ABSTRACT

GENETICALLY EVOLVED DYNAMIC CONTROL FOR QUADRUPED WALKING

by
Giorgio Grasso

The aim of this dissertation is to show that dynamic control of quadruped locomotion is achievable through the use of genetically evolved central pattern generators. This strategy is tested both in simulation and on a walking robot. The design of the walker has been chosen to be statically unstable, so that during motion less than three supporting feet may be in contact with the ground.

The control strategy adopted is capable of propelling the artificial walker at a forward locomotion speed of $\sim 1.5\ Km/h$ on rugged terrain and provides for stability of motion. The learning of walking, based on simulated genetic evolution, is carried out in simulation to speed up the process and reduce the amount of damage to the hardware of the walking robot. For this reason a general-purpose fast dynamic simulator has been developed, able to efficiently compute the forward dynamics of tree-like robotic mechanisms.

An optimization process to select stable walking patterns is implemented through a purposely designed genetic algorithm, which implements stochastic mutation and cross-over operators. The algorithm has been tailored to address the high cost of evaluation of the optimization function, as well as the characteristics of the parameter space chosen to represent controllers.

Experiments carried out on different conditions give clear indications on the potential of the approach adopted. A proof of concept is achieved, that stable dynamic walking can be obtained through a search process which identifies attractors in the dynamics of the motor-control system of an artificial walker.

GENETICALLY EVOLVED DYNAMIC CONTROL FOR QUADRUPED WALKING

by
Giorgio Grasso

A Dissertation
Submitted to the Faculty of
New Jersey Insitute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer and Information Science

Department of Computer and Information Science

May 2000

APPROVAL PAGE

GENETICALLY EVOLVED DYNAMIC CONTROL FOR QUADRUPED WALKING

Giorgio Grasso

Dr. Michael Recce, Dissertation Advisor — Date
Professor of Computer and Information Science, NJIT

Dr. James Calvin, Committee Member — Date
Professor of Computer and Information Science, NJIT

Dr. Fadi Deek, Committee Member — Date
Vice Chairperson and Professor of Computer and Information Science, NJIT

Dr. Stephen Hanson, Committee Member — Date
Chairperson and Professor of Psychology, Rutgers University

Dr. James McHugh, Committee Member — Date
Professor of Computer and Information Science, NJIT

Dr. John Ryon, Committee Member — Date
Professor of Computer and Information Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Giorgio Grasso
Degree:	Doctor of Philosophy in Computer and Information Science
Date:	May, 2000

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer and Information Science
  New Jersey Institute of Technology, Newark, NJ, 2000
- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark, NJ, 2000
- Italian Laurea in Applied Physics
  University of Catania, Catania, Italy, 1993

Major: Computer Science

Presentations and Publication:

[1] G. Grasso, "New Trends in Artificial Intelligence", *Invited Lecture*, Scuola Superiore di Catania, (Catania, Italy), 1999.

[2] G. Grasso and M. Recce, "Towards Genetically Evolved Dynamic Control for Quadruped Locomotion", *Connection Science*, vol. 11, no. 3&4, pp 317-330, 1999.

[3] G. Grasso, "Dynamic Control of Walking", *Selected Presentation*, New Jersey Institute of Technology, (Newark, NJ), 1998.

[4] G. Grasso, "Intelligence Through Evolution", *Invited Lecture*, Department of Mathematics, (Catania, Italy), 1998.

[5] G. Grasso and M. Recce, "Scene Analysis for an Orange Harvesting Robot", *Artificial Intelligence Applications*, vol. 11, no. 3, pp 9-15, 1997.

[6] G. Grasso and M. Recce, "Scene Analysis for an Orange Picking Robot", in *Proceedings of 6th International Congress for Computer Technology in Agriculture (ICCTA'96)*, eds. C. Lokhorst, A.J. Udink ten Cate, and A.A. Dijkhuizen, VIAS, The Netherlands, pp 275-280, 1996

Dedicated to Maria

ACKNOWLEDGMENT

There are many people I would like to thank for helping me towards the successful completion of this thesis. First of all Michael Recce for giving me the opportunity to pursue my PhD and for his supervision. I would also like to thank Ken Harris, Adrienne James, Charles King, Asaur Rahman and John Taylor for the fruitful discussions.

I am grateful to David Jack for reading my thesis and for his continuous moral support. He is not only a colleague but also a good friend. A special thanks to David Edwards at UCL for the construction of the mechanical parts of the walking robot.

On a personal note I wish to thank my family for their love and tremendous support. A special thanks to my father for having always been an endless source of wisdom and knowledge. Finally I wish to sincerely thank my wife Maria for her patience, love and incredible support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

In the last 30 years research on artificial walking has received a great deal of attention. The aim has been to learn about the principles of legged locomotion, particularly those connected with dynamic control and adaptive behavior. There are many reasons to study walking machines. These type of vehicles can show a very high mobility in an off-road environment, because they don't need a continuous path of support as wheeled robots do. In addition walking provides an active suspension mechanism, that decouples the movement of the body from the movement of legs. This feature of legged locomotion provides for a smooth motion of the body, despite great variations in the terrain surface. Walking robot capable of a variety of behaviors could prove extremely valuable for a number of applications, from exploration of unstructured environments to the recovery and transportation of hazardous materials. Finally artificial walkers have been employed to test various models of biological motor control.

The usefulness of a legged robot is entirely based on its ability to cope with unstructured surfaces and move through rugged terrain efficiently, as these are the environments in which it would outperform its wheeled counterpart. Controlling a machine capable of flexible behavior and adaptation to different conditions poses a variety of theoretical as well as technical problems.

Mainly two approaches have been taken in solving the problem of legged locomotion control, leading to the development of two classes of systems successful at walking. Those statically stable with at least three feet on the ground at all times during motion (insect-like) and those intrinsically unstable that have to actively provide for constant balancing, moving without a stable base of support. Most of the work in the literature is devoted to the first class of problems, for which the control is limited to the kinematics and does not pose fast time-scale limitations to the controller architecture.

The work devoted to the investigation of the properties of legged locomotion has produced many different controller types. The following sections will guide the reader through some of the history of artificial walking.

1.1 Background

The control of artificial walking was originally treated as a kinematic optimization problem for statically stable systems and the first experiments on walking were carried out under human control. When digital computers become widespread within the scientific community, a computer-controlled walking robot started to be feasible. In 1970s McGhee's group at Ohio State University started one of the first projects on a six-legged robot. This research resulted in the construction of an insect-like hexapod robot [1], which was able to walk on flat surfaces, following a cyclic gait. Later work focused on adding more flexibility to deal with rugged terrain and lead to the implementation of a heuristic solution to overcome *deadlock* (those configurations in which the robot is not able to place a leg in the planned position) [2].

A lot of effort has been spent in the attempt to gain a general solution for the control of walking. Early in the development of artificial walkers, an optimization approach for multi-legged robots has been outlined by Kugushev [3]. This work presents a general method to search for an optimal walking gait. A sequence of steps that optimize the energy consumption and the corresponding static stability during motion is defined for 4,6 and 8-legged robots. However this strategy turned out not to be easily extended to handle rugged terrain, because of the over-simplifying assumptions adopted in the description of the system, in particular the definition of the stability measure as a function of the gait.

In 1984 Hirose constructed a quadruped robot, implementing a statically stable gait. The robot lifted one foot at the time off the ground and tilted the body to keep the center of mass within the base of support provided by the remaining three feet. This turtle-like machine was able to walk and overcome small obstacles. The work also focused on the energy optimization problem, which is considered to be a crucial point for autonomous walkers. A new leg pantograph mechanism (PANTOMEC) was proposed, and a robust approach for stable walking outlined. The quadruped robot was able to walk upstairs and used touch sensors on its feet, for obstacle avoidance. Robust control was achieved for walking on a flat surface and the energy optimization led to an average power consumption of only 25 W [4]. The static nature of this solution prevents this method from being extended to systems which do not keep at least three supporting feet on the ground (e.g. fast walking quadrupeds, bipeds).

A completely different approach was taken in 1983 by Raibert, for his single-legged hopping machine. The robot was able to stabilize itself while moving towards a stated location at constant speed. This work provided an important breakthrough in the field of walking machine research, because it demonstrated the importance of a dynamically stable system. The agility and flexible behavior of many animals is achieved through a dynamic control of the limbs rather than a statically stable step sequence. Later Raibert proposed a general approach to running, as a dynamical control problem, reducing the 2 and 4-leg running to a hopping sequence [5]. The running of multi-legged robots, can actually be described as a sequence of jumps, and the symmetry of the system allows a reduction of the equations of motion to a single *virtual leg* problem. This idea has been supported by more recent theoretical work [6], which includes a detailed analysis of hopper-terrain interaction. Note that this robot was able to run, following a sequence of jumps, but did not address the problem of articulated walking.

In 1992 Hirose implemented both statically stable walking, and dynamically controlled running on the same robotic platform and in simulation. The results of this work, though encouraging, suggest that a more accurate analysis of the foot-ground interaction is fundamental for further developments [7]. In this research a statically stable gait is used for walking and a dynamically stable one for running (sequence of jumps). The system does not integrate the two separate control issue into a single framework of control.

A new strategy for implementing adaptive control of walking systems based upon learning, was provided in 1992 by Min & Bien, who successfully applied an optimization method for foot placement on a simulated statically stable quadruped. Their work focused on the search for foot placements, that minimizes the energy cost for locomotion in the specific conditions of the terrain surrounding the robot [8]. In the same year Chiel suggested a neural architecture for controlling a purpose-built hexapod robot. The author showed how a simple neural network can solve the problem of adapting to changes in the operating conditions. The robot was able to still perform efficiently when a leg control unit failed. A limitation of this work is that the mechanical design of the robot was developed to maximize static stability (short leg and wide body), resulting in a low degree of flexibility on non flat surfaces [9].

In the attempt of generalizing learning strategies some new paradigms have been proposed in the last few years. In 1992 Lewis proposed a genetic algorithm (GA) to train a neural network controller for an hexapod walking robot. The genetic algorithm was used to change synaptic weights. One neural network was trained to produce an oscillatory pattern for the movements of each leg, then an assembly of this neural nets was used to coordinate oscillations of different legs. During the evolution process parameters defining the nature of oscillations were also changed [10]. The work clearly indicates the usefulness of GAs combined with neural networks for control, but suffers from some disadvantages. Firstly, a substantial support from an external operator is required to ensure the convergence of the learning process. Secondly, the neural network described in the paper is a relatively simple system in which the search space of the genetic optimization has low dimensionality and the optimization function exhibits a relatively smooth dependence on the parameters. The same approach is not easily extended to dynamically stable systems, where often cost functions are extremely sensitive to small changes in parameter values.

1.1.1 The Biological Analogy

More recently observation of biological walkers has provided inspiration for new paradigms of control. Normal walking behavior in animals can be described as a sequence of periodic leg movements. Evidence from the physiology of walking suggests that basic control of limb movements is provided by rhythmic patterns of activity in assemblies of neurons in the spinal cord, without intervention of the higher motor cortex [11]. When an animal walks, a specific pattern of muscle activation is generated by these neuronal assemblies, called central pattern generators (CPG). Generally a CPG is activated at the same point in time that locomotion behavior begins, thereafter the neural activity sustains the periodic muscle activation. Observations of the time evolution of joint angles during walking suggests that the periodic nature of limb movements tends to remain stable until a perturbation occurs (e.g. an obstacle is encountered, or the terrain conditions change) [12]. A cyclic trajectory in joint space is followed, without significant sensory feedback occurring at fast time scale other than joint position estimation, until an event occurs which brings the system out of balance. This is comforted by the evidence that spinal CPGs can provide open loop control over the muscular system, with little sensory feedback and no descending control from the brain [13, 14]. In recent work this hypothesis has inspired the development of legged robots, controlled without feedback or very limited sensory input, and essentially relying on the intrinsic stability of the robot's mechanical structure under the periodic action of joint actuators [15]. An example of this approach focused on a "stick leg" based design where a special rocking gait provided dynamic stability. The mechanical structure of this system and the gait chosen greatly simplifies the control problem as the robot's dynamics is essentially planar. Although similar results for stable dynamic walking in articulated legged robots exist in the literature [7], it is not obvious what a suitable set of CPGs is that provide stable open loop control given a particular mechanical design.

Observation of newborn animals suggests that CPGs are in many cases present at birth [11, 16]. Animals learn relatively quickly to walk, and in some cases a few minutes after birth. This suggests that much of the motor control for walking is learned through evolution rather than acquired through experience. The precoded portion of the motor control system could be thought of as a set of initial CPGs that the animal can access at birth. This set of CPGs is the result of evolution through adaptation. It is worth noting that these quadruped biological walkers implement gaits which are not statically stable, thus not controllable through kinematic optimization only.

Natural selection can be described as a search strategy applied to a very large parameter space, where the optimization criterion is imposed by the environment. This search process results in a set of cyclic patterns of movements, capable of producing stable locomotion right after birth.

1.2 Dynamic vs Static Stability

As mentioned in earlier sections, in the pursuit of an efficient strategy for controlling walking machines two main approaches have been followed: *static* (or passive) balance and, in fewer cases, *dynamic* (or active) balance. The first strategy was mainly inspired by insects which have a relatively small nervous system and yet they are able to walk very efficiently and adapt to many different environments. The body structures and control systems of many different types of insects have been the base of numerous walking robot projects. The second approach is based on the observation that momentum is in many cases a source of stability for systems non-intrinsically stable. As one observes when riding a bicycle or when bouncing a basket-ball, motion helps the control task. The major consequence of motion based stability is the fact that absence of motion results in loss of controllability (e.g. it is hard to keep one's balance sitting on a still bicycle). This implies that constant actions are required for keeping a system balanced when motion constitutes its source of stability.

1.2.1 Insect-Like Robots

The choice of an insect-like architecture is mainly due to the simplifications it brings to the corresponding control system. An insect has an intrinsically stable body structure, which allows it to stay balanced at any one time during its motion. With six legs (or more), it is able to keep at least three of them in contact with the ground, providing a triangular supporting base. In fact during walking insects alternate three supporting legs (providing stability) with three swinging legs (moving forward to new positions). When these operation is repeated in a cyclic fashion walking occurs. During locomotion the main problem to be solved for this type of walkers is to work out the sequence of foot placements that optimize the stability of the system in presence of obstacles or forbidden foot positions. This problem is known as the *gait optimization* problem. The study of this kind of artificial walkers has given some very useful insights into the motor control systems of insects.

Several models of control have proved successful for robots inspired by insects. One biologically plausible example is the work developed by Randall Beer and colleagues [17]. They proposed a neural oscillator system which was composed of coupled leg-unit controllers. The individual controllers were based on the model proposed by

Pearson and colleagues [18], where six neurons are connected through inhibitory and excitatory synapses (see Figure 1.1).

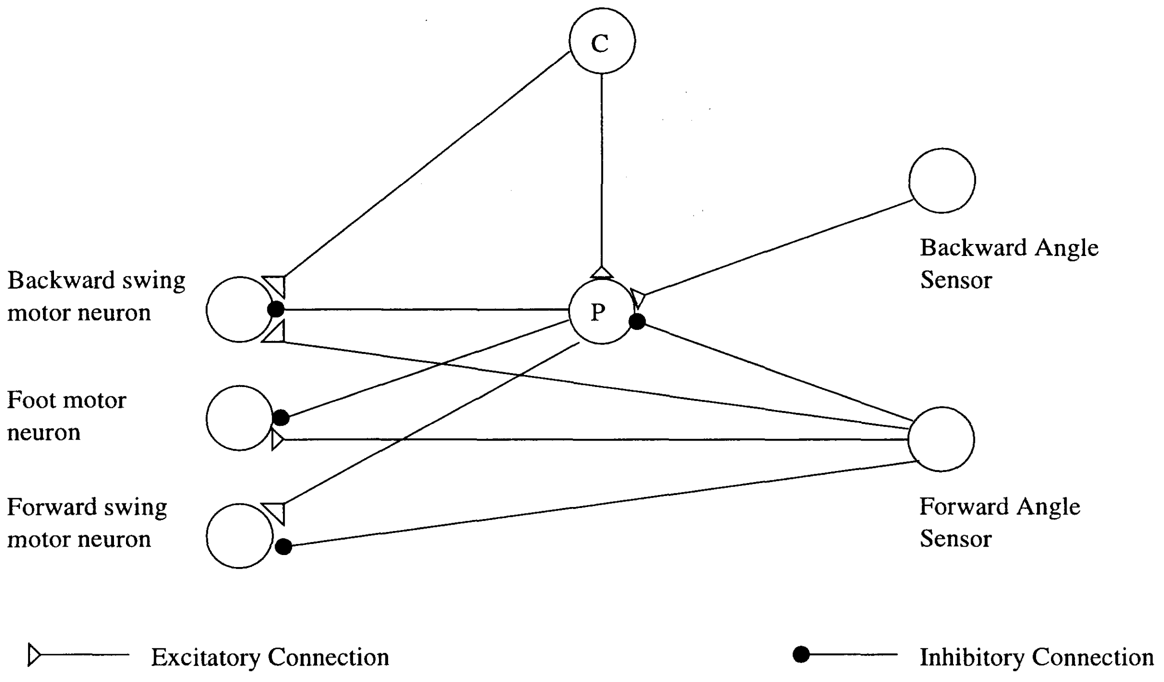


Figure 1.1: Beer's hexapod leg controller. C: Command neuron. P: Pacemaker neuron. Adapted from Chiel *et al.*, 1992. Six units of this type have been employed for the control of an hexapod walker.

The overall controller of walking is composed of six leg controllers activated by a single command neuron. The state of each neuron is described by the following relation:

$$C_i \frac{dV_i}{dt} = -\frac{V_i}{R_i} + \sum_{j=0}^{n} w_{ij} f_i(V_i) + INT_i + EXT_i \tag{1.1}$$

where V_i, R_i and C_i represent the voltage, membrane resistance and membrane capacitance of the i^{th} neuron; w_{ij} represent the strength of the connection between neurons j and i; f is an activation function that rises linearly above threshold until it reaches saturation; ETX_i is the external current injected into the neuron and INT_i

is a an intrinsic current that causes the pacemaker neurons to oscillate, providing for synchronization similarly to the clock of a microprocessor.

Individual leg controllers are coupled through inhibitory synapses with adjacent neighbors, shutting the corresponding pace-makers when active. This produces a wave of stepping which traverses the body of the robot from rear to front (i.e. the rear leg swing before the middle one and the middle leg swings before the front one).

This relatively simple neuronal system can produce various gates, when the inhibitory pattern is varied. Also, it proved robust to the failure of single controllers or to disrupted communication between them. The only sensory information used by the controller is to determine the direction of swing and whether or not a foot is touching the ground. This work shows that in cases where a kinematic description of the system is sufficient, quasi-open-loop control is feasible and coupled oscillatory behavior leads to stable locomotion.

Another example of control strategy for insect-like robots was proposed by Parker in 1997. His research outlines a framework to develop stable walking for an arachnid robot (i.e. eight legs) [19]. A cyclic genetic algorithm is used as learning strategy to acquire optimal gaits in the case of six- and eight-leg walking. A purpose-build spider-like robot (Stiquito) was used to test the learning method. The aim of this work was to show that stable walking can be achieved through an optimization of foot placement sequences, with no a priori knowledge of how to walk. The only information given to the system is embedded in the representation of the genetic material. The cyclic behavior is enforced in the definitions of genes for a controller, where a part of the genome is considered to encode repetitive control.

In Figure 1.2 a representation of the foot sequence found by the algorithm, is depicted. The optimization converged after 2000 iterations and each generation consisted of 64 individuals. Two types of cross-over operations were used: a) genes where picked entirely from a random position in the genome of either one individual;

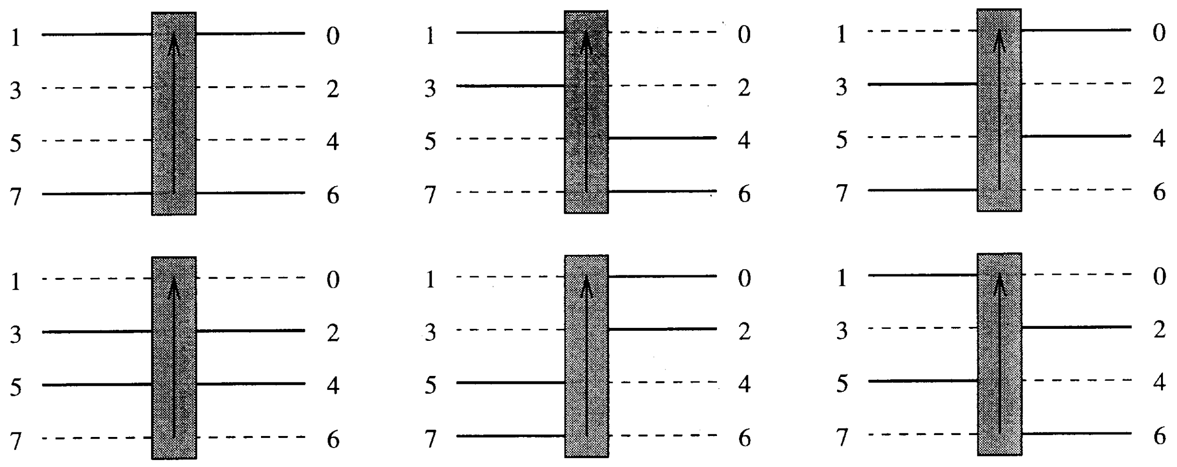


Figure 1.2: Parker's arachnid eight-legged gait. Adapted from Parker and Rawlings, 1997. The dashed lines indicate legs in the swing phase, whereas the solid lines represent the support legs, in touch with the ground.

b) a gene-by-gene cross-over picked sub-gene sequences of every gene in the genome. The genetic representation allowed variable length genes for the cyclic section of the genome. This part of the genetic material is considered cyclic in nature and positions are referenced cyclically - when the last gene is reached, the genome would restart to be referenced from the first cyclic gene.

This work provides a good example of genetic optimization as a means to learn walking behavior in statically stable systems. It clearly shows that no knowledge of the walking process, aside from its cyclicity, is needed to achieve stable optimal locomotion.

1.2.2 Dynamic Stability: A Costly Feature

The limit of insect-like walking research lies in the lack of a detailed model of the system's dynamics - in the case of statically stable systems this level of description is not a requirement. This restricts the applicability of the methods developed for these systems to robots with where dynamical effects can be ignored. Though it does not extend easily to systems with complex dynamics, the various learning frame-works found in the literature for statically stable control have proved effective, both theoretically and in experiments.

The goal of dynamically stable control is considerably more difficult than gait optimization. A dynamically stable legged robot is one that has to constantly apply some force or torque to its joints, in order to keep its balance when moving. This is because during locomotion, a dynamically stable walker, such as a quadruped, does not always keep at least three feet on the ground at all times to provide a stable support. According to the speed it is moving it may have a minimum number of four (still), three (slow walking), two (fast walking) or no legs (running) in contact with the supporting surface. The problems posed by the control of such a system are very complex, because of the real time computation constraints (i.e. the system cannot stop to compute what the next move, because it would fall over) and the highly non-linear behavior of its dynamics. On the other hand there are several advantages in favor of dynamically balanced systems. From the observation of walking animals, it clearly appears that a dynamically stable walker is capable of flexible, smooth and fast locomotion. Also the ability to quickly and continuously correct their posture, and the high peak power of their muscles, allow them to efficiently overcome complex situations. Biological walkers can jump over obstacles, climb slopes and walk upstairs with apparently little effort. Of course all of these abilities come at high cost in terms of sophistication of actuation mechanisms and complexity of control. Animals have an exceptionally favorable power to weight ratio, their nervous system is capable of highly sophisticated computations in a matter of milliseconds. To match such high specifications with an artificial walker is as yet out of reach. One can, though, try to reverse engineer some of the properties of biological systems. Especially appealing, for their immediate feasibility, are the ones connected with the low level motor control.

The simplest form of dynamic control of walking could be considered to be the one of a four legged robot - a machine with the maximum number of legs to be

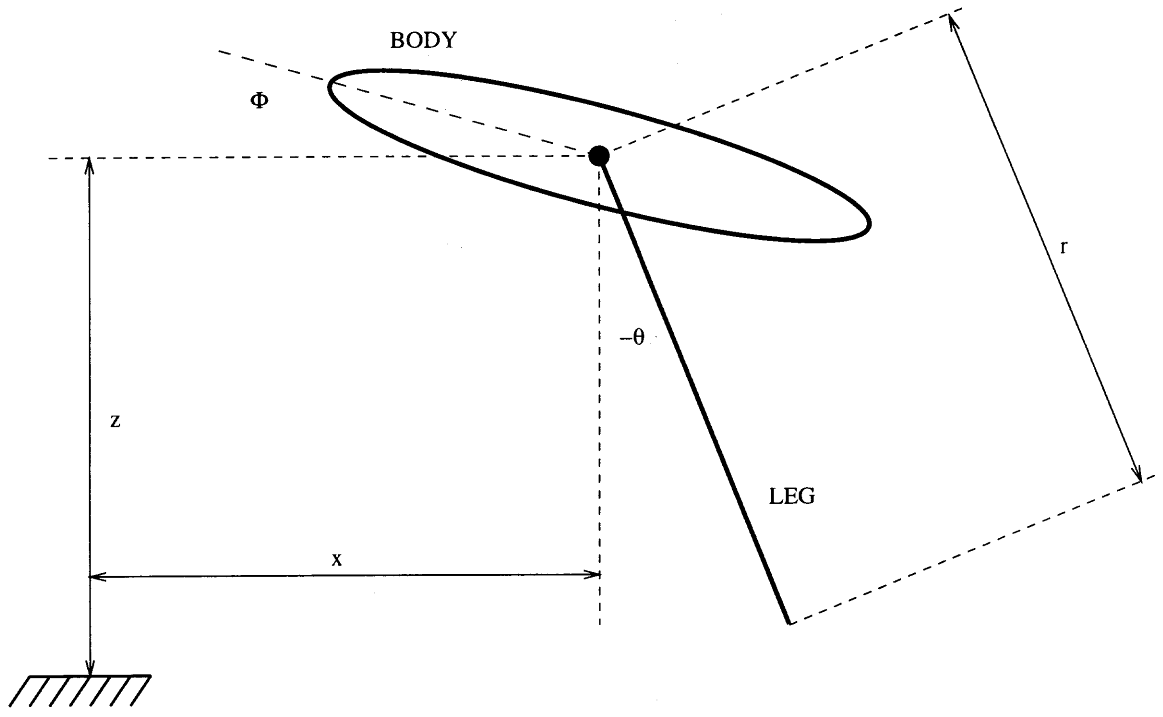


Figure 1.3: Schematic representation of the planar hopping machine. Adapted from Raibert, 1986.

statically unstable. In actuality it has been demonstrated by Raibert that a single leg hopping machine can be controlled using a simple straightforward strategy. The idea here is that the machine bounces continuously spending most of the time off the ground. Corrections to the posture are applied only during the short periods of contact with the ground. Trajectories during bounces are considered ballistic and a simple predictive model of the dynamics can be used to describe motion [20].

For the planar case the equations of motion of a hopping machine can be expressed as follows - the leg is assumed to have no mass, which is a reasonable assumption since its actual mass is negligible with respect to the mass of the body:

$$\begin{aligned} m\ddot{x} &= f sin\theta - \tfrac{\tau}{r} cos\theta \\ m\ddot{z} &= f cos\theta - \tfrac{\tau}{r} sin\theta - mg \\ J\ddot{\Phi} &= \tau \end{aligned} \tag{1.2}$$

where x, y, Φ are the horizontal, vertical and angular positions of the body; r, θ are the leg length and orientation; τ is the hip torque; f is the axial leg force; m, J are the body mass and moment of inertia; g is the gravitational constant.

The hopper's airborne trajectory can be estimated using the above equations and, in the assumption of a massless leg, the time of impact and posture can be determined. The control problem is thus reduced to the computation of the axial force (along the leg), needed to maintain the hop, and the torque required to adjust the subsequent hop trajectory.

Raibert has further extended the theory to a 3D hopper and multi-legged machines. In the case of many legs the walker's description can be reduced to the one of a single virtual leg hopper, exploiting the system's symmetry [5].

The relatively simple and fast control strategy comes at a high cost in terms of mechanical realization of the robot. Firstly a complex pneumatic actuator is utilized to provide for the fast movements of the leg during flight. Secondly an accurate measurement of the position of the center of mass of the body of the robot is necessary for a sufficiently accurate estimate of the time of contact. This work demonstrates that dynamic control is not only feasible, but can provide high mobility and flexibility of behavior. Raibert's hoppers can produce a variety of movements, including somersaults, and have been tested carrying loads and up to speeds of several meters per second.

The major limitation of this robots is the high stress transmitted to the ground at contact (soft materials are not usable as supporting surface) and the inability of changing course between hops. Substantially the major result of this research is to prove that dynamic control of running, which can be viewed as a sequence of hops, can be achieved through an efficient and simple algorithm. The high level of mechanical complexity of the resulting walking machines is somewhat expected from the observation of biological runners. For this systems it is clear that a very favorable power to weight ratio and high peak power are of paramount importance.

1.3 Statement of the Problem

The goal of this dissertation is to demonstrate that stable control of a quadruped walking robot can be obtained through a parametric optimization process, in the absence of higher level information about legged locomotion. A suitable parameterization of the walking behavior is developed and a genetic search is adopted to achieve stable locomotion with a walking robot. The only assumption made is the cyclicity of the behavior.

A set of coupled oscillators are chosen as the basis for the control of the walking periodic behavior. An algorithm for searching stable patterns of oscillation in a simulated environment is developed. The analysis of the experimental results is aimed at proving that stability of walking arises from a favorable coupling of periodic patterns of oscillation.

Many examples are present in the literature that demonstrate how walking behavior can be implemented in quadruped as well as biped artificial walkers. In spite of this results much has to be investigated in the area of adaptive/reactive behavior and learning. The mechanisms underlying motor control in animals are still largely obscure.

The framework chosen here does not propose to improve or elaborate previous attempts at locomotion control of mobile robots, but rather it aims at investigating the possibility of achieving quasi open-loop control of a dynamical system exploiting its natural stable limit cycles [21].

1.4 Structure of the Thesis

Chapter 1 has reviewed the literature on artificial walking. The discussion has shown the progress in the field and outlines different strategies to achieve stable control of walking. Chapter 2 states the intent of this thesis and the framework in which the work has developed. Chapter 3 describes previous work in the area of dynamic simulation for robotic applications. It presents theoretical background on simulation together with some considerations on the choice of the specific method used throughout this thesis. The software tools which are developed as part of the present work are presented together with a description of their use. Chapter 4 compares different optimization strategies and covers specific advantages and disadvantages of them. It introduces the use of Genetic Algorithms and their specific application to the walking problem. Chapter 5 gives a detailed description of the mechanical design of the walking robot, together with its electronic hardware and software interface. Chapter 6 presents the experimental results for the simulation method adopted, with particular attention to its ability to capture the dynamics property of a real system. Also details of the experiments with the search algorithm are discussed, with comparisons amongst the several variants implemented. Chapter 7 draws conclusions about the achievements of the work and the possible future developments.

CHAPTER 2

ALTERNATIVES AND DESIGN CHOICES

2.1 The Assumption of Open Loop Control

From the previous discussion it might seem that for statically stable walkers very little sensory feedback is required, at least in the case of flat surfaces. Whereas for dynamically stable machines a substantial amount of input from a variety of sensors is required for stable control. Although this sounds like a reasonable assumption, it is not the case for biological systems, where very little sensory input is used during locomotion over regular surfaces. For these systems it appears that stability is achieved through repetitive a actuation that couples with their dynamics in a stable periodic fashion. Basically walking appears to be the repetition of a certain pattern designed to produce stability with little feedback [22].

Some work has been devoted to explore this assumption, and results exist in the literature both for simulations and experimental robots. As stated earlier there is sufficient evidence to suggest that limit cycles exist in the dynamics of quadrupeds. These limit cycles attractors can be though of as the result of a process of search for an optimal mechanical structure coupled with an optimal control strategy. In the case of biological system the search can be identified in evolution through selection of the fittest.

The objective of this work is to demonstrate that, using a suitable framework of simulation and an experimental robot, it is possible to identify stable attractors for the dynamics of walking. The assumption of open loop control, at the basis of the method used here, can be summarized in the following: walking can occur with simple sensory information deriving from joint angle position and vestibular information, without a detailed model of the system's dynamics used as a predictive tool for feedback, as long as the system moves within the basin of attraction of a specific limit cycle.

The basin of attraction is defined by the dynamics of the mechanical system as well as by the environment the walker moves in. Flat surface, randomly ragged terrain, regular structures (e.g. stairs) are examples of conditions within which specific limit cycle could produce stable locomotion. When the conditions change a different attractor has to be exploited.

2.2 Optimization as a Learning Strategy

The search for stable walking patterns for legged robots using CPGs involves a suitable optimization strategy. The idea here is to apply a learning by optimizing strategy. Instead of designing a learning algorithm, which acquires the appropriate control parameters through experience, an optimization process selects parameters that lead to suitable behavior. This is similar to say that a controller does not learn to walk, but is, instead, provided with some a priori, blind knowledge by an external evaluation process. This knowledge (i.e. the set of parameter values) is then used, unchanged, throughout the operation of the controller.

Many approaches are available for parameter optimization from different variations of gradient-based, to simulated annealing, to dynamic programming [23, 24]. Gradient descent is generally easily implemented when the functional dependency of the cost function from the optimization parameters is known. It can still be applied when there is no explicit knowledge of the cost function, although the corresponding algorithm reveals computationally costly. Dynamic programming [25] is a technique which requires a detailed model of the system, together with a criterion to establish optimality and sub-optimality. In the case of the problem at hand here, this poses serious limitations due to the mathematical complexity which is implied, together with the unavoidable approximations in the description of the system.

For the purpose of this work a genetic algorithm (GA) [26, 27] has been selected for several reasons described in detail in Chapter 4, of which the most important is its ability to efficiently cope with large parameter spaces. Also, GAs have proved suitable for optimization problems where local maxima exist with very narrow peaks in the cost function space. As it will be shown later the walking stability is very sensitive to the control parameters.

Previous work has focused on GAs as a means to search for optimal walking gaits in hexapods [10, 28]. In this case, though, optimality was intended as the best sequence of foot placements leading to forward stable locomotion. In most of the work on gait generation the dynamics of the mechanical system of robotic mechanisms is ignored. This has been justified by the static stability of insect-like robots, where gravity and other dynamic effects can be ignored.

In the study of systems with dynamic stability, where constant active changes of posture are required to ensure balance, a kinematic analysis is not sufficient to completely describe the behavior of the system. In this case a more detailed description of the physical world is required. In particular the dynamics of the mechanical system, its interaction with objects in the surrounding environment, gravity and friction have to be taken into account. This type of description can be implemented in a dynamic simulator, which can be used to evaluate the behavior of a range of different robot designs. Control strategies can then be tested in simulation before any mechanism is implemented. In addition small changes to a successful strategy are potentially harmful to the mechanical components of an actual robot. Though potentially very useful, this approach depends on a significant level of accuracy in the simulation. A good estimate of the interaction between the robot and the real world has to be captured by the simulation.

2.2.1 A Convenient Choice of Parameters

The cyclic nature of walking makes it possible to represent the parameters of locomotion as a set of coupled oscillators. In this assumption a convenient choice of parameter space is the discrete frequency domain. A controller is thus completely described by its frequency content (i.e. a set of Fourier coefficients). The number of required Fourier components can be experimentally determined, as a mechanical system can only respond to a limited range of frequencies (i.e. its frequency response).

Walking, as other cyclic behaviors, can be thought of as the generation of a specific pattern of coupled oscillations of the limbs. Several models are available in the literature, which model neural coupled oscillators [29, 30], found in the nervous system and thought to be responsible for CPG generation. Although the problem of modeling realistic neural oscillations is a very interesting one, here the aim is to investigate the idea that coupled oscillations of limbs can lead to stable open loop control of walking. The biologically realistic models of CPG oscillators have the disadvantage, at least for optimization purposes, of including a large set of parameters. With this in mind a general oscillator model composed of simple harmonic components is used.

The Fourier parameterization of the problem leads to a small set of numbers describing the properties of a controller entirely. This set of numbers constitutes the parameter space in which to search for optimal stable walking.

2.3 A Framework for Artificial Walking

The framework chosen to implement the control of quadruped walking has gone through several stages of implementation. Many of these stages are not independent from each other and some prerequired phases of development have had to be completed before any useful testing could be carried on the rest of the system. The planning of the complete system has progressed keeping in mind the following list of steps:

- Choose of a suitable representation for CPGs and their codification into a sequence of real valued numbers
- Develop a dynamic simulator for the evaluation of different control parameters
- Develop a search method to optimize the control parameters according to a fitness criterion
- Construct a four-legged robot, together with the necessary electronics and interface software, for the validation of the overall strategy
- Perform a set of validation experiments to estimate accuracy of simulation
- Test the optimized walking patterns and analyze their stability

The order of this list does not reflect the order of implementation, but rather the logical sequence followed in planning towards the final implementation. The main planning effort has been to keep a clear view of what the final setup would look like and how the building blocks would fit together.

The idea of CPG-based control has been a driving force, reinforced over the years by the building evidence in the literature and the ever-improving results of research efforts in the field.

Particular attention has been devoted to the development of the simulation part of this work to ensure flexibility of use and speed of computation. In the early phases of development of the actual walking robot, the simulator has been a valuable tool providing information on different mechanical designs and on the suitability of actuators.

The search method has proved efficient and has worked well since its early implementation. Many improvements have been added but the original design still stands for the most part.

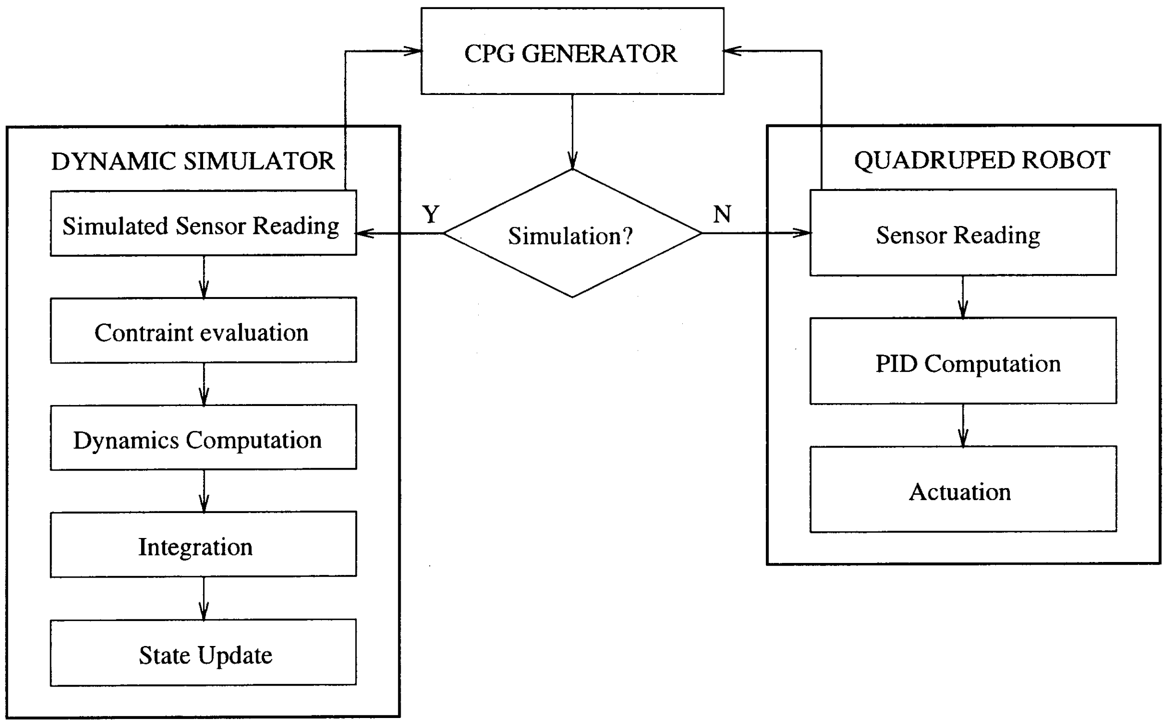


Figure 2.1: Schematic of the interaction between the controller and the simulated or actual robot.

Figure 2.1 shows how the controller interacts with the simulator and the robot. It has to be noted that the controller does not distinguish between the simulated robot and the actual one, treating them in the same way. This connection scheme between the controller and the actual/simulated robot has allowed to easily and quickly test on the quadruped robot the control strategies obtained in simulation. In Chapter 6 an experimental validation of the assumption of simulation correctness is presented in detail.

Results obtained in simulation have to be validated carefully before any conclusion on the correctness or usefulness of the method can be drawn regarding the actual robotic walker. A set of validation experiments is therefore required as an intrinsic part of the development process to ensure that the right design choices are made and that optimization strategies are implemented correctly.

CHAPTER 3

DYNAMIC SIMULATION

3.1 Introduction

Multi-body dynamic simulation has been for many years a very active research area. Recent developments, with ever increasing availability of inexpensive computational power, have produced a number of new approaches for multi-body simulation. The two most important selection criteria among these simulation methods are speed of computation and accuracy. A simulation strategy can be slow though very accurate, which is in some cases desirable (e.g. car-crash tests are simulated for periods of few milliseconds, but high accuracy is crucial). In other applications speed is crucial - for example in virtual reality systems - and in most cases achieved by sacrificing accuracy.

When the goal of simulation is parameter optimization, a fast simulator is of paramount importance, since the number of individual systems to be evaluated is usually very large. This is especially true when the dimensionality of the parameter space is high and the system has a non-linear behavior. The non-linearity makes the granularity of the search small, as even a slight change in parameter values can drastically alter the behavior of the system. Implementation choices in this case are driven by speed of simulation, as presently available computational models and affordable computer hardware make high accuracy unobtainable in a reasonable time-scale.

Several approaches have been proposed for simulating multi-body dynamics for robotic applications. Methods have been developed with computational complexities $O(n)$ in the number of degrees-of-freedom (DOFs) and in some cases as low as $O(logn)$ [31]. In some cases, this performance has been achieved by exploiting parallelism of new computer architectures. In spite of encouraging progress, there is a need for further development in cases when contact between bodies and collisions occur [32].

In general there are two classes of simulation paradigms for the computation of the forward dynamics of mechanical systems. The first one is based on the computation of the varying inertia dyadic where constraints are imposed to the equations of motion in the generalized coordinate system through explicit constraint equations. The second one considers the constraint forces as external forces acting on free moving bodies. In the following sections a brief introductions to these different approaches is presented, pointing out their advantages and disadvantages.

3.2 Constraint Based Dynamics

Constraint based simulation of multi-body dynamics has received a great deal of attention from researchers in different fields. Primarily mathematicians have built a theoretical framework on which to base computational models. The main idea behind this approach is the calculation and reduction of the equation of motion in some suitable generalized coordinate system. A multi-link structure can be approximated, in most cases, as a set of rigid bodies connected by joints. In this view the number of degrees of freedom is reduced, from a potentially high dimensional space into a space with cardinality in the order of the number of joints. The rigidity assumption leads to a compact description of bodies in terms of six scalars describing the position and orientation. The assumption that the work along constrained directions and rotation axis is zero, brings a further reduction in the number of "free" variables.

The reduced description of the dynamics can then be expressed through reduced equations of motion. Several methods exist to compute the reduced expression of the multi-body dynamics of a mechanical system, both iterative-sequential and suitable for parallel implementation. In order to then integrate such equations it is necessary to compute the reduced inertia dyadic. Once the dyadic is computed it needs to be inverted. These operations are generally quite costly. Although recent methods can perform them with computational complexity $O(n)$, in the case of tree-like structures, the computational overhead associated with the extra data structure required by these algorithms remains quite high [33, 34].

3.2.1 Featherstone Algorithm

One of the most frequently used methods for the computation of the equations of motion for a multi-body system is the articulated-body inertia algorithm by Roy Featherstone [33]. This method has a computational cost of $O(n)$ in the number of bodies constituting the mechanical system. The algorithm works iteratively through a chain of n bodies from the last to the first. Initially only body n is considered as part of an *articulated body* and the rest of the system is described as the *handle* located at joint i. At each iteration one more body is added to the articulated body sub-chain. The key concept is to work out a relation between the accelerations in the bodies before the handle and a resulting total acceleration acting, through the handle onto the articulated body. This allows the description of the system to be reduced to the description of the articulated body on which an external solicitation acts. The initial base case reduces to the solution of the single free body n dynamics expressed by the Newton-Euler equations:

$$\begin{aligned} \mathbf{f}_n^I &= m_n(\mathbf{a}_n - \mathbf{g}) \\ \boldsymbol{\tau}_n^I &= \mathbf{I}_n \boldsymbol{\alpha}_n + \boldsymbol{\omega}_n \times \mathbf{I}_n \boldsymbol{\omega}_n \end{aligned} \tag{3.1}$$

where m_n is the mass of body n, and $\mathbf{I}_n$ is the inertia tensor in the body coordinates; $\mathbf{a}_n$ and $\mathbf{g}$ are the spatial gravitational acceleration acting of body n. This equation can also be expressed in matrix form:

$$\begin{bmatrix} \mathbf{f}_n^I \\ \boldsymbol{\tau}_n^I \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{M}_n \\ \mathbf{I}_n & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_n \\ \mathbf{a}_n \end{bmatrix} + \begin{bmatrix} -m_n\mathbf{g} \\ \boldsymbol{\omega}_n \times \mathbf{I}_n\boldsymbol{\omega}_n \end{bmatrix} \tag{3.2}$$

in the case in which $\mathbf{M}_i = m_i\mathbf{1}$*. Equation (3.2) can be rewritten more compactly as follows:

$$\hat{\mathbf{f}}_n = \hat{\mathbf{I}}_n^A \hat{\mathbf{a}}_n + \hat{\mathbf{Z}}_n^A \tag{3.3}$$

where the superscript A refers to the articulated body. These quantities differ from their counterparts without the superscript A, used later, which refer to the body in isolation. For the trivial case of a single body in the articulated chain $\hat{\mathbf{I}}_n^A = \hat{\mathbf{I}}_n$ and $\hat{\mathbf{Z}}_n^A = \hat{\mathbf{Z}}_n$.

In the case in which the articulated body consists of a sub-chain rather than a single isolated object an iterative expression can be set up, which expresses the Newton-Euler equations for body $i-1$ in terms of body i

$$\hat{\mathbf{f}}_{i-1}^I = \hat{\mathbf{I}}_{i-1}\hat{\mathbf{a}}_{i-1} + \hat{\mathbf{Z}}_{i-1} + {}_{i-1}\hat{\mathbf{X}}_i\hat{\mathbf{f}}_i^I \tag{3.4}$$

where ${}_{i-1}\hat{\mathbf{X}}_i$ is the coordinate transform from body $i-1$ to body i. To put (3.4) in a usable form the joint force $\hat{\mathbf{f}}_i^I$ has to be eliminated from the equation expressing $\mathbf{a}_i$ in terms of $\mathbf{a}_{i-1}$. For details on how to solve for $\mathbf{a}_i$ in the case of a revolute or prismatic joint see [35].

The forward dynamics of a multi-body chain can be computed using the above representation following four main steps:

1. Compute velocities iteratively from the first to the last body in the chain.

*$\mathbf{1}$ is the identity matrix.

2. Initialize the articulated inertia and joint forces to the isolated case for each body in the chain.

3. Compute actual articulated inertia iteratively from the last to the first body in the chain.

4. Compute accelerations iteratively from the first to the last body in the chain.

In the above list of computational steps it is clear that although Featherstone's algorithm has computational complexity $O(n)$, it iterates through all the objects in the simulated mechanical system several times. Each iteration adds several algebraic operations on matrices of dimensions greater than or equal to the number of degrees of freedom of individual objects. This makes the algorithm compare favorably against other higher order methods only for a large number of bodies. Also it makes it computationally more expensive than the linear methods described in the later sections, which perform the forward dynamics computation in a single pass through all the objects. In spite of these drawback Featherstone's method is usually more accurate than other linear methods, because of its exact formulation of the joint constraints.

3.3 Free Interacting Body Dynamics

Another approach developed for fast interactive realistic simulations is the one taken by Brian Mirtich in his impulse based method [35, 36]. The idea here is to consider every rigid body as unconstrained and then impose constraints as external impulses. This approach is very efficient for contact calculations and in particular interaction with the ground, though problematic for representing articulated bodies. In fact the description of articulated joints is totally missing in the basic version of the impulse method. Later a hybrid approach was adopted to include joint constraints, which suffered from the same disadvantages of constraint-based methods, described above [37, 38, 39].

The free body idea was used in the development of another algorithm, proved to be efficient for fast simulations. Baraff used Lagrange multipliers to compute the instantaneous reactions at the joints of an articulated mechanism [40, 32]. This method has a complexity of $O(n)$ and proven powerful for robotics applications. Its main computational expenses lie in the calculation of the appropriate multipliers at each step of the simulation. This implies solving a system of linear equations with cardinality equal to the number of constraints.

3.3.1 Penalty Method

A faster approach, based on the free interacting rigid body framework, is the penalty method [38]. This approach computes the free body dynamics for every rigid component of the system. The joint constraint are considered external forces generated by very stiff damped springs. A schematic representation of the idea behind this method is depicted in Figure 3.1. This method is extremely fast, it has a computational cost $O(n)$, and provides for a rough approximation of the effect of vibrations.

The characteristics of vibrational states in the mechanical structure are highly dependent on the parameters of the damped spring enforcing the constraint. The aim here is not to accurately describe vibrations through a chain of bodies, but rather to get a fast, first approximation of the dynamics of a robotic mechanism. As it will be shown later in this thesis, the accuracy of the simulation produced by the penalty method, though not rising to the level of more sophisticated approaches, proved sufficient for the development of stable control of walking. The equations describing the motion of the system using this method can be expressed as follows:

$$\begin{aligned} m^i \hat{\mathbf{I}}^i \ddot{\mathbf{R}}^i &= {\mathbf{F}_e}^i + {\mathbf{F}_c}^i \\ \hat{\mathbf{J}}^i \ddot{\boldsymbol{\theta}}^i &= {\boldsymbol{\tau}_e}^i + {\boldsymbol{\tau}_c}^i \end{aligned} \tag{3.5}$$

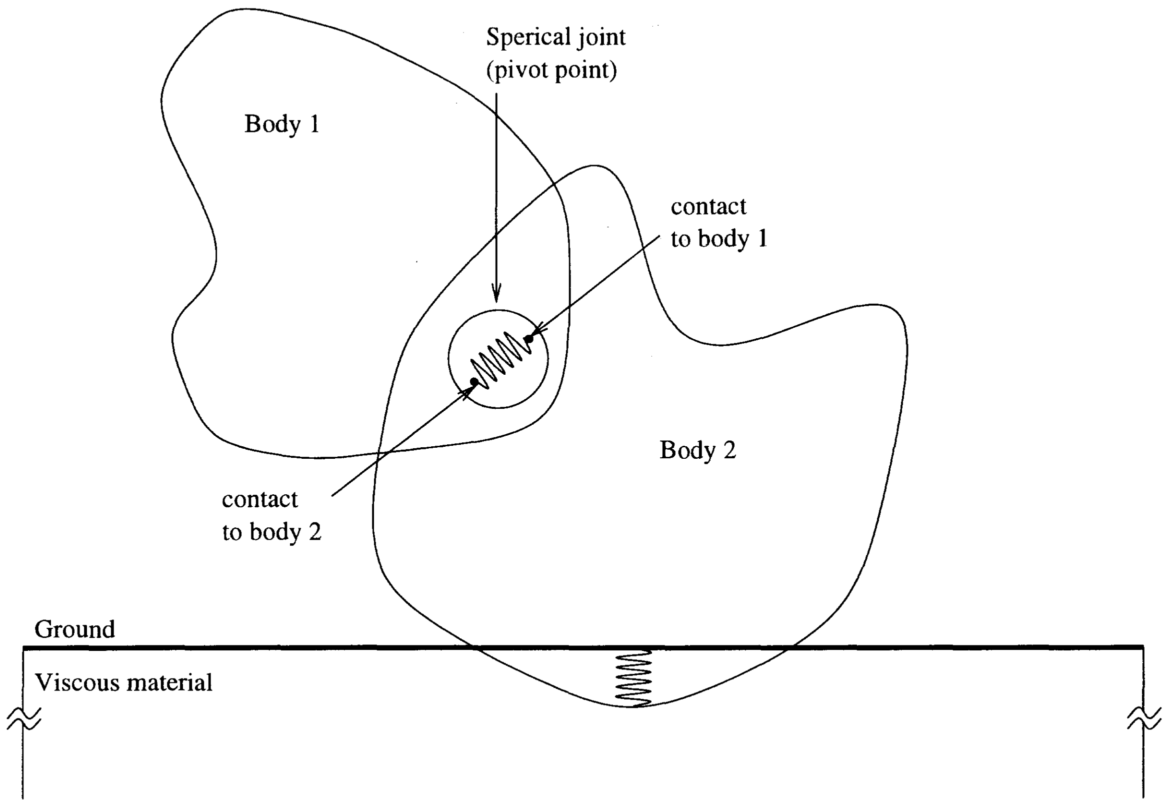


Figure 3.1: A schematic representation of the joint model and ground contact for the free interacting body method.

where $\mathbf{F}_e{}^i$ and $\boldsymbol{\tau}_e{}^i$ are the external forces and torque acting on body i and $\mathbf{F}_c{}^i$, $\boldsymbol{\tau}_c{}^i$ are the constraint forces and torque; $\hat{\mathbf{I}}^i$ and $\hat{\mathbf{J}}^i$ are the inertia and angular momentum dyadic respectively; $\ddot{\mathbf{R}}^i$ and $\ddot{\boldsymbol{\theta}}^i$ are the spatial and angular accelerations of the rigid bodies. The centrifugal term is not present in the torque balance equation because for every rigid body the center of mass (CM) is chosen as the reference point.

The joint constraint are modeled as external forces, expressed by the following relations:

$$\mathbf{F}_c{}^i = \sum_{j=0}^{n} \alpha(\mathbf{p}_j - \mathbf{p}_i) + \beta(\mathbf{v}_j - \mathbf{v}_i) \tag{3.6}$$

where $(\mathbf{p}_j - \mathbf{p}_i)$ is the position of the joint pivot point in body j relative to joint pivot in body i; $(\mathbf{v}_j - \mathbf{v}_i)$ is the relative velocity of the joint pivot points in the two bodies; α and β are positive constants.

3.4 Simulation Algorithm

A general purpose dynamic simulator was developed as part of the present research. It can reproduce the dynamics of fairly general tree-like structures in presence of gravity and contact friction. The software includes several components: a dynamics engine which carries out the actual computation of the forward dynamics; an input/output interface, which reads description files and control parameters. The code is written in the `C` language and uses a purpose developed library for matrix operations.

3.4.1 Dynamics Engine

The forward dynamics is computed by an algorithm based on the penalty method as described above. The algorithm uses the adaptive Euler's method to integrate the equations of motion, starting from the initial conditions specified in a description file. The algorithm works through the following stages for each step of the simulation:

1. Initialize data structures for every object in the simulation
2. Compute positions and velocities of every joint pivot point in each object
3. Evaluate the differential position and velocity between pivot points of adjacent objects (i.e. objects connected through a joint)
4. Compute joint elastic and damping forces (penalties)
5. Detect collision with the ground
6. Add gravity to the total external forces acting on each object
7. Adapt the integration time step to the accelerations present in the system

8. Integrate and update velocities and positions

3.4.2 Collision Detection

Collisions with the ground are computed on the basis of the object geometry and, once a contact is detected, the resulting ground reaction and friction components are computed. Every object is represented through a data structure holding the physical characteristics as well as lists of points used for the 3D rendering. These lists exist for vertices, lines connecting vertices and faces of each object in the simulation. In particular the list of vertices can be used to determine a collision between an object and the ground. Once a collision is detected the elastic reaction of the ground is computed on the basis of the following formula:

$$f_r = -A(z - z_g) - Bv_t \tag{3.7}$$

z is the coordinate of the contact vertex along the normal to the surface; z_g is the coordinate, along the same direction, of the ground surface; v_t is the tangential velocity of the contact vertex of the body; A and B are positive constants. To compute the reaction force at a contact point it is necessary to compute the tangential velocity v_t and the penetration of the contact vertex into the ground $(z - z_g)$. During the simulated experiments on walking the values of A and B have been 100000 Kg/sec^2 and 1000 Kg/sec respectively.

The collision algorithm does not, at present, detect impacts between bodies. Although this feature is desirable, it is not essential to the study of the problem of walking, as there are no objects that can collide, given the mechanical structure of the robot. For the purpose of inter-object collisions, after an impact is detected, the technique adopted for simulating the ground reaction could be used for estimating the contact solicitation. For more details on collision detection see [38, 32, 35].

3.4.3 ODE Integration

The equations of motion of the system are integrated using a simple and fast adaptive-time-step variant of the Euler's method. The forces acting on the bodies being simulated vary largely depending on the situation. Collisions in particular act as impulsive forces that make the ODE stiff. To overcome somewhat the stiffness of the equations the time-step of integration is varied inversely proportionally to the amplitude of the highest force acting on the system. For the walking robot experiments the average time-step settles to $5\ 10^{-5}$ *sec* approximately. At the occurrence of collisions this value reduces to a minimum value of 10^{-6} *sec*, increasing in static cases to the maximum allowed value of $5\ 10^{-4}$ *sec*.

3.5 User Interface

The simulator includes a graphical 3D interface, which allows the user to follow the motion of the system under the controller actions. It also provides a friendly interface, written using a freely available package for graphical user interfaces (GUI) called `XForms` ©. The graphical rendering code is written in the `C` language on top of the `X11` libraries, included with most flavors of `Unix` ©. For the solid rendering a non-specular flat shading model is adopted, and the painter's algorithm is used for polygon sorting [41]. A detailed description of the various components of the software package is reported in the following sections.

3.5.1 IO Formats

The software can be configured through two input files: the description (`.wl`) file, containing the mechanical structure configuration and initial conditions; the control (`.ctl`) file containing the control parameters. It generates, at the end of the simulation, a file containing the parameters of the controller being simulated together with the corresponding computed fitness. The fitness value is evaluated according to the definition of an external function provided by the user. The description file contains definitions for objects, compounds made up of several objects, joints between object-s, general configuration parameters and log statements to save the time evolution of state variables. An example of typical description statements, extracted from a `.wl` file defining a walking robot, are reported below:

```
...
ColorDef        Brown   .7 .8 .5
ColorDef        Blue    .4 .5 .7
ColorDef        Silver  .7 .7 .8

TimeMax         8.0
Simulate        OFF
Display         ON
LogRate         20
Friction        2.5
...
Object BodyBase {
        Type            Cube
        Color           Brown
        Mass            2.5
        Dimensions      .25 .6 .02
}
...
Compound Body {
        BodyBase
        GearBoxPR
        GearBoxPL
        GearBoxAR
        GearBoxAL
        Position        .30 -.56 .322
        Orientation     .0 .0 .0
}
...
Joint HipPR {
        Body            .14 -.22 .03
        LegUPR          .0 .0 .09
        Orientation     .4 .0 .0
        Limits          -1.5 1.5
        ControlX
}
...
```

```
Log      Body      body.log
...
```

The control file for an eight-DOF system such as the walking robot would look something like the following - where the first value is the fitness value and the other numbers are the coefficients of sines and cosines of a Fourier expansion for each DOF:

```
0.012402
0.400000  0.255476  0.032754  -0.030112  0.013921
0.034243  0.039600  0.014926  -0.007822  0.000000
0.000000  0.364647  -0.319100  0.000000  0.000000
0.000000  0.000000  0.000000  0.030025  0.032101
0.000000  0.004799  -0.372179  -0.332543  0.000000
-0.044946  0.003917  0.000000  0.000000  -0.011471
0.000000  0.039063  0.000000  -0.442531  0.300000
0.037914  0.005913  -0.029453  0.012439  0.000000
```

3.5.2 Graphical Interface

In order to make the interaction with the simulation process possible, a 3D graphical interface has been included with the software package. It allows for interactive rotation and movement of the view-point, as well as zooming in and out. The simulated mechanical structure can be rendered in solid (i.e. flat shading) or wire-frame mode. The gravity constant can also be varied continuously between 0 and 9.81 m/sec^2 (the Earth average gravitational constant), the latter being the default value. The simulation can be started and stopped and the action of the controller can be added and removed interactively.

The GUI was designed using the tools for code generation provided with the `XForms` © package. This package allows the quick implementation of graphical user interfaces, through the use of an interactive `WYSIWYG` program. Various widgets can be dragged onto a panel, positioned, dimensioned and labeled. After the design stage is complete `C` code is generated.

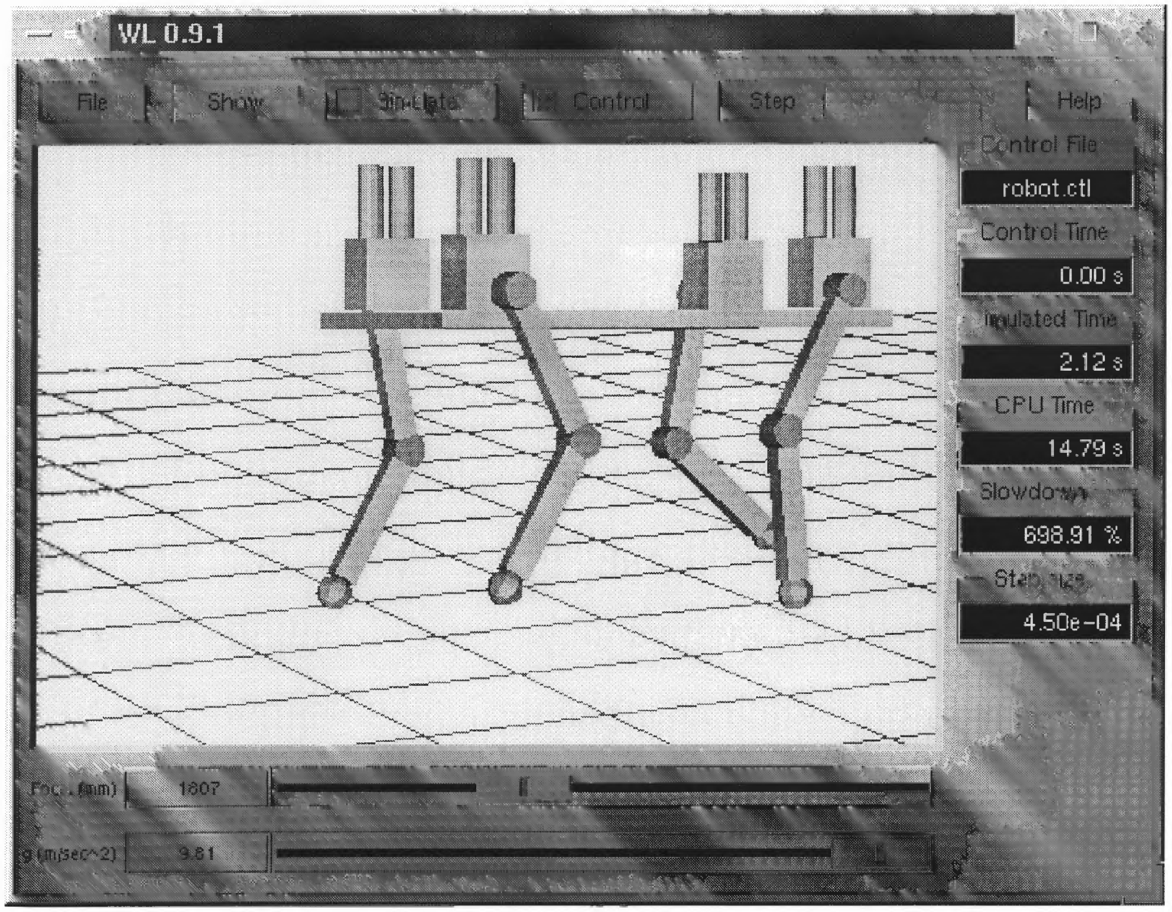


Figure 3.2: A typical 3D projected view of the simulator during an interactive session.

The 3D rendering uses a flat shading model, with a single fixed diffused light source. A pin-hole camera projection model is adopted to obtain the plane projection to be displayed in a window. Polygon sorting is carried out using the painter's algorithm. This algorithm sorts polygons in 3D according to their distance from the projection plane, providing for successive disambiguation of their overlapping. A z-buffer algorithm [41] would provide for more general shape rendering, but in this case would just impose an additional computational burden, since all objects in the simulation are made up of convex regular solids.

3.5.3 Batch Processing

The software can be used in non-interactive or batch mode. To enable this feature the `TextOnly` flag can be defined in the description file. When this flag is activated the simulation is started immediately after execution of the program and carried out until the `TimeMax` value - also defined in the description file - is reached. This mode simulates the system considerably faster than the interactive GUI mode. Also, several instances of the simulator can be run simultaneously through a simple shell script, allowing for the evaluation of several mechanical designs or different control parameters.

The batch mode of the simulator allows the implementation all of the operators of the genetic algorithm through the use of shell scripts. Also the low level of dependence from the underlying computer architecture of the non-graphical mode has allowed the use of more computational resources. The software has been successfully tested on Intel™ , Sparc™ and IBM™ Risc architectures.

3.5.4 Logging

All objects in a simulation can be tracked individually and their position and orientation logged into text files. This feature can be enabled through the simulation configuration file (`.wl`) with a `LOG` statement. The `LogRate` can also be specified in the configuration file giving the number of samples per unit of simulated time. This parameter is global and affects the logging of all the objects in the simulation. What follows is a typical example of a log file produced during the simulation of the walking robot.

```
# Log file for compound Body
# walking-robot.wl walking-robot.ctl
#   T        X          Y          Z       theta      phi       psi
0.000016 0.390000 -0.330000 0.394941 0.001000 0.000000 0.000000
0.050149 0.385982 -0.342738 0.402905 0.050781 -2.003869 2.035580
0.100355 0.380585 -0.354432 0.398617 0.075912 -1.775839 1.820709
```

```
0.150594 0.378196 -0.362336 0.397248 0.091686 -1.994487 2.060207
0.200768 0.377189 -0.362352 0.398658 0.069194 -2.034926 2.113570
0.250871 0.381735 -0.351832 0.400693 0.024297 -2.337590 2.412617
...
```

In each line of a log file is reported the simulation time at which the sample was taken, the X, Y and Z coordinated of the object and the three angles defining the orientation of the object. The # sign at the beginning of a line indicates a comment. The first two lines of every log file contain comments indicating the name of the object being logged and the configuration and control files used for the simulation.

CHAPTER 4

OPTIMIZATION AND EVOLUTION

Once the mechanical design of the walking robot is chosen and a parameterization of the control problem is defined through central pattern generators, parametric optimization is needed to identify stable control for the desired behaviors of the system.

Here a survey of parametric optimization methods is presented together with their relevance to the problem of walking. The reader is guided through the choices made during the design phase in identifying genetic algorithms as best suited for the problem at hand.

As stated in Chapter 2 optimization can be adopted as a learning method for motor control. A search in parameter space corresponds to the exploration of different solutions to the walking problem, obtained varying a set of parameters.

4.1 Review of Optimization Strategies

In the framework of the parameterization chosen for the problem of quadruped locomotion, an optimization strategy is needed to find a good set of parameters which produces stable walking. Many techniques have been proposed and widely used in the past for parametric optimization. In the following sections a brief review of these methods is presented together with some sample applications. The advantages and disadvantages of different strategies are presented to illustrate the choices made for the optimization of the control parameters in the development of quadruped walking.

4.1.1 Hill-Climbing

This technique is based on the idea that the optimization of a certain function against a set of parameters is analogous to a point moving on a hyper-surface looking for a hill-top or a valley. To illustrate this idea consider the problem of finding the best fit of a calorimetric spectrum resulting from chemical reaction experiment. We can assume that every compound reacting in the process releases heat with a Gaussian distribution in temperature. In this assumption the spectrum can be modeled as a sum of Gaussian functions with different shape parameters. In the case of n compounds involved in the process the theoretical model of the reaction heat restitution can be expressed by the following relation:

$$H_t(T) = \sum_{i=0}^{n} a_i e^{-b_i(T-T_i)^2} \tag{4.1}$$

Given this model, the negative of the square fitting error, which constitutes the optimization function, would be given by the following formula:

$$E(a_1, b_1, T_1, ..., a_n, b_n, T_n) = -\sum_{k=0}^{N}[H_t(T_k) - H_e(T_k)]^2 \tag{4.2}$$

where N is the number of data points taken during the experiment; $H_t(T_k)$ and $H_e(T_k)$ are the theoretical and experimental values of the heat restitution function at the temperature of the k^{th} experimental point.

At this point one could try to find the global maximum of this error function analytically. Unfortunately this problem has no analytic solution. Though it is possible to define the components of the gradient of (4.2) as follows:

$$\begin{aligned}
\nabla_{a_i}E(a_1, b_1, T_1, ..., a_n, b_n, T_n) &= -2\textstyle\sum_{k=0}^{N}[H_t(T_k) - H_e(T_k)]\frac{\partial}{\partial a_i}H_t(T_k) \\
\nabla_{b_i}E(a_1, b_1, T_1, ..., a_n, b_n, T_n) &= -2\textstyle\sum_{k=0}^{N}[H_t(T_k) - H_e(T_k)]\frac{\partial}{\partial b_i}H_t(T_k) \\
\nabla_{T_i}E(a_1, b_1, T_1, ..., a_n, b_n, T_n) &= -2\textstyle\sum_{k=0}^{N}[H_t(T_k) - H_e(T_k)]\frac{\partial}{\partial T_i}H_t(T_k)
\end{aligned} \tag{4.3}$$

Starting with a random value of the parameters $P^r(a_1^r, b_1^r, T_1^r, ..., a_n^r, b_n^r, T_n^r)$ the error function can be climbed moving P in the direction expressed by (4.3). This process can be thought of as an iterative algorithm which updates the value of the generic point P in the parameter space according to the following relation:

$$\mathbf{P_{t+1}} = \mathbf{P_t} + \alpha \nabla E \tag{4.4}$$

where α is a positive constant < 1.

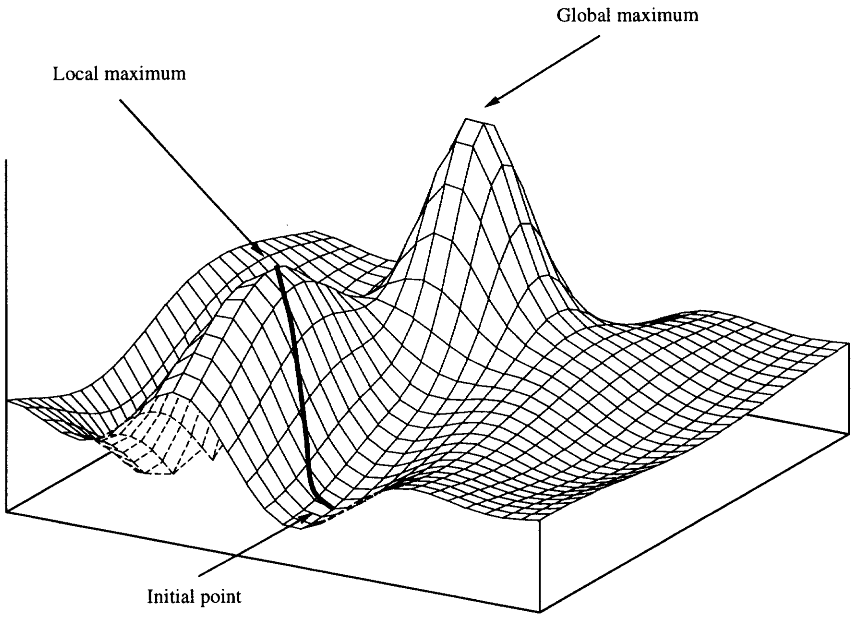


Figure 4.1: Illustration of the hill-climbing algorithm. The parameters are varied according to the direction of maximum slope of the optimization surface.

A generic error function for a two parameter problem can be visualized in a 3D plot and could look something like Figure 4.1. The thick line in this figure represents the direction of the gradient at the generic point P on the surface. If the point is moved in the positive direction of the gradient, expressed by the + sign in (4.4), it climbs the error surface. P follows the positive gradient on the thick solid path shown in Figure 4.1, until it reaches the top of a hill on the surface, corresponding to a zero of the gradient.

One major limitation of this algorithm is evident from Figure 4.1 as the maximum found by the moving point is a local one. This limitation is intrinsic to this method, it can be reduced but not eliminated. Some variants of the technique exist which perform several independent searches starting from different initial points chosen randomly. This provides for a better exploration of the parameter space and generally increases the chance of finding a global maximum.

In the case in which the functional dependence of the optimization function from its parameters is not explicitly known a second limitation is imposed by the need of the numerical computation of the derivatives. This limitation becomes more evident for the problem concerning this thesis, where the estimation of the values of the optimization function is computationally expensive. In fact the computation of each value of this function implies the full simulation of the system, through the dynamical simulator, for a given period of time. A step of the hill-climbing algorithm, therefore, corresponds to the computation of one value of the function for each parameter, in oder to obtain the partial derivatives.

4.1.2 Conjugate Gradient

A variation of the above algorithm allows to save computation in some cases, reducing the number of derivatives to be calculated to one, no matter how many parameters the optimization process depends on. The idea is to follow the largest component of the gradient vector until it becomes zero (i.e. at a saddle). Then the next largest component is followed, until all the gradient is null. The drawback is that usually the path to a maximum becomes longer than the one found by a hill-climbing algorithm. This is evident in Figure 4.2 which shows the same optimization function described in the previous section climbed by the conjugate gradient algorithm.

This algorithm, though generally faster then its original counterpart, suffers from the same limitations in terms of the locality of its solutions as well as the

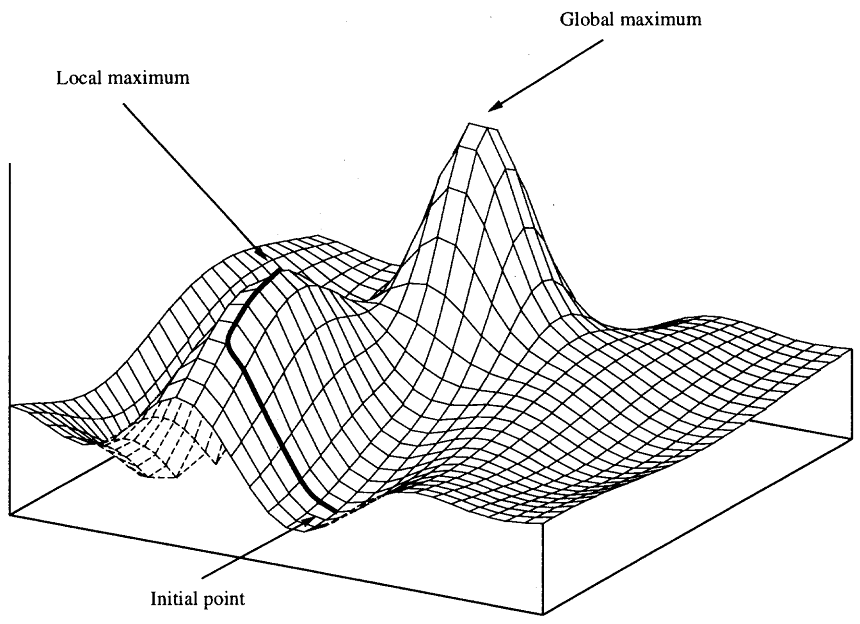


Figure 4.2: Illustration of the conjugate-gradient algorithm. The path on the optimization surface is usually longer than the one defined by hill-climbing.

numerical computation of the derivatives, even thought only one partial derivative is needed at each step.

4.1.3 Thermal Relaxation (Boltzmann Machines)

Another global optimization method is modeled after the natural collapse of physical system of particles to a minimal energy state when the temperature is lowered to the absolute zero. This strategy uses a population of states - each one of which corresponds to a given set of values of the parameters of the system. To this population a total energy level is assigned which allows each state to move over the optimization function, i.e. the greater the energy the more mobile the states are.

When the total energy is lowered - this corresponds to lowering the temperature of a physical system of particles - the mobility of the points over the surface becomes impaired as they become trapped in valleys of the optimization function. The probability of moving a point to a new location on the surface is proportional to the Boltzmann coefficient $e^{-\frac{\Delta E}{KT}}$, where ΔE is the difference in the error values of the two locations and K and T are the Boltzmann constant and the temperature of the system, respectively. In the case in which the population is sufficiently large and the temperature (energy) is lowered very slowly (quasi-statically) it is possible to mathematically prove that some of the solutions end up in the global minimum of the negative of the optimization function, corresponding to the global optimal value of the parameters.

The process is illustrated in Figure 4.3 where four steps of the algorithm are graphically shown, the population of possible solutions can only move in the portion of the surface visible in the 3D plots. When the temperature is lowered the population is forced into smaller and smaller regions, which eventually collapse into the global minimum.

This method is usually very effective for systems where the computational overhead connected to the evaluation of the states of the system is low, as a very large number of states has to be considered. One area in which this method has grown popular is the optimization of pathways in electronic circuits. This because the extreme computational requirements are justified by the production of millions of identical pieces in a factory. Still in some cases its use is infeasible, due to the short-time development constraints of modern electronic manufacturing, which would not allow months of computational optimization and forces the use of "any" non-fully optimized working solution.

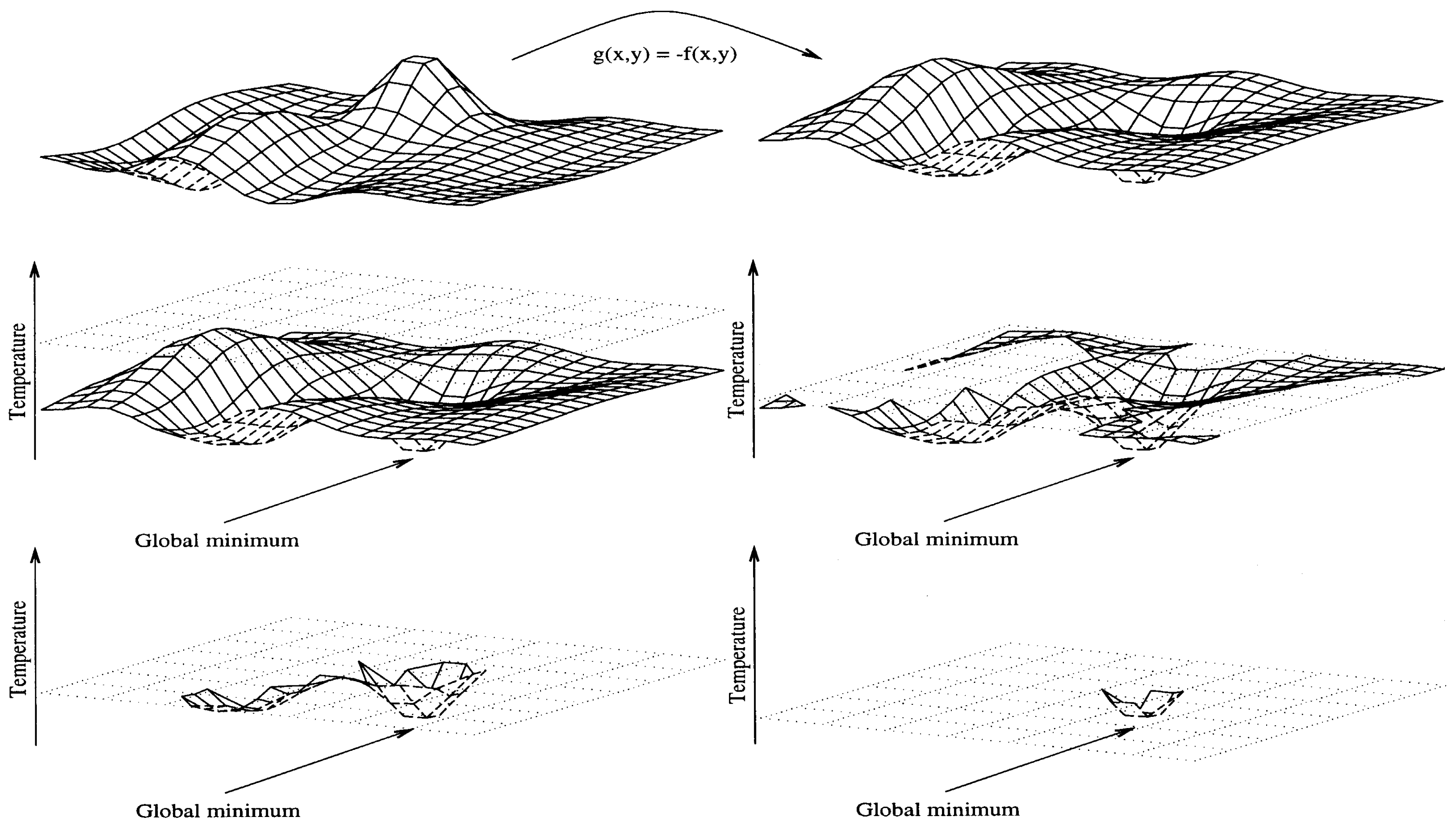


Figure 4.3: Illustration of the thermal relaxation algorithm. The algorithm finds minima of the optimization function, which has to be reversed before the analysis. The energy level is lowered giving access to portions of the surface defined by the optimization function, until only the global minimum is accessible.

4.1.4 Dynamic Programming

This technique is a very powerful method to compute optimal solutions to complex problems. It works by combining the results computed for sets of sub-problems. In order to apply this method it is necessary to demonstrate that optimality of solutions is scalable (it carries from sub-solution to the full solution). It also needs a strategy for choosing an optimal combination of sub-solutions out of all the possible ones [25, 24, 42].

The method, although extremely powerful for a number of applications, shows its weakness when the system cannot be described entirely. In general a complete knowledge of all the aspects of a problem is needed. In the case of walking control the optimality of a solution cannot be defined mathematically in a straight forward way. Also the relationship of scalable optimality between sub-solutions and the full solution is hard to determine, due to the fact that the performance of the system is computed by the simulation software.

A complete description of the mathematics governing the functional dependency of the fitness function from the parameters of the system, could be determined through a predictive model of the forward dynamics of the robot. This approach would be applicable, in spite of the complexity of the mathematics behind it, if the system under study here would not show high sensitivity to initial conditions. This characteristic renders the necessary approximations of a predictive model chain totally unreliable and solutions considered optimal for the predictive model could result in an highly unstable behavior when applied.

4.1.5 Genetic Algorithms

A relatively new class of optimization techniques has been inspired by the theory of evolution by natural selection or survival of the fittest. Living beings are the result of a long process of evolution. Individual creatures are evaluated by the environment they live in and the more suited they are to this environment the more likely they are to reproduce and carry their features to offsprings. The characteristics of an individual are completely described in its DNA or genome, which is simply a long string of molecules used similarly to a tape on which all the features of a living being are recorded. When reproduction occurs some of the features from two individuals are extracted from their genomes and combined together into a new DNA.

Genetic algorithms try to mimic the evolution process in a simulated environment, where the constraints to survival are imposed artificially. A simulated evolution of an artificial system is then possible when a measure of its "fitness" to a particular task can be defined. The choice of the fitness function, used to evaluate the system, depends on the aim of the optimization required. For example a routing algorithm used to plan the paths of airplanes for intercontinental flights could be optimized against time of flight, fuel consumption, passenger comfort, or a combination of these parameters. A genetic algorithm could be used for this optimization task, choosing a fitness which takes into account the criteria defined above.

The first step towards the implementation of a genetic algorithm is the definition of a genome. In other words it is necessary to define a way to code the parameters in a single object, which usually consists of a string of contiguous numerical values.

Subsequently the definition of a fitness measure is required, which in general is a function that does not explicitly depend on the parameters of the system, but rather increases in value with the ability of a particular system to carry out the required task.

The algorithm proceeds from an initial set of individuals, that is a set of genome strings, containing the parameters of several instances of a system. This initial set is chosen randomly from a predefined range of each parameter. This population of individuals is then evaluated using the chosen fitness measure. Once the evaluation is complete some individuals out of the entire population are chosen for reproduction.

The selection criterion is usually tailored to give a better chance of reproduction to the fittest individuals of the population, trying at the same time to preserve some genetic diversity. The preservation of diversity is of paramount importance to avoid the locality of solutions, which affects the generality of some of the methods described in previous sections.

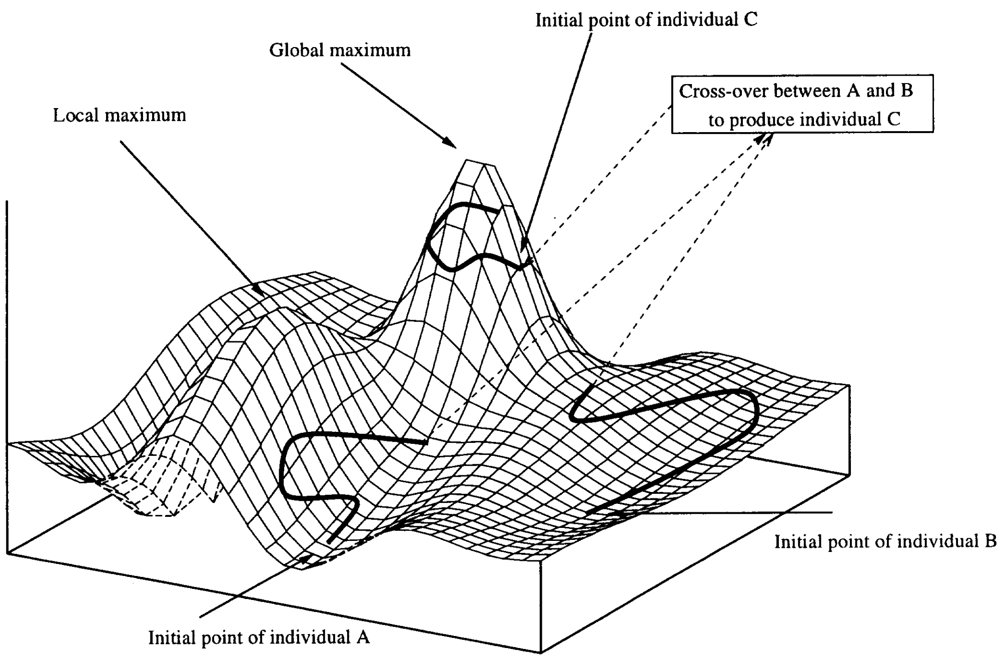

Initial point of individual C
Global maximum
Cross-over between A and B to produce individual C
Local maximum
Initial point of individual B
Initial point of individual A

Figure 4.4: Illustration of the genetic algorithm. Several paths proceed on the optimization surface through mutation and cross-over. Occasionally the cross-over between two independent search paths produces a new individual, which outperforms both of the parents.

Genetic search strategies are similar to the hill-climbing methods, as they tend to move up over a fitness surface in parameter space. One of the advantages of these techniques is that of searching, in parallel, several portions of the space. More importantly they provide a way for combining partial solutions to the problem through cross-over of distinct individuals (see Figure 4.4). This feature is totally missing in the gradient-based searching methods explained above and infeasible in the case of dynamic programming.

The choice of a genetic algorithm for the optimization of quadruped walking has been driven by the advantages explained above. It allows to perform a wide search over a parameter space minimizing the chance of finding local minima and avoiding the definition of partial derivatives of a fitness function with respect to the parameters of the system. It also permits to combine partial solutions through cross-over, similarly to dynamic programming, but in the absence of a detailed knowledge of the mathematical dependence of the optimization function from the parameters.

4.2 Applications of GAs

Genetic algorithms have been employed in a variety of applications, where they have proved suitable for many parameter search tasks. In particular for those cases in which the dimensionality of the search domain is high and fitness maxima are localized in regions far apart from each other.

Some examples of optimization problems solved using GAs are: printed circuit wiring length optimization; routing of vehicles for public transportation; coverage maximization of fire department over urban areas; parametric optimization of neural network design; neuro-genetic learning; approximation of nonlinear functions; fitting of experimental data. In all of these applications GAs have performed comparably or better than other parametric search strategies.

4.3 Implementation

The genetic algorithm used for the optimization of CPGs is based on two genetic operations and a stochastic selection criterion. The optimization process is carried out on simulated walkers, through the use of the dynamic simulation software described in Chapter 2. Initially a generation of N individual controllers is produced from a single set of parameters. The single initial controller consists of a set of individual 0.5 Hz sinusoidal oscillation for each of the joints. Different control files (describing the oscillatory pattern of CPGs) differ by a random amount for each control parameter.

All the individual control files are evaluated using the simulator in batch mode and results of the corresponding fitness values are stored in files. The fitness values computation is described in the next section. After the evaluation process has completed for all of a generation individuals a reproduction operation takes place. Controllers are individually duplicated with a probability proportional to the value of their fitness. This process ensures that high performing controllers will propagate to to the next generation with higher probability, though worse performing controllers can still be selected, which allows different lines of evolution.

Once the reproduction operator has produced at least N new individuals, mutation takes place. Mutation operates on individual parameters stored in the control files. The probability of mutation if fixed for all controllers in a single generation, but can be varied as the evolution process progresses.

A different reproduction strategy has also been implemented which mimics DNA cross-over. It can be applied as an alternative reproduction scheme before mutation takes place and consists of the creation of a new individual, whose parameters are the composition of the parameters of two parent controllers.

4.3.1 Fitness Function

The fitness function chosen for a particular genetic algorithm has a crucial effect on the performance of the optimization process. In the case of walking a reasonable choice of fitness function is one which maximizes the speed of locomotion, while keeping the system stable. For this reason the evaluation function was constructed as follows. The function is linearly increasing with the average speed of locomotion in a chosen direction in the simulated environment, and it depends on the square of the cosine of the average orientation of the robot's body relative to the vertical direction. This function can be expressed as follows:

$$f(v_y, \theta) = \frac{v_y \ \delta \ cos(\theta)^2}{(\delta - 1) \ cos(\theta)^2 + 1} \tag{4.5}$$

where v_y is the average velocity in the axial direction of the robot's body; θ is the angle between the normal to the robot's body and the vertical direction; δ is a positive constant. According to this formula, defining the instantaneous fitness, the average value of the fitness, expressed by (4.6), is always less than or equal to the average locomotion speed of the robot.

$$\langle f \rangle = \frac{1}{T} \int_T f(v_y, \theta, t) \ dt \tag{4.6}$$

The cosine dependence from θ has the advantage that, for small angles, the orientation of the robot's body weights very little on the fitness, being the cosine very close to 1 in a range of ± 10 *deg* from the zero of its argument. This allows a robot which rolls slightly when it walks to be considered, in terms of its fitness, almost as stable as one which maintains a fixed body orientation throughout the motion. Without a non-linear dependence from the orientation, the stability part of the fitness function would have too little or too much effect on the overall final fitness value. This would produce highly unstable walkers, or very slow moving ones. The width of the angular interval allowed in the fitness function can be tuned with the parameter δ: the larger δ the wider the angular range over which the robot is considered flat.

The influence that the parameter δ has over the fitness function is visible in Figure 4.5, where plots are shown for values of $\delta = 0.5, 2, 8, 64$. Through the definition of fitness in (4.5) and (4.6) the maximum allowed value of the average fitness, used as

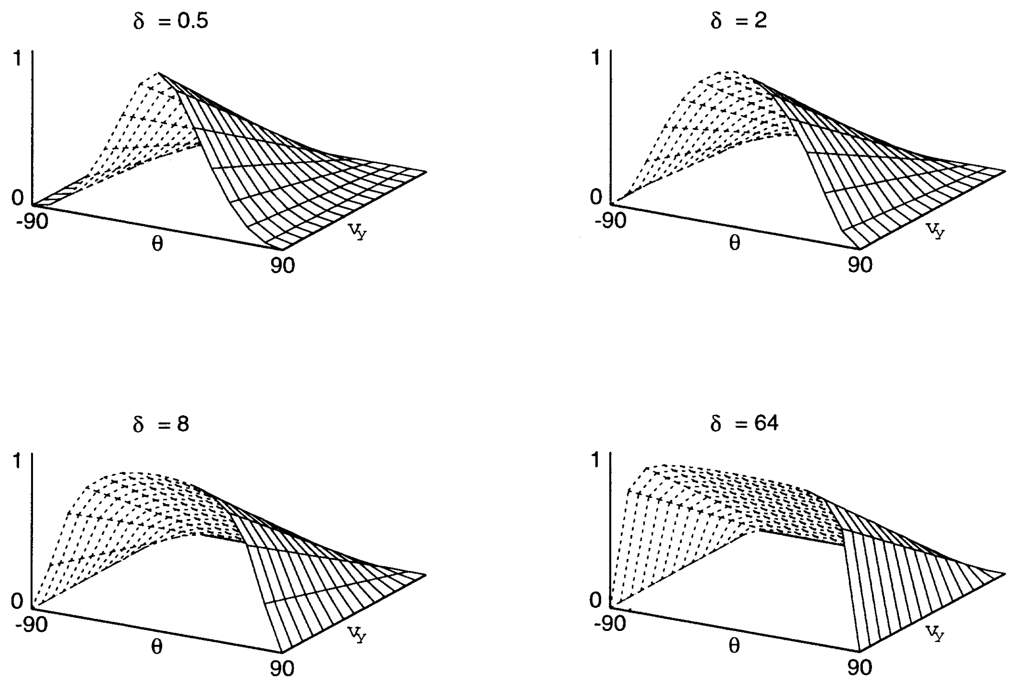


Figure 4.5: Illustration of the effect that the parameter δ has on the fitness function. Note how increasing the value of δ the angular range where the fitness function is little affected by the orientation (θ) widens.

optimization overall measure, is in all cases equal to the average velocity, from which the fitness depends linearly.

4.3.2 Stochastic Selection

The selection of individuals from a population of controllers deals with a twofold constraint. Firstly it has to ensure that well performing controllers are carried to the next generations; secondly that part of the "unfit" genetic material is preserved to guarantee diversity in the population. Using a purposely devised stochastic criterion both of these requirements can be satisfied. The selection of individuals for the offspring of a generation is carried out as follows: 1) the fitness value is read in from the files holding the control parameters together with the result of the simulated evaluation; 2) a random number is generated in the range [0 : 1] from a uniform distribution; 3) if the random value is below the value of the fitness the individual is duplicated. In actuality the process does not deal with absolute probability, but rather with relative ones, as the upper bound of the fitness values is in the cases of interest less than 1. This does not constitute a limitation of the algorithm as the number of duplication attempts is repeated until at least the minimum number (N) of new controllers is created - in the cases analyzed here this minimum number was set to at least 10 for all simulation experiments.

4.3.3 Mutation

Every control file, containing the CPG parameters of a controller, is parsed by the mutation program. For every one of the parameter values a random number is generated in the range [0 : 1] and if that value is less than a specified threshold the parameter is modified. In this case the threshold value determines the probability of reproduction. This can be understood by the following argument: if the threshold value is 0.1 then - in the case of a uniform random number generator - 10 times out of 100 attempts the mutation is, on average, successful.

The mutation rate is the parameter which tunes the diversity of a population. If the value of the mutation threshold is too high, offsprings are very different from previous generations. This translates to a search that has no directionality in the parameter space. If the mutation rate is too low most of the individual controllers end up, after a few generation, in the same portion of the parameter space, therefore loosing one of the core feature of genetic algorithms: the concurrent exploration of many portions of the parameter ranges. An illustration of the mutation operator is visible in Figure 4.6.

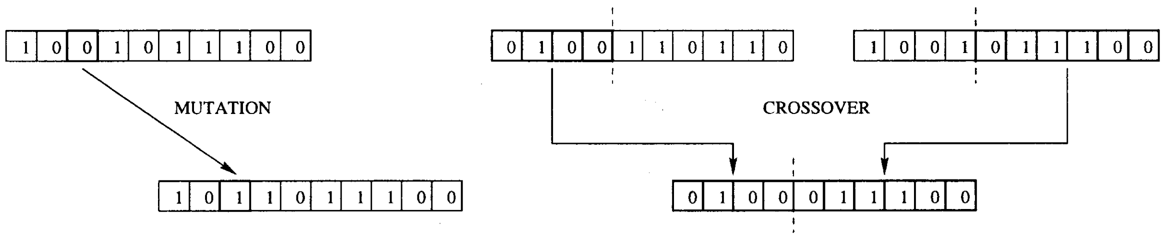


Figure 4.6: Illustration of genetic reproduction operators. Mutation changes the value of one gene in the genome with a specified probability. Crossover takes two complete genomes and generates a new one as a combination of them.

4.3.4 Cross-Over

A sophistication to the stochastic selection is the addition of cross-over between individuals. This operation (depicted in Figure 4.6) is carried out by taking pairs of control files and generating a new set of control parameters with values from both files. The choice of the parents is made using the same stochastic process described earlier, where the probability is now proportional to the sum of the fitness values of the parents. The criterion for choosing parameters from the two parent control files is as follows: 1) an integer number σ is generated in the range $[1 : N_{param}]$, where N_{param} is the total number of control parameters; 2) the first σ values are taken from the first file and the remaining $N_{param} - \sigma$ are taken from the second file. This operation is repeated until N new control files are created. In choosing the control files for merging, the ones used in previous cross-overs are left in the pool from which pairs are taken. The type of cross-over operator implemented here is also called uniform cross-over.

4.3.5 Pocket Algorithm

The genetic diversity maintained by the previous selection methods can lead to an oscillatory behavior of the fitness function as the number of generations increases. This is due to the diversity within populations becoming too broad. To avoid this effect a preselection step can be applied to a population of controllers before the reproduction operator. This process would select a subset of the individuals consisting of a fixed number of the best performing controllers. This preselection method, also known as pocketing, generally improves the speed of convergence of a genetic algorithm. One drawback of the method is the excessive loss of genetic diversity, which may occur when the size of the pocket (the number of preselected individuals) is too small. If the pocket size is 1 both of the selection criteria described above degenerate into the reproduction of the single fittest individual. When this occurs the advantages of the stochastic selection are lost.

CHAPTER 5

WALKING ROBOT

5.1 Robot Design

For the experimental part of this thesis a 4-legged walking robot has been built at University College London. The walker has eight degrees of freedom and is equipped with a variety of sensors. It has been used for the validation of the simulation software described in Chapter 3. The robot is controlled by an external computer and uses purpose built electronics for implementing PID control of the joints.

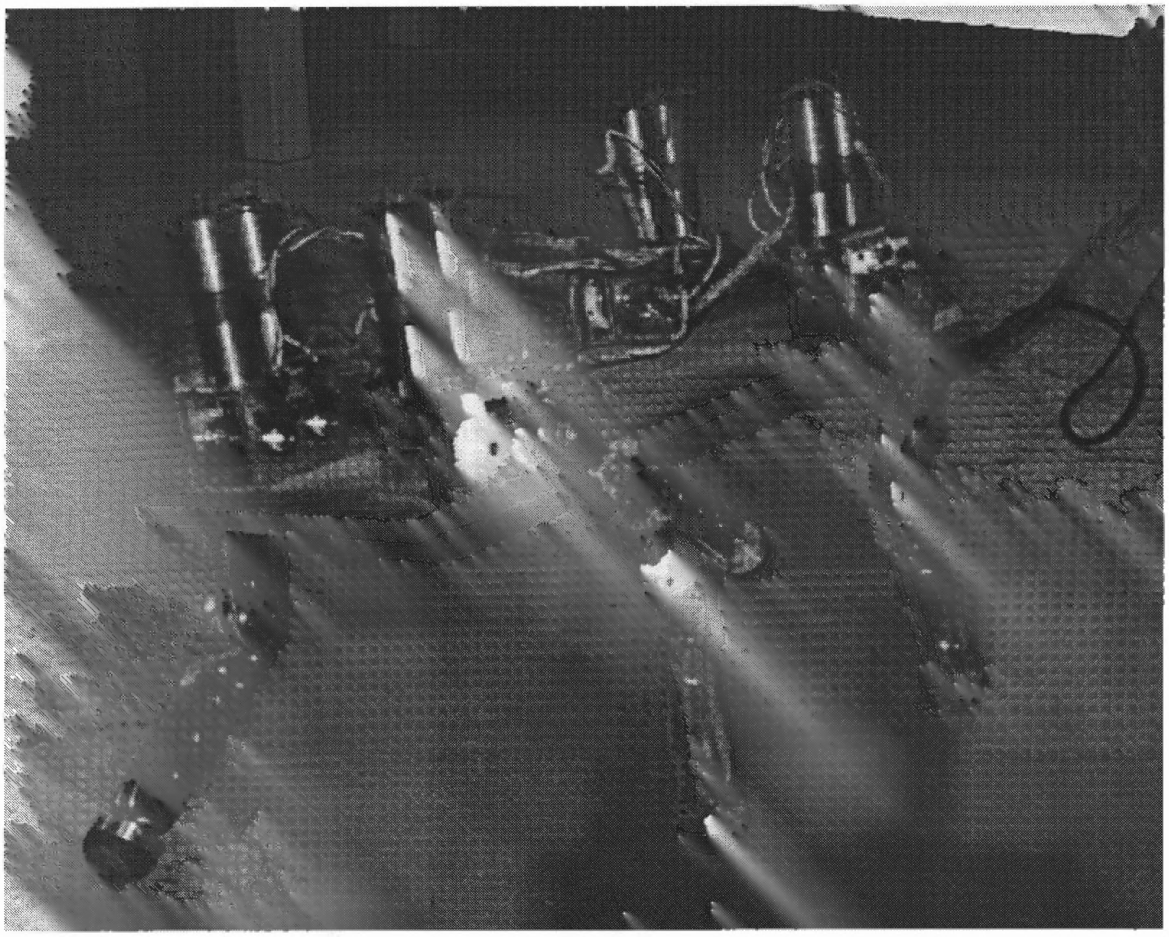


Figure 5.1: Snapshot of the robot during a walking experiment. A pattern optimized by the genetic algorithm was being tested when the photograph was taken.

The robot is connected to the control system and to a power unit through an umbilical cord, which carries current to the motors and sensor readings back to the external computer. In order to supply the high current required by the DC motors used to actuate the joint a purposely built amplifier has been constructed. It amplifies the control currents coming from the PID control chips. The maximum nominal power of the amplification system is 350 W, although the average consumption is much lower.

This chapter describes the mechanical characteristics of the robot and the design choices made during its development.

5.1.1 Leg Design

The walking robot has four identical legs, each with two degrees of freedom at the knee and the hip. The material chosen for the construction of the leg's structure is aluminum to keep the total weight low. The upper and lower section of an individual leg consists of a sandwich of aluminum sheet 3 mm thick with a reinforcement aluminum block in between 10 mm thick, that ensures rigidity. The upper leg carries a steel chain, which transmits torque to the knee joint. The tension of the chain is regulated through an adjustable tensioner gear. In the initial implementation of the leg a rubber belt with steel reinforcements was used instead of the chain, this solution proved to be ineffective due to the high degree of backlash as well as the inadequate resistance to traction. In fact two of the rubber belts snapped during the first week of walking experiments with the robot. The steel chain solution eliminated the problem of resistance to load on the transmission, but did not completely eliminate the backlash problem, which is still noticeable and constitutes the major limitation to the achievable accuracy of control.

A passive joint at the ankle is also included in the design, although it was not used during experiments. The ankle joint has been tested in the preliminary phases of the project. It was intended to act as a passive shock absorber during walking.

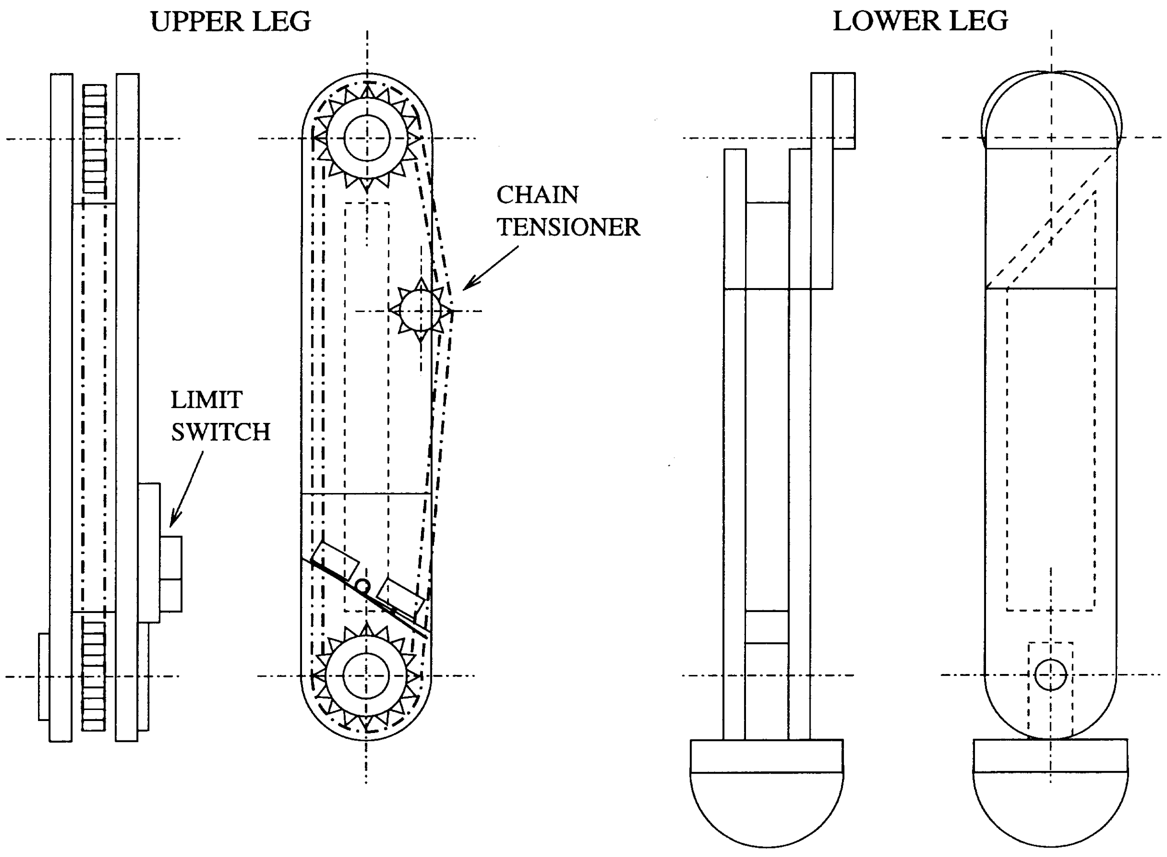


Figure 5.2: Drawing of the mechanical design of an individual leg. In the figure the steel chain transmitting the motion to the knee joint is visible together with the chain tensioner. The ankle pivot joint is shown, though during all experiments it has been kept fixed.

Good friction with the ground is ensured by a rubber hemisphere-shaped tip glued to the bottom of each foot. A schematic drawing of the leg's mechanical structure is visible in Figure 5.2.

The joints at the hip and knee are equipped with limit switches that work in two stages. When the joint approaches the mechanical limit of the joint a signal is sent to the control electronics, if the software fails to stop the motion a second switch cuts off the power to the joint when the actual mechanical limit is reached. This system allows for control actions to be undertaken when a joint approaches the limits of its motion range, but also prevents damage to the leg structure when the control software fails to work properly.

5.1.2 Motors and Transmission

The motors used in the walking robot are DC motors with a voltage range of $\pm 30\ V$ capable of producing a maximum torque of $0.350\ Nm$. Each motor is equipped with a gear-box to provide higher torque to the joint, at low RPM. Torque is transmitted to the legs through right-angle gear couples, visible in Figure 5.3.

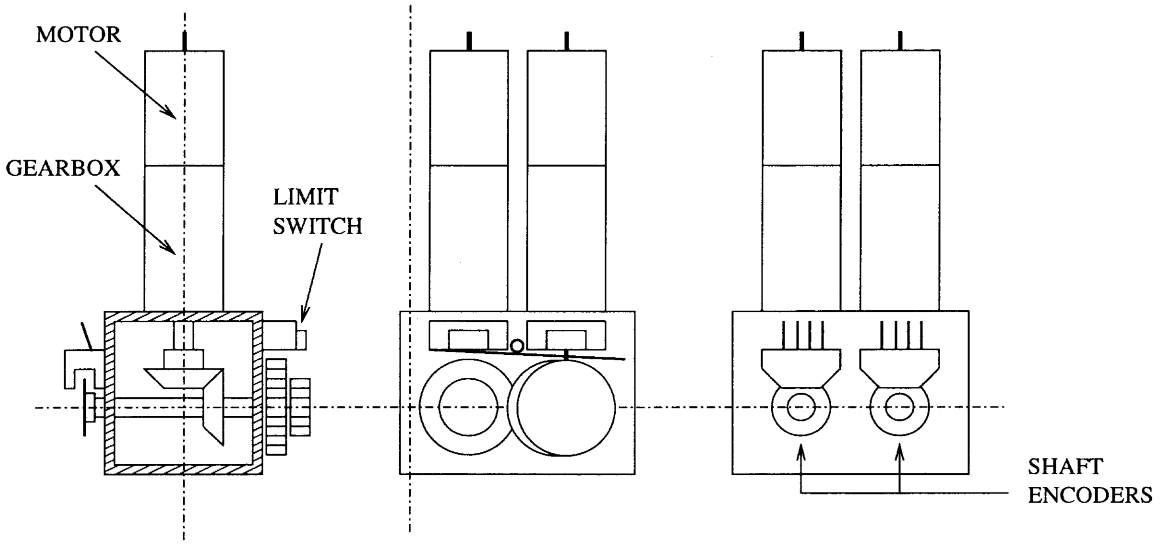


Figure 5.3: Drawing of the mechanical design of a motor unit. In the figure the 90 degree gear couples are shown.

The total weight of one complete motor unit, consisting of two DC motors, two gear-boxes and the 90 degree transmission is $1.9\ Kg$. The robot carries four complete motor units, which together with the body chassis place the total body weight of the walker around $10.5\ Kg$.

5.2 Sensors

In order to obtain information about the posture of the robot several sensors are mounted on the walker. Position feedback for the PID control is provided by optical encoders and posture and position measurements are implemented using a vestibular system and a magnetic range sensor.

5.2.1 Joint Position

The joint position feedback to the PID control chips is provided through optical shaft encoders (see Figure 5.3). Joint position is computed by counting the number of impulses coming from the optical sensor into the PID chips. An absolute zero pulse is used to set the reference for angular position and corresponds to the leg section in the vertical position. The number of impulses corresponding to a full 360 *deg* revolution is 500, giving an absolute resolution of 0.72 *deg*. In actuality the accuracy of measurement is limited by the amount of backlash present in the transmission chain. The hip joint angular position measurement precision is 3 *deg* and the precision for the knee joint is 5.5 *deg*. These values have been experimentally measured, determining the maximum angle for which no optical encoder pulses are generated at different angular positions for each joint. They represent an average estimate as different joints have slightly different amounts of backlash. The higher value of imprecision in the knee joints angular position measurements is due to the additional amount of backlash introduced by the chain transmission.

5.2.2 Vestibular System

To estimate the orientation of the robot it is necessary to measure the roll and pitch of its body. A sensory system which provides this kind of information is sometimes called a vestibular system. A dual pendulum structure has been attached on the body of the robot to provide information about the angular tilt along the longitudinal and transversal axis of the robot's body. The angular positions of the pendulums correspond to the angles between the direction perpendicular to the body and the direction of the total acceleration, which is mainly gravitational. More precisely each pendulum measures the angle between the gravitational acceleration and the projection of the body's normal onto a vertical plane. The projection planes being oriented along the sides of the rectangular body of the walking robot.

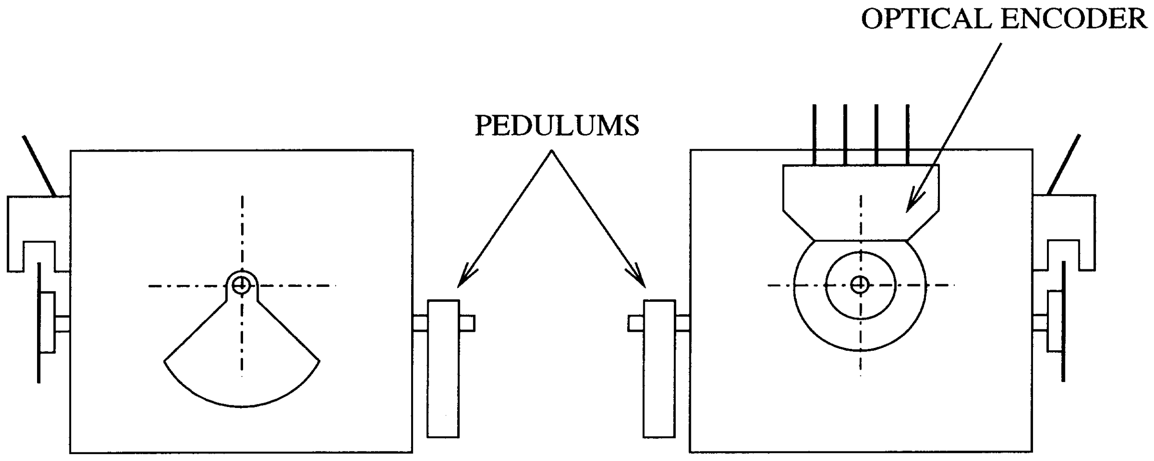


Figure 5.4: Drawing of the mechanical design of the vestibular system. The two pendulums are mounted at a 90 degree angle from one another. Two optical encoders are used to measure roll and pitch angles.

The dual pendulum system showed several deficiencies during preliminary testing. Firstly its static friction was too high - in spite of bearings used to fixate the pendulum axis - which limits greatly the precision of its measurements. Secondly for big angles the values returned by the optical encoders are not decoupled from each other, impairing the validity of the readings. The main limitation of the vestibular sensor lie in its low precision, as most of the time the system has small roll and pitch angles. In addition the inability to provide consistent information for big angles affects the control system's ability to detect dangerous instability.

For the reasons mentioned above a different sensor has been employed for the estimation of roll and pitch, during walking experiments. A range magnetic sensor by Ascension Technology Corporation was employed instead to determine the absolute spatial localization of the robot as well as its orientation. The sensory system consists of a magnetic probe located on the object being tracked, and a receiver unit which triangulates the position of the probe. The receiver has to be within a 1.5 m range of the probe and the angular measurements have to be initialized to a known orientation.

5.3 Interfacing

Several components make up the interface hardware which permits the interaction between the walking robot and the control PC. A custom built add-on board mounted on the computer carries 8 LM628 chips, that implement closed-loop PID control, using as position feedback the pulses generated by the optical encoders. The control currents fed to the power amplifier, used to drive the motors, are produced through 8 DAC integrated circuits, individually connected to each LM628. The digital to analog conversion is necessary as the PID controller chips output binary values through a digital data port.

The power amplifier used to drive the DC motors has been constructed to suite the specifications imposed by the weight and mechanical structure of the robot.

5.3.1 Electronics

PID control is implemented using specialized integrated circuits. Eight LM628 by National Semiconductor are arranged on a prototype PC add-on board. The board interfaces directly with the host PC bus, allowing communication between the control software and the DATA and COMMAND ports on board the LM628. A schematic connection diagram, shown in Figure 5.5, explains how the control chips interact with the sensors and the motors.

The D/A converter is required as the LM628 outputs digital data proportional to the control current. To drive the amplifiers a National Semiconductor DAC0808 8-bit converter chips has been employed.

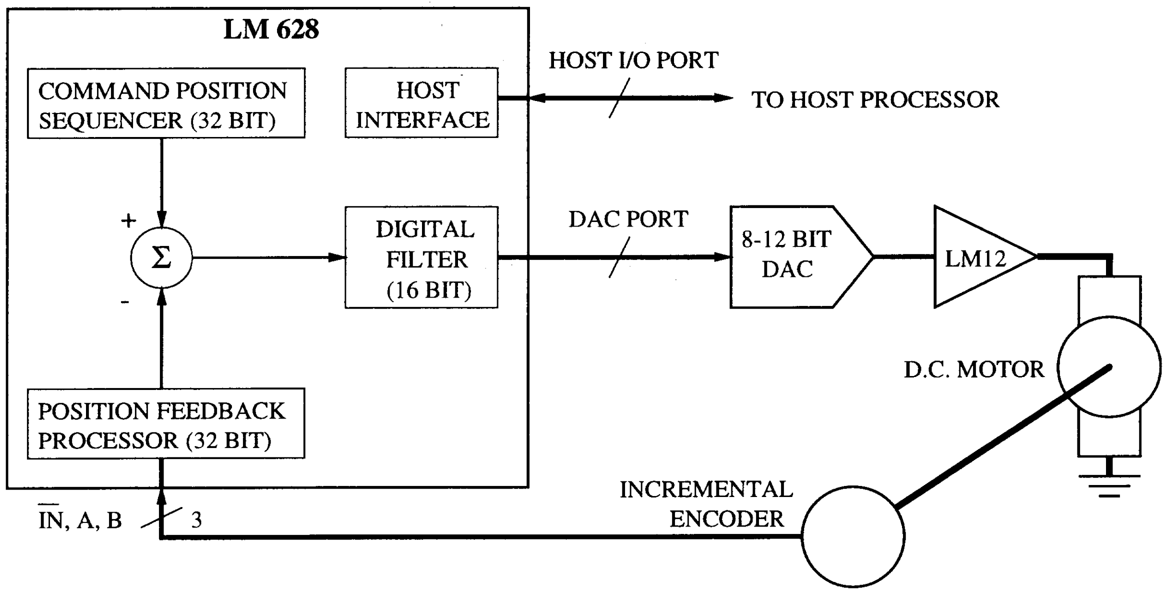


Figure 5.5: This sketch shows how the LM628 controller is connected to the optical encoder and to a linear amplifier, used to drive the DC motors.

5.3.2 Power Amplifier

A high current amplification system is used to adapt the control currents generated by the DACs to the range of currents required to drive the robot's DC motors. In Figure 5.6 a schematic of the connections and electronic components used in one amplification unit is pictured. The complete amplification system consists of an AC adapter and a circuit board holding eight units of the type shown in Figure 5.6.

The motor driving signals coming from the control computer carrying the LM628 PID controllers are connected to the amplifier box through a 25-pin D connector in the back of the amplifier box. On the front of the box there are connectors for the wires bringing the output currents to the motors, passing through the limit switches. The amplifier has a linear (within 1%) transfer function in the operating range of motors.

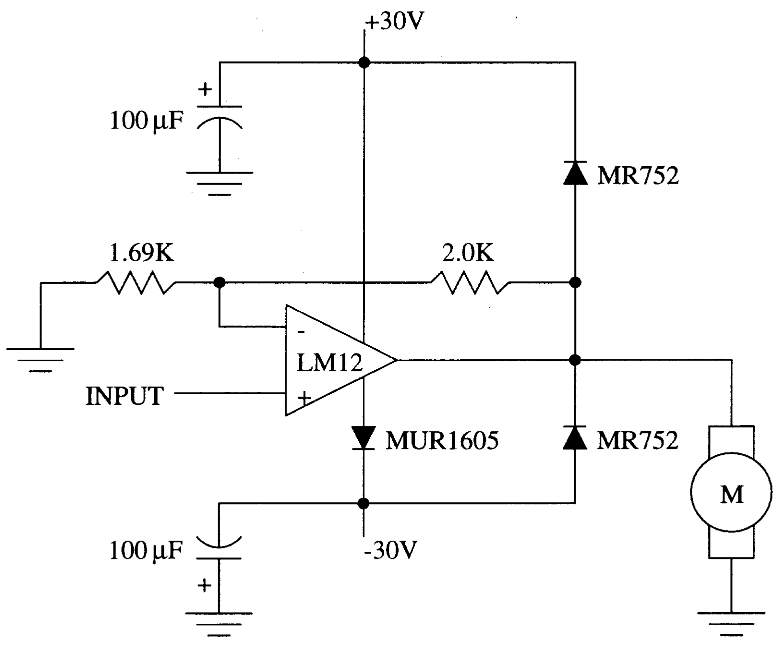


Figure 5.6: The diagram shows the schematic used to build one unit of the power amplifier. The power supply system has 8 of this units, each driving one DC motor.

5.4 Future Developments

In the development of the walking robot a number of problems had to be overcome, sometimes with a satisfactory solution and other times leaving room for improvements. There are a number of changes that could be made to improve the mechanical design of the walking robot. Firstly a better transmission could be designed, which minimizes the amount of backlash in the joints. For example if the motors where mounted horizontally the 90-degree gear couple would be eliminated. Secondly the steel chain transmitting torque to the knee joints could be equipped with a self-adjusting tensioner. Also a better material could be employed for the foot tips, in order to increase the contact friction. Aside these improvements to the current design there are a number of additional features that would greatly broaden the range of applicability of the robot. The following sections suggest some of these improvements, which have been planned for the near future.

5.4.1 Higher Flexibility

One intrinsic limitations of the walker lie in the low flexibility of movements it can produce. Although a broad variety of behaviors can be implemented by changing the control patterns, there are many other interesting types of movements which are worth investigating. In particular the rigidity of the body of the robot does not permit the testing of the body motion correction strategies observed in animals.

5.4.2 Touch

One of the main objectives of this work has been to prove that quasi open-loop control is feasible for the implementation of legged locomotion. In the future view of a system able to interact with its environment some representation of the surrounding features would be necessary. Although the identification of limit cycles permits the control of walking with limited sensory feedback, it does not solve the problem of detecting discontinuities in the supporting surface or changes in the gripping conditions of the terrain. A controller which can access information on the foot-terrain contact conditions could be trained to recognize dangerous changes in the environment. Also, a pattern recognition system could activate a certain gait when a known sensory input condition is detected.

In order for the controller to interact with varying terrain conditions, an artificial touch sensor could be added to the robot. There are several types of sensors which can provide pressure measurements, contact forces and torque. To measure pressure the most sensitive and cost effective type of sensors are the one based on piezo-crystals, which come in different shapes and sizes. To measure force there are resistive as well as capacitance-based sensors. At the time of this writing piezo sensors are under development to be added to each foot of the walking robot.

5.4.3 Sensors for Higher Control

Low level motor control has been the focus of this thesis, therefore higher functions such as motion planning, navigation and pattern recognition are not of direct interest here. Nevertheless the ability to gain information about the environment would allow a more thorough investigation of the applicability of the methods presented here. Once a set of stable behaviors have been isolated, they could be used as the building blocks of a robotic agent, which could react to its environment activating the appropriate pattern of movements, suited for the surrounding situation.

Many sensors are available to provide a control system with information about environmental features, ranging from sonar to infrared to laser to vision systems. One of the planned features for future developments of the walking robot is the addition of sonar and vision sensing to the current set of sensors. The sensory capability the robot currently has could be defined proprioceptive as it gives information about the posture of the robot, but not about the surrounding environment.

CHAPTER 6

RESULTS

A crucial step towards a successful simulation strategy is the validation of the computational model. In Chapter 3 a survey of different dynamic simulation approaches was presented and the motivations of the specific choice of the penalty method, used to implement the dynamic simulator, were expressed. The method is, like any other simulation model, just an approximation of the physical world. To find out to which extend the information produced by the simulation software is usable for the description of an actual robotic mechanism it is necessary to perform validation experiments on actual mechanisms. For this reason several experiments have been carried out on the simulator and on the actual walking robot in order to compare qualitatively and quantitatively the respective behaviors.

6.1 Simulator Validation

A first set of experiments aims to gain information on the PID control mounted on the walking robot. The goal of these tests is to investigate the performance of the low level hardware position control of individual joints. These tests are performed on single joints in the absence of loads, on single joints with loads and on coupled joints moving in opposition of phase.

A second set of validation tests are performed on simple walking behaviors and various characteristics of the movement of the simulated and actual robot are analyzed. Results are shown in the later sections for the body position and orientation as well as the time dependence of individual and coupled joint angles.

For all the experiments a measure of agreement between the simulated and actual data is presented. In addition two criteria to evaluate the stability of motion, during walking, based one on the projection of the center of mass of the body and the other on orientation of the body with respect to the vertical direction, used in the definition of the fitness function (see section 4.3.1).

6.1.1 PID Validation

The behavior of the walking robot is determined by the pattern of movements imposed by CPG defined in the different controllers, but depends also on the characteristic of the low level actuation of joints. In other words the paths in phase space defined by a set of coupled oscillation has to be faithfully followed by the mechanical system. The ability to follow a certain pattern can be impaired by the frequency response of the system and by the backlash in the joints. Both of these factors intervene during normal walking and they have to be characterized separately. The frequency response can be evaluated through experiments where single oscillations, decreasing in period (thereby increasing in frequency), are fed to the controller to determine the point at which amplitude is severely attenuated.

This has to be repeated with and without load on the joints as the response of the system is affected by the inertial characteristics of its mechanical structure. In this section data are presented for single oscillations in different load conditions.

In Figure 6.1 (A) the joint angles of the anterior right hip of the walking robot are plotted against the oscillation phase, for a single sinusoidal pattern at 0.5 Hz, together with the corresponding desired joint angle trajectory (target). The differences between the two plots are due to the fact that the PID control electronics, when coupled with the motors of the robot can lag behind or precede the correct desired position for different reasons: 1) the frequency response of the system; 2) backlash in the transmission. In panels (B) and (C) of the same figure the absolute errors for the two experimental conditions is depicted.

To determine the cut-off frequency at which the PID chain stops responding to the position updates coming in too quickly, the single oscillation experiment described

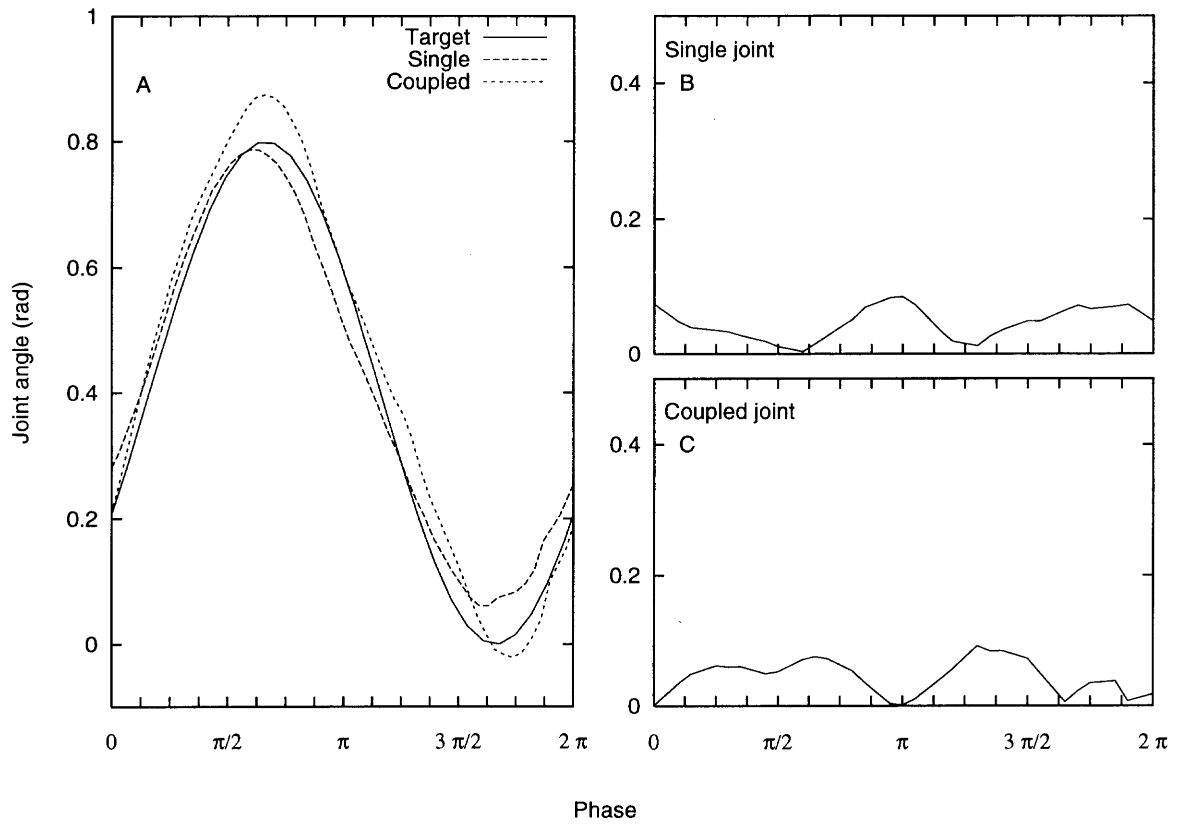


Figure 6.1: Errors in the low level PID controller of a hip joint on the robot. (A) shows one period of a 0.5 Hz drive signal and the resulting set of angular positions of the hip joint in two different conditions. In the first condition (dashed line) the knee joint is not being driven, and in the second condition the knee is being driven out of phase with the hip joint. (B) and (C) show the absolute value of the difference between the target and actual positions of the hip joint in these two conditions.

above has been repeated at different frequencies. The data has been reorganized into a plot showing the average amplitude of the error against the frequency of oscillation. This is visible in Figure 6.2 in which the cut-off frequency can be placed around $0.74Hz$, where the slope of the average absolute error reaches its maximum value. The cut-off value has been obtained fitting the frequency analysis data with the sigmoid function $[1+e^{-a(x-b)}]^{-1}$ and then computing the maximum of its first derivative.This result constitutes a limit of the robot's design, but still allows the implementation of several quadruped gaits, which in animals are observed to have frequencies below the cut-off value [43].

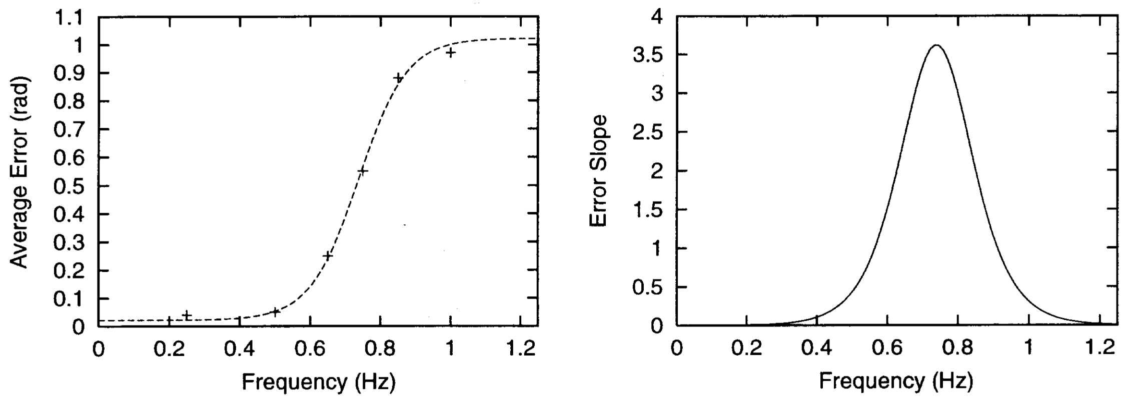


Figure 6.2: Average error over one oscillation as a function of the frequency of the driving pattern. The cut-off frequency can be placed around 0.74 Hz, corresponding to maximum slope of the average absolute error.

6.1.2 Walking Behavior

The overall agreement between the simulation and the walking robot has to be ultimately shown during walking, as this is the behavior of interest. To collect data for the validation of the overall simulation strategy an additional set of sensors has been employed, which is not part of the robot's on board sensors, but was only used during the validation phase. The absolute position and orientation of the walking robot is tracked using a set of range magnetic sensors (Ascension Technology Corporation, Flock of Birds). These sensors are capable of a spatial resolution of 1.7 mm and can acquire data at over 100 Hz.

The walking gait chosen for the walking simulation validation experiment was taken from of a population of controllers during a run of the genetic algorithm with the stochastic mutation operator only. Figure 6.3 shows the joint angle comparison between the simulator and the walking robot during normal walking under the action of the chosen control parameters.

In the two panels values for the both joints in the front right leg are presented, knee (A) and hip (B). The walking pattern corresponds to a fitness value of 0.096121, which resulted in an actual average forward speed of 0.15 m/sec for the simulated walker and 0.19 m/sec for the walking robot. The agreement between the plots of the

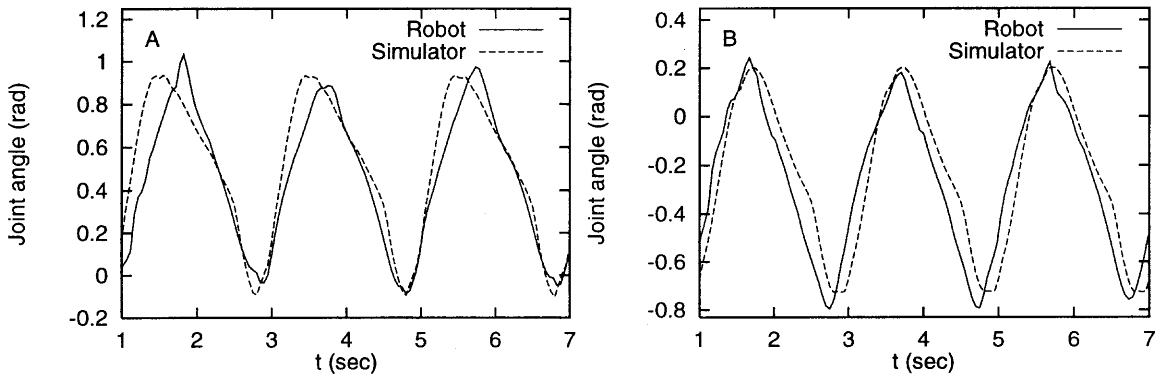


Figure 6.3: Correspondence between the movement of one of the legs during walking in the robot and in the simulation simulation. (A) shows the movement pattern in the simulator and of the robot for the front right knee joint, and (B) shows the same values for the right front hip joint.

joint cycles computed by the simulator and the one measured on the robot is within the error limitations of the PID control described in the previous section. Most of the discrepancy can be attributed to the backlash of the transmission, which is not entirely captured from the simulator. Another source of difference can be found in the friction coefficient which is not uniform on a real surface, whereas it is kept constant in the simulated environment. The friction non uniformity resulted, in this particular experiment, in less overall slippage occurring with the walking robot, which accounts for the slower locomotion speed of the simulated walker.

A comparison of the overall behavior of the robot is clearer in Figure 6.4, where the Y coordinate of the CM of the walker is plotted against time (A). Note the difference in the forward locomotion speed, also due to the non uniformity of the floor friction. In spite of this the movement pattern appears very similar for the actual an simulated robot. This is consistent with the aim of the simulation, which was to capture the overall behavior rather than match exactly the actual motion. More motion structure can be seen in Figure 6.4 (B) where the CM movements are plotted in the $X - Y$ plane. The plot has been scaled to match the average velocity of the walkers. Again the agreement between the movement patterns is evident.

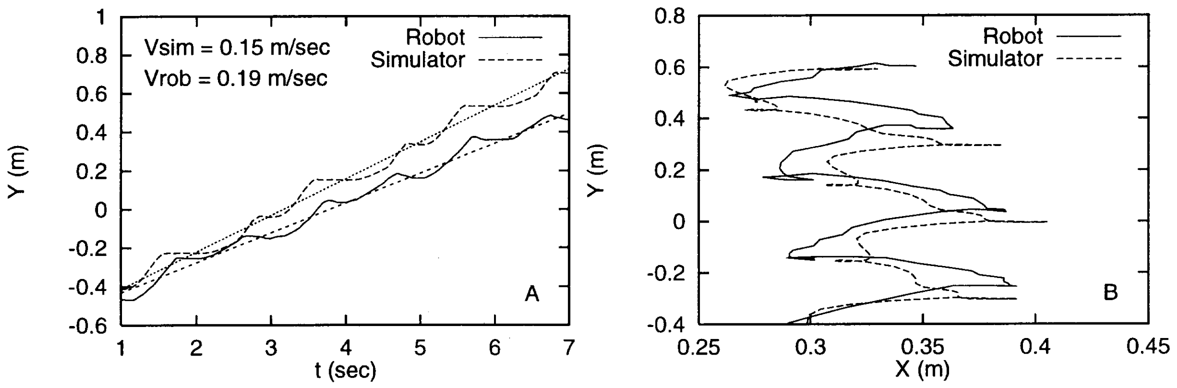


Figure 6.4: Correspondence, between robot and simulation, of the path of the CM of body of the walker during normal locomotion. Panel (A) shows the Y coordinate as a function of time, the linear best fit is used to extrapolate the average locomotion speed. Note that the simulated robot has a slightly lower walking speed, this is due to the non uniformity of the friction between the floor and the feet. Panel (B) shows the $X - Y$ coordinates of the CM of the walker. The plots have been scaled to match locomotion speed.

6.2 Evolution Results

In the following sections results are presented for different versions of the genetic algorithm applied to the walking optimization problem. Starting from a simple version, which includes only the reproduction operator and stochastic selection, features are added one by one and their effect is shown. A description and a quantitative evaluation of influence of every operator on the performance of the parametric optimization is presented.

6.2.1 Optimization Performance

The genetic algorithm (described in Chapter 4) used for learning the oscillation patterns for walking, has several parameters which can be tuned. This section examines the effects on the performance of the optimization resulting from changes in the mutation rates and the strategy chosen for reproduction. Several experiments on mutation rates are presented. Mutation occurrence is initially fixed throughout generations, then varied continuously as a function of the generation.

A comparison between different selection and reproduction criteria is presented, showing how refinements of the GA lead to considerable improvements in the overall performance of the search. The effects of reproduction by duplication, pocketing and cross-over are investigated and a detailed evaluation is carried out.

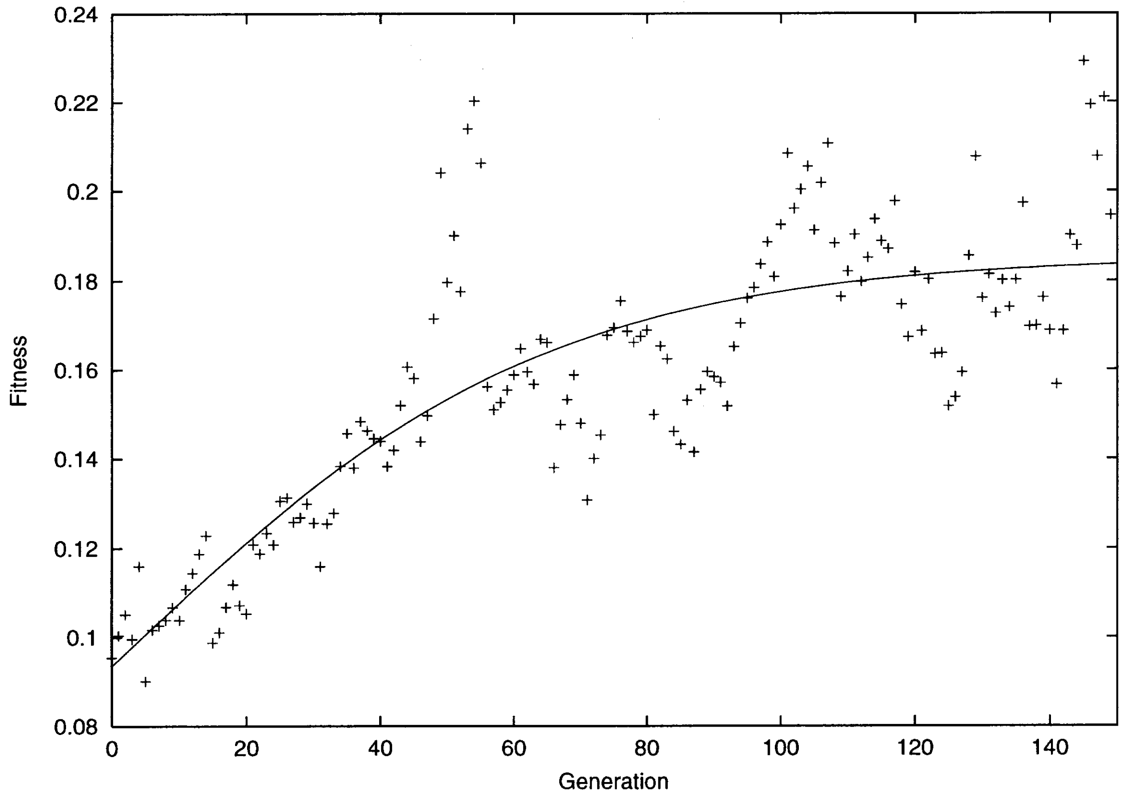


Figure 6.5: Evolution for a population size of 20 individuals and a fixed mutation probability of 20%. Note the amount of oscillation within the data points, when the fitness value approaches saturation.

Data is shown for simulated evolution of at 150 generations, each with population sizes ranging from 10 to 20 individuals.

The simplest genetic algorithm considered uses only mutation with a fixed mutation rate of 20%, that is on average 1 gene is changed out of five within a genome. This simple scheme leads to high genetic diversity but tends to carry what is called, using a biology term, macro mutations to late stages of the evolution. This simply means that within a population many individuals can be very far away from a condition of stability up to the last generation of the evolution process. Also in the absence of cross-over combinations of different genomes are not considered. The average resulting performance gain during evolution is shown, for this first method, in Figure 6.5.

6.2.2 Changing Mutation Rate

To refine the search process as it approaches its performance saturation condition a varying mutation rate can be used. For the implementation of this feature a mutation rate depending on the generation being evaluated was introduced in the algorithm. Initially an exponentially decreasing function was used, but a function inversely proportional to the generation number was found to give better results. This is due to the fact that lowering the mutation rate restricts the diversity of populations and therefore a function dropping too quickly with the generation number produces very similar individuals after few generations. The dependence of the mutation rate from the generation number is expressed by the following formula:

$$\mu = \frac{\mu_{max}}{1 + \gamma GEN} \tag{6.1}$$

where μ is the mutation rate, μ_{max} is the initial mutation rate at generation 0, γ is a positive constant in the range $[0 : 1]$, 0 corresponding to a fixed mutation rate and 1 to the steepest decreasing mutation rate function; GEN is the generation number.

Evolution has been performed using different values of γ to determine the optimal dependence of the fitness from mutation rate. Results of simulation are presented in Figure 6.7 for $\gamma = 0.0, 0.01, 0.1, 1.0$. In the graph it can be noticed how for high values of γ the fitness value saturates early in the evolution process reaching a plateau. This is the effect of the genetic uniformity. As the value of γ decreases the

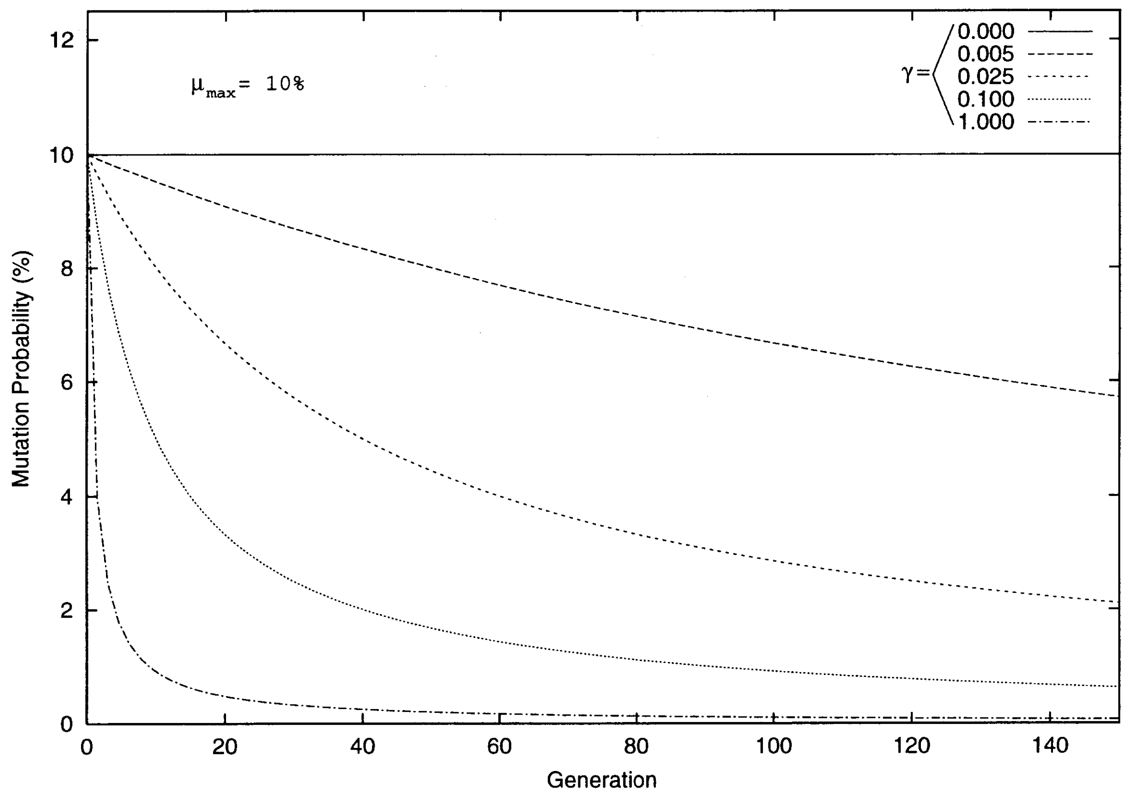


Figure 6.6: Mutation rate decreases with generations during the evolution process according to (6.1). Here plots for a few values of the the parameter γ are shown. In these examples the initial mutation rate μ_{max} has a value of 10%, different values of μ_{max} would scale the plots linearly.

fitness increases monotonically up to higher plateaus, until over a certain value of γ it starts oscillating around a maximum value or decreases.

It has to be pointed out that evolution of the controllers is governed by a stochastic process, therefore for the same mutation rate function one can obtain varying performance results. The effects described in this section are therefore to be considered average effects and in fact several simulations have been performed for each value of γ, in order to establish a significant sample for different values of this parameter.

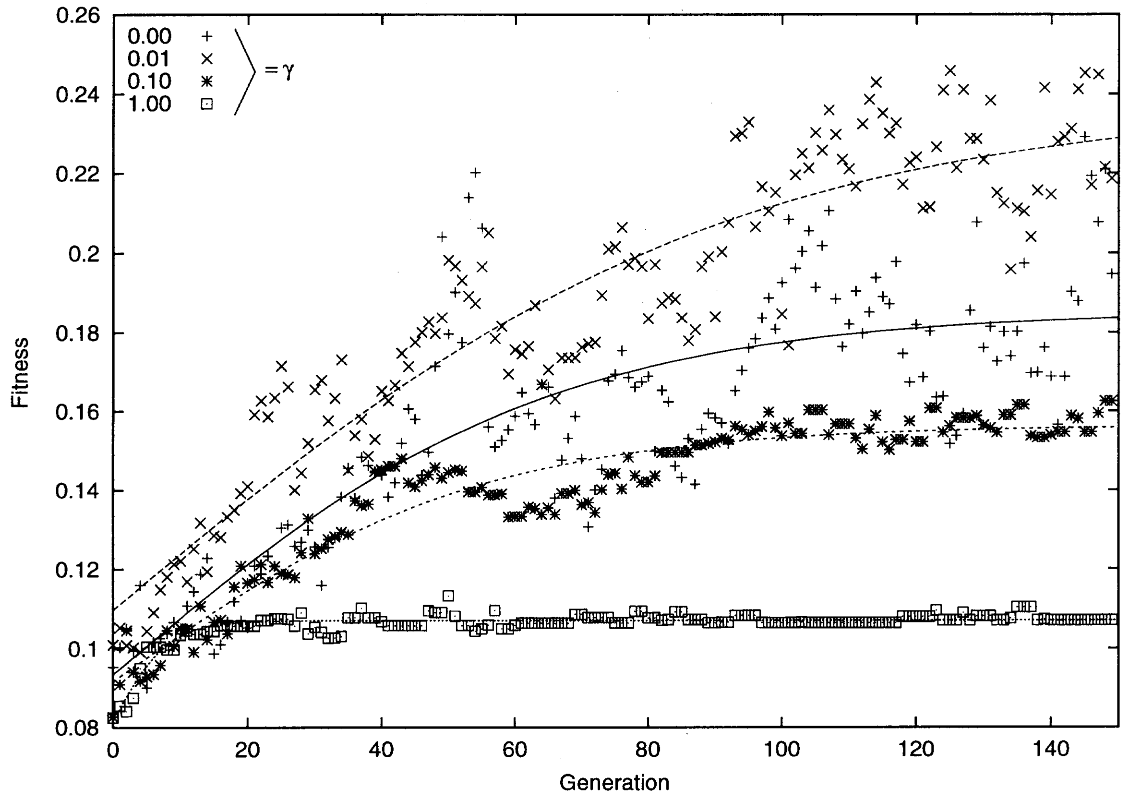


Figure 6.7: Comparison between runs of the GA using different values of γ, in the case of simple reproduction by replication. For all the evolution trials the population size was 20 and the initial mutation was rate $\mu_{max} = 20\%$.

6.2.3 Effect of the Pocket Algorithm

A large genetic diversity in the population of controllers, due to a high value of the mutation rate, can lead to wide oscillations in the fitness values though generations, this effect is visible in Figure 6.7. To contain this effect the pocket algorithm was employed with a pocket size of 4 (out of 20 individuals). The variation in the performance of the GA has been dramatic as can be viewed in Figure 6.8. The plateau level of the fitness has more than doubled and the amount of residual oscillations is much reduced, in relative terms, even in the case of fixed mutation rate.

The clouds of points generated by oscillations in the fitness function for values near the saturation are much less overlapped when pocketing is used. This is due

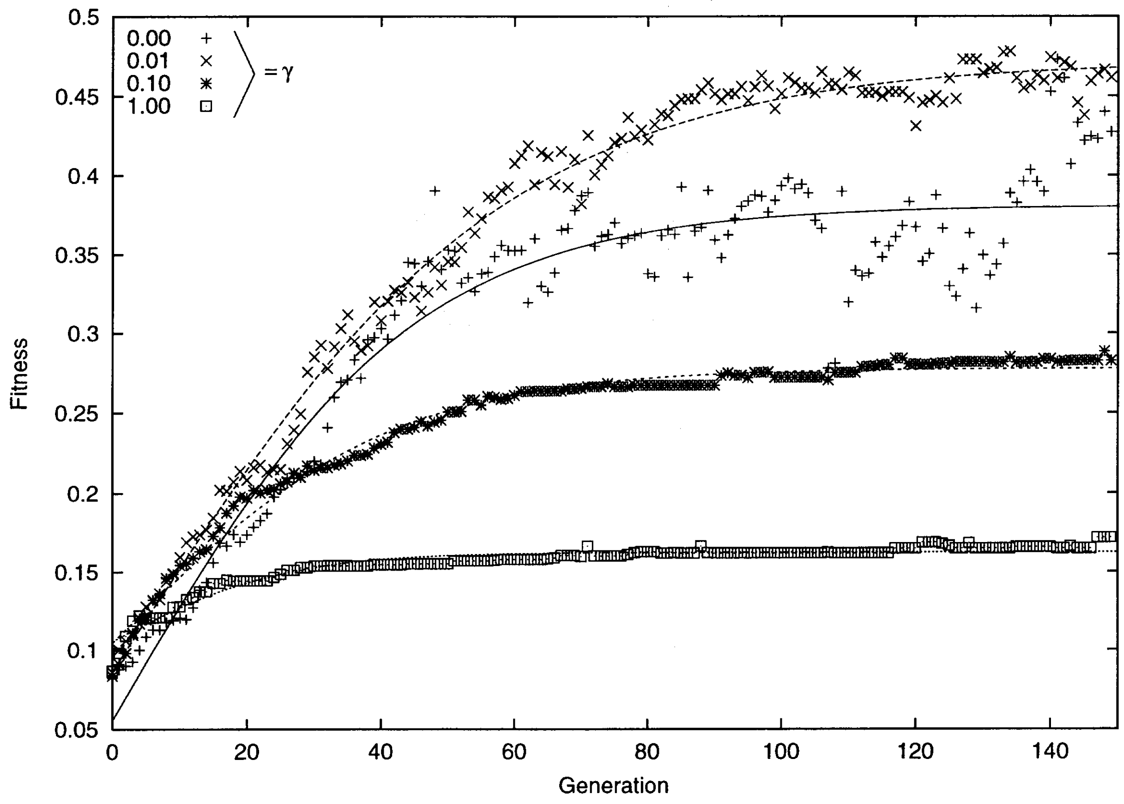


Figure 6.8: Comparison between runs of the GA using different values of γ, in the case of simple reproduction by replication and pocketing. The algorithm parameters are: population size of 20 individuals; initial mutation rate $\mu_{max} = 20\%$ and pocket size of 4.

to the fact that evolution trends are further apart from each other, although the oscillations absolute amplitude remains unchanged.

6.2.4 Adding Cross-over

In the previous sections the effects of varying the mutation rate and forcing selection through the pocket algorithm during evolution have been explored. Now the changes in performance caused by the addition of the cross-over operator are taken into account. The importance in the design of a genetic algorithm of the cross-over operator has already been stated, as it is responsible for combining partial solutions. The operator (described in section 4.4) adopted here uses an asymmetric cut in the genome. Although other criteria can be used to split genomes of controllers for cross-over, such as the random repeated choice of genes performed alternatively from parents, there is little actual evidence of any increase in performance as a result of different cut strategies.

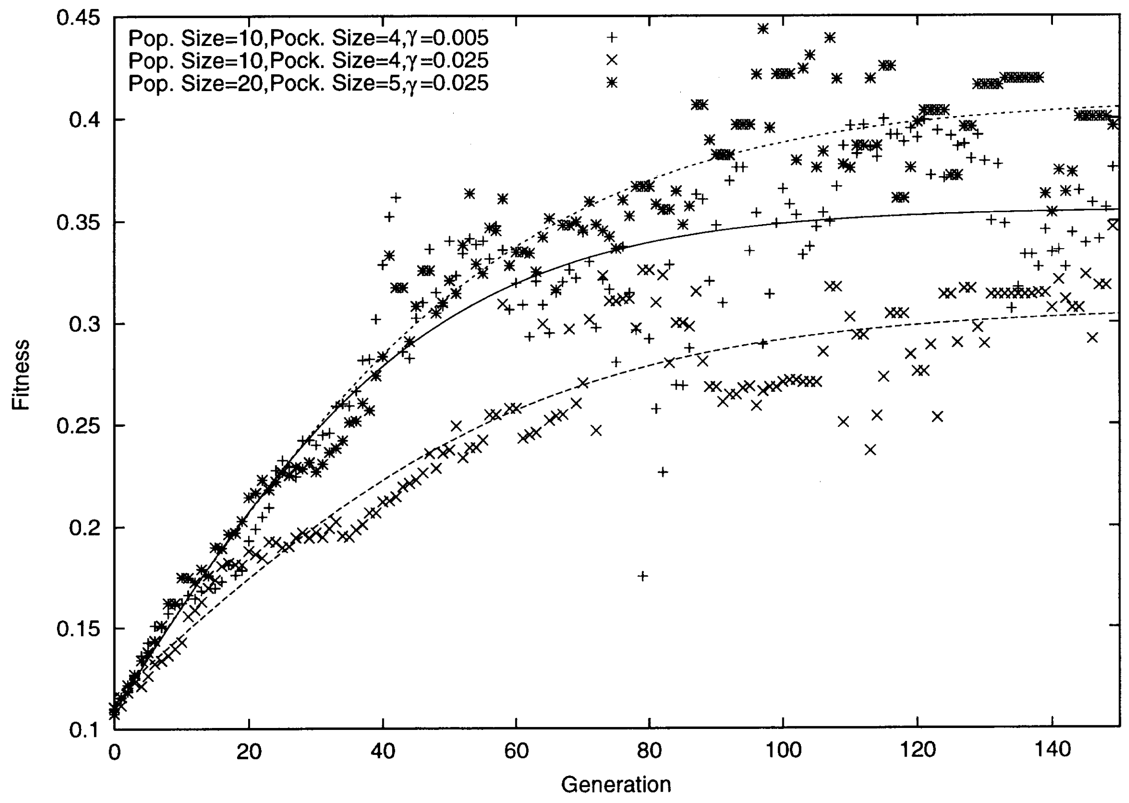


Figure 6.9: Comparison between trials of the GA reproduction by cross-over, mutation, and pocketing. All trials have been carried out with a value of $\mu_{max} = 20\%$.

The trends of the optimization process after the addition of the cross-over operator is visible in Figure 6.9. In this plot it is clearly visible the effect of combining individuals with partially optimal CPGs, which produces oscillations in the value of the fitness for values near the saturation level, even for small mutation rates. This effect is attributed to the introduction, by the cross-over operator, of macro-mutations all the way to the end of the evolution process. The oscillations would stop if genetic uniformifity subsided, therefore the oscillatory behavior is in this case desirable.

Although the crossover operator does not improve the performance of the algorithm over the time of one trial, the addition of macro-mutations and the presence of diversity in the population could lead to better results for longer evolution time. Also controllers that constitute the population of the last few generations of the GA, exhibit a broader variety of stable behaviors than the ones produced without the cross-over operator. In the case of simple mutation the stable controllers where clustered around a single stable set of parameters.

6.2.5 Random Search

In order to best asses the performance of the GA, data has been collected for a random search in the space of the parameters. The fitness of 3000 sample parameter vectors randomly taken from within a hypercube of side 1.6 centered in the origin of the parameter space has been evaluated in simulation. The choices of the domain of the search and the number of samples have been made to carry out a fair comparison between the different optimization algorithms. The GA explores values of the parameters in the range [-0.8:0.8] and evaluates a maximum of 20 individuals per generation for a total of 150 generations, corresponding to 3000 samples.

In Figure 6.10 the result of the random search algorithm is presented. The plot shows the best estimate against the number of iterations of the algorithm. The overall average value (horizontal line) is also plotted to show how the random nature of the process leads to a near zero mean fitness.

Although the random search algorithm is known to be inefficient it is almost ubiquitously used as a baseline against which to compare more sophisticated algorithms. The results produced through random samples give a performance for the best individual which is roughly half the maximum fitness achieved by the GA. The stability analysis carried later in this chapter will show that the controllers produced by this algorithm are highly unstable. This is due to the inability of this algorithm to refine the value of the parameters around a near-optimal solution.

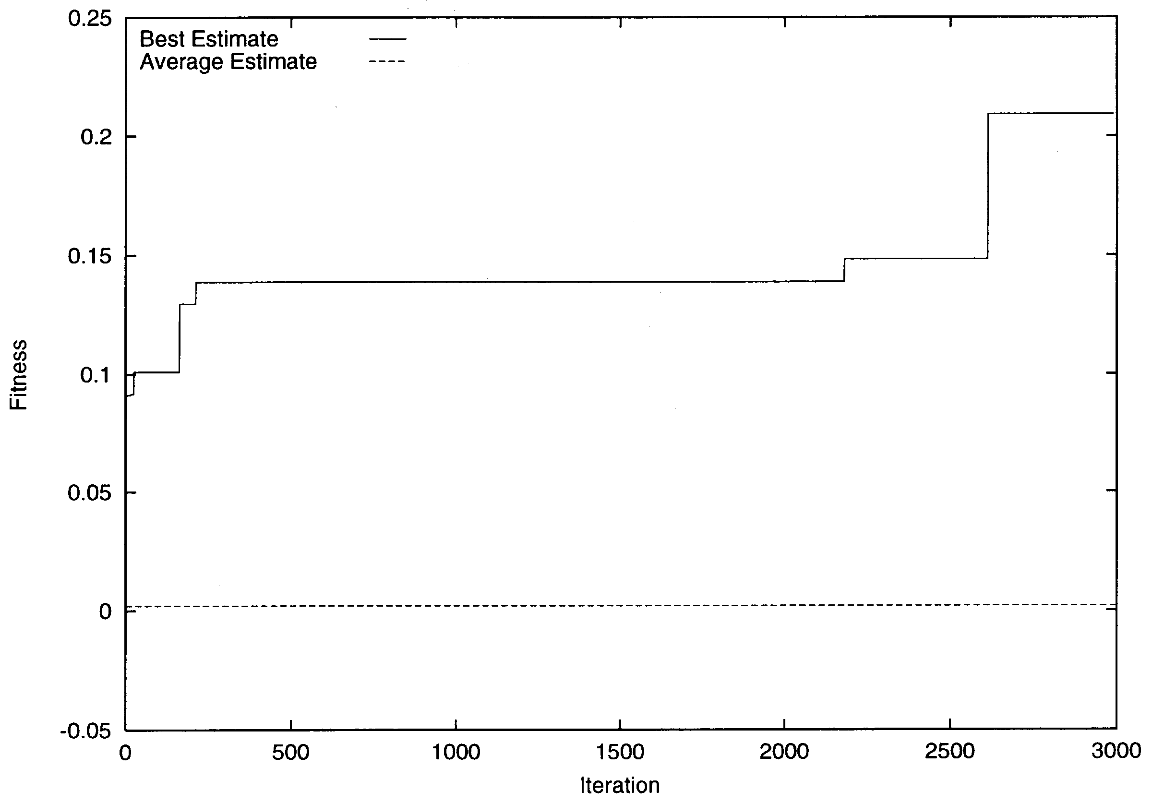


Figure 6.10: Results from a random search performed within a hypercube of side 1.6 in the parameter space. The best and the average estimates of the fitness are plotted against iterations of the algorithm.

6.2.6 Adaptive Monte Carlo Search

A second comparison has been carried out to evaluate the performance of the GA against an independent optimization method. The algorithm is an adaptive version of the Monte Carlo search. Here an array of M independent random variables is used to search the space of the parameters. The estimates of parameter values is based on a fixed number of previous estimates. Successive estimates distributions are restricted around the best estimate of the previous ones [44].

It can be proved that the classes of algorithms based on the distribution properties of uniform random variables converge to the absolute maximum of the optimization function, given that the number of estimates is large enough [45]. In the case of

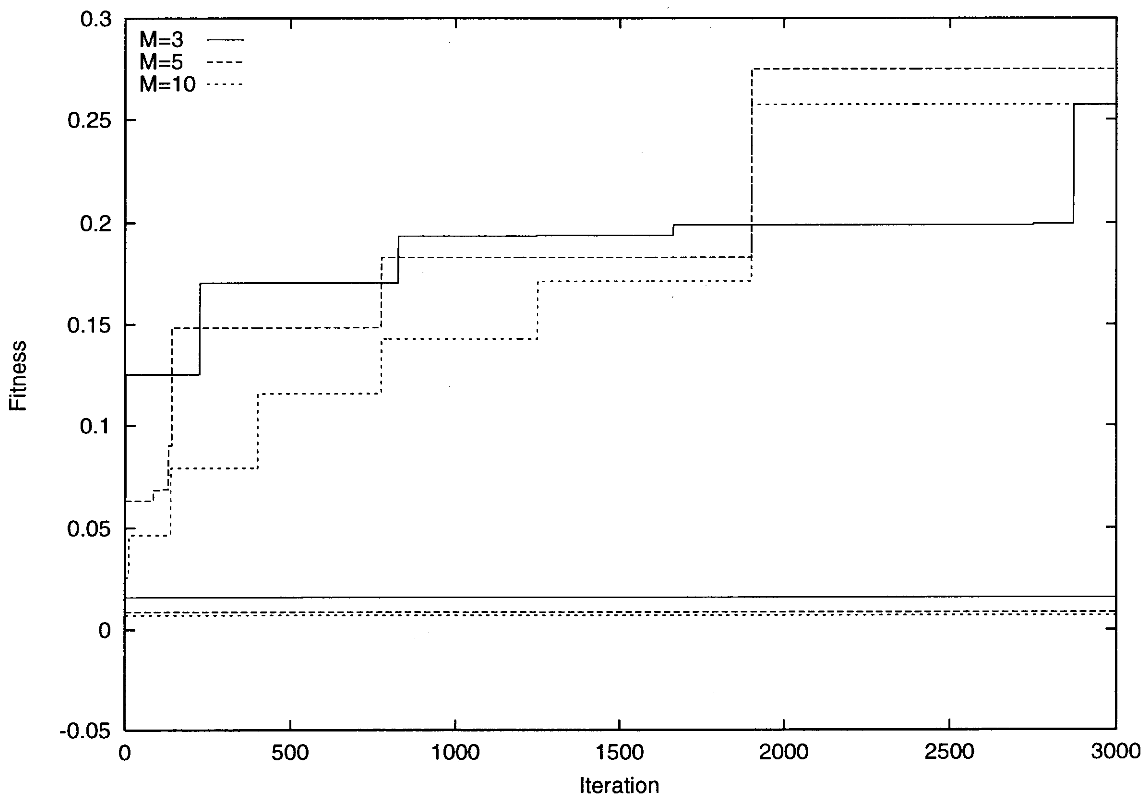


Figure 6.11: Results from a Monte Carlo search performed within a hypercube of side 1.6 in the parameter space. The best and the average estimate of the fitness are plotted against iterations of the algorithm.

the particular algorithm used here it is necessary that the number M of independent random variables is also large enough [44].

Optimization experiments have been carried out using the adaptive Monte Carlo search for values of $M = 3, 5, 10$. Results are shown in Figure 6.11, where the best estimate of the algorithm is plotted against the number of samples. The average of the sample estimates is also show as an horizontal line. Note that the mean value of the estimates is positive, differently from the zero mean estimated found for the uniform random search.

6.3 Algorithm Comparison

This section presents a comparison of the results of the three algorithms examined for the optimization of the walking patterns. The best trials for the different variants of the GA, the Random Search and the Adaptive Monte Carlo methods are compared.

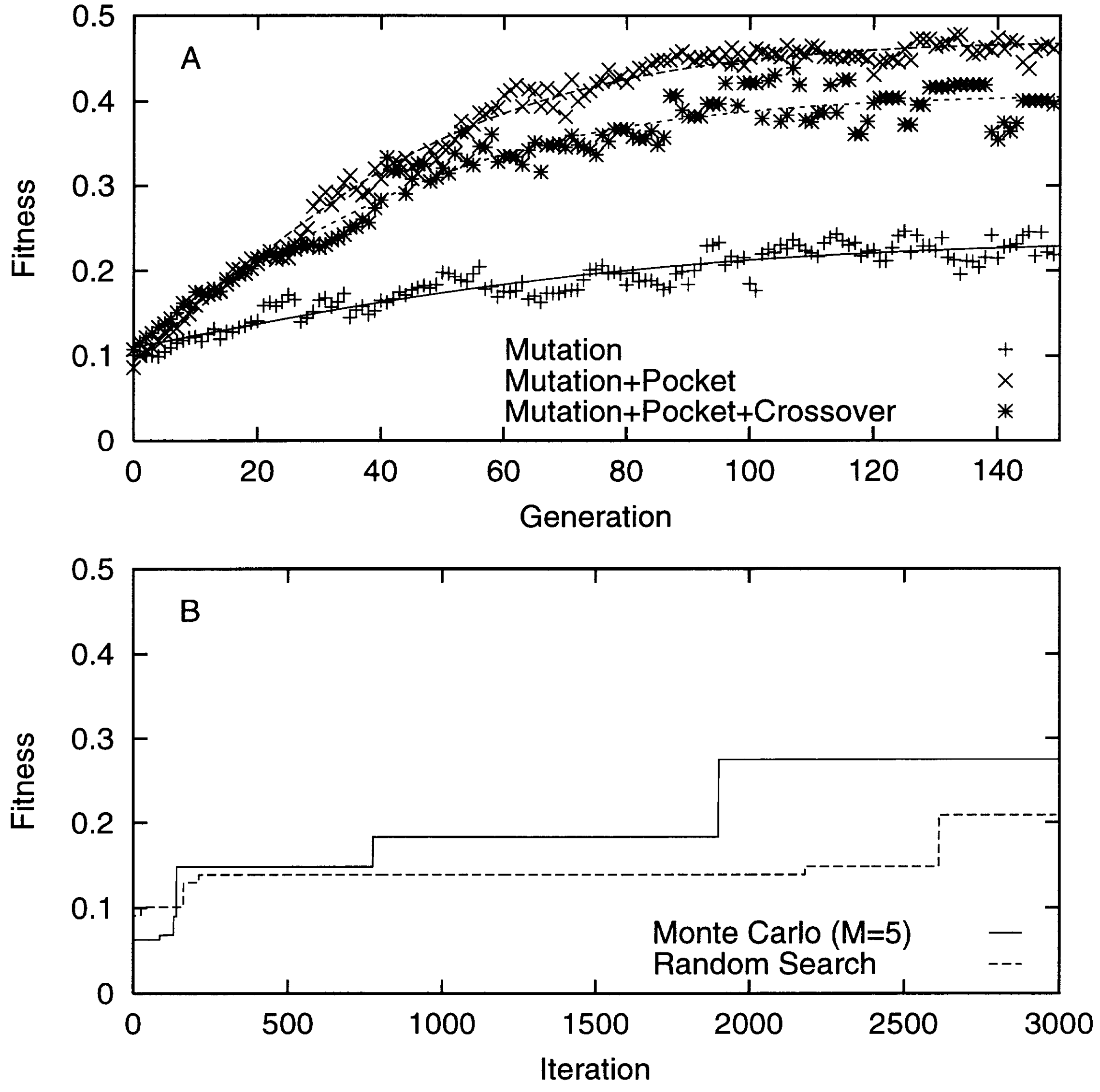


Figure 6.12: Comparison among different optimization algorithms. (A) shows the best case trial for each implementation of the GA; (B) direct comparison between the Random Search and the Adaptive Monte Carlo methods.

In Figure 6.12 (A) the best performance achieved by each variant of the GA is shown. The effect of the three genetic operators used in the implementation of algorithm is visible. Panel (B) shows the direct comparison between the Random Search and Adaptive Monte Carlo methods, where for the latter the best trial has been chosen with a value of $M = 5$.

The Fitness scale in both panels of Figure 6.12 is the same to facilitate the comparison. The scale of the X axis cannot be compared directly as the GA evaluates runs for a total of 150 generations, whereas the number of iterations for the other two methods is 3000. The total number of individual controllers evaluated by the GA as well as the other methods is 3000, therefore the two plots can be compared in terms of number of parameter samples.

From the results obtained by all the algorithms examined can be concluded that the GA outperforms the other methods roughly by a factor of 2. The main reason for the low performance of the Adaptive Monte Carlo method is that it is based on the statistical properties of random variables. In order to construct a sensible statistics for this variable a rather large number of samples has to be evaluated. In some sense this drawback is similar to the major limitation of the Thermal Relaxation method described in Chapter 4. For the optimization of walking problem, the high computational cost of evaluating samples and the large parameter space render the Adaptive Monte Carlo method not entirely applicable to the problem.

6.4 Stability Analysis

Oscillatory patterns derived through parametric optimization tend to maximize the speed of locomotion maintaining the stability of the system. The following sections investigate the sensitivity to perturbations of the resulting controllers using two independent measures. The CPGs extracted through optimization are evaluated in simulation in the presence of perturbations of the supporting surface. Two types of perturbative tests are carried out:

1. A bump of given height is placed half way through the walking path to test the ability to recover stability.
2. Random roughness is added to the height of the floor, to mimic unstructured terrain and test the adaptivity of the walking gait.

The aim of this analysis is to establish whether the system falls into a limit cycle of its dynamics and, as a consequence, tends to return naturally to the unperturbed state or unrecoverable looses its stability. From the experimental results a better understanding of the sensitivity to external perturbation can be inferred and a realistic definition of robustness outlined.

6.4.1 Vestibular Method

The first measure of stability used to verify the sensitivity to perturbations of the walking patterns is the same used in the definition of the optimization fitness function (see Equation (4.5)). It ranges from 0 to 1 and expresses the "flatness" of the body of the robot during locomotion. The following formula gives the dependence of the stability from the orientation of the robot.

$$S(\theta) = \frac{\delta\ cos(\theta)^2}{(\delta - 1)\ cos(\theta)^2 + 1} \tag{6.2}$$

where δ is a positive constant.

First a walking pattern with high stability is taken into account. Its vestibular stability is computed as a function of time during normal walking on a flat surface to use it as a baseline of comparison for the behavior in the presence of perturbations.

The first type of perturbation examined is the addition of a bump half way through the path of the robot. Three different heights of the bump are examined and the corresponding stability during walking is computed. The values of the bump height are 1, 2 and 3 *cm*. Results for this perturbation experiment are visible in

Figure 6.13, where the solid line is the stability values in the perturbed environment whereas the dashed line is the stability in the unperturbed flat surface. In both cases when the bump height is 1 and 2 *cm* the walker quickly recovers from the perturbation. Also the stability values do not significantly differ from the unperturbed ones.

For the case in which the bump height is 3 *cm* the walker shows quite a different stability pattern right at onset of the perturbation (around .4 *sec*), to unrecoverably loose stability shortly after. The robot actually flips over in the attempt of overcoming the bump.

The second perturbative action is the addition of random values to the surface height. This to simulate the roughness of an outdoor or unstructured terrain. Here 4 different random ranges are considered: 1, 2, 3, 4 *cm*. The random values are distributed uniformly over the terrain and over the height range. During this second experiment the stability of the perturbed walker changes much less than in case of the bump examined earlier. The results are visible in Figure 6.14, where the solid line is the unperturbed stability and the dashed line is the stability of the walker over rugged terrain. In this case the stability collapses quite dramatically for a range of roughness of 4 *cm* remaining almost unchanged for other values.

These results suggest that the limit cycle embedded in the control pattern is robust to perturbations both localized and evenly distributed in the range [0:3] *cm*. This experimental evidence constitutes an indication for the existence of stable attractors in the dynamics of CPG-based control of walking, though a more in depth analysis could provide details on the characteristics of such attractors as well as an estimate of the number of different stable behavior the system can exhibit.

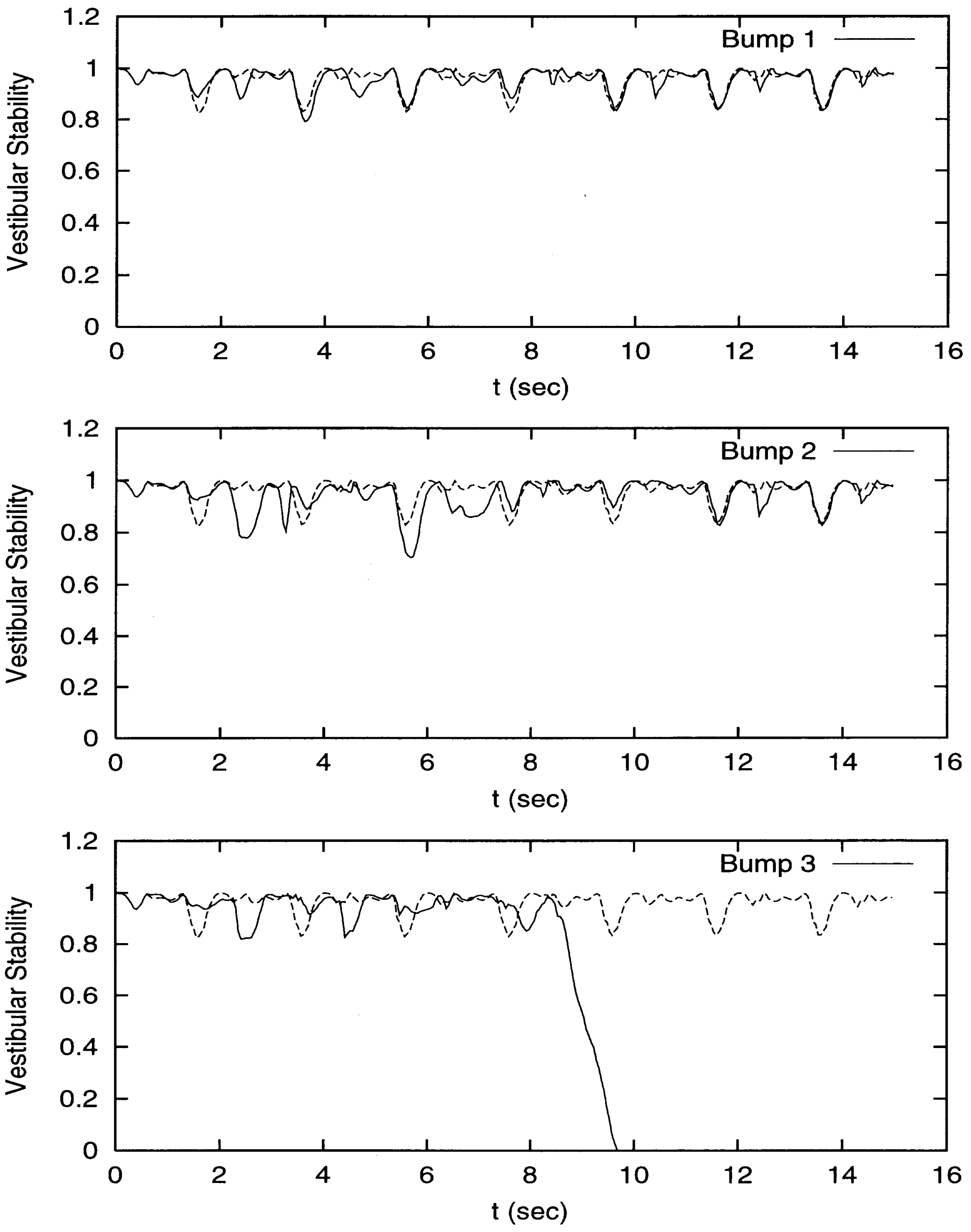


Figure 6.13: A perturbation is added to the simulated environment to test the stability of the control patterns. In this figure results are presented for a highly stable controller. The solid lines refer to walking experiments in the presence of the perturbing bump on the terrain 1, 2 and 3 *cm* high, whereas the dashed lines are the stability during walking of a flat surface.

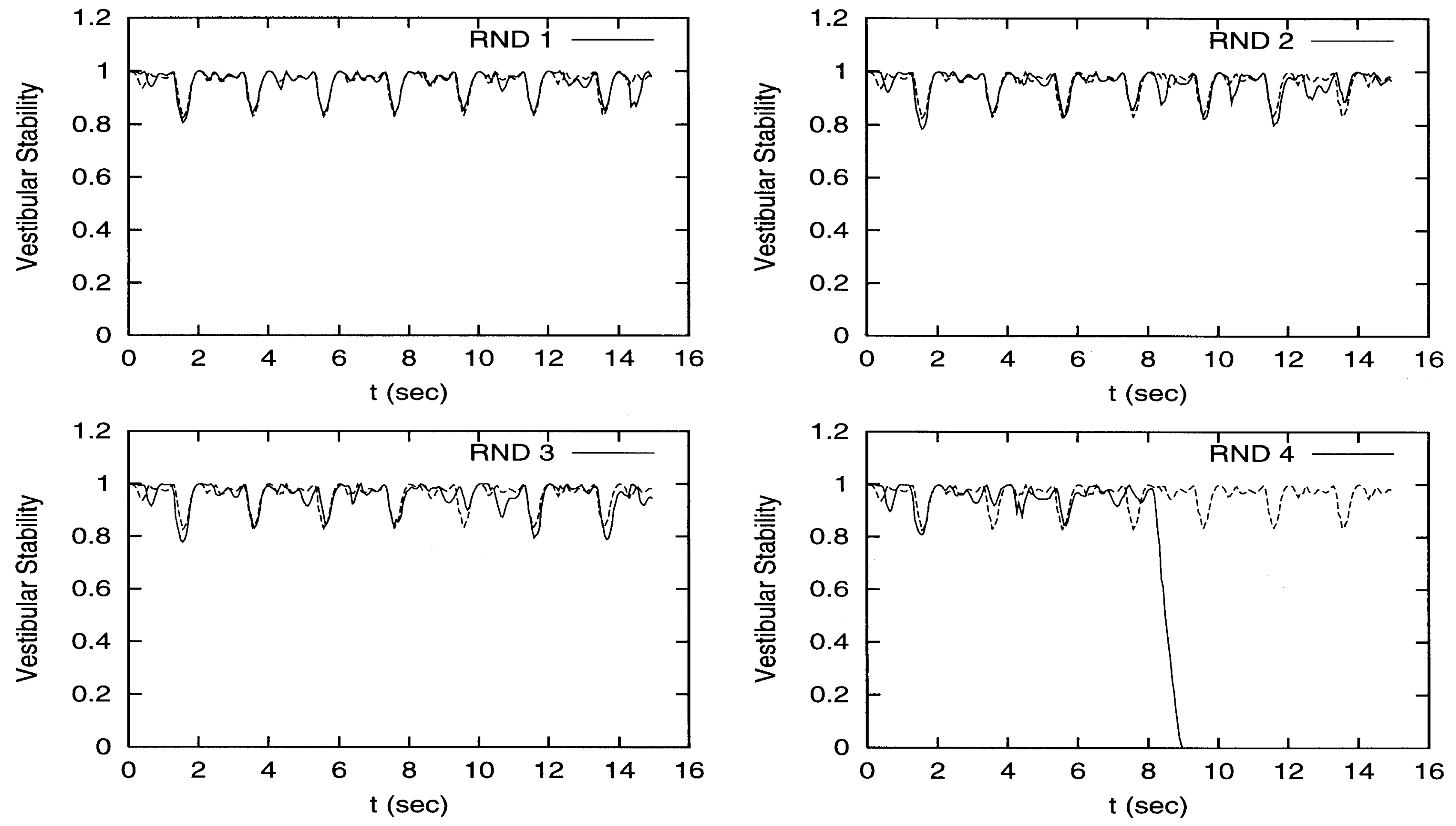


Figure 6.14: In this case the perturbation is a uniformly distributed roughness of the supporting surface. The four panels refer to the four different ranges of random values of 1, 2, 3 and 4 *cm*. Note how the stability values do not differ much from the unperturbed case for the smaller values of the perturbation.

The same perturbation experiments have been repeated on a controller with much lower vestibular stability to show how in this case the sensitivity to perturbations is dramatically increased. Results for this unstable controller are shown in Figure 6.15 and Figure 6.16. As can be seen in Figure 6.15 the walker completely looses its stability in the attempt to overcome the smallest bump of 1 *cm*. For this reason the higher bumps have not been considered.

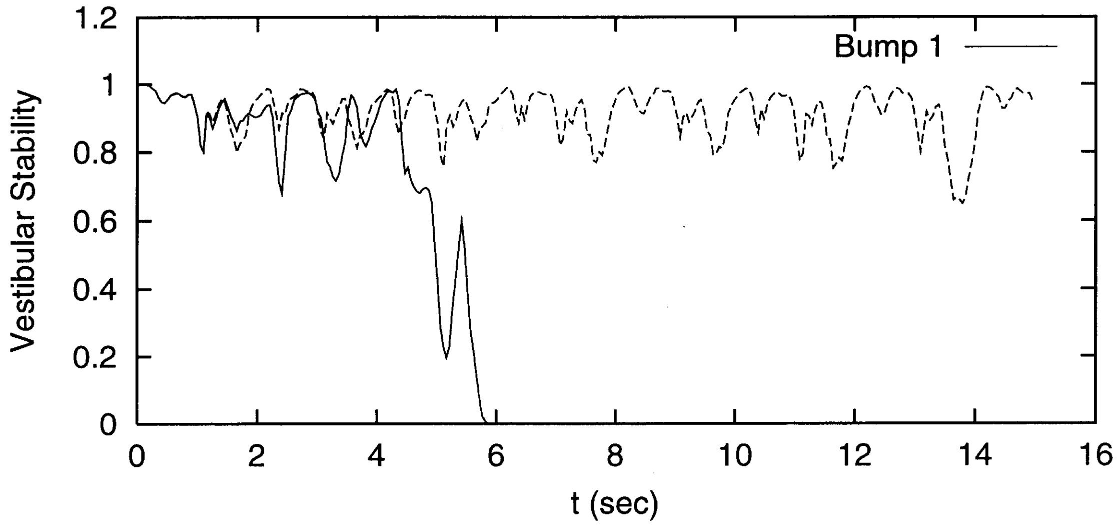


Figure 6.15: Result from the perturbation experiment on a highly unstable controller. The walker looses its stability and falls as it attempts to overcome the obstacle. In this case the height of the bump located in front of the simulated walking robot is 1 *cm*.

In the case of random perturbation the controller manages to maintain a stable pace throughout the whole duration of the first three experiments. A more pronounced difference with the unperturbed state is evident when compared to the previous, more stable, controller. In the case of rugged terrain (Figure 6.16) the walker does not collapse for small values of the random range. This can be explained by the fact that the actual height difference between the foot placements is, on average, much less that the range of random values.

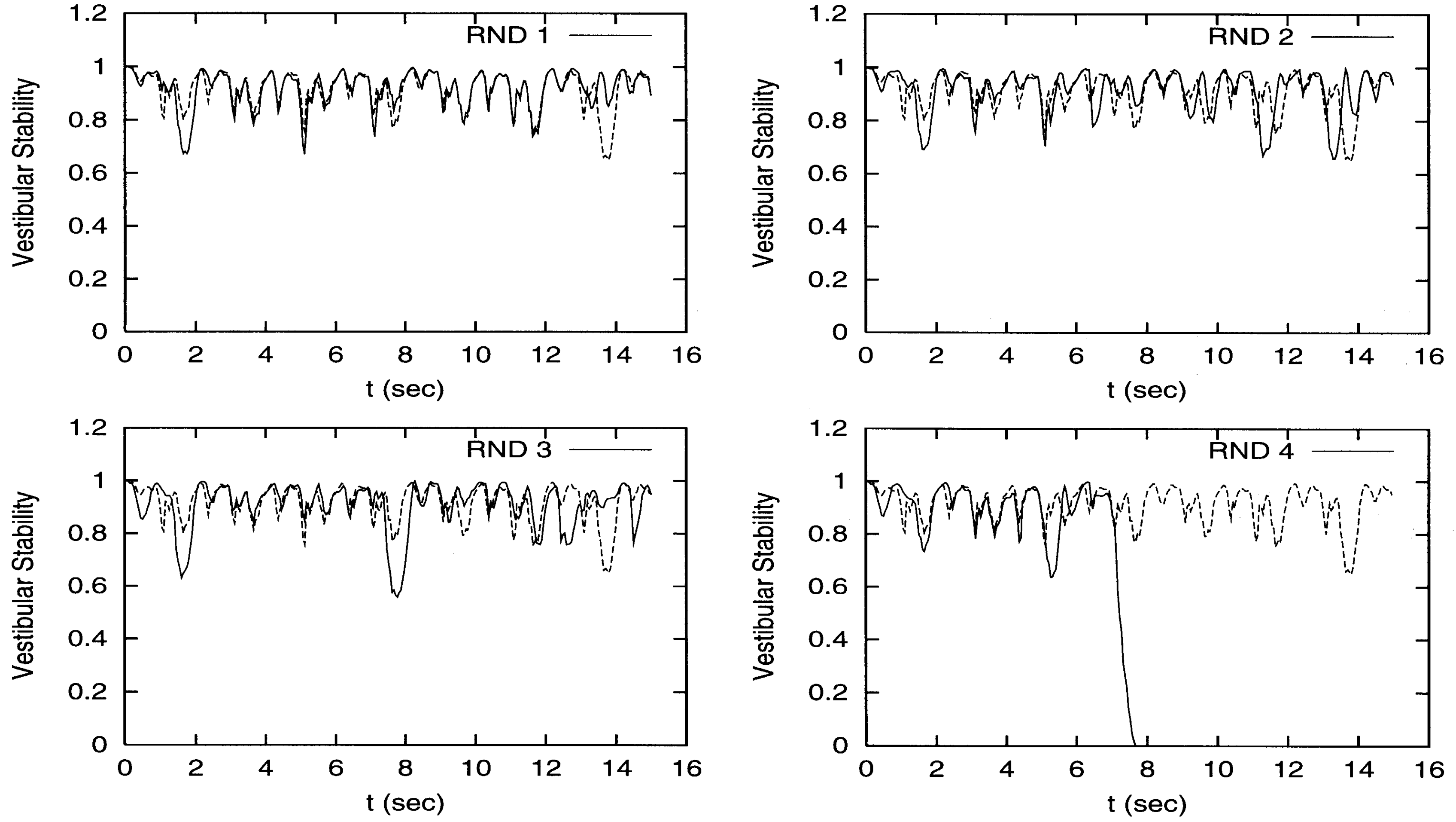


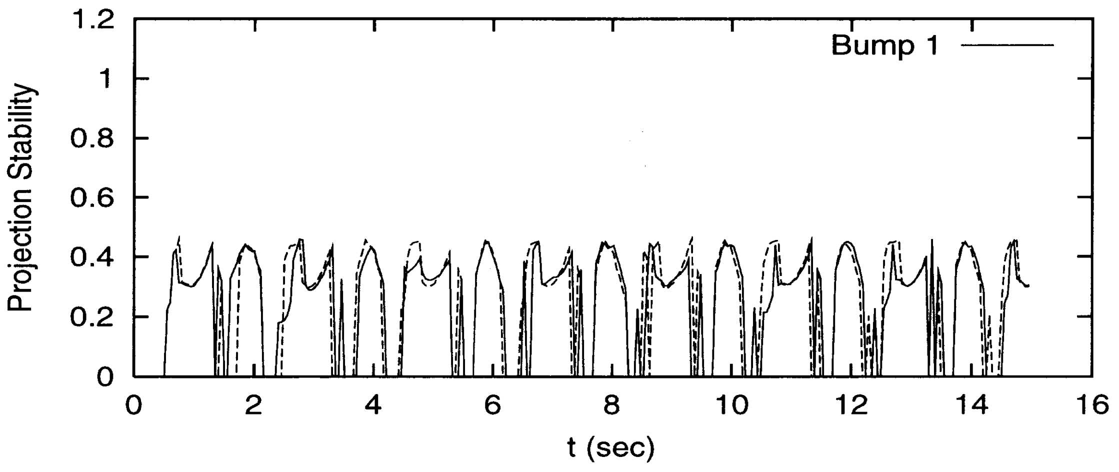
Figure 6.16: Effect of random perturbations to the behavior of controller with low stability. There are noticeable differences from the unperturbed behavior even for a range of random values of 1 *cm*.

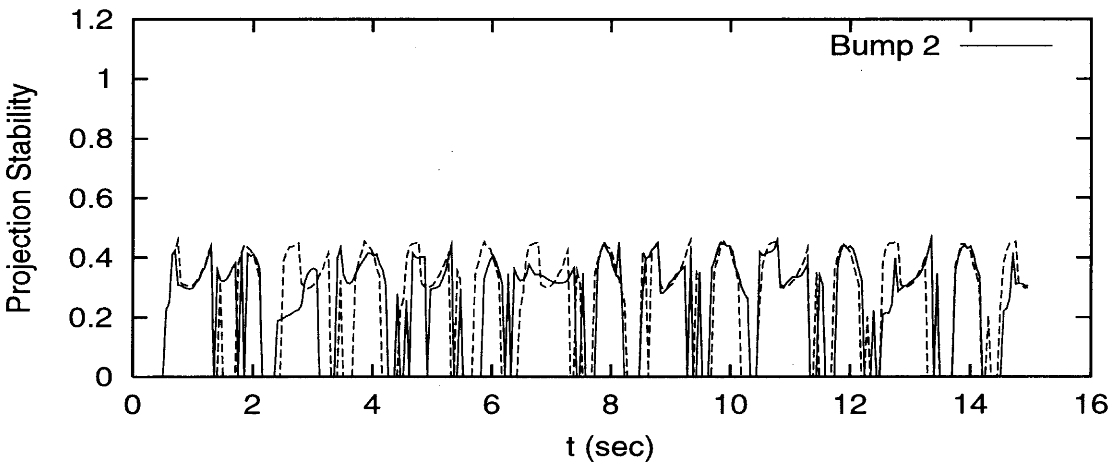
6.4.2 Projection Method

Another definition of stability, which is widespread in the robotics literature, is based on the minimum distance of the projection of the center of mass of the body of robot from the boundary of the support polygon defined by the feet in touch with the ground. The stability is defined as the ratio between this distance and the height of the center of mass. Of course this measure of stability is undefined when less than three supporting feet are in touch with the ground as the support polygon degenerates into a segment. Mainly for this reason this stability measure criterion is not best suited for systems intrinsically unstable such as the quadruped robot. As it is shown in this section controllers with good performance would have been greatly penalized in the optimization process by this measure of stability.

The same perturbation experiments examined in the previous section have been repeated to estimate the stability measure using the projection method. The information gained with this method is similar to the one provided by the vestibular method and it is presented in Figure 6.17 and Figure 6.18. A major difference exist in the way the stability information is linked to the desired actual stable behavior of the walking robot. In the previous section a clearer indication of the stability of a controller could be inferred from the unperturbed experiment. Using the projection method gives results not easily connected, in absolute terms, with the overall stability. Confronting the flat terrain experiments with the experiments in the presence of a perturbation it becomes evident which controller has less sensitivity to external disturbances.

The main reason for the qualitative difference in the level of description of the two methods lies in the fact that the projection method was initially developed for intrinsically stable walkers, where a minimum triangular support base was always defined. Obviously this is not the case with the quadruped robot, that often during walking keeps only two, or even one foot on the ground.





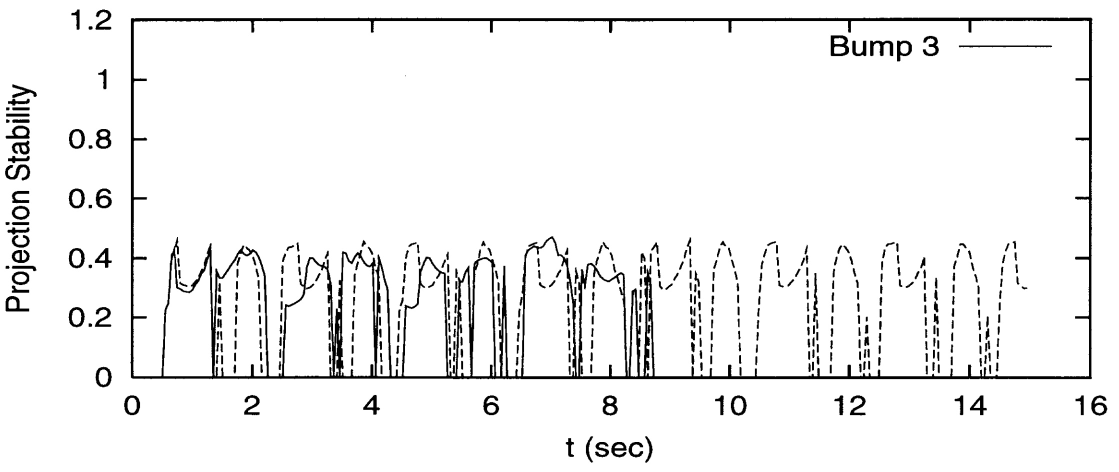


Figure 6.17: Result from the perturbation experiment on a stable controller. The information obtained from this data is similar to the one given by the vestibular method, although here the stability cannot always be defined as less than three feet are occasionally in contact with the ground.

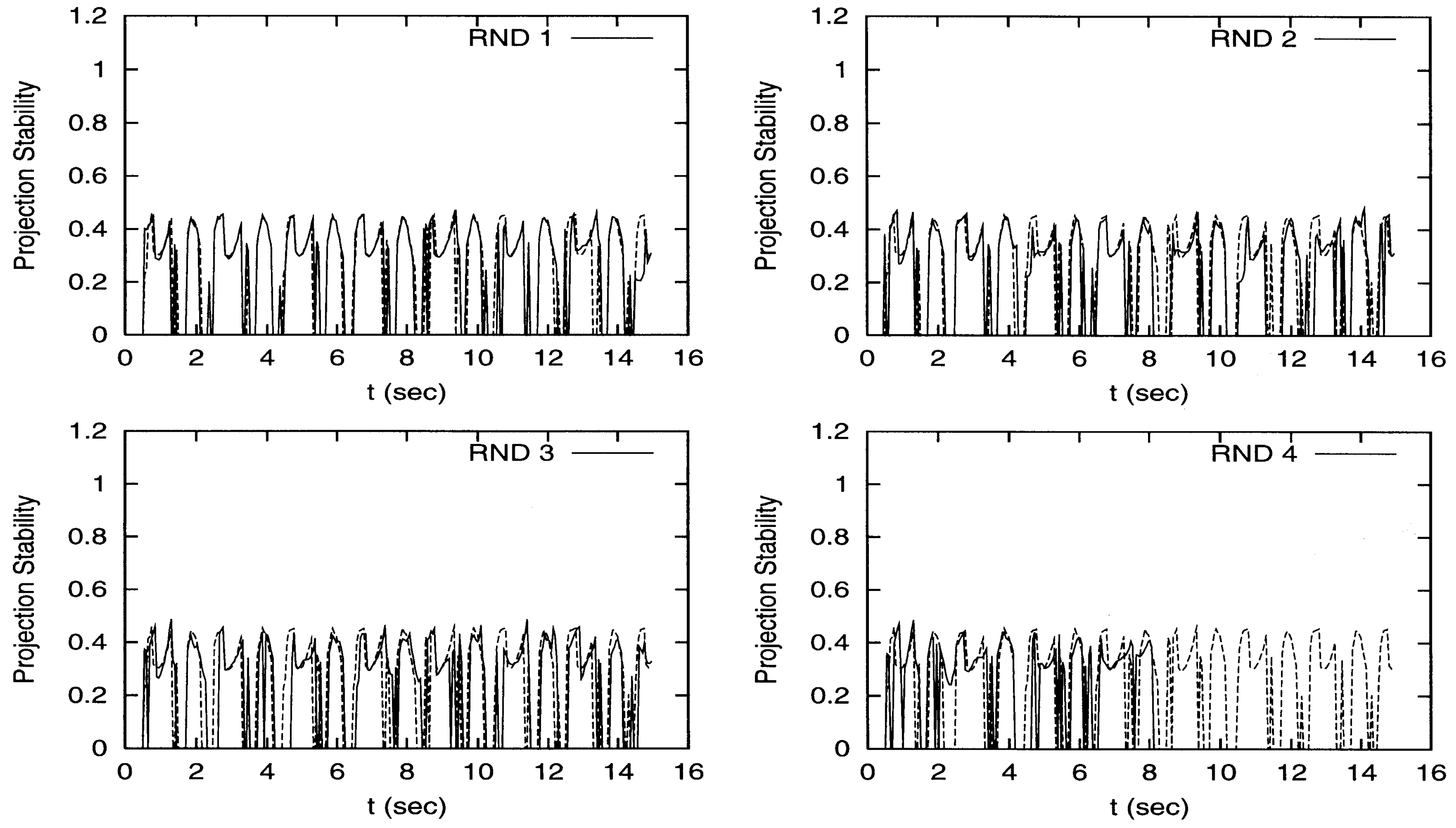


Figure 6.18: Effect of random perturbations to the projection stability for a stable controller. With this definition of stability, similarly to the vestibular case, there are noticeable differences from the unperturbed behavior even for a range of random values of 1 *cm*.

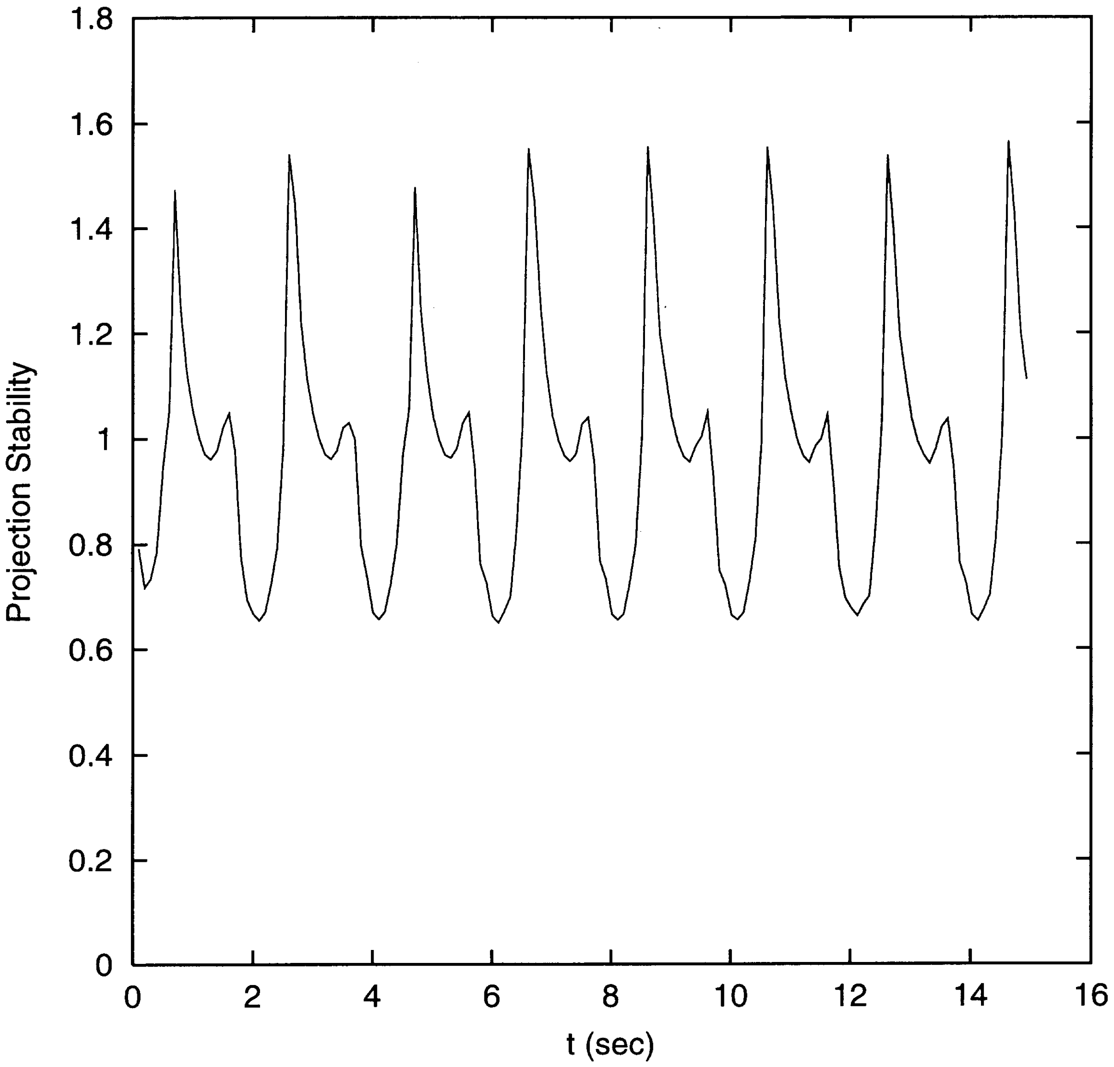


Figure 6.19: Stability of an hexapod walker on flat terrain. Note how the value of the stability is always positive as the measure is defined at every given point in time because of the triangle formed by the supporting feet.

This concept is evident in Figure 6.19 where the stability values during normal walking are reported for an hexapod walker. The value of the projection stability is always defined for this type of walking robots as they maintain three supporting feet on the ground at all times. The function plotted in of Figure 6.19 is positive throughout the whole duration of the experiment and the stability values are much higher than the ones measured on the quadruped. This difference in values is also due to the different ratio between the height and width of the two walking robot mechanisms. The hexapod has a better height to width ratio being closer to the ground (see Appendix A).

In Figure 6.20 and Figure 6.21 the projection stability results are shown for the perturbation experiments carried out with a highly unstable controller. Although it is possible to distinguish between the projection stability patterns of this controller and the previously examined more stable one (Figure 6.17 and 6.18), the difference is less evident than for the vestibular stability measure. This is due to the problems connected with the definition of this stability measure for systems without continuous stable support.

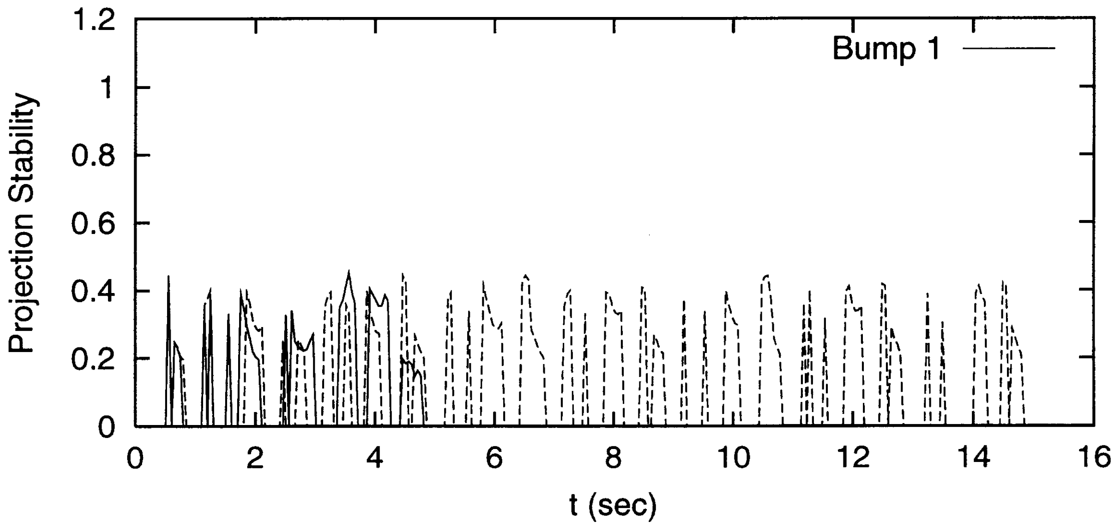


Figure 6.20: Projection stability results from the perturbation experiment on a highly unstable controller. The walker looses its stability and falls as it attempts to overcome the 1 *cm* high obstacle.

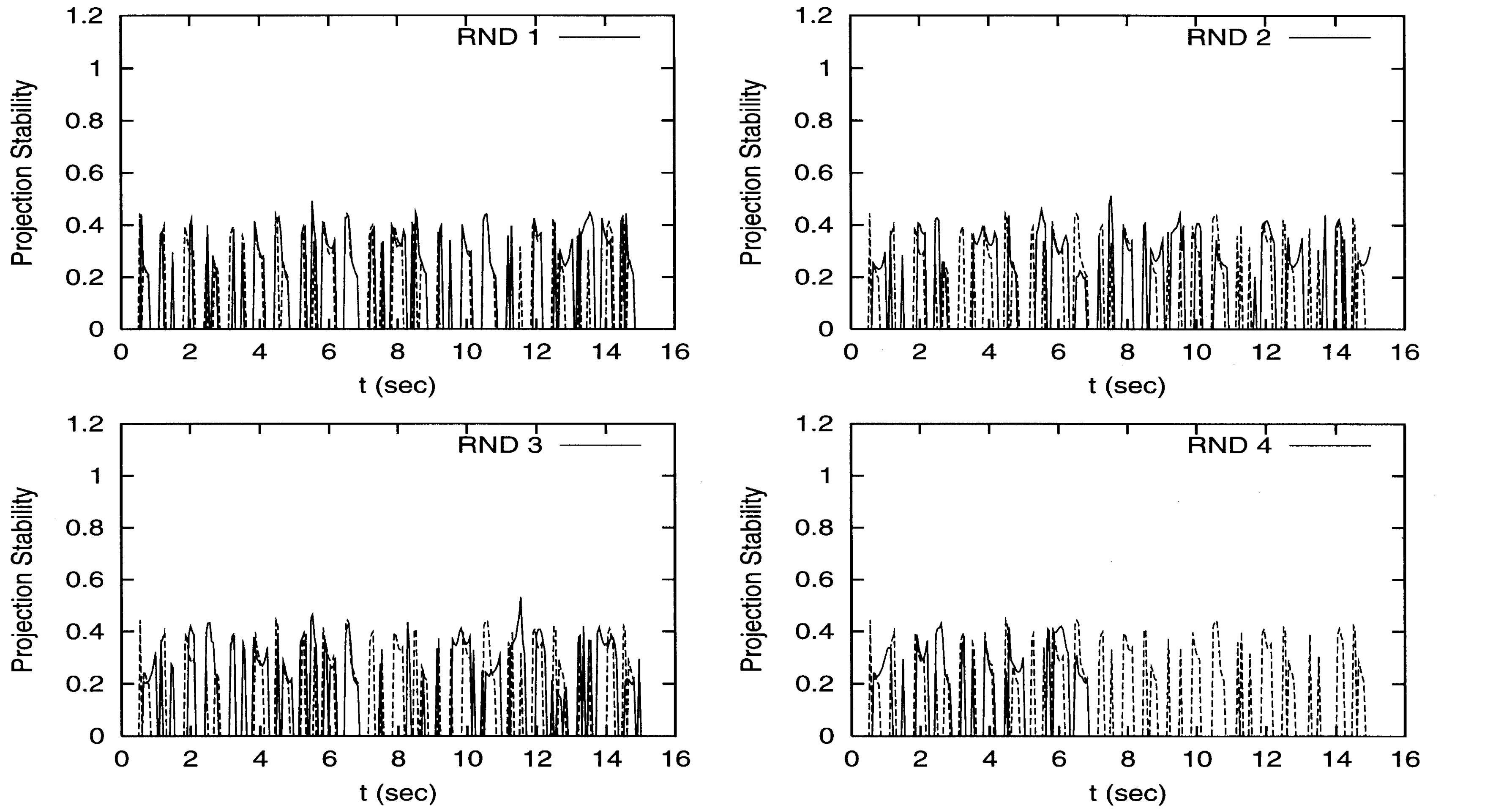


Figure 6.21: Effect of random perturbations to the behavior of controller with low stability. There are noticeable differences from the unperturbed behavior even for a range of random values of 1 cm.

CHAPTER 7

DISCUSSION

7.1 Proof of Concept

The idea of CPG-based dynamic control of walking has been presented and tested throughout this thesis. An optimization strategy has been used to achieve stability of locomotion for an intrinsically unstable walking robot. The optimization method used no a priori knowledge on the walking process except its cyclicity. The only sensory information used by the control system is the angular measurements of the joints of the robot.

In spite of the lack of detailed model of the target behavior, the optimization process finds stable patterns that produce walking locomotion up to 1.5 Km/h over rugged terrain. This result indicates that the system, once optimized falls into an attraction basin for a particular limit cycle of the dynamics of the mechanical structure. The limit cycle is, obviously, linked to the patterns of movements followed by each individual joint, thus it is the system coupled with the control action which defines completely the dynamics of system and its limit cycles.

The main message that can be extrapolated from the results of this work is that the implementation of low level control of quadruped walking can be considered a search process, which seeks stable attractors in the parameter space of a mechanical system. This approach to motor control can lead to important advances in the understanding of walked locomotion, as well as, providing a new general method for dynamic control, which is not based on detailed descriptions of robotic mechanisms or a knowledge base defining the behavior of interest.

7.2 Simulation: A Useful Tool

In developing the framework for the work carried out in this thesis, dynamic simulation has been extensively used. It has proved useful at design stage, suggesting appropriate solutions for the mechanical structure, all the way to the evaluation of the final experimental results. Its use was paramount for the implementation of the optimization strategy, which would have been infeasible with just the actual robotic mechanism.

The latest version of the simulation software is a fairly general tool for the simulation of multibody systems. The interface included with the software has allowed several people in Neural Computation and Robotics Lab to run their own experiments. The overall accuracy and performance of the simulator, which has been a crucial issue in its development compares favorably with other software tools available both commercially or open source.

A particularly important aspect of the experimental results is the validation of the simulation accuracy against an actual mechanical system. This, though missing in many research papers on robotics, is a crucial step in the development process of any control strategy of a dynamical system carried out using simulation. The information gained in simulation is only useful together with a quantitative measure on how faithful the numerical results are to the real system.

7.3 Stable Walking Through Open Loop Control

The ability to extract stable behaviors from a dynamical system without high level sensory information is a very important aspect of this work. Although examples exist in the literature for systems, where particular assumptions are made to constrain the nature of the control task, such as the exploitation of symmetries or the reduction of the dimensionality of the problem (e.g. planar case), the method presented here has general applicability and deals with articulated systems in three-dimensional space.

7.4 Gaits as Attractors for the Dynamics of Walking

With the stability analysis carried out in Chapter 6 some evidence has been built that attractors for the dynamics of walking exist in the dynamics of quadruped robots. This result not only opens the way to a variety of new strategies for dynamics control of robotic mechanisms, but may also provide evidence of the underlying motor control algorithms in biological systems, where much of the low level motor control is implemented without the intervention of the cerebral cortex.

7.5 Future Direction

There are several aspects of motor control, not fully addressed in this thesis, which are worth studying in more detail. A first set of issues is connected to the optimization method adopted here, a second set has to do with the experimental setup.

One interesting future development of the strategy presented here is the analysis of its applicability to a broader set of behaviors as well as a variety of different environments. The main concern here was to prove the suitability of the method for achieving stable walking with a quadruped robot, but other interesting scenarios could be investigated, such as climbing stairs or jumping over obstacles.

Another development which would be worthy of an in depth study is the application of the CPG method to systems with higher flexibility and richer dynamics. With a set of different mechanical systems an analysis of the characteristics of their dynamic attractors could be carried out and a better understanding of their potential behaviors gained.

APPENDIX A

SIMULATION EXAMPLES

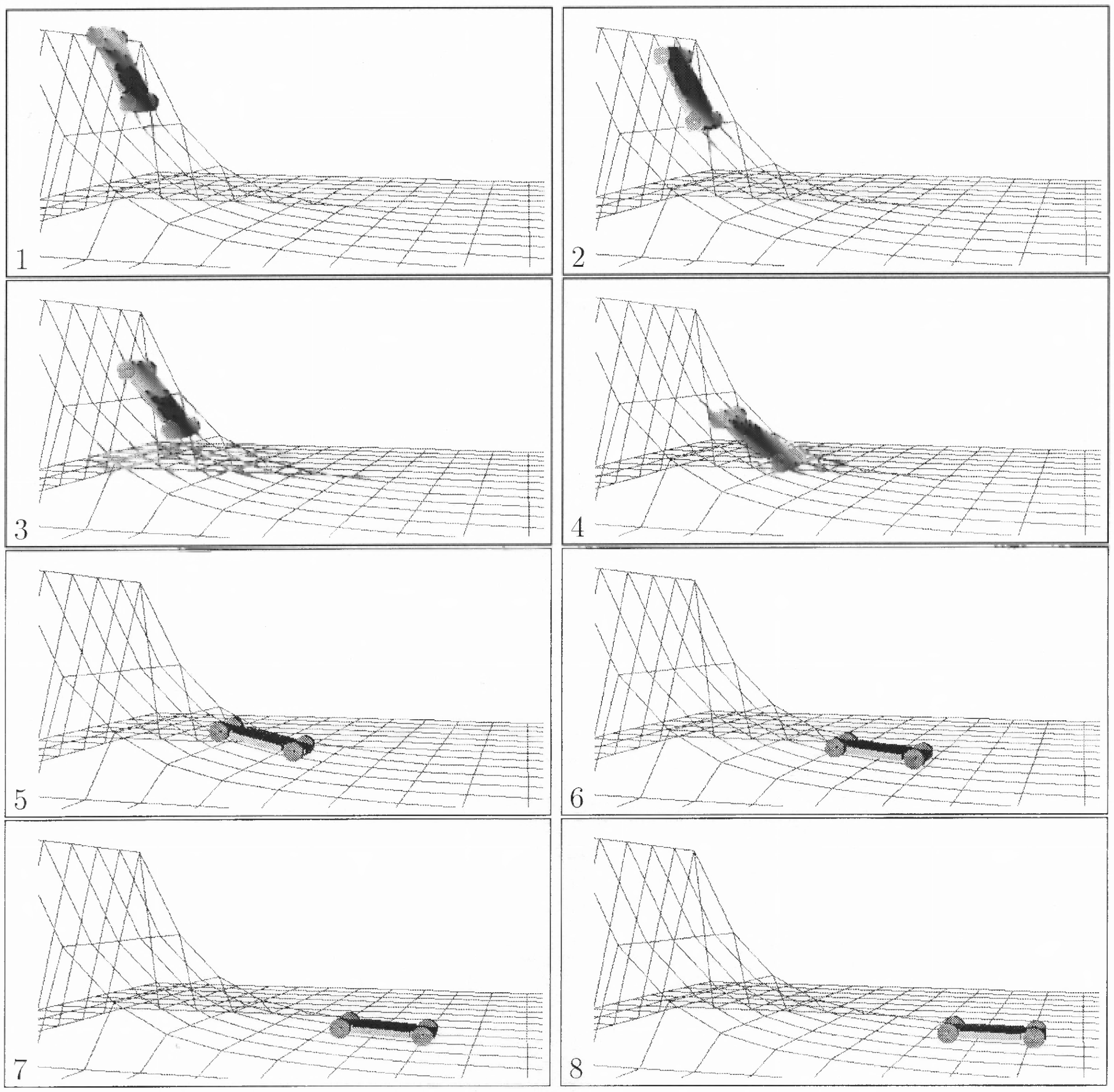


Figure A.1: Simulation of a small toy car going down a slope in the absence of control. This example shows how to utilize the terrain shape definition to change the simulation environment.

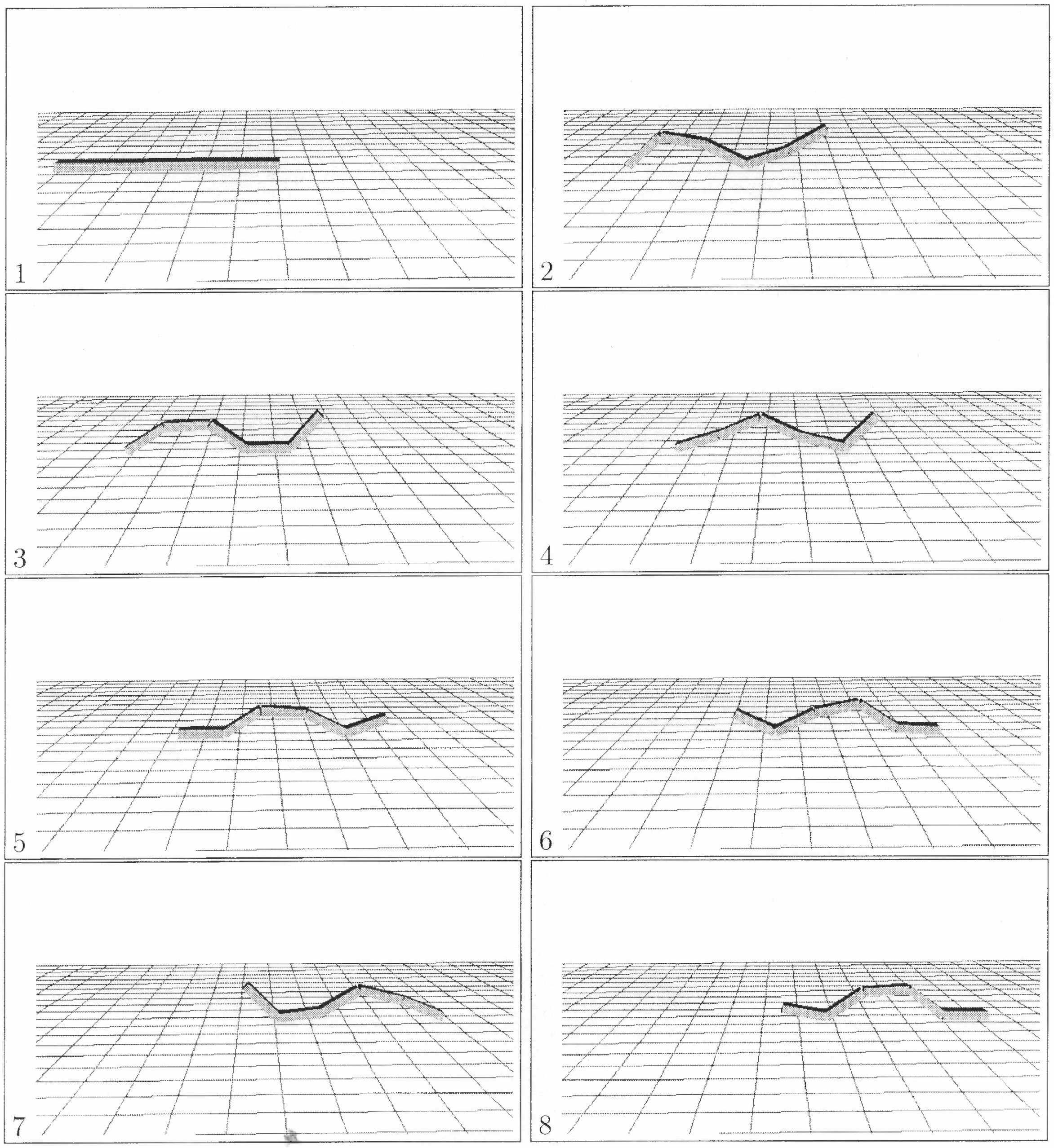


Figure A.2: In this simulation an example of simple periodic pattern applied to the control of a mechanical system is presented. The worm is moved using a single sinusoidal pattern for each joint. Each joint pattern has a $\frac{\pi}{2}$ phase shift with the next.

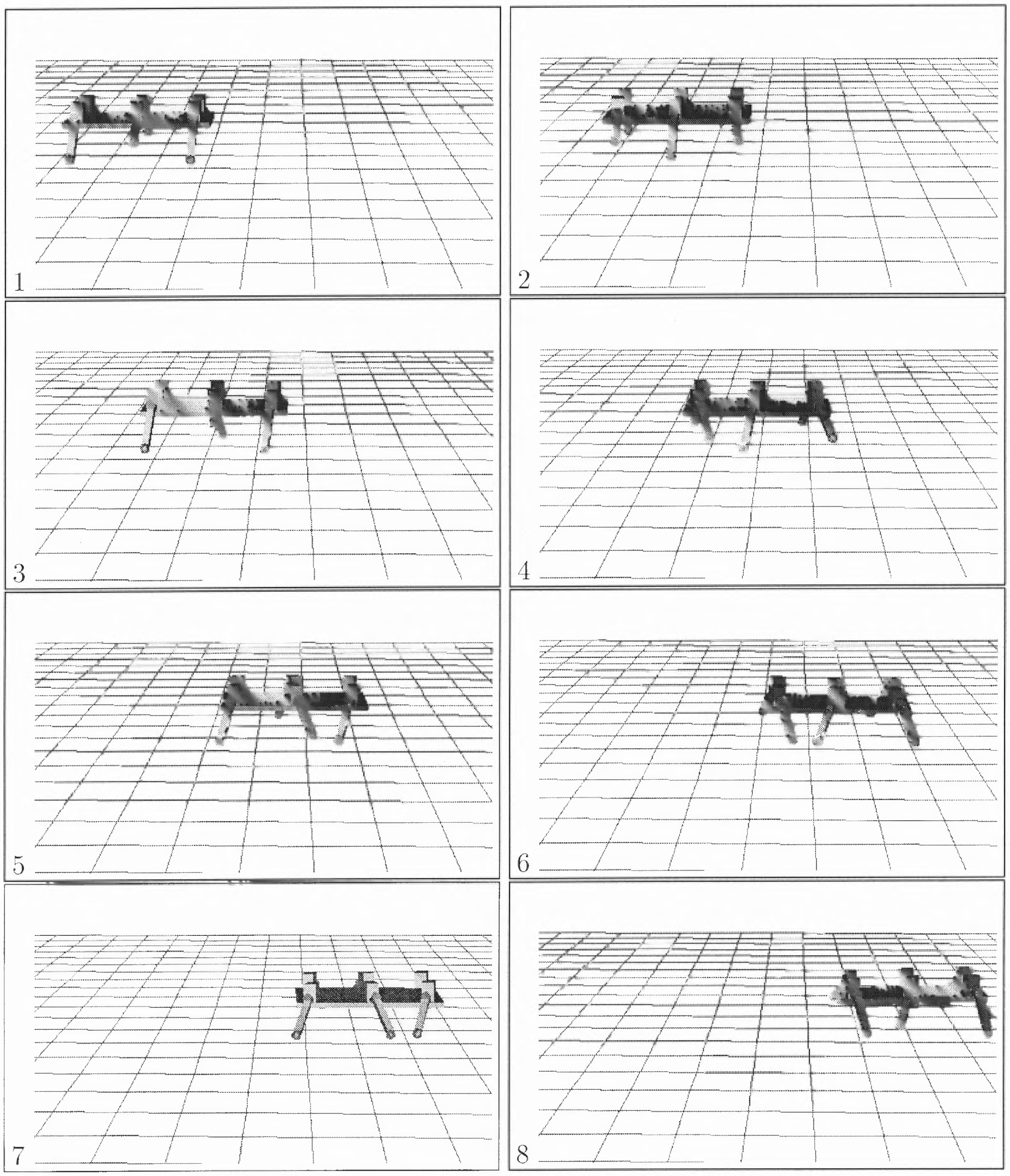


Figure A.3: A statically stable walker is examined in this example. For each joint a single sinusoidal pattern is applied. The upper joints have a $\frac{\pi}{4}$ phase shift relative to the corresponding lower ones. With these patterns each foot goes through circular paths. The two tripods defined by anterior-right, central-left, posterior-right and anterior-left, central-right, posterior-left legs have a π phase shift.

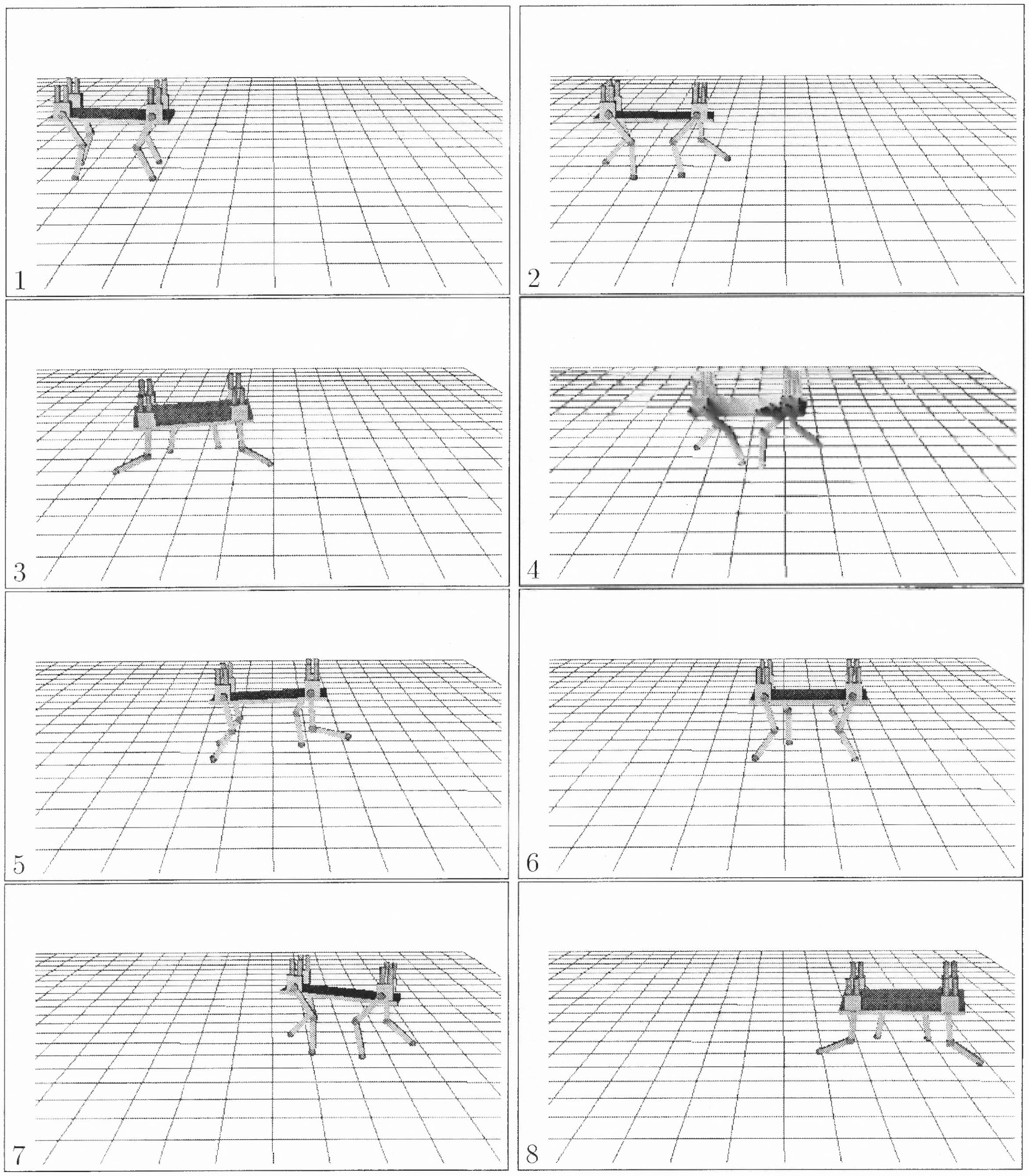


Figure A.4: This simulation shows the quadruped walking robot on a flat terrain robot under the action of a fairly stable controller obtained through the GA. As it it visible in the snapshots the body of the walker goes through wide oscillations during locomotion.

The examples shown in the previous pages illustrate the capabilities of the dynamic simulator. The terrain definition feature is demonstrated in the toy car simulation where a ramp is defined through a matrix of surface heights which is scaled to fill the whole workspace. The heights in between given values are bilinearly interpolated. The wheels of the toy car are attached to the chassis through passive (i.e. not controlled) revolute joints.

In the worm example shows how to apply a simple control strategy consisting of a single sinusoidal pattern with constant phase shift between joints. The sinusoid travels through the body of the worm producing forward motion.

For the third example a walking robot has been chosen to show the applicability of the software to robotic walkers. In this case an hexapod is chosen, as the control of this statically stable system is rather simple. A single sinusoidal pattern is applied here, similarly to the previous example. The the phase shift between upper and lower joints is $\frac{\pi}{4}$, whereas two tripods move in opposition of phase. The tripods consist of the anterior-right, central-left, posterior-right and anterior-left, central-right, posterior-left legs respectively.

Name	DOFs	Time (sec)	Slowdown	Friction	Terrain
car	4	3.5	1.8	1.0	Ramp
worm	4	15.0	1.0	2.5	Flat
hexapod	12	10.0	8.6	1.0	Flat
quadruped	8	10.0	9.3	2.5	Flat

Table A.1: Summary of simulation examples. DOFs is the number of degrees of freedom of the system; Time is the total simulated time; the Slowdown value is the ratio between actual time required to run the simulation and the total simulated time.

The last simulation shows the quadruped walker moving on a flat surface under the action of a stable controller evolved with the genetic algorithm. Note that even for this fairly stable control pattern the body of the robot swings from side to side considerably.

Table A.1 summarizes the parameters of the simulation examples. Note that the Slowdown ratio does not always scale up with the DOFs, as the integration time-step depends on the external forces acting on the system as well as on the velocity of the bodies being simulated.

APPENDIX B

DYNAMIC SIMULATION ALGORITHM

```
/***********************************\
         INITIALIZATION
\***********************************/
for all bodies
    Initialize-Body(body)
end

for all joints
    Initialize-Joint(joint)
end

/***********************************\
        JOINT CONSTRAINTS
\***********************************/
for all joints
    PP1 = Pivot-Position(body1(joint))
    PP2 = Pivot-Position(body2(joint))
    DP  = PP1 - PP2
    PV1 = Pivot-Velocity(body1(joint))
    PV2 = Pivot-Velocity(body2(joint))
    DV  = PV1 - PV2
    JP  = JOINT-PENALTY(DP,DV)
    Add-To-Force(body1(joint),JP)
    Add-To-Torque(body2(joint),JP,PP1)
    Add-To-Force(body1(joint),-JP)
    Add-To-Torque(body2(joint),-JP,PP2)
end

/***********************************\
        GROUND COLLISIONS
\***********************************/
for all bodies
    BB = Bounding-Box(body)
    if (Impact(BB)) then
         CP = Contact-Position(body)
         CV = Contact-Velocity(body)
         ND = Normal-Direction(CP)
         PD = Penetration-Distance(CP,ND)
         GP = GROUND-PENALTY(PD,CV)
        if (Static Friction) then
            FP = STATIC-FRICTION-PENALTY(CV,ND)
        else
            FP = DYNAMIC-FRICTION-PENALTY(CV,ND)
        end
    end
    Add-To-Force(body,GP+FP)
    Add-To-Torque(body,GP+FP,CP)
end

/************************************\
            GRAVITATION
\************************************/
for all bodies
    G = Gravity(body)
    Add-To-Force(body,G)
    Add-To-Torque(body,G,Position(body))
end

/************************************\
          ODE INTEGRATION
\************************************/
if (Adams-Bashforth)
    for all bodies
        Save-Previous-Step(Position(body),Velocity(body))
        Adams-Bashforth-Integrate(body,Nsteps)
    end
else
    MA = Max-Acceleration()
    TS = Time-Step(MA)
    for all bodies
        Euler-Integrate(body,TS)
    end
end
```

APPENDIX C

WALKING ROBOT SPECIFICATIONS

General

Description	Value	Unit
Total weight	13.8	Kg
Body dimensions	250x600x20	mm
DOFs	8	

Motors

Description	Value	Unit
Type	DC	
Voltage range	±30	V
Max torque	0.35	Nm
Gearbox reduction	50:1	

Legs

Description	Value	Unit
Upper weigth	0.5	Kg
Lower weigth	0.3	Kg
Upper dimensions	16x35x150	mm
Lower dimensions	21x35x162	mm
Joint type	Revolute	

Encoders

Description	Value	Unit
Type	Optical	
Resolution	0.72	deg

REFERENCES

[1] R. B. McGhee and G. I. Iswandhi, "Adaptive locomotion of a multilegged robot over rough terrain," *IEEE Trasactions on Systems, Man and Cybernetics*, vol. 9, no. 4, 1979.

[2] F. Ozguner, S. J. Tsai, and R. B. McGhee, "An approach to the use of terrain-preview information in rough-terrain locomotion by a hexapod walking machine," *The International Journal of Robotics Research*, vol. 3, no. 2, 1984.

[3] E. I. Kugushev and V. S. Jaroshevskij, "Problems of selecting a gait for an integrated locomotion robot," in *Proc. of the 4th IJCAI*, (Tiblisi, USSR), p-p. 789–793, 1975.

[4] S. Hirose, "A study of design and control of a quadruped walking vehicle," *The International Journal of Robotics Research*, vol. 3, no. 2, 1984.

[5] M. H. Raibert, "Running with symmetry," *The International Journal of Robotics Research*, vol. 5, no. 4, 1986.

[6] J. K. Kearney and S. Hansen, "Generalizing the hop: Object-level programming for legged motion," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[7] K. Yoneda and S. Hirose, "Dynamic and static fusion gait of a quadruped walking vehicle on a winding path," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[8] B. Min and Z. Bien, "Neural computation for adaptive gait control of the quadruped over rough terrain," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[9] H. J. Chiel, R. D. Beer, R. D. Quinn, and K. S. Espenschied, "Robustness of a distributed neural network controller for locomotion in a hexapod robot," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[10] M. A. Lewis, A. H. Fagg, and A. Solidum, "Genetic programming approach to the construction of a neural network for control of a walking robot," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), 1992.

[11] S. Grillner, *Handbook of Physiology*, vol. II, sec. 1, part 2, pp. 1179–1236. V.B. Brooks Ed., 1981.

[12] L. Sheh and R. E. Poppele, "Kinematic analysis of cat hindlimb stepping," *Journal of Neurophysiology*, vol. 74, no. 6, 1995.

[13] K. Pearson, "The control of walking," *Scientific American*, vol. 235, pp. 72–86, 1976.

[14] R. Shadmehr, F. A. Messa-Ivaldi, and E. Bizzi, "Postural force fields of the human arm and their role in generating multijoint movements," *The Journal of Neuroscience*, vol. 13, no. 1, pp. 45–62, 1993.

[15] M. Buehler, A. Cocosco, K. Yamazaki, and R. Battaglia, "Stable open loop walking in quadruped robots with stick legs," in *IEEE International Conference on Robotics and Automation*, (Detroit, MI), 1999.

[16] A. H. Cohen, S. Rossignol, and S. Grillner, *Neural Control of Rhythmic Movements in Vertebrates.* New York, NY: John Wiley & Sons, 1988.

[17] R. D. Beer, H. J. Chiel, R. D. Quinn, K. Espenschied, and P. Larsson, "A distributed neural network for hexapod robot locomotion," *Neural Computation*, vol. 4, pp. 356–365, 1992.

[18] K. G. Pearson, C. R. Fourtner, and R. K. Wong, "Nervous control of walking in the cockroach," in *Control of Posture and Locomotion*, (New York, NY), R. B. Stein, K. G. Pearson, R. S. Smith and J. B. Redford, Eds., 1973.

[19] G. Parker and G. Rawlings, "Learning gaits for the stiquito," in *Proceedings of the 8th International Conference on Avanced Robotics (ICAR'97)*, pp. 285–290, 1997.

[20] M. H. Raibert and I. E. Sutherland, "Machines that walk," *Scientific American*, vol. 248, no. 1, 1983.

[21] G. Grasso and M. Recce, "Towards genetically evolved dynamic control for quadruped locomotion," *Connection Science*, vol. 11, no. 3&4, pp. 317–330, 1999.

[22] D. C. Shapiro, R. F. Zernicke, G. R. J., and D. J. D., "Evidence for generalized motor programs using gait pattern analysis," *Journal of Motor Behavior*, vol. 13, pp. 33–47, 1981.

[23] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Englewood Cliffs, NJ: Prentice Hall, 1995.

[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge, MA: The MIT Press, McGraw Hill, 1990.

[25] R. Bellman, *Dynamic Programming.* Princeton, NJ: Princeton University Press, 1957.

[26] J. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection.* Cambridge, MA: MIT Press, 1992.

[27] S. Baluja, "An empirical comparison of seven iterative and evolutionary function optimization heuristics," tech. rep., School of Computer Science, CMU, Pittsburgh, PA, 1994.

[28] R. Beer and J. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 1, pp. 91–122, 1992.

[29] K. A. Sigvardt and T. L. Williams, "Effects of local oscillator frequency on intersegmental coordination in the lamprey locomotor cpg: Theory and experiment," *Journal of Neurophysiology*, vol. 76, no. 5, 1996.

[30] B. Berns, W. Ilg, J. A. Deck, J. Albiez, and R. Dillmann, "Mechanical construction and computer architecture of the four-legged walking machine bisam," *IEEE/ASME Transations on Mechatronics*, vol. 4, no. 1, pp. 32–38, 1999.

[31] K. S. Anderson, "An efficient formulation for the modeling of general multi-flexible-body constrained systems," *International Journal of Solids and Structures*, vol. 30, no. 7, pp. 921–945, 1993.

[32] D. Baraff and A. Witkin, "Dynamic simulation of non-penetrating flexible bodies," *Computer Graphics*, vol. 26, no. 2, pp. 303–308, 1992.

[33] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *International Journal of Robotics Research*, vol. 2, no. 1, pp. 13–30, 1983.

[34] D. Baraff, *Dynamic Simulation of Non-Penetrating Rigid Bodies.* PhD thesis, Cornell University, Ithaca, NY, 1992.

[35] B. Mirtich, *Impulse-based Dynamic Simulation of Rigid Body Systems.* PhD thesis, University of California at Berkeley, Berkeley, CA, 1996.

[36] B. Mirtich and J. Canny, "Impulse-based dynamic simulation," in *Symposium on Interactive 3D Graphics*, (New York), ACM Press, 1995.

[37] K. Lilly, *Efficient Dynamic Simulation of Robotic Mechanisms.* Norwell, MA: Kluwer Academic Publishers, 1993.

[38] M. Xie, *Flexible Multibody System Dynamics*, ch. 8, pp. 190–203. Washington, D.C.: Taylor & Francis, 1994.

[39] A. A. Shabana, *Computational Dynamics*, ch. 4, pp. 299–304. New York, NY: John Wiley & Sons Inc., 1994.

[40] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Computer Graphics Proceedings, Annual Conference Series*, 1996.

[41] A. Watt and M. Watt, *Advanced Animation and Rendering Techniques, Theory and Practice.* New York, NY: Addison Wesley Longman Inc., ACM Press, 1998.

[42] W. Rytter, "On efficient parallel computation for some dynamic programming problems," *Theoretical Computer Science*, vol. 59, pp. 297–307, 1988.

[43] N. C. Heglund, C. R. Taylor, and T. A. McMahon, "Scaling stride frequency and gait to animal size: Mice to horses," *Science*, vol. 186, pp. 1112–1113, 1974.

[44] J. Calvin, "Adaptive monte carlo global search with bounded memory," tech. rep., New Jersey Institute of Technology, Newark, New Jersey, 1996.

[45] J. Calvin, "Average performance of a class of adaptive algorithms for global optimization," *Annals of Applied Probability*, vol. 7, pp. 711–730, 1997.