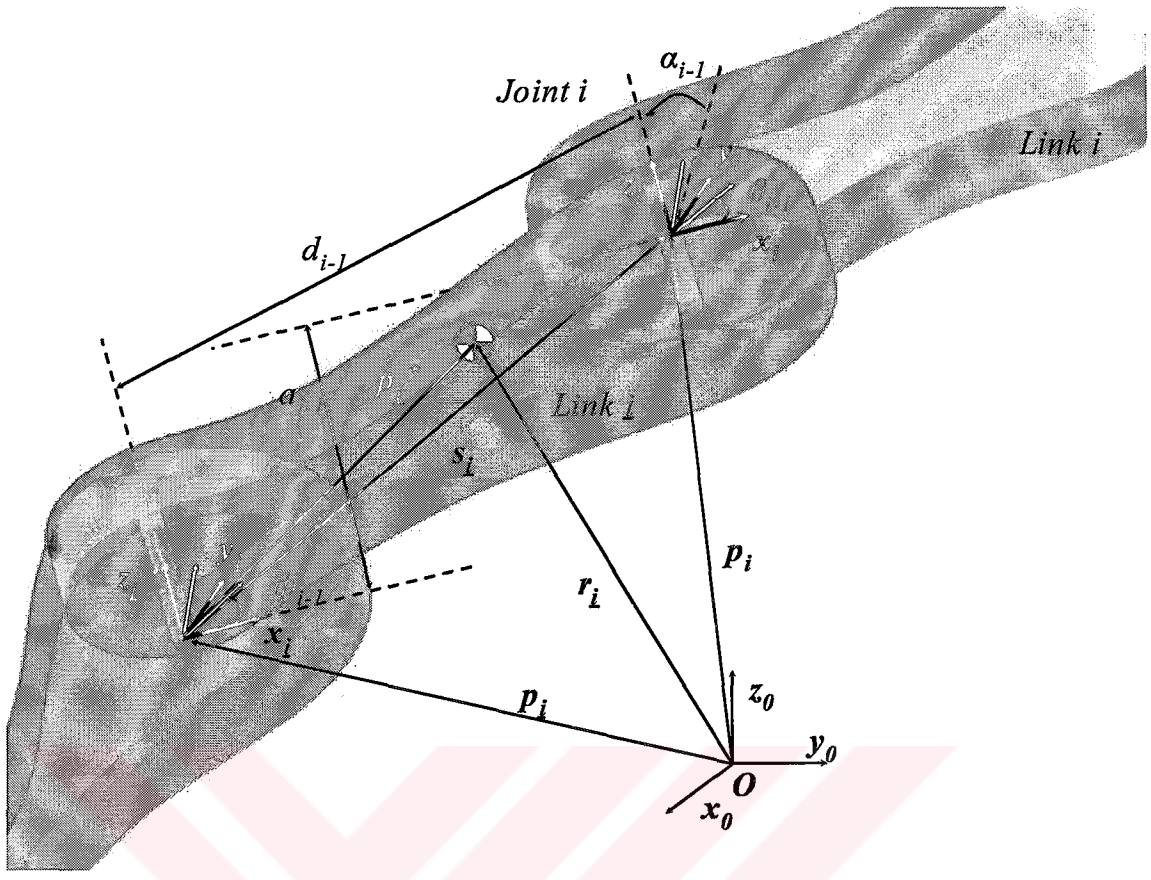

Figure 3.6. Coordinates and parameters of links.

To obtain the motion of the biped robot, inverse dynamics of the robot should be calculated by the following recursive equations. The formulation is performed in the algorithm frame of the [28, 29] with some difference. To compute inverse dynamics efficiently, the link-fixed coordinates shown in Fig. 3.6 are introduced.

The i^{th} coordinates are defined as

$$ {}^{0}A_i = {}^{0}A_1 {}^{1}A_2 \ldots {}^{i}A_i \tag{3.8} $$

$$ {}^{0}A_i = A_B \tag{3.9} $$

where

$$ {}^{i}A_i = \left[{}^{i}x_i, {}^{i}y_i, {}^{i}z_i \right]. \tag{3.10} $$

${}^{i}A_i$ transforms any vector with reference to i^{th} coordinate system to a new coordinate system whose coordinates are parallel to $\underline{i}^{th}$ coordinate system. Note that ${}^{i}A_i^{-1} = {}^{i}A_i^{T} = {}^{i}A_{\underline{i}}$ since coordinate system is orthogonal. The transformation can be parameterized by

$$ {}^{i}A_i = \begin{bmatrix} c_{\phi_{i-1}}c_{\theta_{i-1}} - s_{\phi_{i-1}}c_{\alpha_{i-1}}s_{\theta_{i-1}} & -c_{\phi_{i-1}}s_{\theta_{i-1}} - s_{\phi_{i-1}}c_{\alpha_{i-1}}c_{\theta_{i-1}} & s_{\phi_{i-1}}s_{\alpha_{i-1}} \\ s_{\phi_{i-1}}c_{\theta_{i-1}} + c_{\phi_{i-1}}c_{\alpha_{i-1}}s_{\theta_{i-1}} & -s_{\phi_{i-1}}s_{\theta_{i-1}} + c_{\phi_{i-1}}c_{\alpha_{i-1}}c_{\theta_{i-1}} & -c_{\phi_{i-1}}s_{\alpha_{i-1}} \\ s_{\alpha_{i-1}}s_{\theta_{i-1}} & s_{\alpha_{i-1}}c_{\theta_{i-1}} & c_{\alpha_{i-1}} \end{bmatrix}. \tag{3.11} $$

When $\dot{v}_B, \dot{w}_B, w_B, \ddot{q}_i, \dot{q}_i$ and $q_i, 1 \le i \le N$, are given, for i=1, the angular velocity ${}^{i}w_i$, the angular acceleration ${}^{i}\dot{w}_i$ and the acceleration of the origin ${}^{i}\ddot{p}_i$ of i^{th} link referenced to its own link coordinates can be recurrently obtained as follows.

$$ {}^{1}\ddot{p}_1 = {}^{1}A_0(\dot{v}_B + g) \tag{3.12} $$

$$ {}^{1}w_1 = {}^{1}A_0 w_B \tag{3.13} $$

$$ {}^{1}\dot{w}_1 = {}^{1}A_0 \dot{w}_B \tag{3.14} $$

and for $2 \le i \le N$

$$ {}^{i}w_i = {}^{i}A_{\underline{i}}\,{}^{\underline{i}}w_{\underline{i}} + z_0\dot{q}_{i-1} \tag{3.15} $$

$$ {}^{i}\dot{w}_i = {}^{i}A_{\underline{i}}\,{}^{\underline{i}}\dot{w}_{\underline{i}} + z_0\ddot{q}_{i-1} + ({}^{i}A_{\underline{i}}\,{}^{\underline{i}}w_{\underline{i}}) \times z_0\dot{q}_{i-1} \tag{3.16} $$

$$ {}^{i}\ddot{p}_i = {}^{i}A_{\underline{i}}\left[{}^{\underline{i}}\dot{w}_{\underline{i}} \times {}^{\underline{i}}p_i^* + {}^{\underline{i}}w_{\underline{i}} \times \left({}^{\underline{i}}w_{\underline{i}} \times {}^{\underline{i}}p_i^*\right) + {}^{\underline{i}}\ddot{p}_{\underline{i}}^*\right]. \tag{3.17} $$

Thus for $1 \le i \le N$, the acceleration of the center of mass $\ddot{r}_i$, the total force F_i, and the total moment N_i of the i^{th} link can be calculated as follows.

$$ {}^{i}\ddot{r}_i = {}^{i}\dot{w}_i \times {}^{i}s_i + {}^{i}w_i \times ({}^{i}w_i \times {}^{i}s_i) + {}^{i}\ddot{p}_i \tag{3.18} $$

$$ {}^{i}F_i = {}^{i}m_i\,{}^{i}\ddot{r}_i \tag{3.19} $$

$$ {}^{i}N_i = {}^{i}J_i\,{}^{i}\dot{w}_i + {}^{i}w_i \times ({}^{i}J_i \times {}^{i}w_i), \tag{3.20} $$

where

$$ {}^{\underline{i}}p_i^* = p_i - p_{\underline{i}} = [a_{i-1}\cos\phi_{i-1}, a_{i-1}\sin\phi_{i-1}, d_{i-1}]^T. \tag{3.21} $$

z_i denotes the direction of the joint i-1, and s_i denotes the center of mass with respect to the i^{th} coordinate system origin (Fig. 3.6). The gravity effect can be considered by adding a gravitional acceleration on one of the base links' $\ddot{v}_B$.

Then, for $1 \le i \le N$ f_i and n_i, the force and moment exerted on link i by inner link $\underline{i}$ can be calculated as follows.

$$ {}^{i}f_i = {}^{i}F_i + \sum_{j \in O_i} {}^{\underline{j}}A_j\,{}^{j}f_j + {}^{i}A_0 \sum_{j \in M_i} {}^{0}f_{E_j} \tag{3.22} $$

$$ {}^{i}n_i = {}^{i}N_i + \sum_{j \in O_i}\left[{}^{\underline{j}}A_j\,{}^{j}n_j + {}^{\underline{j}}p_j^* \times ({}^{\underline{j}}A_j\,{}^{j}f_j)\right] + {}^{i}s_i \times {}^{i}F_i - \sum_{j \in M_i}\left[{}^{j}c_j \times ({}^{i}A_0\,{}^{0}f_{E_j})\right] \tag{3.23} $$

where

f_{E_j} : 3×1 vector specifying j^{th} external force

M_i : A set of index numbers of external forces which are imposed on link i

c_j : 3×1 position vector of j^{th} contact point with respect to the origin of its own link-fixed coordinates as shown in Fig. 3.7.

As a result, given x, v and $\dot{v}$, for $1 \le i \le N-1$ the solution of inverse dynamics $u^T = \left[f_B^T, n_B^T, \tau_B^T\right]$ is

$$\tau_i = {}^{i+1}n_{i+1}^T z_0 + D_i q_i, \tag{3.24}$$

where τ_i denotes the torque on i^{th} joint, D_i denotes the viscous damping coefficient of joint i; and for i=1,

$$f_B = {}^0A_1 \, {}^1f_1 \tag{3.25}$$

$$n_B = {}^iA_1 \, {}^1n_1 \tag{3.26}$$

where f_B and n_B are the force and the moment exerted on the origin of the base link, respectively.

In the simulation the trajectories of the joint torques and the interaction forces from the ground given the position, velocity and acceleration of the joint angles and

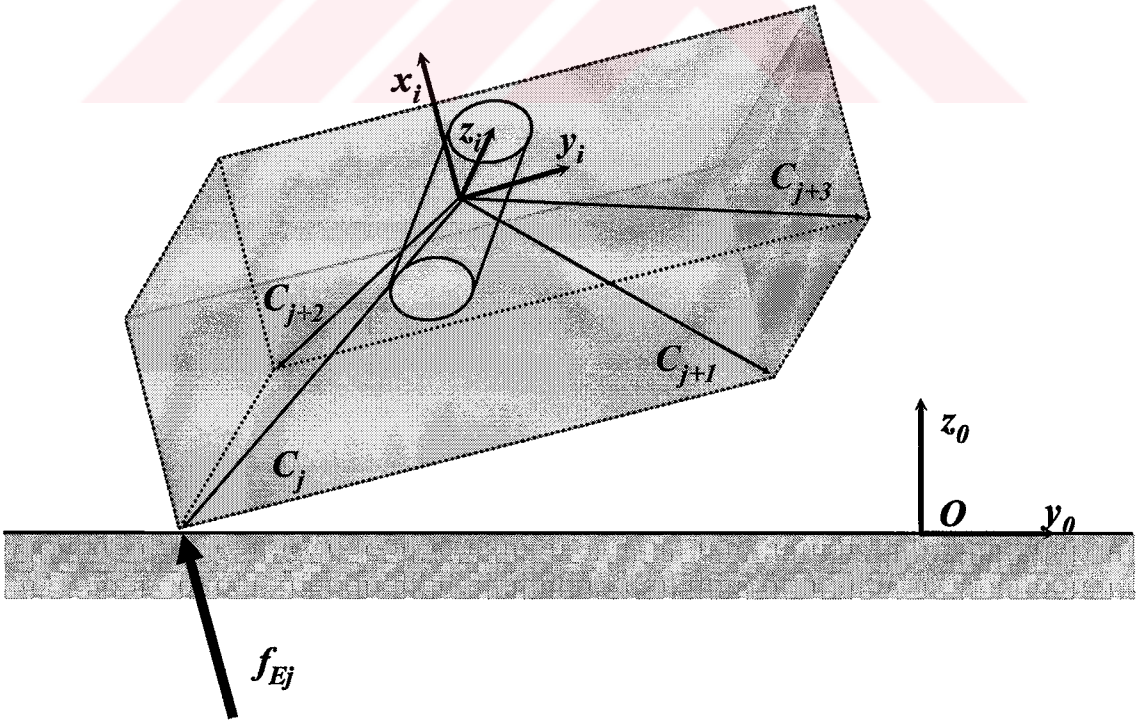

Figure 3.7. Contact points of the foot.

those of the body are calculated. In other words, the problem is the solution for the joint torque $\boldsymbol{\tau}$ and the external force $\boldsymbol{u}_E$ given the generalized coordinates $\boldsymbol{x}$, generalized velocities $\boldsymbol{v}$ and generalized acceleration $\dot{\boldsymbol{v}}$ in equations (3.1)-(3.7). When $\boldsymbol{x}$, $\boldsymbol{v}$ and $\dot{\boldsymbol{v}}$ are given, left side of (3.7) can be calculated using the Newton-Euler method described above by setting external forces $\boldsymbol{f}_E = 0$. Let the results of the calculation of the left side be equal to $\boldsymbol{u}_a$,

$$\boldsymbol{u}_a(\boldsymbol{x},\boldsymbol{v},\dot{\boldsymbol{v}}) = \boldsymbol{H}(\boldsymbol{x})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g}(\boldsymbol{x}) \tag{3.27}$$

which corresponds to the generalized forces generated by the inertial force, centrifugal forces, Corioli's forces and gravity effects.

$$\boldsymbol{u}_E = \sum_{j \in M_A} \boldsymbol{K}_j \boldsymbol{f}_{E_j} = \boldsymbol{K}\boldsymbol{f}_E \tag{3.28}$$

where

$$M_A = \bigcup_{i=1}^{N} M_i \tag{3.29}$$

$\boldsymbol{K}_{Ej}$: $(N+6)\times 3$ matrix specifying transforms from j^{th} external force to generalized forces

M_A : A set of index numbers of all active contact points

$\boldsymbol{f}_E$: $(3M)\times 1$ vector which contains active contact forces

$\boldsymbol{K}$: $N \times (3M)$ matrix specifying transforms from $\boldsymbol{f}_E$ to generalized forces

M : Number of time-variant active contact points.

From Equations (3.7) and (3.28)

$$\boldsymbol{u}_a = \boldsymbol{u} + \boldsymbol{K}\boldsymbol{f}_E . \tag{3.30}$$

$\boldsymbol{K}$ can be calculated by solving inverse dynamics mentioned above with setting $\boldsymbol{x}$ to the current state, $\boldsymbol{f}_E = \boldsymbol{e}_j$, $\dot{\boldsymbol{v}} = 0$, and ignoring gravity, centrifugal and Corioli's effects. (3.30) can be rewritten as,

$$\begin{bmatrix} \boldsymbol{f}_a \\ \boldsymbol{n}_a \\ \boldsymbol{\tau}_a \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_B \\ \boldsymbol{n}_B \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{K}_f \\ \boldsymbol{K}_n \\ \boldsymbol{K}_\tau \end{bmatrix} \boldsymbol{f}_E \tag{3.31}$$

where

$$\boldsymbol{u}_a^T = \begin{bmatrix} \boldsymbol{f}_a^T & \boldsymbol{n}_a^T & \boldsymbol{\tau}_a^T \end{bmatrix} \tag{3.32}$$

$$\boldsymbol{K}^T = \begin{bmatrix} \boldsymbol{K}_f^T & \boldsymbol{K}_n^T & \boldsymbol{K}_\tau^T \end{bmatrix}. \tag{3.33}$$

Since there is no actuation force f_B and torque n_B on the biped, they must be zero:

$$f_B = n_B = 0. \tag{3.34}$$

Then (3.31) becomes,

$$\begin{bmatrix} f_a \\ n_a \end{bmatrix} = \begin{bmatrix} K_f \\ K_n \end{bmatrix} f_E \tag{3.35}$$

$$\tau_a = \tau + K_\tau f_E. \tag{3.36}$$

If,

$$K_{fn} = \begin{bmatrix} K_f \\ K_n \end{bmatrix} \tag{3.37}$$

has full row rank, the solution for the external force f_E exists in (3.35). The minimal norm solution of f_E is obtained by

$$f_E = K_{fn}^T (K_{fn} K_{fn}^T)^{-1} \begin{bmatrix} f_a \\ n_a \end{bmatrix}. \tag{3.38}$$

When the matrix K_{fn} has full column rank and does not have full row rank, there might be no solution for (3.35). In this case approximate solution which has minimal norm error is given by

$$f_E = (K_{fn}^T K_{fn})^{-1} K_{fn}^T \begin{bmatrix} f_a \\ n_a \end{bmatrix}. \tag{3.39}$$

Then the joint torque generated by the actuator can be obtained from (3.36) as

$$\tau = \tau_a - K_\tau f_E. \tag{3.40}$$

In order to calculate the joint torques the above mentioned procedure is followed in the written computer simulation program. The followed calculation loop for every step time can be listed such as:

1. Calculate the generalized force u_a numerically by solving inverse dynamics equations (3.8)-(3.26) and (3.27) with setting $(x, v, \dot{v})$ to current state and ignoring external forces.
2. Calculate the transformation matrix K numerically by solving inverse dynamics equations (3.8)-(3.26) with setting x to the current state, $f_E = e_j$ for $1 \le j \le M$ and $\dot{v} = 0$, and ignoring gravity, centrifugal, and Corioli's effects.

3. Calculate the external force $\boldsymbol{f}_E$ by solving (3.38) or (3.39) depending on the rank of $\boldsymbol{K}_{fn}$.

4. Calculate the joint torque $\boldsymbol{\tau}$ by solving (3.40).

5. Return to 1 for next simulation cycle.

Using this procedure, we can completely analyze joint torques of the biped and interaction forces from the ground when the trajectories of the position, the velocity and the acceleration of the joint angles, those of the body, and those of the body attitude are given.

With the calculated variables, independent joint PID controllers are employed to control the joint angles.

$$u = K_p \cdot \theta^{err} + K_d \cdot \frac{d\theta^{err}}{dt} + K_i \cdot \int \theta^{err} dt \tag{3.41}$$

Constants used in the PID control for the joints, which are numbered at Fig. 3.5., are tabulated at Table 3.3. As can be seen in the upper most plot in Fig. 3.3 the foot references are shifted in the x-direction by 11 centimeters back with respect to the body coordinate frame. This value is obtained through a number of simulations on a trial and error basis. This shift is termed the x-reference asymmetry in this work and it is one of the parameters defining the walking pattern. Other parameters are the step height, the step size, the body height, the trunk swing amplitude, the trunk swing offset, the single support period and the double support period [22]. All of these parameters as in the case of x-reference asymmetry are determined via simulations and via experiments on Mari-2 based on trial and error. The values of these parameters that are found are given in Table 3.4. These parameters values differ from the actual biped robots',

Table 3.3. PID Controller gains for the leg joints.

Joint Number	$\mathbf{K_p}$	$\mathbf{K_d}$	$\mathbf{K_i}$
1	12000	1	40
2	12000	1	40
3	20000	1	40
4	60000	1	40
5	60000	1	40
6	12000	1	40

Table 3.4. Some of the important simulation parameters

Parameter	Value	Parameter	Value
x-reference asymmetry	-0.11 m	body height	0.50 m
y-reference asymmetry	0.00 m	trunk swing amplitude	0.13 m
step height	0.06 m	trunk swing offset	0.11 m
step size	0.08 m	double support period	1.0 s
step time	0.0005 s	single support period	1.4 s

Mari-2, parameter values with ±5%, which means that built simulation is very close to the reality.

The previous version of the simulation was written in Matlab and simulation periods were relatively long. In order to increase the speed of calculation, an advanced matrix library is used in C++ environment and the simulation code that executes the above mentioned simulation scheme is rewritten. Some of the important functions of this code are given at the Appendices. The simulation time is decreased more than ten

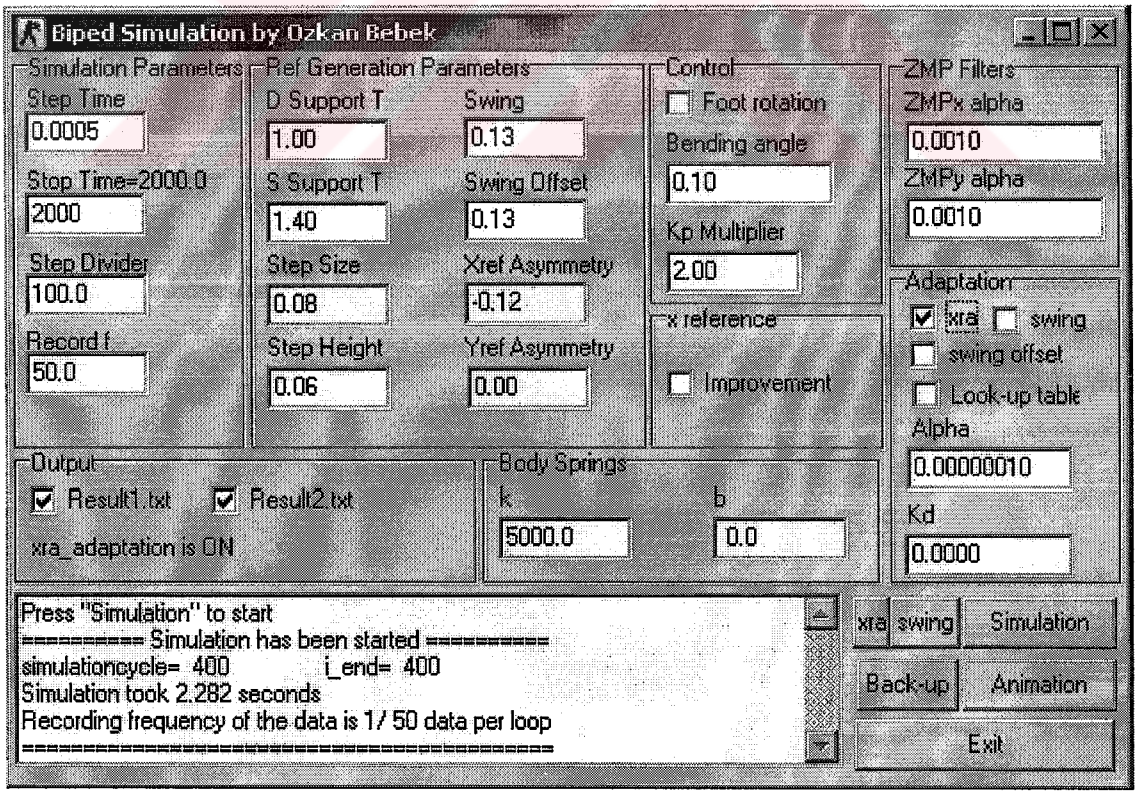

Figure 3.8. Screen shot of the Biped Simulation GUI

times. A 10-second-walk of the biped is simulated in 65 seconds using a personal computer equipped with an Intel Pentium ® IV – 2.0 GHz CPU.

In order to create an easy-to-use simulation package, a simple GUI that executes the simulation and animation is build. A screen shot is shown in Fig 3.8.

The GUI enables entering the desired parameter values to the value boxes before simulation. By this way successive simulations can be done faster. After simulation is complete, an informative message is displayed on the log window, giving some specifications of the executed simulation. These data and all simulation variables are also written to a log file after simulation is complete. The results of the simulation can be animated by clicking the animation button of the GUI. A screen shot from the animation window is shown in Fig. 3.9.

Simulation software also keeps the data of selected 128 different variables in the result files in order to be inspected later. These data are usually examined using Matlab© software.

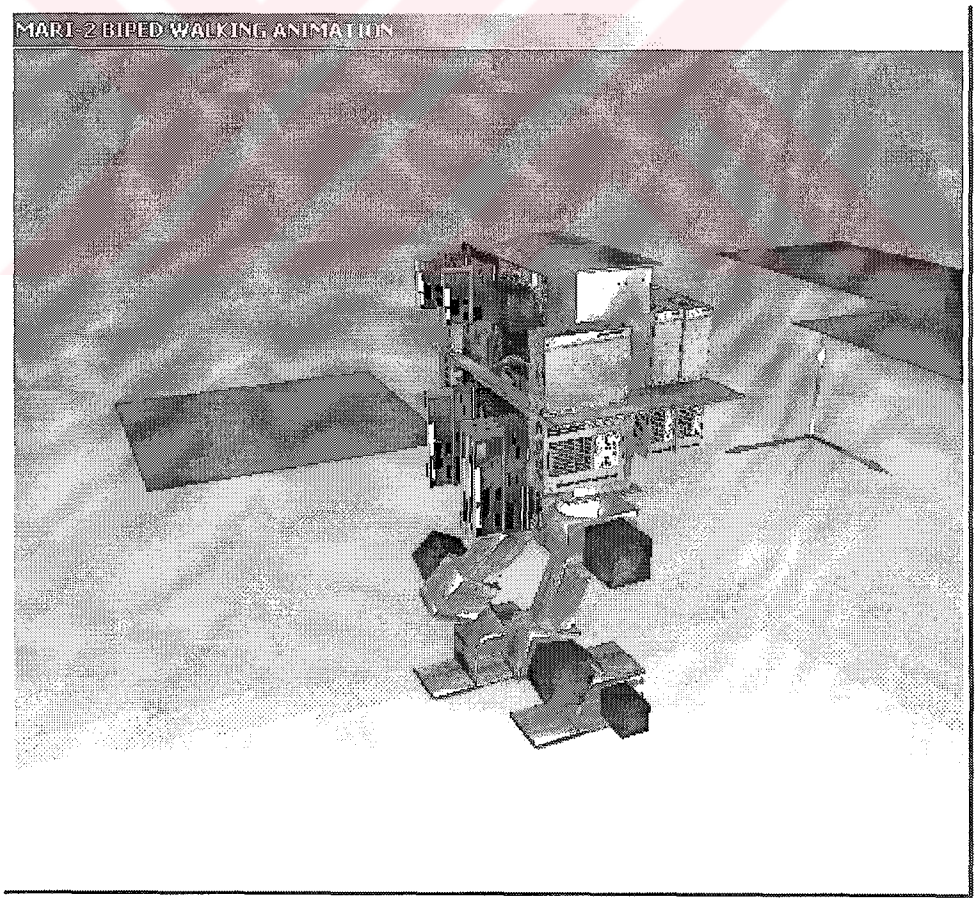

Figure 3.9. A screen shot from the Biped Animation

In the next chapter the idea of attaching virtual torsional springs will be described. Improvements gained after the attachment of the springs are shown by figures. Lastly online tuning ideas are discussed.

4. VIRTUAL SPRINGS FOR TUNING WALKING PARAMETERS

In the case of conventional trial and error tuning, if the tested parameter is inadequate the robot loses its balance unrecoverably and it falls. It is observed that in this kind of tuning, suitable ranges for gait parameters are very narrow, in the order of few millimeters. Both in implementations with the real robot and in simulation, it is impossible to test with values differing low than 1 mm. When the simulation starts with an unfeasible value of the parameter the robot falls even before completing the first

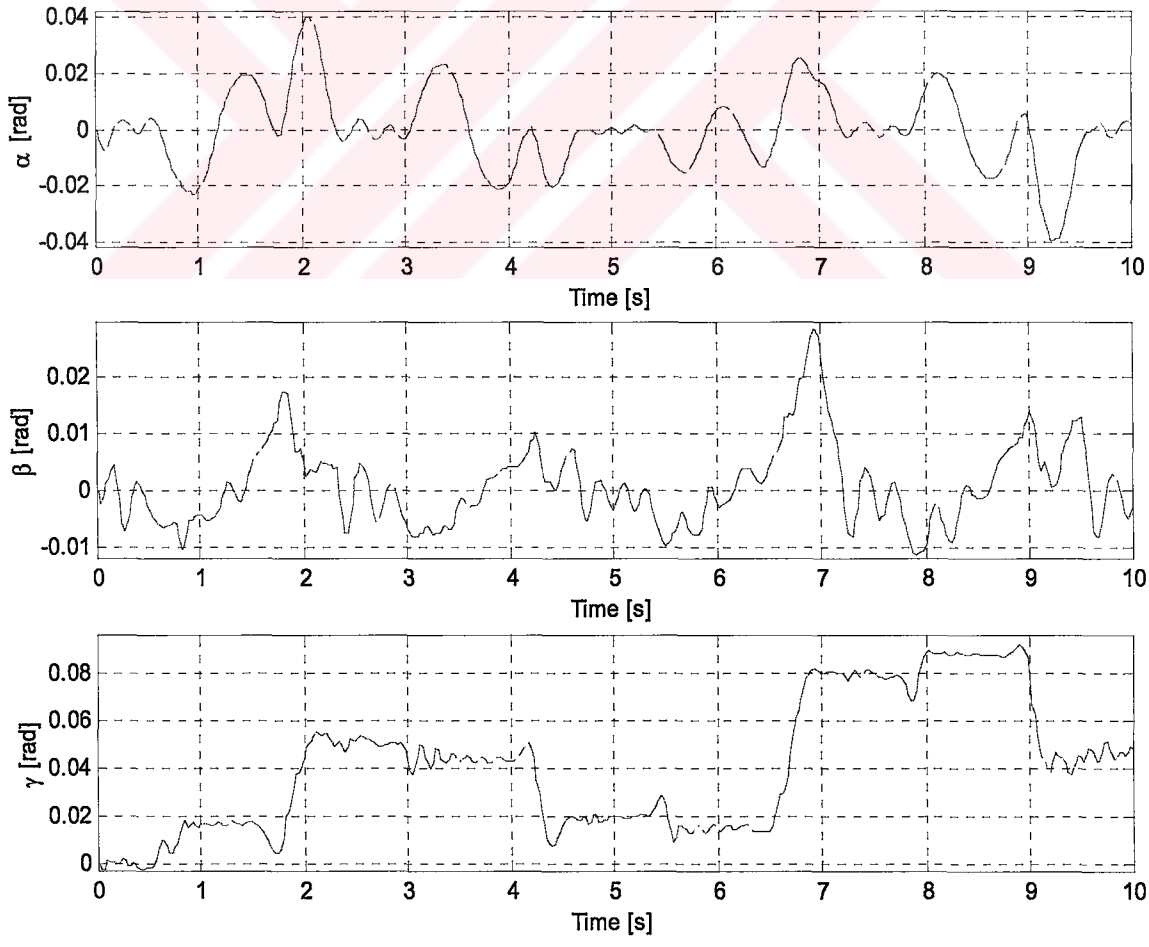

Figure 4.1. α,β and γ angles, with the trial and error based tuning.

reference gait cycle. An on-line tuning scheme however can only be successful if there is sufficient time for training without falling, which might last in some cases hundreds of cycles. This implies that a mechanism for keeping the robot in continuous walk, even when the parameter settings are totally wrong, is necessary during training.

A measure of the robot's balance is its body orientation with respect to a fixed coordinate frame. This orientation can be described by a set of yaw, pitch and roll angles denoted by γ, β and α respectively. Typical angular deflections of the trunk during a successful walk are shown in Fig. 4.1. In order to stabilize the motion of the biped, it is aimed to pull down these angles to zero. In this work, virtual torsional springs resisting against deviations of these angles from zero are employed as the mechanism for keeping the robots balance during training. The torsional springs (Fig 4.2) are attached to the body origin of the biped robot along the x, y and z axes (Fig 3.2). Fig. 4.3 shows the angular deflections of the trunk after the connection of the virtual

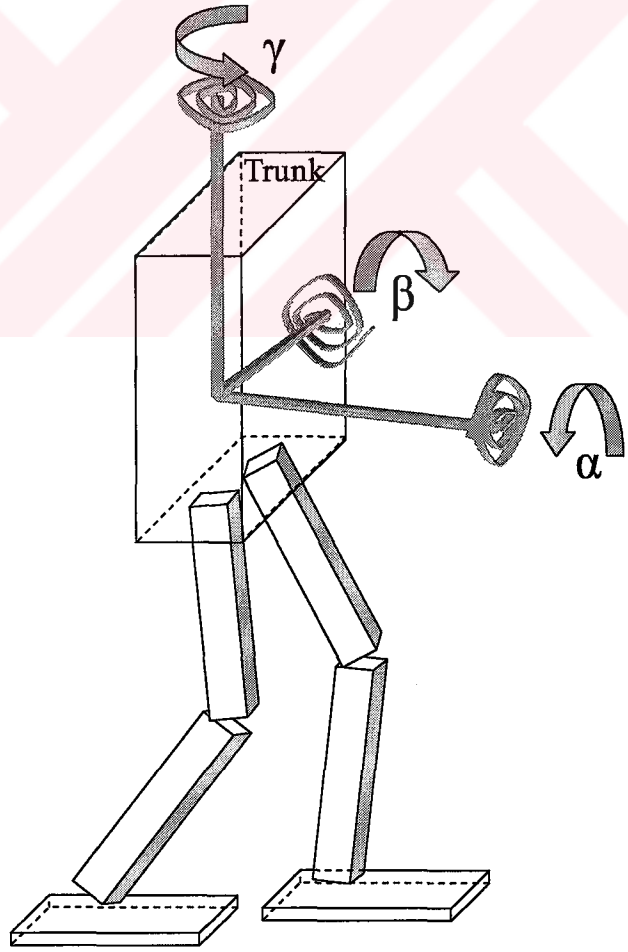

Figure 4.2. Virtual Torsional Springs attached to the Biped Trunk.

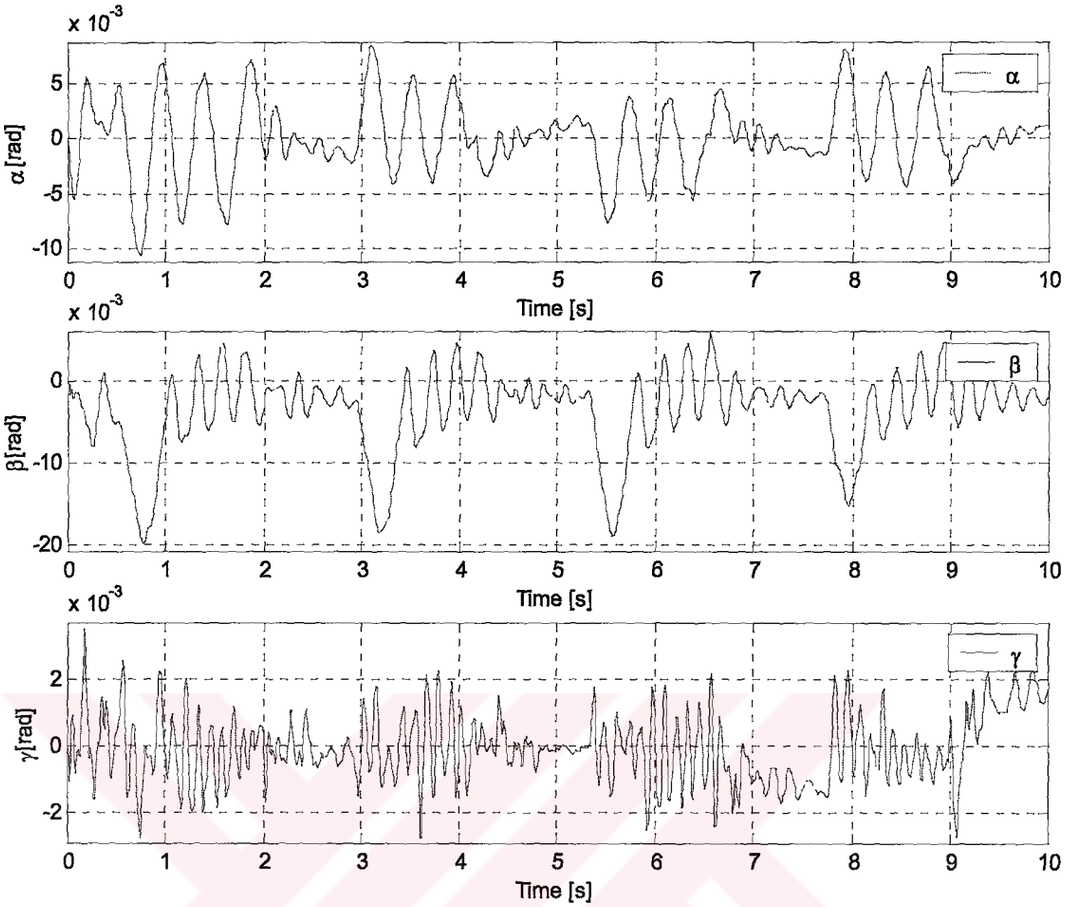

Figure 4.3. α,β and γ angles after the connection of virtual torsional springs.

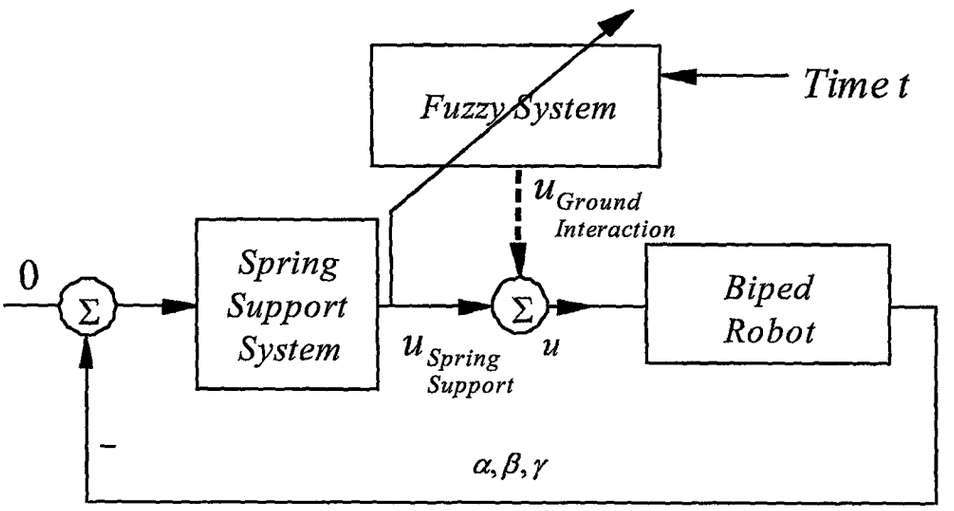

Figure 4.4. On-line identification and control with the feedback error-learning scheme.

torsional springs. The spring constant is adjusted to be to 5000 N/m by trial and error such that letting some movement space to the robot trunk but also decreasing the unwanted motion of the trunk to the desired level. The effect of the spring to the system can be described by the following equation,

$$u_{SpringSupport} = K_{spring} \cdot \beta . \tag{4.1}$$

This scheme can be described as, keeping the system in its balance orientation by external aid and letting system to reach the balanced orientation by tuning its parameters. This kind of approach is novel in the literature [30] and can be applied to other control systems that involve parameter tuning.

At this point it should also be noted that keeping the body angles as close as zero is not the unique way of improving the balance. Reported are studies in which deliberate variations of the trunk angles can serve as tools for the balance control [31,32].

In this thesis an online parameter adaptation system or alternatively a fuzzy system generating a functional over the reference cycle are employed for tuning. The fuzzy identifier scheme used in the tuning mechanism for the x-reference asymmetry is shown in Fig. 4.3. In this figure the $u_{Ground\ Interaction}$ is the torque induced on the trunk by the interaction of the robot feet and the ground. This torque depends on the selection of the various gait parameters. The fuzzy system is responsible to compute the so called x-reference asymmetry and therefore it contributes to the torque induced on the trunk. Also contributing are the virtual spring support torques.

The x-reference asymmetry parameter determines the average position of the feet with respect to the body coordinate frame during the walk. In order not to fall to the back or front this parameter should be adjusted.

In addition of supporting the robot and enabling a sufficient training period without falling, the pitch support torque (henceforth termed as β-torque) serves another purpose: It is used to tune the x-reference asymmetry parameter or the fuzzy system for the x-reference asymmetry computation.

This structure is similar to the one in [33] that is feed back error Neural Network learning scheme, also implemented via fuzzy identifiers in [34].

In this tuning scheme, the virtual springs are removed after the convergence of the x-reference asymmetry functional over the reference gait cycle.

The next chapter describes the three schemes used for the online tuning of the x-reference asymmetry parameter of the biped model.

5. THE BIPED GAIT PARAMETER TUNING SYSTEMS

In order to tune the x-reference asymmetry parameter satisfying the balance conditions of the biped robot, the following adaptation schemes are employed.

Firstly a constant value is searched using a simple adaptation scheme. Next, Neuro-Fuzzy systems are applied for functional approximation for a less conservative design. In order to test the validity of the Neuro-Fuzzy system, a one membership fuzzy identifier is used for adaptation in the second tuning scheme. In the last scheme, the tuned parameter is obtained as a functional over the reference cycle using adaptive fuzzy systems with 15 membership functions.

5.1. A Simple Adaptation Scheme for Tuning

First scheme used in adaptation is a simple application for parameter tuning.

$$xra(k+1) = xra(k) - \eta \cdot E \tag{5.1}$$

where

$$E = [\tau(\beta) - d]. \tag{5.2}$$

The error E selected for this purpose is β-torque with desired value d zero. η is the constant step size. This scheme is used for parameter tunings for the control systems. In this case, a constant value for the x-reference asymmetry parameter was searched.

5.2. Adaptive Fuzzy Systems in Tuning the Biped Gait Parameter

In this section, a fuzzy logic system is employed for the on-line computation of the x-reference asymmetry parameter. Further, on-line tuning of the fuzzy system is carried out via back-propagation.

The fuzzy systems are of the form (5.3).

$$f(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \left[\prod_{i=1}^{n} a_i^l \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} a_i^l \exp\left(-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]} \tag{5.3}$$

This function characterizes a fuzzy system with center average defuzzifier, product inference rule, singleton fuzzifier and Gaussian membership functions. Here M is the number of rules, $\bar{y}^l$ stands for the output constant of rule l, n is the number of input variables, x_i is the ith input variable, $\bar{x}_i^l$ is the center of the membership function for x_i for rule l, σ_i^l represents the width and a_i^l the height of this membership function. Gaussian membership functions are differentiable. This feature is required in the back-propagation algorithm. The function in (5.3) can be represented with a three-layer feed-forward neural network structure shown in Fig. 5.1 [35].

In Fig. 5.1, μ stands for the membership functions described above. Triangles represent gains.

With the motivation that systems of the form (5.3) are universal approximators [35], [35] develops a back-propagation training algorithm for this class of fuzzy systems as in the following.

For a given input-output pair $(\underline{x}^p, d)$ with $\underline{x}^p \in R^n$ and $d \in R$, a measure of the modeling error of a fuzzy model $f(x)$ of the form above can be defined by

$$E = \frac{1}{2}\left[f(\underline{x}^p) - d \right]^2. \tag{5.4}$$

In order to minimize this error, assuming that all the a_i^l terms are equal to 1, fuzzy system parameters will be varied according to the back-propagation rules below.

$$\begin{aligned} \bar{y}^l(k+1) &= \bar{y}^l(k) - \eta \left. \frac{\partial E}{\partial \bar{y}^l} \right|_k \\ &= \bar{y}^l(k) - \eta \frac{f - d}{b} z^l \end{aligned} \tag{5.5}$$

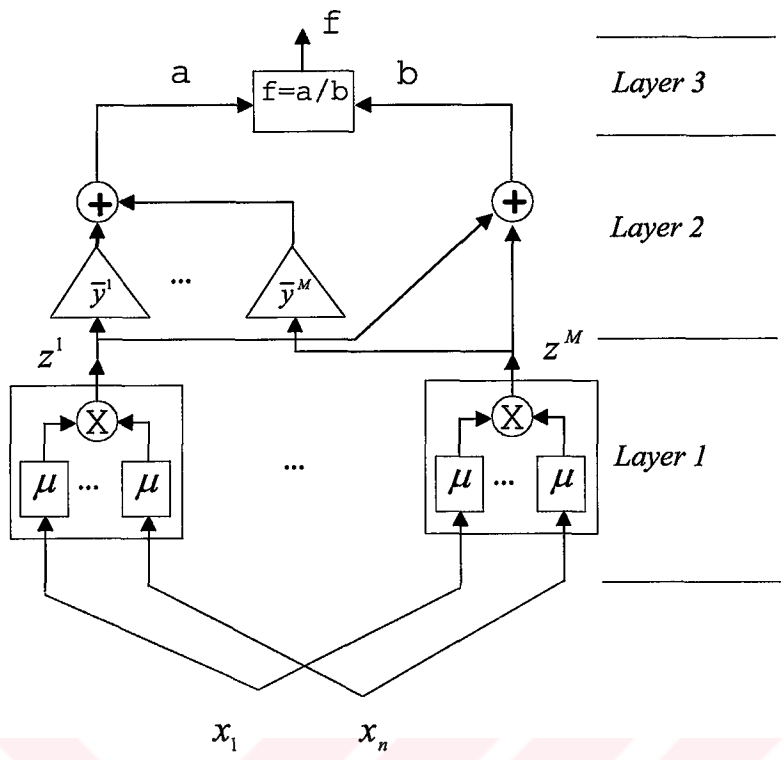

Figure 5.1. The three-layer feed-forward Neural Network architecture

$$\bar{x}_i^l(k+1) = \bar{x}_i^l(k) - \eta \left. \frac{\partial E}{\partial \bar{x}_i^l} \right|_k$$

$$= \bar{x}_i^l(k) - \eta \frac{f-d}{b}\left(\bar{y}^l - f\right) z^l \frac{2\left(x_i^p - \bar{x}_i^l(k)\right)}{\sigma_i^{l\,2}(k)} \tag{5.6}$$

$$\sigma_i^l(k+1) = \sigma_i^l(k) - \eta \left. \frac{\partial E}{\partial \sigma_i^l} \right|_k$$

$$= \sigma_i^l(k) - \eta \frac{f-d}{b}\left(\bar{y}^l - f\right) z^l \frac{2\left(x_i^p - \bar{x}_i^l(k)\right)^2}{\sigma_i^{l\,3}(k)} \tag{5.7}$$

Here, η is a constant step size. The variable b is defined in Fig. 5.1 and f stands for the function $f(\underline{x}^p)$ in (5.4).

In the used test bed robot, motion is achieved with open loop walking, although it suffers from a number of problems. Firstly it lacks the feedback to compensate uncertainties in the environment. Another drawback is that the values of the parameters

are constant over the gait cycle and hence this results in a conservative design. The last but not the least is the difficulty in carrying out the numerous simulations to obtain a set of parameters which harmoniously accomplish a stable walk. Nonetheless, this approach is reported as successful in obtaining gait parameters in both simulation and experimental studies [25]. Specifically, a functional over the gait cycle rather than a constant is sought for the x-reference asymmetry parameter for a less conservative design, and an automatic tuning algorithm is developed to replace the trial and error tuning method.

The input of the fuzzy system is selected to be the time in the reference cycle since all the reference functions are based on periodic time functions that repeat themselves. An optimum x-reference asymmetry functional (over the reference gait cycle) that makes walking the most stable is aimed. The difference term in the fuzzy identifier mechanism, that is, $f - d$ in (5.4), is selected as the β-torque. An indication of the stable walk of the biped is that the contribution of the support springs is kept to a minimum. Since the effect of inadequate x-reference asymmetry values can be characterized by falling back or forward, it is reasonable that, the best performing x-reference asymmetry parameter can be obtained by minimizing the β-torque which directly resists against back or forward falling.

5.2.1. Tuning with One Membership Function

In order to test the applicability of the proposed Fuzzy-Neural system, adaptation is applied using only one membership function. Back propagation is applied for only the output constant of rule $\bar{y}^l$ and width σ_i^l of membership function. Center of the membership function $\bar{x}_i^l$ is placed at 2.4 s, which is the mid point of a reference cycle.

The plot for the used Gaussian function is given in Fig 5.2. Since there is only one membership function, (5.3) and (5.5) becomes:

$$f(\underline{x}) = \bar{y}^l . \tag{5.8}$$

$$\bar{y}^l(k+1) = \bar{y}^l(k) - \eta(f - d) \tag{5.9}$$

which is the same adaptation scheme used in section 5.1. So the results of these two schemes should be the similar to each other.

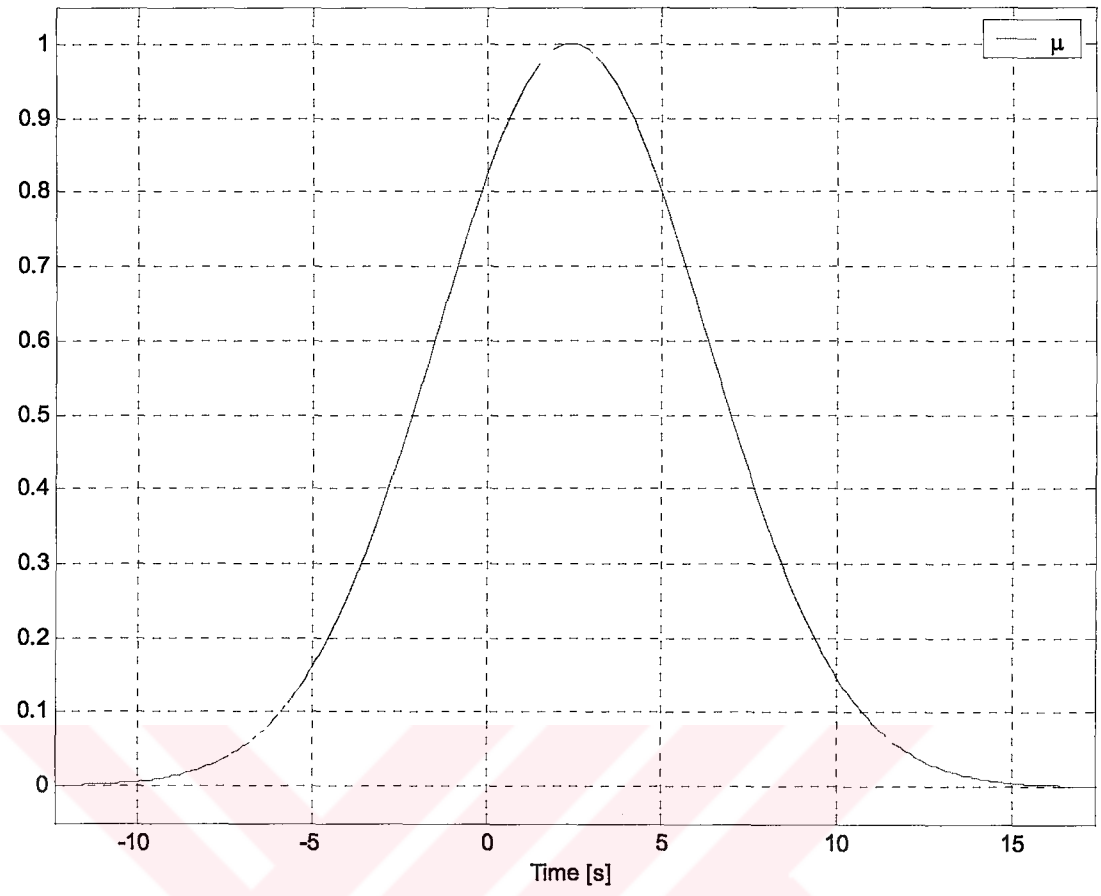

Figure 5.2. Plot of the Gaussian membership function used for test purpose of the Neuro-Fuzzy Scheme before training.

5.2.2. Tuning with More Than One Membership Functions

Tuning using more than one membership functions is described in this section. Again the centers of the Gaussian membership functions $\bar{x}_i^l$ are fixed and the back propagation is applied to output constant of rules $\bar{y}^l$ and width σ_i^l of membership functions. In this scheme, it is desired to obtain a functional of x-reference asymmetry. For this reason, 15 membership functions are used for approximation. This amount of membership functions is a feasible number for both computation speed and accuracy. The centers of the functions are placed between -4.8 s to 9.6 s with equal intervals. Plot of the membership functions is given at Fig. 5.3.

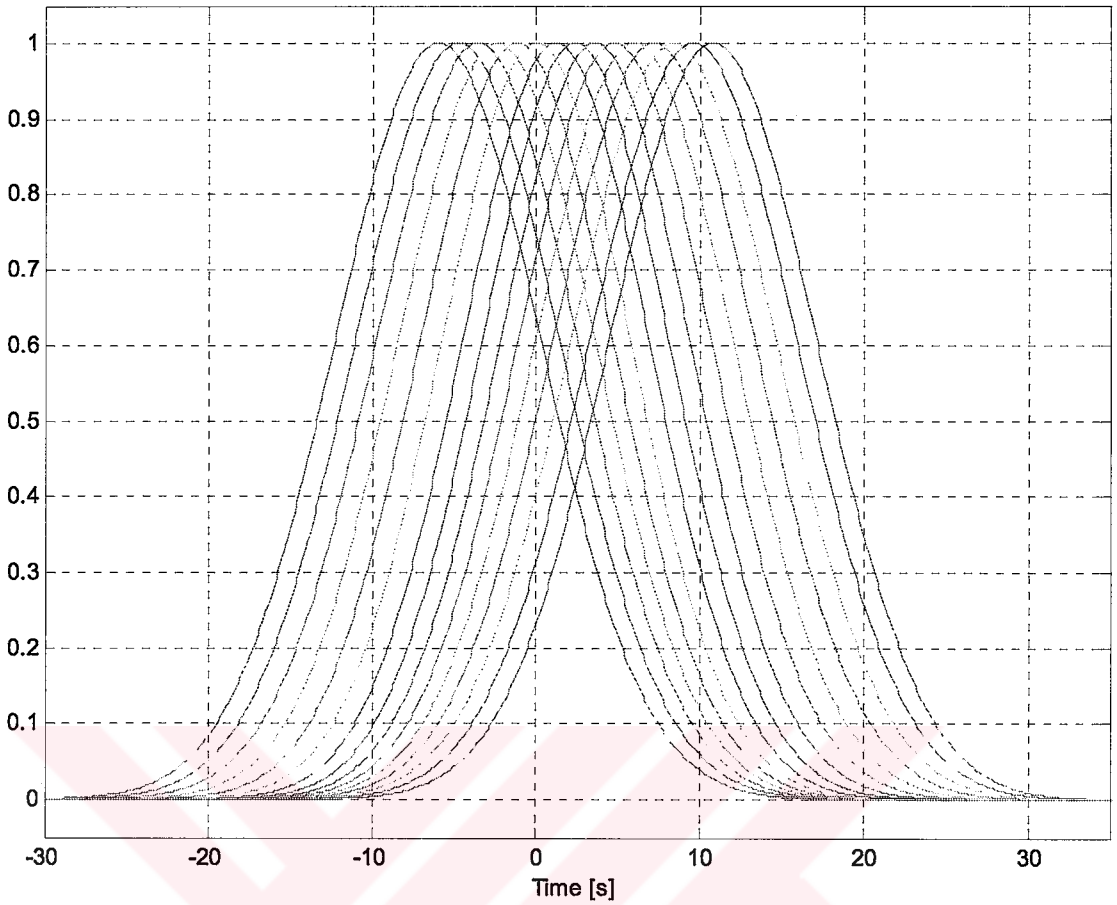

Figure 5.3. Plot of the 15 Gaussian membership functions used for the Neuro-Fuzzy Scheme before training.

In the next chapter the simulation results of the adaptation schemes described above is presented.

6. SIMULATION RESULTS

The simulations are carried out in 3-D using C++ with sampling time of 0.5 milliseconds with Euler integration. The simulation scheme is similar to the one in [36], which generalizes the recursive dynamic modeling method in [28] to the tree structure. In order to visualize the walking, simulation results are animated using an OpenGL based animation environment. A snapshot of the animation is shown in Fig. 3.9, also two steps of motion of the biped with screen shot intervals of 0.2 s are shown in Fig. 6.1.

6.1. Virtual Torsional Springs

In order to test the effect of the torsional springs on the balance of the robot, firstly, simulations without the adaptation are performed with the x-reference asymmetry parameter obtained by the trial and error tests, which is -11 centimeters. The footprints of the robot with and without the virtual spring support are shown in Fig. 6.1 and Fig. 6.2, respectively. It can be observed that the addition of the torsional springs improves the quality of the walk by forcing the robot to walk on a straight line. Further the deflections of the body yaw pitch and roll angles with and without torsional springs are inspected and it is observed that these deflections are reduced when the support springs are attached.

After these simulations, which justify the positive effects of the virtual support springs on the robot balance, a new set of simulations are carried out by turning on the adaptation systems described in Chapter 5.

6.2. Parameter Adaptation Scheme

In the simulations for parameter tuning, where (5.1) - (5.2) are employed, the

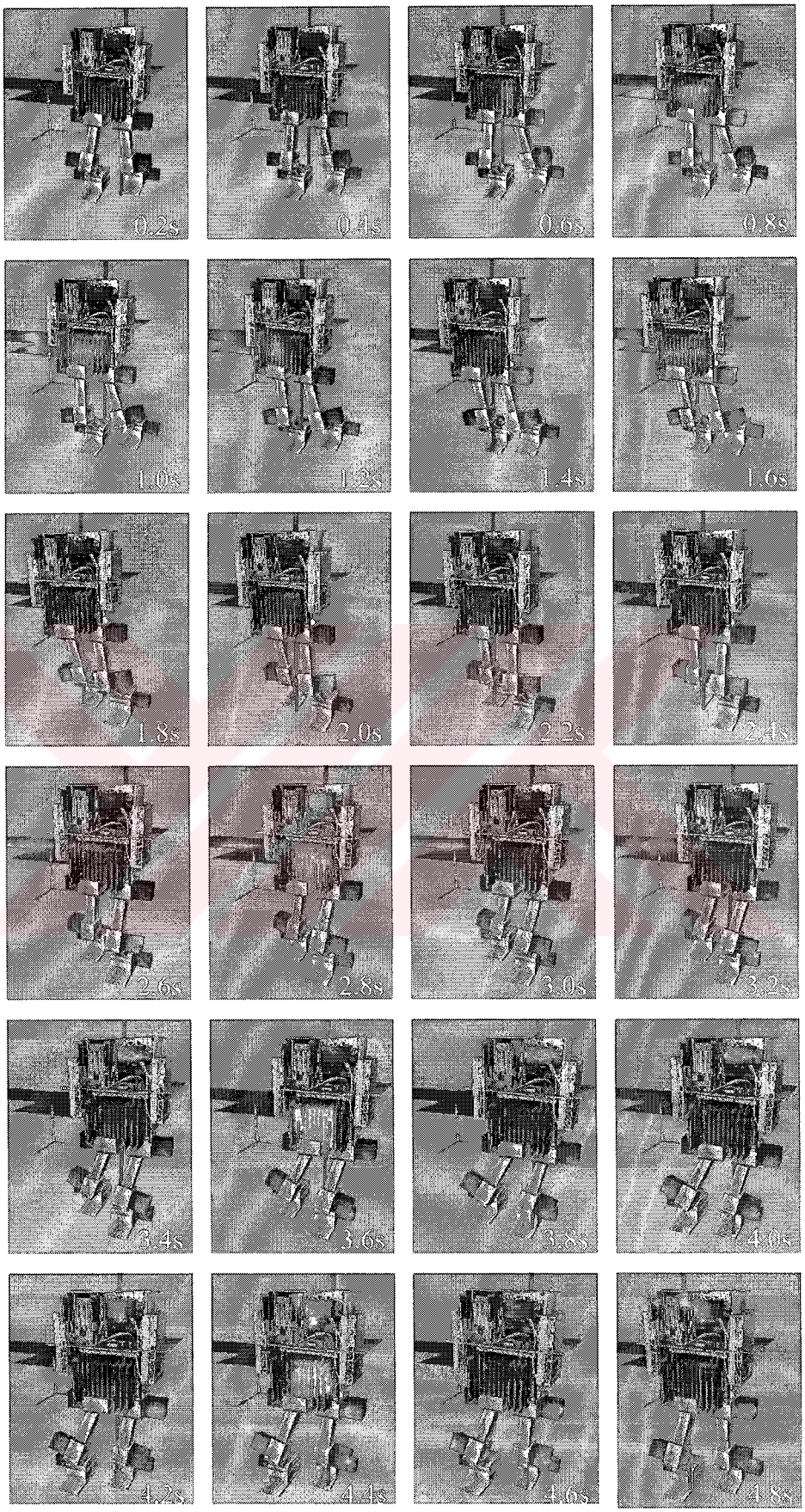

Figure 6.1. Two steps of animation sequence, (4.8 s).

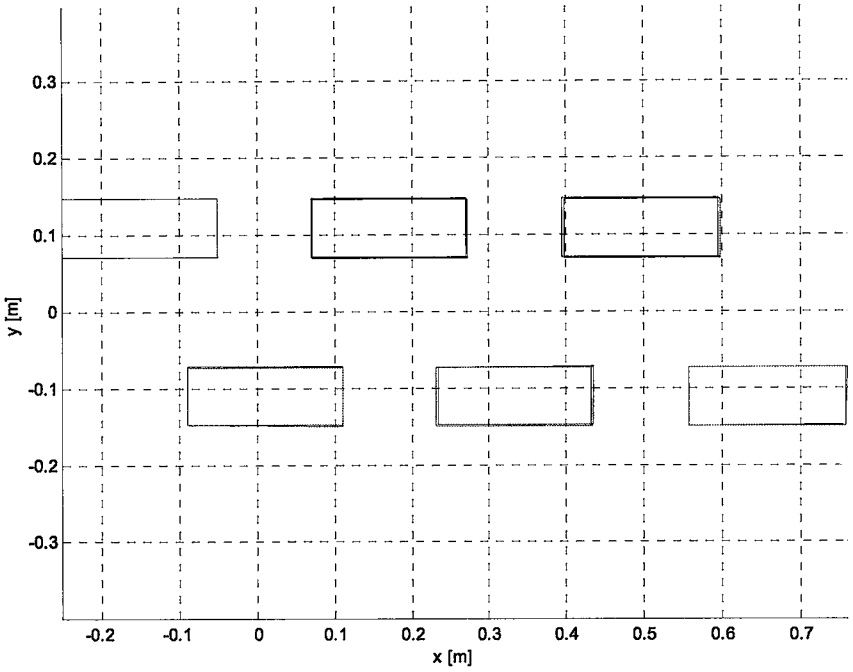

Figure 6.2. Foot positions of the biped in *x-y* plane after the connection of the support springs

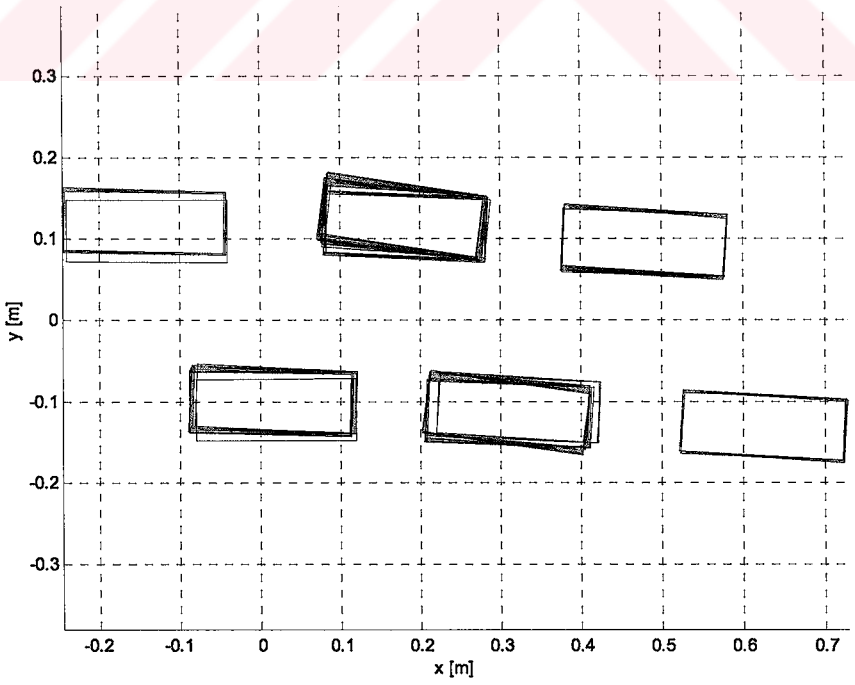

Figure 6.3. Foot positions of the biped in *x-y* plane before the connection of the support springs

initial x-reference asymmetry values are both taken far apart from its suitable value obtained by the trial and error tests. Shown in Fig. 6.4 and in Fig 6.5 are the convergence of the x-reference asymmetry parameter after 1000 seconds simulation time, where the initial value of the parameter used was chosen as zero and -11 centimeters respectively. In both cases the steady state value of the x-reference asymmetry is same with negligible deviations. The total simulation time has been 2000 seconds and no significant deviations from the learned parameter are observed in the last 1000 seconds. The learning rate may seem relatively low, however system does not allow faster learning rates because of the physical constraints. Fast parameter changes may result with positions of the foot centers that are out of the workspace of the feet. Hence constant step size η, also called as learning rate, is selected as 10^{-7} by trial and error. It is worth noting that the initial value zero is much different than the value obtained by trial and error tests that is -11 centimeters.

Considering our aforementioned observation that the parameter is highly sensitive to deviation from that value, it is easy to conclude that without the support springs it would be impossible to keep the 1000 seconds steady walk, which was necessary for the automatic tuning of the parameter. Simulations are carried out with a number of different x-reference asymmetry initial values. In each of these simulations the parameter functional converged to the same trajectory. The variation of the x-reference asymmetry for the last reference period is shown at Fig. 6.6 and Fig. 6.7.

6.3. Adaptive Fuzzy Scheme with One Membership Function

As described in Section 5.2, the adaptive fuzzy scheme is tested firstly with one membership function before going into studies with more membership functions. Similar results with the previous scheme that are explained in Section 5.1, are expected. Using the same initial conditions and constant step size η, simulations are repeated. The plots given at the previous section in the same order are shown in Figures 6.8, 6.9, 6.10 and 6.11. As expected, the results are similar to the simple adaptation scheme that is described in Section 5.1.

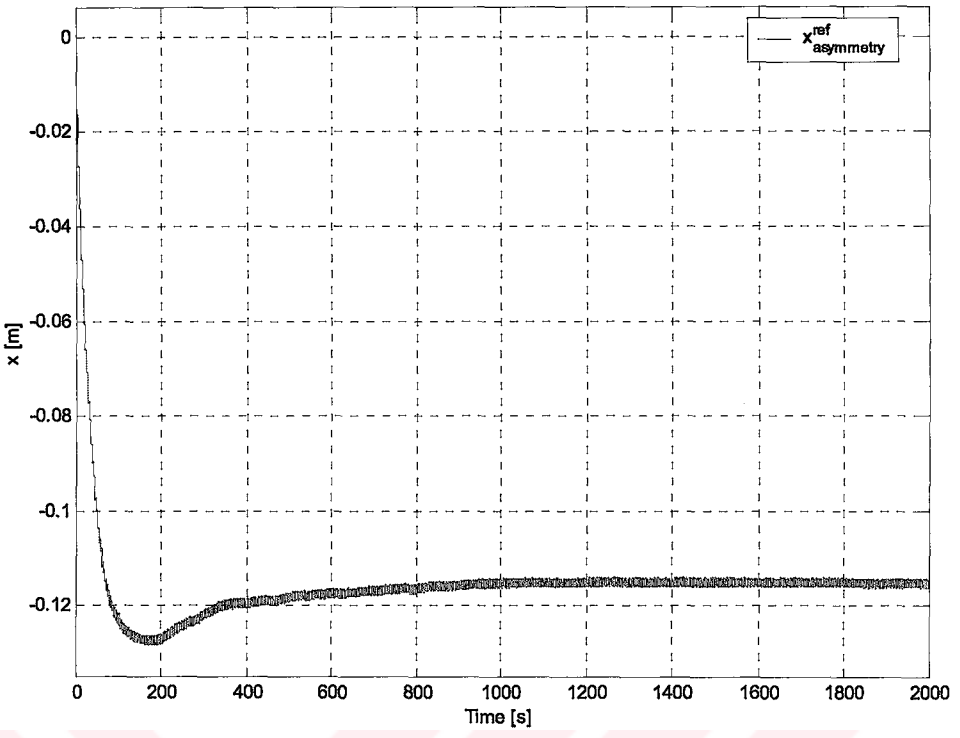

Figure 6.4. *x*-reference asymmetry versus time plot through the learning walk of the biped with initial value xra_0=0.

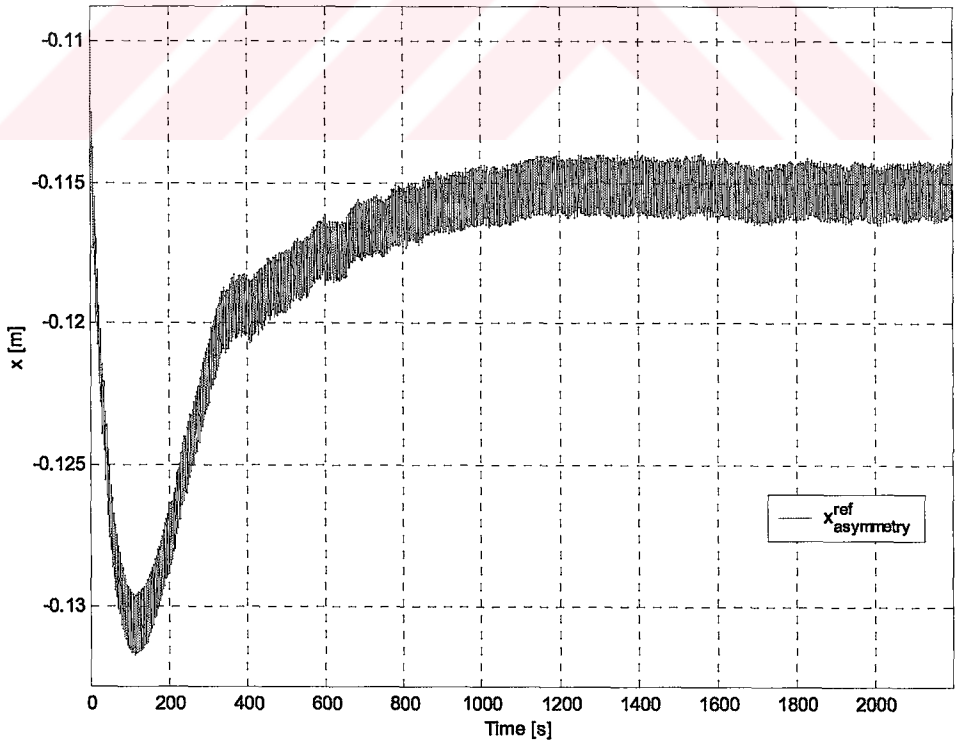

Figure 6.5. *x*-reference asymmetry versus time plot through the learning walk of the biped with initial value xra_0=-11 cm.

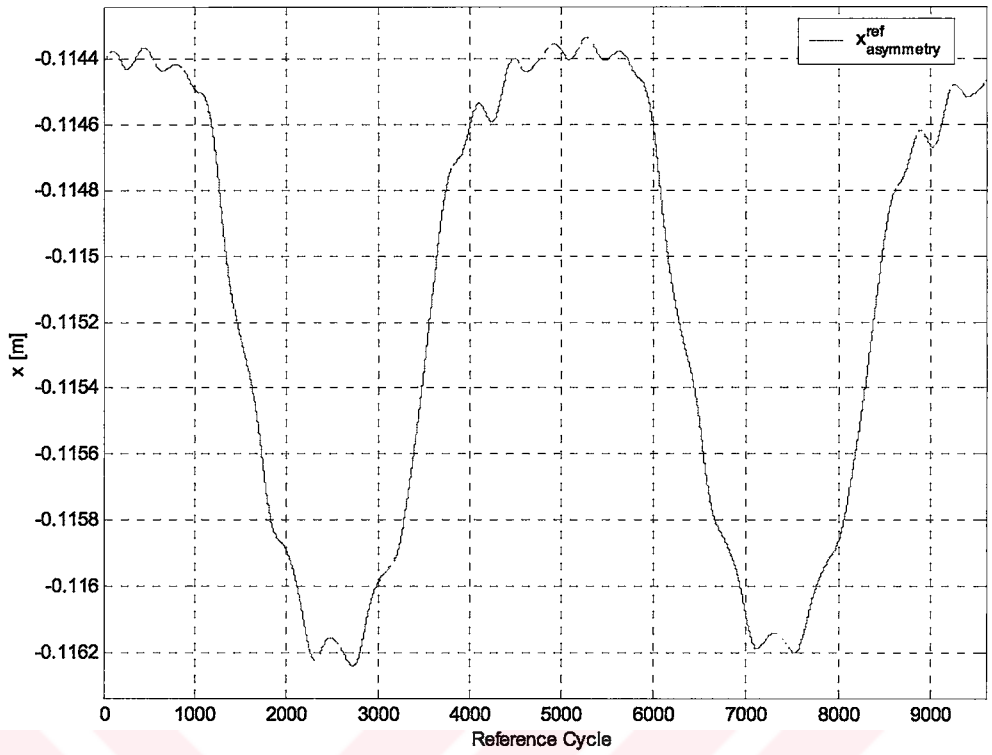

Figure 6.6. Simulation result of x-reference asymmetry for the last reference cycle of the learning walk of the biped with initial value xra_0=0.

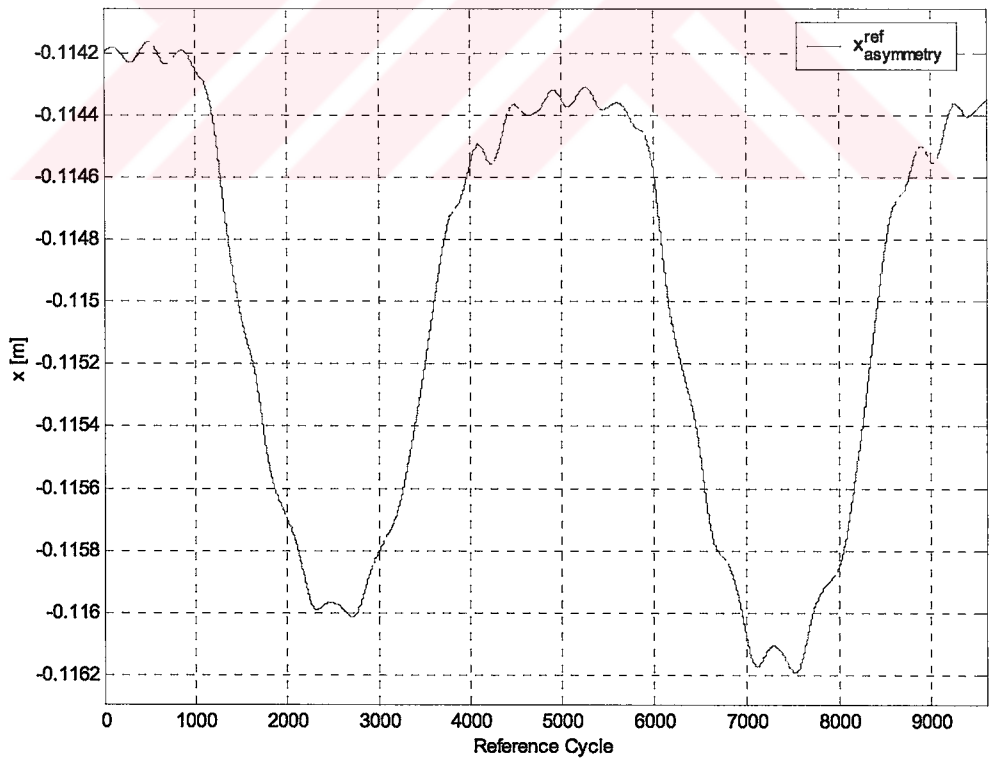

Figure 6.7. Simulation result of x-reference asymmetry for the last reference cycle of the learning walk of the biped with xra_0=-11cm.

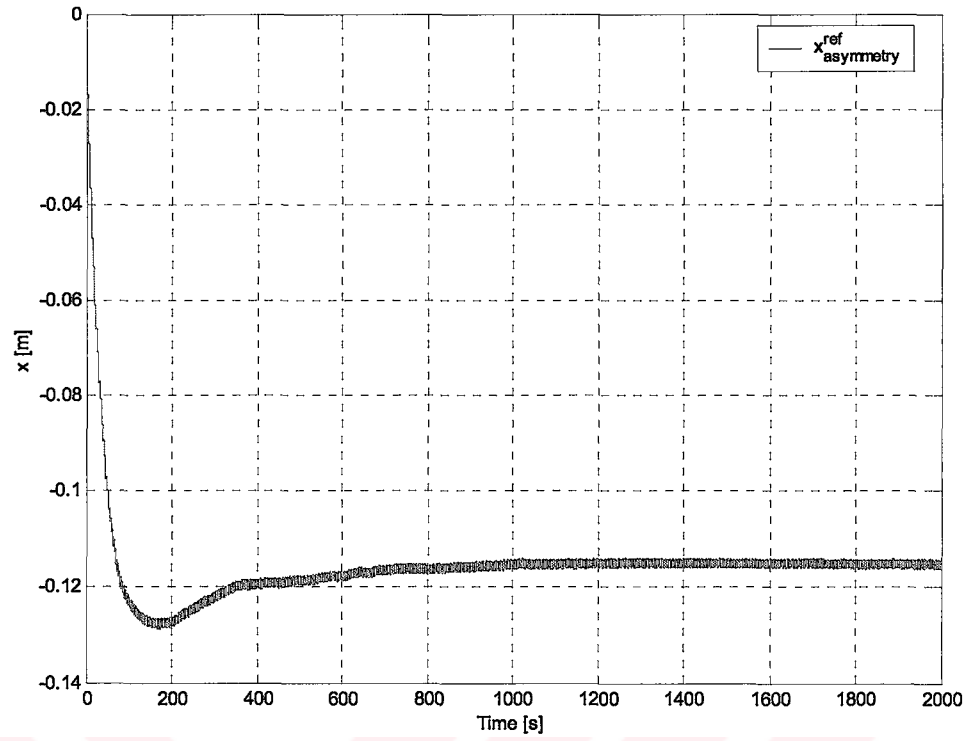

Figure 6.8. x-reference asymmetry versus time plot through the learning walk of the biped using one Gaussian membership function with initial value xra_0=0.

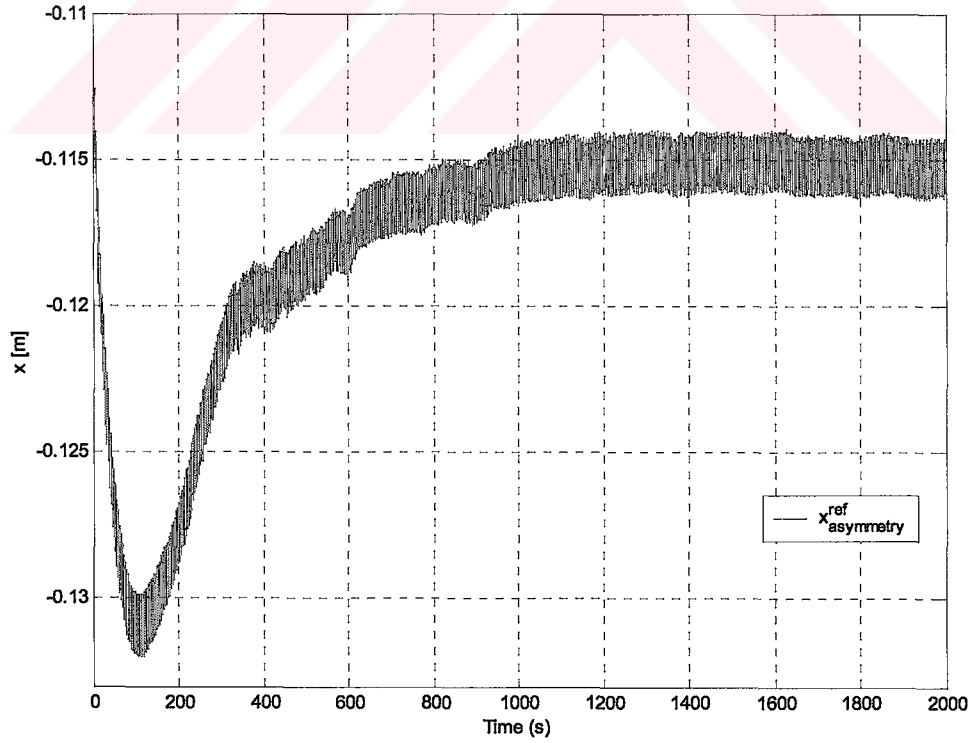

Figure 6.9. x-reference asymmetry versus time plot through the learning walk of the biped using one Gaussian membership function with initial value xra_0=-11 cm.

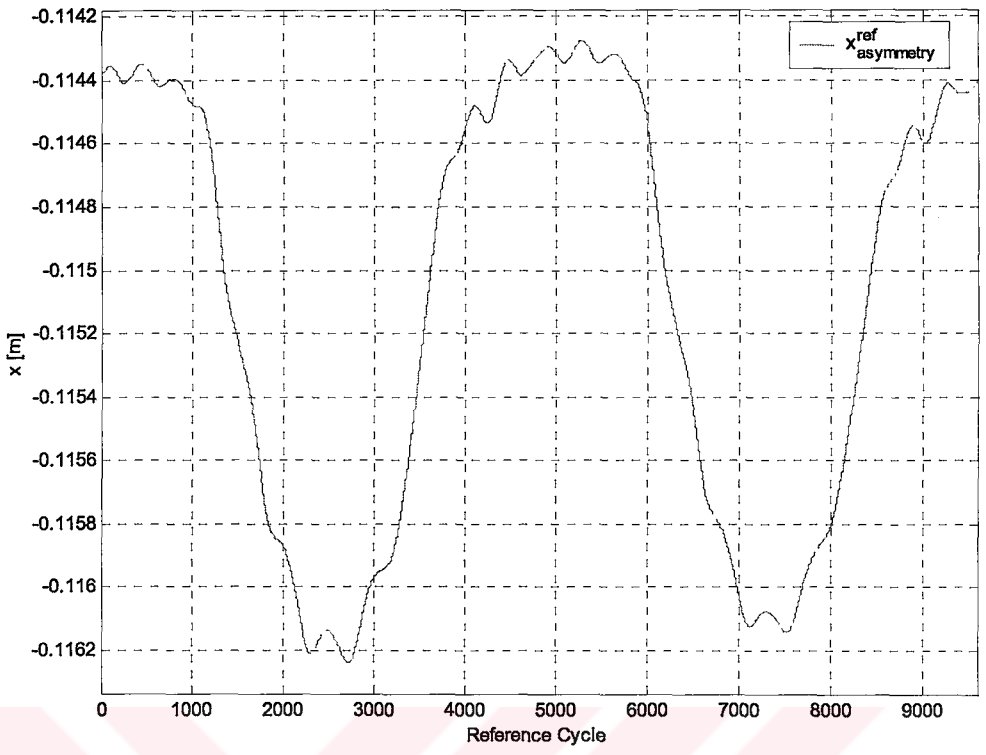

Figure 6.10. Simulation result of x-reference asymmetry for the last reference cycle of the learning walk using one Gaussian membership function with initial value xra_0=0.

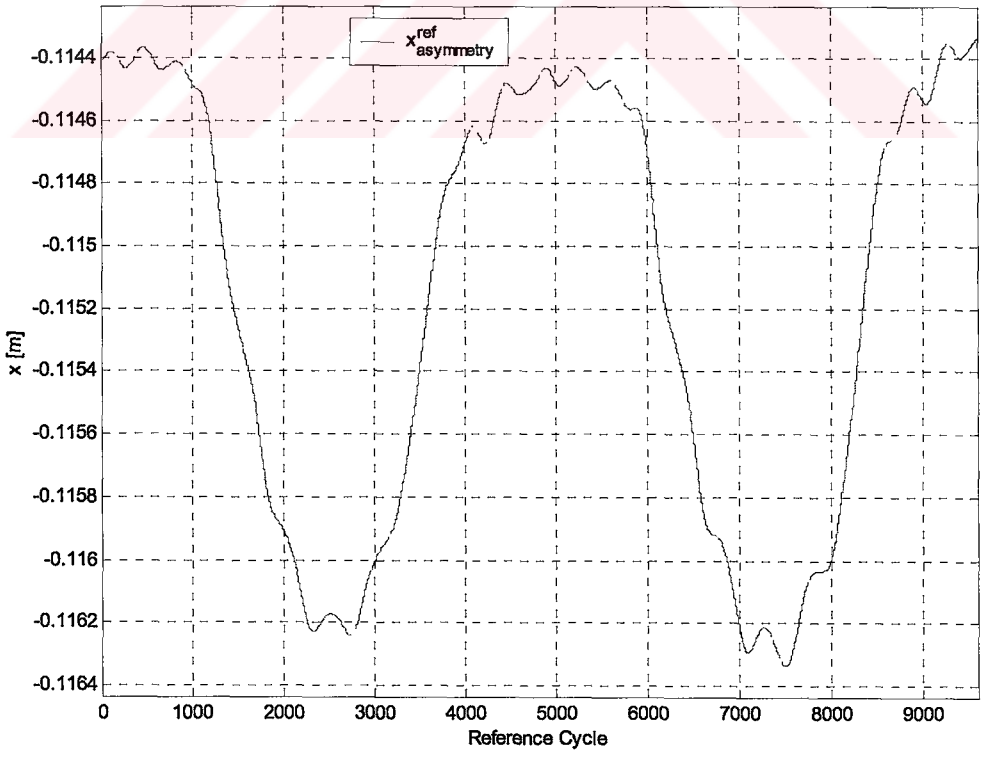

Figure 6.11. Simulation result of x-reference asymmetry for the last reference cycle of the learning walk using one Gaussian membership function with xra_0=-11cm.

6.4. Adaptive Fuzzy Scheme with More Than One Membership Functions

The Neuro-Fuzzy adaptive scheme is applied using 15 membership functions to obtain a functional for the x-reference asymmetry, rather than a scalar parameter, which was sought in Section 6.2. Same parameters with previously described schemes are used in these simulations too.

The simulation results, in Figures 6.12 and 6.13, show the convergence of the x-reference asymmetry functional after 3000 seconds simulation time. The initial value of the parameter used was chosen as zero and -11 centimeters, respectively. The total simulation time has been 4000 seconds and no significant deviations from the learned trajectory are observed in the last 1000 seconds.

The functional learned by the end of training period is shown in Fig 6.14. When the supporting springs are removed it is observed that the discontinuity caused by the difference of the functional values at the beginning and end of the reference cycle causes instability in the biped walk. It can be deduced that the learning with 15 membership functions is only partially successful. This can be explained by the fact that too many membership functions for the same variable, that is, the time, compete with each other when their parameters are adapted. This results in improper setting of the desired functional. This problem is alleviated by using interpolation in which it is assumed that a function symmetric with respect to 2.4 s that is the middle point of the reference cycle is suitable. Specifically the curve generated by learning and its mirror image with the above mentioned symmetry point is averaged to obtain the final shape of the functional to be applied to the robot after training. Other interpolation techniques, which result in the same initial and final x-reference asymmetry values on the reference cycle, could also be applied for the same purpose. The resulting curve is shown in Fig 6.15 and successful walk was achieved with it. However as it can be seen from the Figures 6.14 and 6.15, the difference between the maximum and minimum values in this functional is minute. The functional is bounded into a very narrow range and hence for practical purposes it can be considered as a constant function. The results suggest that using the parameter x-reference asymmetry as a constant rather than a functional would not deteriorate the walking performance. The functional can be considered to be a constant function at -11.5 cm.

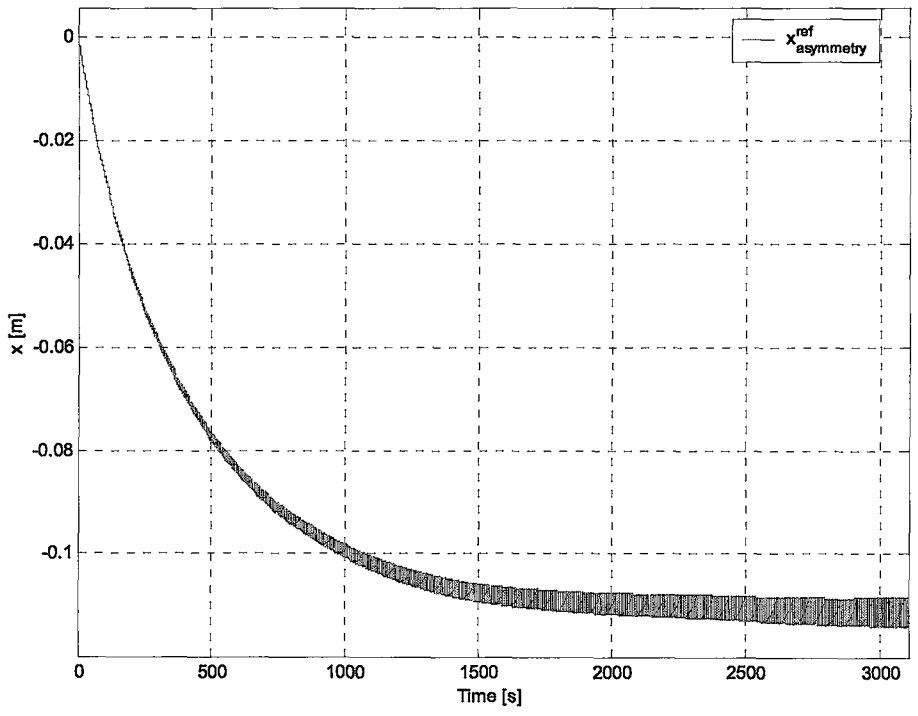

Figure 6.12. x-reference asymmetry versus time plot through the learning walk of the biped using 15 Gaussian membership functions with initial value xra_0=0.

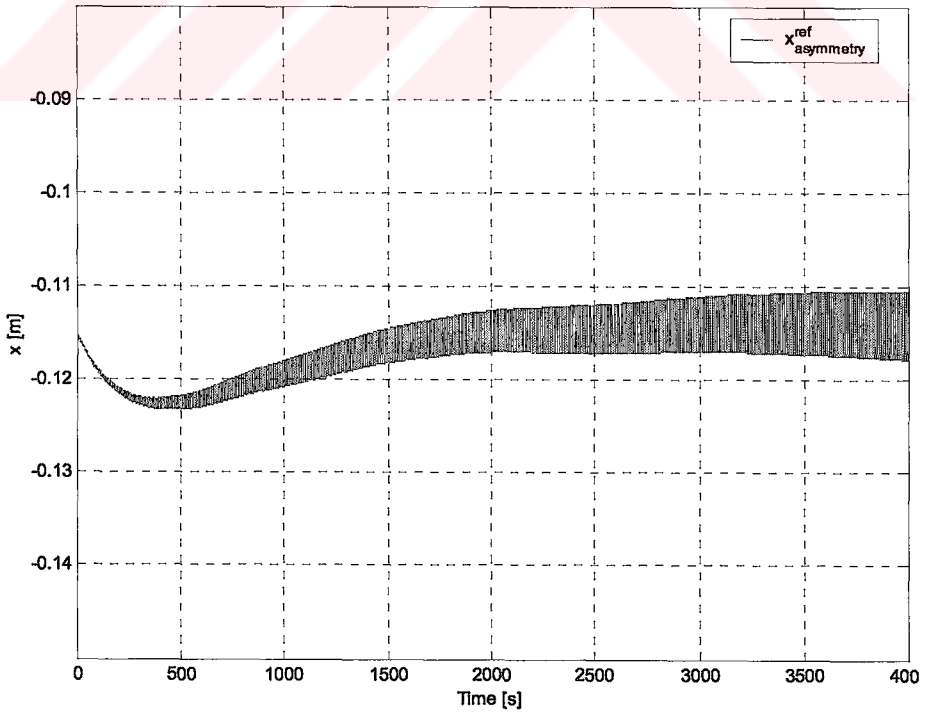

Figure 6.13 x-reference asymmetry versus time plot through the learning walk of the biped using 15 Gaussian membership functions with initial value xra_0=-11.5 cm.

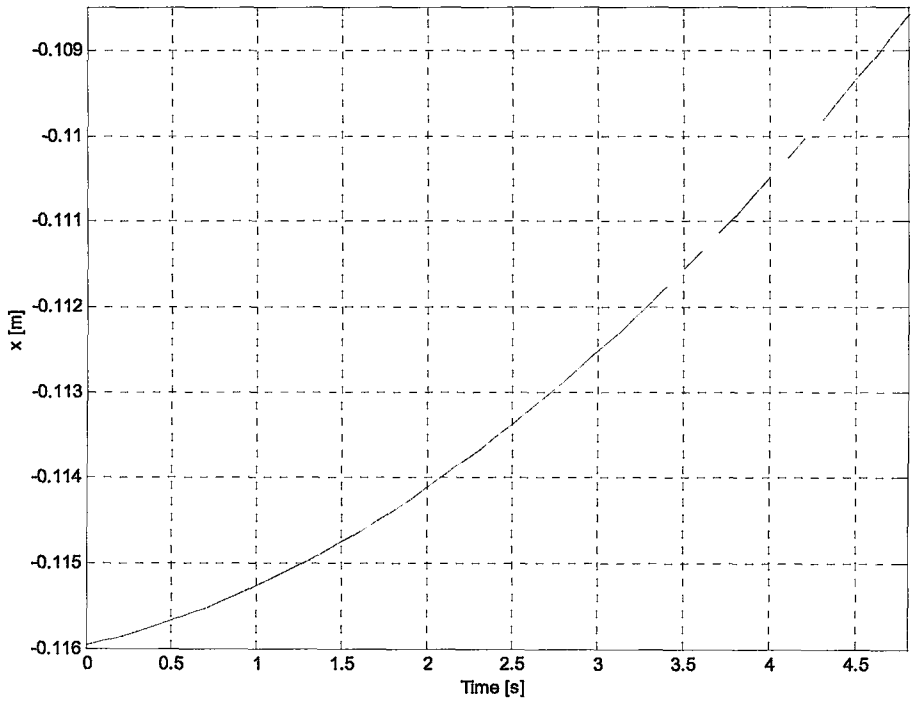

Figure 6.14 The functional created by the Neuro-Fuzzy Scheme over one reference cycle period, 4.8 s.

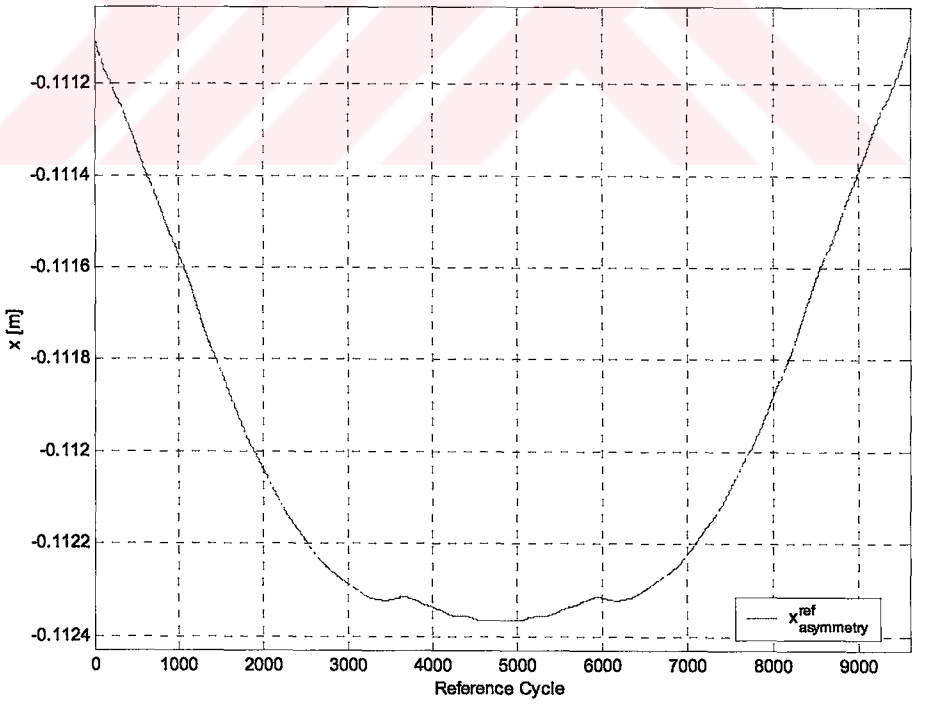

Figure 6.15. Simulation result of x-reference asymmetry for the last reference cycle of the learning walk using 15 Gaussian membership functions with initial value xra_0=0 after manual modification.

6.5. Discussion of the Simulation Results

The parameter value and fuzzy systems obtained as a result of the tuning processes are tested by turning off the adaptation mechanism and by removing the support springs. It is seen that the oscillation of the body pitch angle (β-angle) was decreased with respect to the case with the constant x-reference asymmetry parameter obtained by trial and error. The pitch, yaw and roll angles using the generated functional is shown in Fig 6.16. It should be noted that the fuzzy systems trained do not produce x-reference asymmetry parameters, which are far apart from the trial and error value. Significant, however, is that similar parameter values are obtained automatically without the elaborate work of carrying out numerous trial simulations.

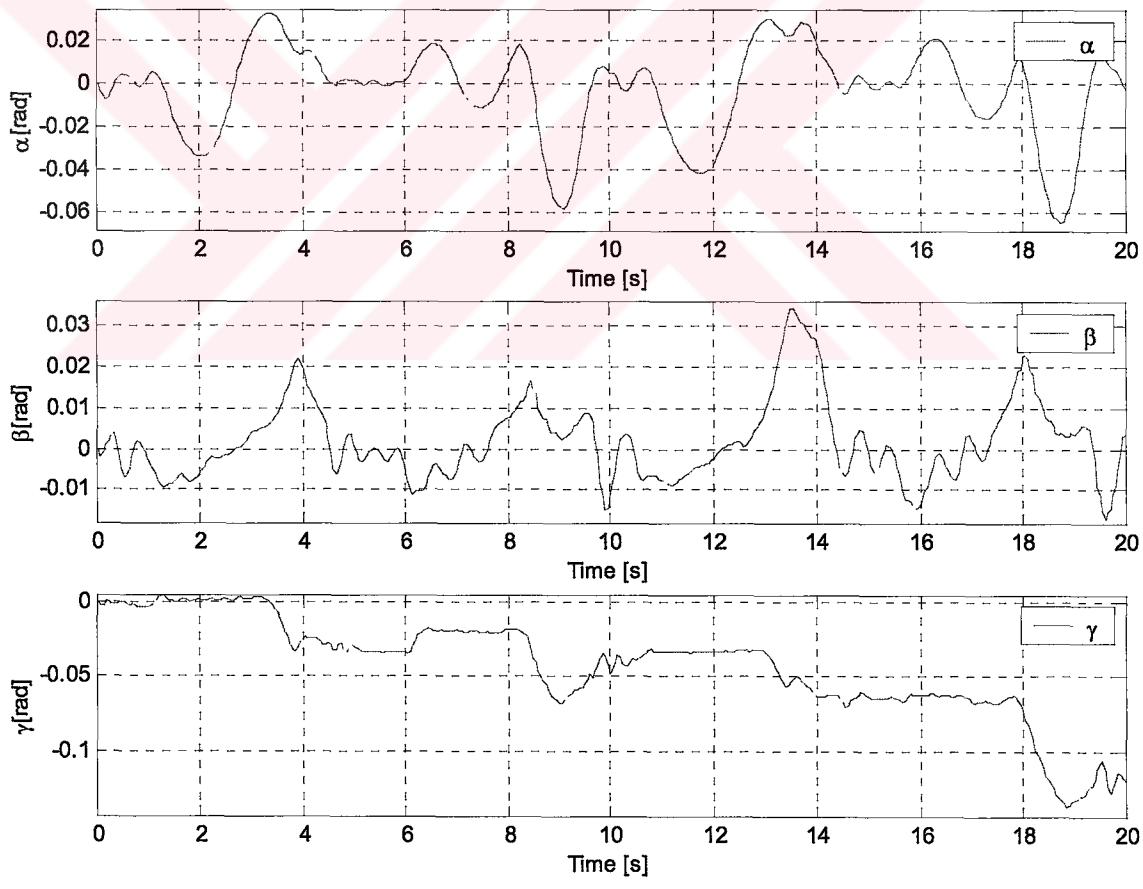

Figure 6.16. α,β and γ angles after the removal of the virtual torsional springs.

7. CONCLUSIONS

Biped robots attract the attention of researchers because of their possible uses in the human environment. Achieving complex tasks that human can do is a challenging work for the bipeds, and many problems regarding these are waiting to be solved. The open loop walking control of the biped robots yields satisfactory walking solutions up to some extend. Keeping the balance of the biped robot is the most important problem during walk. In the open loop walking scheme, this can be realized only by tuning the reference generation parameters precisely, since change of the parameters in even in millimeter ranges can cause the robot to fall.

In this work, automatic tuning of one of the reference generation parameters is applied via the use of virtual support springs attached to the robot trunk. Parameter and functional learning schemes are applied. As tools of functional learning fuzzy identifiers, which are universal approximators, are employed. Simulation studies revealed that automatic tuning of a scalar parameter produces directly applicable results whereas fuzzy identifier learning needs manual adjustments on the resulting functional after the training process. Using virtual springs, long and stable walks are realized and online tuning is done.

Simulation results indicate that the methods are successful and that a suitable x-reference asymmetry value or function can be obtained. Correctness of the tuning is justified by using different initial conditions for the x-reference asymmetry parameter and achieving the same value or functional over the reference cycle, eventually. To be noted is that, during the training phases, all of the other walking pattern parameters were kept at their constant values obtained by the trial and error method.

The success of the proposed on-line tuning method for one of the many parameters of the walk pattern is promising. However, it leaves the tuning of all of the parameters simultaneously, with more dimensional rule bases, as a problem to be addressed in the future.

8. APPENDICES

Appendix A

```
void PARAT()
{
SIMULATION PARAMETERS
steptime=0.0005;
starttime=0;
stoptime=5;
simulationcycle=-1;
stepdivider=100;
i_end=stoptime/steptime;
recordfrequency=10;   //Recordig frequency to the output file

//Opening a data for animation
ofstream fout2;
fout2.open ("result2.txt",ios::out);
fout2.precision(7); // Set 7 digits past the decimal
fout2.flags(ios::right+ios::fixed);
fout2<<stoptime<<endl;
      fout2.close();

//Open a data file for storing some of the selected data
      ofstream fout1;
      fout1.open ("result1.txt",ios::out);
      fout1.precision(7); // Set 7 digits past the decimal
      fout1.flags(ios::right+ios::fixed);
      fout1.close();

//simulation time addition parameters
Matrix groundforcecontainer;
impulse_container=0.0;
alphabetagamma=0.0;
alphabetagamma_dot=0.0;
alphabetagamma_old=0.0;

//Gravity - Eq.31
gx=0;
gy=0;
gz=9.80621;

//LINK PARAMETERS.
//LINK LENGTHS MASSES, GENERAL INFORMATION (BLOCK SHAPED LINKS
ASSUMED).
//Trunk
Lb=0.2; Wb=0.4; Hb=0.5; mb=50;

//Thigh 1 (Axes for lateral plane)
```

```
L1=0.1; W1=0.1; H1=0.1; m1=5;

//Thigh 2 (Axes for sagital plane)
L2=0.1; W2=0.1; H2=0.1; m2=3;

//Thigh 3
L3=0.27; W3=0.05; H3=0.1; m3=4;

//Calf
L4=0.22; W4=0.05;H4=0.1; m4=0.5;

//Foot 1
L5=0.1; W5=0.1; H5=0.1; m5=0.5;

//Foot 2
L6=0.087; W6=0.12; H6=0.25; m6=5;

//LINK INERTIA MATRICES ABOUT CENTER OF MASS IN THE ASSOCIATED LINK
FRAME.
double afil1[] = {pow(Wb,2)+pow(Hb,2), 0.0, 0.0,
            0.0, pow(Lb,2)+pow(Hb,2), 0.0,
            0.0, 0.0, pow(Lb,2)+pow(Wb,2) };
Matrix mfil1( 3,3,afil1);
Ib = Matrix(mfil1);
Ib *=(mb/12);

double afil2[] = {pow(W1,2)+pow(H1,2), 0.0, 0.0,
                  0.0, pow(L1,2)+pow(H1,2), 0.0,
                  0.0, 0.0, pow(L1,2)+pow(W1,2) };
Matrix mfil2( 3,3,afil2);
I1 = Matrix(mfil2);
I1 *=(m1/12);

double afil3[] = {pow(W2,2)+pow(H2,2), 0.0, 0.0,
                  0.0, pow(L2,2)+pow(H2,2), 0.0,
                  0.0, 0.0, pow(L2,2)+pow(W2,2) };
Matrix mfil3( 3,3,afil3);
I2 = Matrix(mfil3);
I2 *=(m2/12);

double afil4[] = {pow(W3,2)+pow(H3,2), 0.0, 0.0,
                  0.0, pow(L3,2)+pow(H3,2), 0.0,
                  0.0, 0.0, pow(L3,2)+pow(W3,2) };
Matrix mfil4( 3,3,afil4);
I3 = Matrix(mfil4);
I3 *=(m3/12);

double afil5[] = {pow(W4,2)+pow(H4,2), 0.0, 0.0,
                  0.0, pow(L4,2)+pow(H4,2), 0.0,
                  0.0, 0.0, pow(L4,2)+pow(W4,2) };
Matrix mfil5( 3,3,afil5);
I4 = Matrix(mfil5);
I4 *=(m4/12);

double afil6[] = {pow(W5,2)+pow(H5,2), 0.0, 0.0,
                  0.0, pow(L5,2)+pow(H5,2), 0.0,
                  0.0, 0.0, pow(L5,2)+pow(W5,2) };
Matrix mfil6( 3,3,afil6);
I5 = Matrix(mfil6);
I5 *=(m5/12);
```

```
double afil7[] = {pow(W6,2)+pow(H6,2), 0.0, 0.0,
                  0.0, pow(L6,2)+pow(H6,2), 0.0,
                  0.0, 0.0, pow(L6,2)+pow(W6,2) };
Matrix mfil7( 3,3,afil7);I6 = Matrix(mfil7);I6 *=(m6/12);

//DENAVIT-HARTENBERG PARAMETERS (CONVENTIONAL NOTATION).
alpha=0.0; a=0.0; d=0.0;

alpha[0][0]=pi/2.0; a[0][0]=0; d[0][0]=0;

alpha[1][0]=-pi/2; a[1][0]=0; d[1][0]=0;

alpha[2][0]=pi/2; a[2][0]=L3; d[2][0]=0;

alpha[3][0]=0; a[3][0]=L4; d[3][0]=0;

alpha[4][0]=-pi/2; a[4][0]=0; d[4][0]=0;

alpha[5][0]=0; a[5][0]=L6; d[5][0]=0;

//LINK RELATED POSITION VECTOR COMPUTATIONS FOR THE LUH-WALKER-PAUL N-
E MECHANISM.
//NEXT COORDINATE FRAME ORIGIN LOCATION IN THE CURRENT FRAME.
A0ipiSTAR=0.0;       //Equation 47 of Luh-Paul-Walker paper
for (int i=1;i<n+1;i++)
{
      double VV1[]={a[i-1][0],d[i-1][0]*sin(alpha[i-1][0]),d[i-
1][0]*cos(alpha[i-1][0])};
      A0ipiSTAR[mslice(0,i-1,3,1)] = Matrix(3,1,VV1);
}

//CENTER OF MASS LOCATION OF THE CURRENT LINK IN THE CURRENT FRAME.
Matrix A0isHAT(3,n,0.0);            //Eq 41

double VV2[]={0,0,0};   A0isiHAT[mslice(0,0,3,1)]=Matrix(3,1,VV2);
double VV3[]={0,0,0};   A0isiHAT[mslice(0,1,3,1)]=Matrix(3,1,VV3);
double VV4[]={-a[2][0]/6,0,0};
      A0isiHAT[mslice(0,2,3,1)]=Matrix(3,1,VV4);
double VV5[]={-a[3][0]/6,0,0};
      A0isiHAT[mslice(0,3,3,1)]=Matrix(3,1,VV5);
double VV6[]={0,0,0};   A0isiHAT[mslice(0,4,3,1)]=Matrix(3,1,VV6);
double VV7[]={-a[5][0]/6,0,0};
      A0isiHAT[mslice(0,5,3,1)]=Matrix(3,1,VV7);

//LINK MASS VECTOR, DEFINED FOR PROGRAMMING CONVENIENCE.
double VV8[]={m1,m2,m3,m4,m5,m6};m=Matrix(6,1,VV8);

//LINK INERTIA ARRAY, DEFINED FOR PROGRAMMING CONVENIENCE.
I=0.0;
I[mslice(0,0,3,3)]=I1;
I[mslice(0,3,3,3)]=I2;
I[mslice(0,6,3,3)]=I3;
I[mslice(0,9,3,3)]=I4;
I[mslice(0,12,3,3)]=I5;
I[mslice(0,15,3,3)]=I6;

//GENERAL CONSTANTS
double VV9[]={0,0,1};   z0=Matrix(3,1,VV9);

//joint VISCOUS FRICTION COEFFICIENTS
double Bfriction1=5;
```

```
double Bfriction2=5;
double Bfriction3=5;
double Bfriction4=5;
double Bfriction5=5;
double Bfriction6=5;
B_joint_friction_matrix=0.0;
B_joint_friction_matrix[0][0]=Bfriction1;
B_joint_friction_matrix[1][1]=Bfriction2;
B_joint_friction_matrix[2][2]=Bfriction3;
B_joint_friction_matrix[3][3]=Bfriction4;
B_joint_friction_matrix[4][4]=Bfriction5;
B_joint_friction_matrix[5][5]=Bfriction6;
B_joint_friction_matrix[6][6]=Bfriction1;
B_joint_friction_matrix[7][7]=Bfriction2;
B_joint_friction_matrix[8][8]=Bfriction3;
B_joint_friction_matrix[9][9]=Bfriction4;
B_joint_friction_matrix[10][10]=Bfriction5;
B_joint_friction_matrix[11][11]=Bfriction6;

//contact parameters
B_K_ratio=0;
Kspring=20000;
Bdamper=2000;
damper_threshold=0.005;
Bhorizontalfriction=0.1;
mu=0.3;
minimumKspring= 1000000;
maximumKspring =10000000;
elasticity_coefficient=1;
fixedKspring=5000;
horizontal_on=1;
ZLayerThickness=0.5;
integral_force=0.0;
velslip=0.00001;
touch_on_off=0.0;
touch_on_off_OLD=0.0;
touch_xy_coordinates_array=0.0;
Khorizontalspring=1000;
spring_threshold=0.000;

//*********************************************************************
//obstacle contact parameters
Kxobstacle=50000;
Bxobstacle=10000;
Byobstacle=0.3;
xobstacle=0.06;
obstacle_length=0.0;
obstacle_height=-1;

//reference generation parameters
double_support_period=1;  //s
single_support_period=1.4;//s
stepsize=0.08;    //m
step_height=0.03;     //m
swing=0.13;    //m
swing_offset=0.11;    //m
xref_asymmetry=-0.11; //m
yref_asymmetry=0;     //m

RaddX=0;
RaddY=0;
```

```
RaddZ=0;
LaddX=0;
LaddY=0;
LaddZ=0;

stepsize_old = stepsize;
stepsize_add = -0.04;
k_stepsize = stepsize_add/(0.2/steptime);
//*****************************************************************

//CONTROLLER PARAMETERS
Kp=0;Kd=0;Ki=0;
Kp[0][0]=6000*Kp_multiplier;
Kp[1][1]=6000*Kp_multiplier;
Kp[2][2]=10000*Kp_multiplier;
Kp[3][3]=30000*Kp_multiplier;
Kp[4][4]=30000*Kp_multiplier;
Kp[5][5]=6000*Kp_multiplier;
Kd[0][0]=1;
Kd[1][1]=1;
Kd[2][2]=1;
Kd[3][3]=1;
Kd[4][4]=1;
Kd[5][5]=1;
Ki[0][0]=40;
Ki[1][1]=40;
Ki[2][2]=40;
Ki[3][3]=40;
Ki[4][4]=40;
Ki[5][5]=40;

qRinterror =0.0;
qLinterror =0.0;

//LEG ATTACHMENT FRAME TO BODY FRAME ROTATION MATRICES
double VV10[]={1,0,0, 0,1,0, 0,0,1}; bARo=Matrix(3,3,VV10);
bALo=Matrix(bARo);

//CORNER COORDINATES IN FOOT LINK FRAME
double VV11[]={0,0.038,-0.0503};    A0nc1=Matrix(3,1,VV11);
double VV12[]={0,0.038,0.1503};      A0nc2=Matrix(3,1,VV12);
double VV13[]={0,-0.038,-0.0503};       A0nc3=Matrix(3,1,VV13);
double VV14[]={0,-0.038,0.1503};    A0nc4=Matrix(3,1,VV14);

//P_STARS FOR THE VIRTUAL LINKS BETWEEN BODY ORIGIN AND LEG ATTACMENT
POINTS

double VV15[]={0,-0.085,0};     Ro_pSTAR_Ro=Matrix(3,1,VV15);
double VV16[]={0, 0.085,0};     Lo_pSTAR_Lo=Matrix(3,1,VV16);

//THE BODY S_HAT
double VV17[]={-0.05,0,Hb/2};b_sHAT_b=Matrix(3,1,VV17);

body_height=0.50;

//ADDED BY OZKAN
///xref_asymmetry adaptation
y_l_xra[0][0]= 0.0;
y_l_xra[1][0]= xref_asymmetry;
y_l_xra[2][0]= 0.0;
y_l_xra[3][0]= 0.0;
```

```
y_l_xra[4][0]= 0.0;

x_center_xra[0][0]= 0.0;
x_center_xra[1][0]= 2.4;
x_center_xra[2][0]= 0.0;
x_center_xra[3][0]= 0.0;
x_center_xra[4][0]= 0.0;

sigma_xra[0][0]= 0.0;
sigma_xra[1][0]= 3.0;
sigma_xra[2][0]= 0.0;
sigma_xra[3][0]= 0.0;
sigma_xra[4][0]= 0.0;

///swing adaptation
y_l[0][0]= 0.20;
y_l[1][0]= swing;
y_l[2][0]= -0.20;

x_center[0][0]= 0;
x_center[1][0]= 2.4;
x_center[2][0]= 0;

sigma_l[0][0]= 4;
sigma_l[1][0]= 4;
sigma_l[2][0]= 4;

///swing_offset adaptation
y_l_o[0][0]= 0.20;
y_l_o[1][0]= swing_offset;
y_l_o[2][0]= -0.20;

x_center_o[0][0]= 0;
x_center_o[1][0]= 2.4;
x_center_o[2][0]= 0;

sigma_l_o[0][0]= 4;
sigma_l_o[1][0]= 4;
sigma_l_o[2][0]= 4;

/////////////////LOOKUP TABLE & ADAPTATION PARAMETERS///////////////
if (!lookxratable && xra_adaptation)
            xref_asymmetry=adaptation_xra(0);

if (!lookxratable && swing_adaptation)
            swing=adaptation_swing(0);

if (!lookxratable && swing_offset_adaptation)
            swing_offset=adaptation_swing_offset(0);

if (lookxratable)
      {
            xref_asymmetry=lookedupmatrix(0,0);
            swing=lookedupmatrix(0,1);
            swing_offset=lookedupmatrix(0,2);
      }
//*********************************************************************

double VV18[]={ 0,0,1, xref_asymmetry+stepsize-Ro_pSTAR_Ro[0][0],
            0,1,0,-swing_offset-Ro_pSTAR_Ro[1][0],
             -1,0,0,-body_height-Ro_pSTAR_Ro[2][0],
```

```
            0,0,0, 1};
TRight=Matrix(4,4,VV18);

dummyrefR_old=i_kine_analytic(TRight);

double VV19[]={0,0,1, xref_asymmetry-stepsize-Lo_pSTAR_Lo[0][0],
             0,1,0, swing_offset-Lo_pSTAR_Lo[1][0],
            -1,0,0,-body_height-Lo_pSTAR_Lo[2][0],
             0,0,0, 1};
TLeft=Matrix(4,4,VV19);
dummyrefL_old=i_kine_analytic(TLeft);

//initial conditions
//avector=0.0;
vvector = 0.0;

initialflatangle=pi*20/180;
initialbackangle=acos(2*cos(initialflatangle)-1);

double VV20[]={0, 0, body_height, 1, 0, 0, 0, 1, 0, 0, 0, 1,

dummyrefR_old[0][0],dummyrefR_old[0][1],dummyrefR_old[0][2],dummyrefR_
old[0][3],dummyrefR_old[0][4],dummyrefR_old[0][5],

dummyrefL_old[0][0],dummyrefL_old[0][1],dummyrefL_old[0][2],dummyrefL_
old[0][3],dummyrefL_old[0][4],dummyrefL_old[0][5]};
xvector=Matrix(24,1,VV20);

rotmatR=0.0;
rotmatL=0.0;

for (int i=1;i<7;i++)
{
    rotmatR[mslice(0,3*(i-1),3,3)]=rotat(i,dummyrefR_old[0][i-1]);
    rotmatL[mslice(0,3*(i-1),3,3)]=rotat(i,dummyrefL_old[0][i-1]);
}

uEvector=0.0;
jfvector=0.0;

/////by ozkan for spring force
f_support=0.0;

qRref=0.0;
qRdotref=0.0;
qLref=0.0;
qLdotref=0.0;

qref =0.0;

//INERTIA MATRIX INITIALIZATION
H=treehinitializer(xvector);
inverseH=!H;

//external force transformer matrix
K=treekinitializer(xvector);

//bias vector
b=0.0;

//control
```

```
uvector=0.0;

//MINIMIZATION VARIABLES
minimisation_on=1;
checkenergycurve =0;

//walking zones and reflex triggers
zone1=0;
zone2=1;
//forwardzmpsteptrigger=0.05;

referencecycle=0;
referencecycle_counter=0;

ZMPx=0.0;
ZMPy=0.0;
ZMPx_b=0.0;
ZMPy_b=0.0;

ZMPx_f=-0.13349;
ZMPy_f= 0.083757;

ZMPx_bf=-0.13349;
ZMPy_bf= 0.083757;

//avoid slippping in x direction => initialization for new start
rightsteploss=0.0;
leftsteploss=0.0;
}
```

Appendix B

```
#include "refgencartesian_analytic.h"
void controller()
{
Matrix result;

//INPUT VARIABLES
//velocity_vector=vvector;     //defined in biped_vel_integrator.h
position_vector=xvector;
pb=position_vector[mslice(0,0,3,1)];;
wAblist=position_vector[mslice(3,0,9,1)];
q=position_vector[mslice(12,0,2*n,1)];
vb=velocity_vector[mslice(0,0,3,1)];
wb=velocity_vector[mslice(3,0,3,1)];
qdot=velocity_vector[mslice(6,0,(6+2*n)-7+1,1)];
double VV0[]={wAblist[0][0],wAblist[3][0],wAblist[6][0],wAblist[1][0],
wAblist[4][0],wAblist[7][0],wAblist[2][0],wAblist[5][0],wAblist[8][0]};
wAb=Matrix(3,3,VV0);

//joint actual variables
qR=q[mslice(0,0,n,1)];
qL=q[mslice(n,0,n,1)];
qRdot=qdot[mslice(0,0,n,1)];
qLdot=qdot[mslice(n,0,n,1)];

//joint error variables
qRerror=0.0;
qRdoterror=0.0;
qLerror=0.0;
```

```
qLdoterror=0.0;

if (!lookxratable)  ////If look-up table is used don't enter.
{
lookupmatrix(referencecycle,0)=xref_asymmetry;
lookupmatrix(referencecycle,1)=swing;
lookupmatrix(referencecycle,2)=swing_offset;
}

referencecycle=referencecycle+1;

if( referencecycle>=2*(double_support_period+single_support_period)/st
eptime)
{
      referencecycle=0;
referencecycle_counter++;

if (!lookxratable) ///If look-up table is used don't enter.
{
//Look-up table to the file
      ofstream fout3;
            fout3.open ("result3.txt",ios::out);
            fout3.precision(7); // Set 7 digits past the decimal
            fout3.flags(ios::right+ios::fixed);
      fout3<<lookupmatrix<<endl;
      fout3.close();
}
}

Matrix PID_control_R(6,1),PID_control_L(6,1);
//joint pos, vel references
qref=refgencartesian_analytic(qR,qL);
qRref=qref[mslice(0,0,n,1)];
qLref=qref[mslice(n,0,n,1)];
qRdotref=qref[mslice(2*n,0,n,1)];
qLdotref=qref[mslice(3*n,0,n,1)];

//p d i errors
qRerror=qRref-qR;
qLerror=qLref-qL;

qRdoterror=qRdotref-qRdot;
qLdoterror=qLdotref-qLdot;

qRinterror=qRinterror+steptime*qRerror;
qLinterror=qLinterror+steptime*qLerror;

PID_control_R = Kp * qRerror + Kd * qRdoterror + Ki * qRinterror;
PID_control_L = Kp * qLerror + Kd * qLdoterror + Ki * qLinterror;

double UU1[]={0,0,0,0,0,0,
PID_control_R[0][0],PID_control_R[1][0],PID_control_R[2][0],PID_contro
l_R[3][0],PID_control_R[4][0],PID_control_R[5][0],
PID_control_L[0][0],PID_control_L[1][0],PID_control_L[2][0],PID_contro
l_L[3][0],PID_control_L[4][0],PID_control_L[5][0]};
result= Matrix(18,1,UU1);
uvector=result;
}
```

Appendix C

```
void treeanglecomp()
{
Matrix VV1,VV2,Amatrix(3,3);
double compalpha,compbeta,compgamma;
position_vector=xvector;
VV1=position_vector[mslice(3,0,9,1)];
double UU1[]={VV1[0][0],VV1[3][0],VV1[6][0],VV1[1][0],
      VV1[4][0],VV1[7][0],VV1[2][0],VV1[5][0],VV1[8][0]};
Amatrix=Matrix(3,3,UU1);
compalpha=atan2(Amatrix[2][1],Amatrix[2][2]);
compbeta=asin(-Amatrix[2][0]);
compgamma=atan2(Amatrix[1][0],Amatrix[0][0]);

//added by ozkan
alphabetagamma_old=alphabetagamma;

double UU2[]={compalpha,compbeta,compgamma};
alphabetagamma=Matrix(3,1,UU2);

alphabetagamma_dot=(alphabetagamma-alphabetagamma_old)/steptime;

//call adaptation
if  (xra_adaptation)
{
      if (lookxratable)
      {
            xref_asymmetry=lookedupmatrix(referencecycle,0);
      }
      else
      {
                  xref_asymmetry=adaptation_xra((referencecycle)*stept
                  ime);
      }
}

if (swing_adaptation)
{
      if (lookxratable)
      {
            swing=lookedupmatrix(referencecycle,1);
      }
      else
      {
            swing=adaptation_swing(referencecycle*steptime);
      }
}

if (swing_offset_adaptation)
{
      if (lookxratable)
      {
            swing_offset=lookedupmatrix(referencecycle,2);
      }
      else
      {
                  swing_offset=adaptation_swing_offset(referencecycle*
                  steptime);
```

```
            }
      }
}
```

Appendix D

```
double gausian(double x, double center, double sigma )
{
double result=0.0;
result=exp(-((x-center)/sigma)*((x-center)/sigma));
return result;
}
```

Appendix E

```
double adaptation_xra(double x)
{
double a=0.0,b=0.0,z_l=0.0;
      for (int i=0;i<=2;i++)
      {
            a = a +
            (y_l_xra[i][0]*gausian(x,x_center_xra[i][0],sigma_xra[i][0
            ]));
            b = b + (gausian(x,x_center_xra[i][0],sigma_xra[i][0]));
}

//backpropagation
y_l_xra_old = y_l_xra;
x_center_xra_old = x_center_xra;
sigma_xra_old=sigma_xra;

for (int i=0;i<=2;i++)
      {
      z_l=(gausian(x,x_center_xra[i][0],sigma_xra[i][0]));
            y_l_xra[i][0]=      y_l_xra_old[i][0] - alpha_step_size *
            (f_support[4][0] / b)* z_l ;
            x_center_xra[i][0]= x_center_xra_old[i][0] -
            alpha_step_size * (f_support[4][0] / b)* (y_l_xra[i][0]-
            f_support[4][0])* z_l * 2*(x-
            x_center_xra[i][0])/(sigma_xra[i][0]*sigma_xra[i][0]);
            sigma_xra[i][0]=    sigma_xra_old[i][0] - alpha_step_size
            * (f_support[4][0]/ b)* (y_l_xra[i][0]-f_support[4][0])*
            z_l * 2*(x-x_center_xra[i][0])*(x-
            x_center_xra[i][0])/(sigma_xra[i][0]*sigma_xra[i][0]*sigma
            _xra[i][0]);
}
return (a/b);  //swing=> f = a/b
}
```

REFERENCES

[1] Riezenman, M.J., "Robots Stand On Own Two Feet", IEEE Spectrum , Volume: 39 Issue: 8 , pp: 24 -25, Aug 2002

[2] McGhee, R.B., "Finite State Control of Quadruped Locomotion", Proceedings of Second International Symposium on External Control of Human Extremities, Dubrovnik, Yugoslavia, 1966.

[3] Frank, A.A., "Automatic Control Systems for Legged Locomotion", USCEE Report No. 273, University of South California, USA, 1968.

[4] Kalo, I., S. Ohteru, H. Kobayashi, K. Shirai and A. Uchiyama, "Information-Power Machine with Senses and Limbs" in First CISM-IFToMM Symp. on Theory and Practice of Robots and Manipulators, Spring-Verlag, 1974.

[5] Hirai, K., M. Hirose, Y. Haikawa, T. Takenaka, "The Development of Honda Humanoid Robot", Proceedings of IEEE International Conference on Robotics and Automation, pp: 1321 -1326 vol.2, May 1998

[6] Kaneko, K.; F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, T. Isozumi, "Design of Prototype Humanoid Robotics Platform for HRP", IEEE/RSJ International Conference on Intelligent Robots and System, pp: 2431-2436 vol.3, Oct. 2002

[7] Yamasaki F., T. Miyashita, T. Matsui, H. Kitano, "PINO the Humanoid That Walk", Proc. of The First IEEE-RAS International Conference on Humanoid Robots, The Massachusetts Institute of Technology, USA, 2000

[8] Huang, Q., K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi and K. Tanie, "Planning Walking Patterns for a Biped Robot", IEEE Transactions On Robotics and Automation, Jun 2001

[9] Yamaguchi, J., A. Takanishi, I. Kato, "Development of a Biped Walking Robot Adapting to a Horizontally Uneven Surface", Proceedings of the IEEE/RSJ/GI

International Conference on Intelligent Robots and Systems, pp: 1156 -1163 vol.2, Sep. 1994

[10] Park J. H.; H. Chung, "ZMP Compensation by Online Trajectory Generation for Biped Robots", IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings, pp: 960 -965 vol.4, Oct 1999

[11] Park, J. H., H. C. Cho, "An On-line Trajectory Modifier for the Base Link of Biped Robots To Enhance Locomotion Stability", IEEE International Conference on Robotics and Automation, San Francisco, USA, Apr 2000

[12] Jalics, L., H. Hemami, Y.F. Zheng, "Pattern Generation Using Coupled Oscillators for Robotic and Biorobotic Adaptive Periodic Movement", IEEE International Conference on Robotics and Automation Proceedings, pp: 179 -184 vol.1, Apr 1997

[13] Huang Q., K. Kaneko, K. Yokoi, S. Kajita, T. Kotoku, N. Koyachi, H. Arai, N. Imamura, K. Komoriya, K. Tanie, "Balance Control of a Piped Robot Combining Off-line Pattern With Real-time Modification", IEEE International Conference on Robotics and Automation Proceedings, pp: 3346 -3352 vol.4, Apr 2000

[14] Nishiwaki, K., T. Sugihara, S. Kagami, M. Inaba and H. Inoue, "Online Mixture and Connection of Basic Motions for Humanoid Walking Control by Footprint Specification", IEEE International Conference on Robotics and Automation Proceedings, pp: 4110-4115 vol.4, May 2001

[15] Lim, H; Y. Kaneshima and A. Takanishi, "Online Walking Pattern Generation for Biped Humanoid Robot with Trunk", IEEE International Conference on Robotics and Automation Proceedings, pp: 3111-3116, May 2002

[16] Ogura Y., Y. Sugahara, Y. Kaneshima, N. Hieda, H. Lim; A. Takanishi, "Interactive Biped Locomotion Based on Visual/Auditory Information", 11th IEEE International Workshop on Robot and Human Interactive Communication Proceedings, pp: 253 -258, Sept. 2002

[17] Huang, Q., S. Kajita, N. Koyachi, K. Kaneko, K. Yokoi, T. Kotoku, H. Arai, K. Komoriya and K. Tanie, "Walking Patterns and Actuator Specifications for a Biped Robot", IEEE/RSJ International Conference on Intelligent Robots and Systems Proceedings, pp: 1462 -1468 vol.3, Oct 1999

[18] Kun, A., W.T. Miller, III, "Adaptive Dynamic Balance of a Biped Robot Using Neural Networks", IEEE International Conference on Robotics and Automation Proceedings, pp: 240 -245 vol.1, Apr 1996

[19] Chew, C.M., "Biped Locomotion: Augmenting an Intuitive Control Algorithm with Learning", Doctoral Thesis Proposal, Mechanical Engineering Department, MIT, Nov 1999

[20] Magdalena, L., F. Monasterio, "A Fuzzy Logic Controller with Learning Through the Evolution of its Knowledge Base", International Journal of Approximate Reasoning, 1999

[21] Henaff, P., M. Milgram, "Adaptive Neural Control with Backpropagation Algorithm", Proceedings of the International Symposium on Signal Processing, Robotics and Neural Networks, France, Apr 1999

[22] Erbatur, K., A. Okazaki, K. Obiya, T. Takahashi and A. Kawamura, "A Study on the Zero Moment Point Measurement for Biped Walking Robots", Proc. 7th International Workshop on Advanced Motion Control, pp. 431-436, Maribor, Slovenia, 2002

[23] Eberhart, H. D., "Physical Principles for Locomotion", in Neural Control of Locomotion, R. M. Herman et, al., Eds. New Yoek: Plenum, 1976.

[24] Lim, H.O., Y. Kaneshima and A. Takanishi, "Online Walking Pattern Generation for Biped Humanoid Robot with Trunk", ICRA '02 IEEE International Conference on Robotics and Automation, Washington D.C., USA, 2002

[25] Yonemura, A., Y. Nakajima and A. Kawamura, "Experimental Aproach for the Fast Walking by the Biped Robot", Proc. Of the 2000 IASTED Int. Conference Robotics and Aplications, Honolulu, Hawaii, USA, 2000

[26] Zheng, Y.F., "A Neural Gait Synthesizer for Autonomous Biped Robots", Proceedings of IEEE International Workshop on Intelligent Robots and Systems, pp: 601 -608, 1990

[27] Spong, M., M. Vidyasagar, "Robot Dynamics and Control", John Wiley and Sons, 1989

[28] Luh, J. Y. S., M. W. Walker and R. P. C. Paul, "Online Computational Scheme for Mechanical Manipulators", Journal of Dynamic Systems, Measurement, and Control, 1980

[29] Fujimoto, Y, "Study on Biped Walking Robot with Environmental Force Interaction", Doctoral Thesis submitted to Division of Electrical and Computer Engineering, Graduate School of Engineering, Yokohama National University, March 1998.

[30] Bebek, O., K. Erbatur, "A Fuzzy System for Gait Adaptation of Biped Walking Robots", Proceedings of the IEEE Conference on Control Applications, 2003

[31] Park, J.H, Y.K. Rhee, "ZMP Trajectory Generation for Reduced Trunk Motions of Biped Robots" International Conference on Intelligent Robots and Systems, 1998

[32] Qinghua L, A. Takanishi and I. Kato, "Learning Control of Compensative Trunk Motion for Biped Walking Robot Based on ZMP Stability Criterion", International Conference on Proceedings of the Intelligent Robots and Systems, 1992

[33] Kawato, M., Y. Uno, M. Isobe and R. Suzuki, "Hierarchical Neural Network Model for Voluntary Movement with Application for Robotics", pp.8-16, IEEE Control Systems Magazine, April 1988

[34] Erbatur, K, O. Kaynak and I. Rudas, "Fuzzy Identifier Based Inverse Dynamics Control for a 3-DOF Articulated Manipulator", IECON 23rd International Conference on Industrial Electronics, Control and Instrumentation, 1997

[35] Wang, L. X., "Adaptive Fuzzy Systems and Control-Design and Stability Analysis", PTR Prentice Hall, Englewood Cliffs, New Jersey 07632, 1994

[36] Fujimoto, Y., A. Kawamura, "Simulation of an Autonomous Biped Walking Robot Including Environmental Force Interaction", IEEE Robotics and Automation Magazine, June 1998, pp. 33-42