POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in Ingegneria Aeronautica
Corso di Laurea in Ingegneria Spaziale

Decentralized control system for a hexapod robot
by using neural networks

Relatore: Prof. Mauro MASSARI

Tesi di laurea di:
Fabio ORTALLI Matr. 725152
Alessandro SPALLA Matr. 721871

Anno Accademico 2009-2010

Contents

List of Figures

List of Tables

Abstract

The reproduction of what the nature offers us has been for long time one of the most exciting challenges to scientists and engineers. The thrust to copy the animals and their behavior is justified by the fact that biological systems tend to have a better efficiency than those found in their artificial counterparts. This concept can also be applied in space exploration, replacing with legged robots inspired by the animals, the classic rovers. The latters, in fact, show mobility difficulties, particularly on rough terrain. Because of this the choise to design a controller based on observed natural behavior. The main source of inspiration is the stick insect, because of its high level of mobility and of the ease with which it can be studied. This approach led to the creation of a decentralized control system that has proven better performance and flexibility than those of classical control algorithms. A physical platform to test the designed controller has been finally built.

Key words: neural networks, hexapod robot, walking, advanced control, gait, biomimicry.

Sommario

La riproduzione di ciò che la natura ci offre è stato per molto tempo una delle più avvincenti sfide lanciate da scienziati e ingegneri. La spinta a copiare gli animali e i loro comportamenti è giustificata dal fatto che i sistemi biologici tendono ad avere un'efficienza maggiore di quella riscontrabile nelle loro controparti artificiali. Questo concetto può essere applicato anche nell'ambito dell'esplorazione spaziale, sostituendo con robot muniti di zampe ispirati al mondo animale i classici robot a ruote. Questi ultimi, infatti, presentano difficoltà di mobilità, particolarmente su terreni sconnessi. Per questo si è scelto di progettare un controllore basato sul comportamento osservato in natura. La principale fonte di ispirazione è l'insetto stecco, vista la sua ottima mobilità e la facilità con cui può essere studiato. Tale approccio ha portato alla creazione di un sistema di controllo decentralizzato che ha dimostrato prestazioni e flessibilità superiori a quelle dei classici algoritmi di comando. Infine è stata costruita una piattaforma fisica per testare il controllore progettato.

Parole chiave: reti neurali, robot esapode, camminata, controllo avanzato, gait, biomimesi.

Chapter 1

Introduction

This thesis aims to improve the understanding of walking robots for space exploration both creating a control system and realizing an hardware platform to test theoretical results experimentally as part of the development of the Neural Ento-Mechanic System (NEMeSys) project.

Interest on walking robots depends on current trends in space exploration: no human missions on celestial bodies has been planned for the immediate future so this essential part of scientific research relies on semi-autonomous probes only.

1.1 Walking systems

Since the start of the space race only few semi-autonomous rovers have been actually employed and all of them mounted wheels to propel themselves on the ground. This choice can be easily justified by examining their advantages mainly as for legs. A wheeled system is almost always simpler, cheaper and more reliable than a legged one, and it also shows continuous stability, that can be achieved without any control strategy.

The problem is to understand when legs become preferable to wheels. There are two main reasons supporting the first option: legs don't require a continuous area of solid ground for moving and navigation is not constrained.

Investigating the first reason it's possible to say that ground always offers a non-continuous support and the evaluation of motion depends on the relationship between gap distance and wheel size. A wheel is able to cross horizontal discontinuities as larger as its radius, but it can climb only much smaller vertical ones. The other main problem is that a large gap/radius relation drives to a very irregular and energetically inefficient body motion. Furthermore wheel size can't be changed during the motion and this reduces vehicle flexibility. For a leg instead, overcoming either horizontal or vertical gap it's conceptually the same thing and the quality of motion isn't so strictly related to discontinuities dimensions. A leg has also greater flexibility because each step can be adapted to the current gap size thus optimizing energetic efficiency.

The second reason involves the total controlled Degree of Freedom (DoF)s of each system. A wheeled vehicle moving on ground is a non-holonomic system because it controls fewer variables than those defining

its position. The problem descends from the single wheel that is controlled only by two variables, angular velocity and wheel direction. So a wheeled vehicle is able to explore the whole space, but only with a high number of trajectory corrections and an advanced navigation planning. A typical example of such a problem is a car parking maneuver. In opposition, a jointed leg is able to reach every allowed position directly, without any kind of trajectory adjustment. This drives to a simpler and more flexible motion planning.

All the considerations used on wheels can be extended to other kinds of mechanisms to transmit motion on ground, particularly to all the nonholonomic systems such as whegs or oscillators, because their behaviors can be easily reduced to the wheels' ones.

After explaining the advantages in legs usage the attention can be focused on how this kind of systems works. From this point of view the most important issues are stability, complexity, and control structure.

A legged system is not stable per se, like a wheeled one, because its stability depends fundamentally on how many legs touch simultaneously the ground. If during the walk, three or more legs are always in contact, the motion is a mere juxtaposition of equilibrium state and the system is called statically stable. This means the stability is not time dependant and a global stability controller is not required. When only two legs or fewer are in contact at the same time the system requires continuous adjustments to posture and inertia to maintain its stability since it becomes dynamically stable. It's obvious that the former ones are safer and more robust than the latter ones but they require at least four legs and a coordination between all the legs in contact with the ground. On the other hand, dynamically stable systems are more suitable for fast locomotion.

The single jointed leg, as said above, is an holonomic system because the number of controlled variables is the same of the total DoFs in its task space. Extending the problem to a complete robot and considering only static-stable cases, the systems become redundant, because the task is the body motion control producing a 6-dimensional task space, but the control variables, i.e. the joints angles, are 9 or more. This property is a great advantage relating to nonholonomic systems for the previously shown reasons, but it needs also a much more complex controller that is the main disadvantage of legged systems.

The structure of any legged locomotion controllers can be analyzed at three different levels: body trajectory, inter-leg coordination and single leg motion. The first typically depends on the direction and the speed required for the vehicle and it's normally a higher-level command. Once fixed where the body should be, it's possible to calculate the legs' move-

ment and hence joint positioning. However, each leg must take into account the motion of the other one, otherwise the robot could fall or walk inefficiently. To avoid this a coordination among the legs is required. The main advantage of such a structure is the possibility to design each level independently because they are only weakly coupled. This allows the use of a simpler, decentralized control system, since it requires a less performing CPU than a global controller.

From the spatial point of view a global evaluation of the legged vehicles can be done. The exploration of a space environment requires a very robust behavior, so a statically stable robot with an high number of legs is preferable. The complexity of an high DoFs system is balanced by its adaptability to an unknown environment and its flexibility to achieve lots of different tasks. It also makes it more fault tolerant, since the robot can lose a leg without losing its capability for walking. The possibilities offered by using a decentralized control matches the requirements of the low-performances space-certified CPUs.

At the start of the NEMeSys project a trade-off among some multi-legged robots was done, driving to choose the six-legged one. The main reason supporting this choice is the double symmetry of such a system: they are both reflectional and translational symmetric with respect to the walking direction. This simplifies the problem allowing to design only one leg and obtaining the others by symmetry. Another reason is the high number of existing works on these systems also from non-engineering fields such as entomology or neurobiology.

1.2 Biomimicry

In engineering, as in other scientific fields, it's usual to evaluate existing natural models and emulate them on the purpose of obtaining the same results. This process is called biomimicry from the ancient Greek *βίος*, meaning life, and *μίμησις*, meaning imitation.

Nature offers lots of perfectly walking systems and, at this level of development of our science, they work all better than their artificial counterpart. This means that some design ideas can be helpful in the pursuit for building truly autonomous robots.

Obviously the best walking hexapod are insects, the most studied of them are stick insects, particularly the specie *Carausius Morosus*, the 'common', 'Indian' or 'laboratory' stick insect (see figure 1.1). One of the reasons to choose this stick insect as a model is its researcher-friendly morphology. It has a long, straight body allowing to strap it onto different devices for researching it's walking. The stick insect's legs aren't hidden

Figure 1.1: A stick insect *Carausius Morosus*.

by other body parts as in a lot of other insects, allowing to observe them easily. Finally breeding has proved to be fairly simple.

Once that the target of the study is chosen the main question becomes which is the way of learning from insects. The answer depends on the approach to insect walking problem. There are two ways of modeling hexapod locomotion on the basis of biological findings. The former one is to assemble the known components of the nervous and musculoskeletal systems, just to build an incrementally realistic model of a moving insect. The latter one, takes the complementary approach of assembling logical components to reproduce a model of the system properties of the behaving animal.

This work relies extensively on the second approach for the control strategy concept: in previous works (see Cruse et al. [1]) kinematics and dynamics motion parameters were measured, then the results were interpreted by the formulation of rules that were assumed to describe the properties of the underlying control system. The control system of NEMeSys has been developed starting from these rules and adapting them to fulfill the other project requirements.

Another important biomimetic feature involves the type of controller used to implement behavioral rules. Insects show the ability to perform an accurate walking control using a very simple neural network that has a little computational capability compared to a computer. This means it's possible to decrease the computational cost of the problem adopting instead of classic control systems, a biological-like ANN. Neural networks have a decentralized structure, perfect to represent the problem of walk-

ing control that doesn't need a global supervision and can be divided in fairly independent subtasks. Therefore they can show a nonlinear behavior, very useful to simulate the leg motion that is an intrinsically nonlinear physical phenomenon.

1.3 Goals

After an overview of the known walking systems and their biomimetical features, it's possible to identify the higher-level goals of this work. The former of them is the realization of a ANN-based walking controller able to generate the angular reference signal for each of the joints that manage the walk. This goal can be achieved by fulfilling some tasks:

- the fundamental signal is able to produce a stereotypic leg motion. This means that the system is able to walk also with no external input improving fault tolerance;

- the reference signal can be changed depending on sensory feedback. This requirement is necessary to guarantee the motion adaptability to unpredictable external conditions such as: rough terrain, changing slope and obstacles;

- the reference signal can be changed depending on the high-level path control. This allows the possibility to vary walking speed, walking direction or body attitude;

The other main goal is to design and build a walking robot on which to test the control system. Starting from the results of the former NEMeSys works it's possible to identify the tasks as follows:

- the mechanical project need to be redone both to reduce the global mass and to improve the mechanical efficiency of the robot;

- the electronics is already complete, but needs some improvements. The main of them is the introduction of a force sensors to evaluate the ground contact, in the place of the switch sensors previously mounted. Other changes will be done in order to simplify the hardware and to reduce the global weight;

- the software interface exists but has to be changed in order to be adapted to the new control architecture.

1.4 State of the art

To better understand how certain choices have been taken, as described in the next chapters, it can be useful to look at some important examples of legged locomotion in the robotics field. Because this work aims either to build a robot or to project its control system, the overview shown below involves both of these features: each example is an existing walking robot having a particular kind of control structure.

1.4.1 Mobot Lab robots

The Mobot Lab of the MIT developed some interesting space-oriented robots in the first half of the '90s (see figure 1.2). The first of them was Genghis an hexapod with only 2 DoFSs for each leg, but very compact with a length of 35 cm and a weight of 1 kg. The sensors allow to measure high-level path parameters like body orientation or obstacles contact. The controller is mounted on-board.

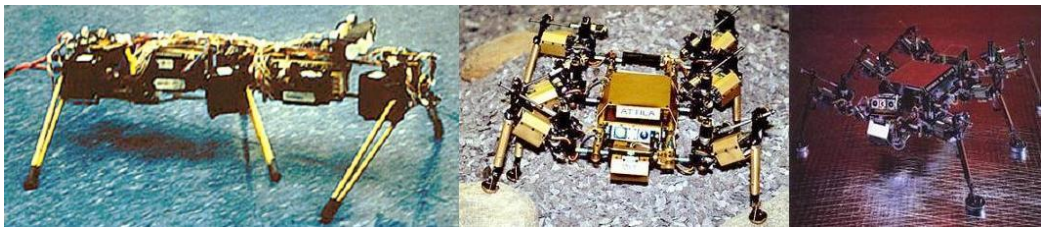


Figure 1.2: Mobot Lab robots. From the left: Genghis, Attila and Hannibal.

The most interesting feature of this robot is its control system: it relies on a network of elementary agents called Finite State Machines or FSM [2]. Each of them managed only a very simple task, depending on few inputs from sensors. Although this isn't an actual neural network it can be considered like a precursor of NEMeSys, since the main ideas behind the projects are the same:

- modularity: the controller is decentralized on various blocks, each of them working only on a limited task, almost independently from the others;
- incremental growth: the project starts fulfilling only the simplest tasks, adding more complex behaviors step-by-step.

After Genghis a couple of more complex robots were developed: Attila and Hannibal [3]. They had 3-DoFs legs and were smaller than their predecessor. The main advantage of these systems was the capability to maintain a stable configuration also with high slopes, thanks to an additional DoF in the body.

Figure 1.3: Randall Beer's robots. From the left: Robot I and Robot II.

1.4.2 Randall Beer's robots

The guidelines of the earlier development of the NEMeSys project had been taken from the work of professor Randall D. Beer [4]. He and his team worked extensively on the theory of neural networks-based controllers and also realized some interesting walking platform.

Their first work is Robot I a hexapod with 2-DoFs legs. The control system is a decentralized neural network with a single net controlling each leg. The inter-leg coordination depends on the mutual influences among the controllers of each single leg, with the aid of a global supervisor. Another interesting feature of Robot I is the design approach: the control parameters has been selected via a large number of simulation, using biological-like evolutionary algorithms.

Robot II, the second model, is quite different: it relies on six 3-DoFs legs and has a completely different controller based on reflexes. It means that walking behaviors emerges only from the interaction between the robot and the environment and is not generated autonomously by the platform.

Robot I and II aren't designed for space exploration but are among the best examples of working legged vehicles that are controlled by ANNs.

1.4.3 SCORPION

The SCORPION is an eight-legged walking robot sponsored by DARPA and NASA and realized by Universität Bremen [5](see figure 1.4). The vehicle is quite compact: mounting sensors, a communication system and batteries, it reaches a global weight of 11.5 kg and a length of 65 cm. Each leg has 3 DoFs actuated with DC motors and features a spring element too.

The control of this robot is based on the models of two biological control primitives: central pattern generators and reflexes. The model is controlled by a higher central control level by means of Rhythmic Motion Patterns (RMPs) that control path and Posture Control Primitives (PCPs) that

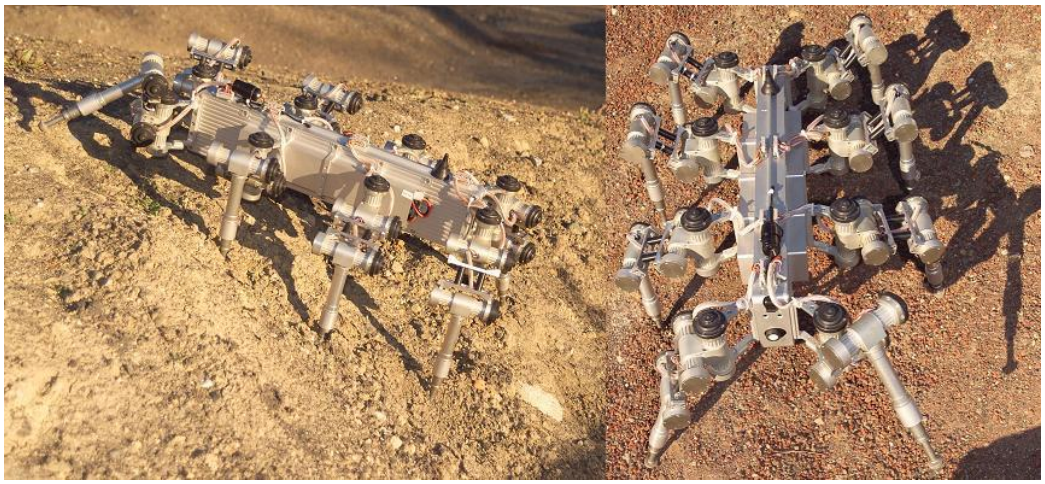


Figure 1.4: The SCORPION robot.

control body attitude. With this approach there exists the possibility of omni-directional walking and smooth and fast crossing between different motion patterns. Moreover the posture and the speed of the robot can be changed while walking.

The robot was successfully tested in rough, sandy and rocky ground. This is a space oriented project: the vehicle is completely autonomous and all the electronics is hidden to protect it by hostile environment. Its only problem is the very high power consumption.

1.4.4 Walknet

Walknet is a controller developed by the Department of Biological Cybernetics and Theoretical Biology of the University of Bielefeld led by professor Holk Cruse [6]. He tries to resume the walking behavior of the stick insect *Carausius Morosus* by means of a limited number of considerations derived from his experiments on living animals:

- the motion of a single leg can be divided into two different phases, the power stroke and the return stroke, that can be controlled independently;
- transition between phases is regulated by two parameters only: the ground contact and the posterior extreme position;
- coordination among legs can be summed up into six rules, by which each leg influences the others.

On the basis of such an hypothesis, it's possible to realize a decentralized ANN that controls a 6-legged walk in a very simple, but flexible way.

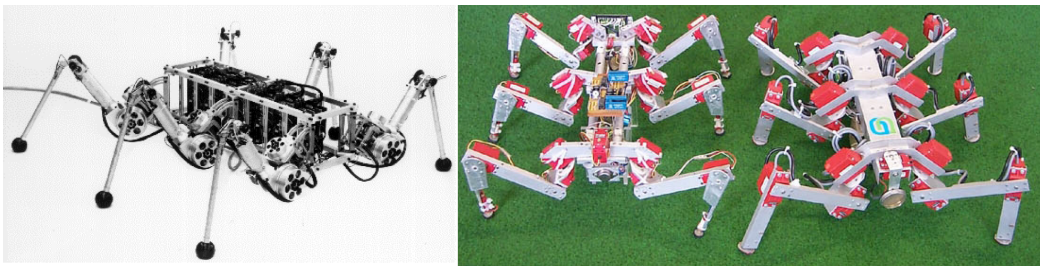


Figure 1.5: Walknet-controlled robots. From the left to the right: TUM, Tarry I and Tarry II

Walknet has been tested in its full version only on two robots, TUM, realized by Dr. Friedrich Pfeiffer at Technische Universität München [7], and the Tarry family, built by the Department of Engineering Mechanics at the University of Duisberg [8][9] (see figure 1.5). On the other hand, the coordination rules have been used in lots of walking platforms, such as Robot I and II, that prove their effectiveness.

Because Walknet has been a guideline for the present work, lots of these features will be discussed later.

1.5 Thesis contributions

The main contributions of this thesis can be divided in both theoretical and practical features. The former ones involve the realization of a decentralized control system able to produce the legs' trajectories (chapters from 2 to 6). The latter ones the design of a legged platform on which test the controller (chapter 7).

This thesis aims to produce a hexapod walking control architecture that incorporates elements of stick insect locomotion that would allow it to outperform current implementations. To obtain such a results ANNs can be a powerful instrument. For this reason a good understand of how this kind of systems works and the knowledge of their design process are necessary. In this work a criterion that allows to chose the most effective ANN to fulfill a given problem has been identified (chapter 2).

The structure of the controller has to be defined, starting from biological findings, by recognizing all the behaviors to reproduce and by assigning to each of them the correspondent control module. For each module the required inputs and outputs and the interactions with other components of the controller have to be identified too (chapter 3).

The decentralized approach allows to study the control of a single leg independently from the rest of the robot. This problem has been solved

by using a innovative solutions, e.g. dynamic neural networks, positive feedback or nonlinear approximation. These ones has been combined with classic control systems, in order to obtain the best results both in terms of performances and reliability (chapter 4)

The single legs motion must be coordinated to obtain the desired global motion. This works want to find a controller for the coordination that ensure the stability of the robot during the entire motion with a structure as simple as possible. It must also guarantee a great flexibility by allowing to change the motion over a wide range of possibilities or to introduce new behaviors, without any changes in the coordination system (chapter 5).

The complete control system needs to be tested. At first it can be evaluated on a multibody model of a legged system in a dynamic simulation environment in order to identify possible problems or lacks into the previous design process (chapter 6).

The last major purpose of this thesis is the design of a legged robot. It has to be a functional 18-DOF hexapod capable of straight-line and curve walking able to respond to external forces through measurements taken from foot-mounted force sensors and joint-mounted angular displacement sensors. Its weight has to be kept down and the dimensions must be limited. Moreover it should has autonomous power and the capacity for autonomous control (chapter 7).

Chapter 2

Neural networks

Biomimetic findings shown in section 1.2 imply that the problem of walking control can be solved in a better way by using an artificial version of the insects' neural network instead of a classic controller. To better understand which is the most adapt network to fulfil a given requirement, it's necessary to know which kind of networks exist, how each type works and how to design it. In this chapter, after an overview of such systems (sections 2.1 to 2.3), the criteria to choice the correct type of neural network will be evaluated (section 2.4) and the calculation methods for its parameters will be explained (sections 2.6).

2.1 Overview

An ANN is a mathematical model that tries to simulate the structure and functional aspects of biological neural networks. It consists of group of units, called artificial neurons, mutually joined by weighted connections, called synapses. Information can be processed using a connectionist approach to computation: this means that, in opposition to a typical computer, tasks aren't accomplished solving a deterministic sequence of operations, but with a distributed, parallel and local processing involving all the units.

These characteristics result in some interesting advantages. Such a structure allows to manage large amounts of data with great accuracy, thus approximating complex mappings. They're fairly independent to every assumption on data's distribution and interaction among components. Fundamentally they are sophisticated statistical systems with a good robustness to noisy, incomplete or totally missing inputs; if some units work incorrectly, the network could suffer a degradation of the level of its performances, but almost never stops its work. They are able to generalize by giving an output also when unknown inputs appear. It's possible to implement them in a parallel hardware optimizing their computational efficiency.

Opposite to advantages, one must note that the model produced by ANNs, although very efficient, can't be explained with an analytical approach: results must be taken as they come and neural networks have to be treated like black boxes. It's not possible to understand how certain inputs cause certain outputs so the only way to obtain an efficient ANN is to start

from a set of well-chosen statistical data. If the problem to solve is complex, the number of data needed to a correct ANN design is very high and the calculation of the network parameters become very computer intensive. Theorems or models that permit to define the best network structure are not available yet, so the final result is obtained with heuristic methods that heavily rely on the experience of their creator.

In this work all the networks aren't realized by a dedicated hardware, but with a software working on a classic computer. This strategy reduces the computational efficiency of the neural approach, but allows to use existing components and avoids to waste resources on problems that are to far from walking control.

2.2 Structure

An ANN consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections. It is called network because the output of each unit is defined as a composition of functions depending on other units. All of the existing ANNs show the same structure, composed by:

- a set of processing units called neurons;

- connections between the units. Generally each connection is defined by a weight w_{ij} which determines the effect which the signal of unit i has on unit j. For positive w_{ij} the contribution is considered an excitation and for negative w_{ij} an inhibition.

Every neuron is characterized by four different features:

- a state of activation y_i, which is equivalent to the output of the unit;

- a propagation rule, which determines the effective input S_i from its external inputs;

- an activation function Φ, which determines the new level of activation based on the effective input S_i and the current activation y_i;

- an activation offset or bias θ_i.

The whole process of outputs generation by the elaboration of inputs is called combination.

2.3 Classification

On the basis of these fundamental assumptions a huge number of different type of ANNs has been proposed in the past, so that a brief classification is necessary to elaborate the discriminating criteria that enable to find the best network that fulfills the requirements of this work.

2.3.1 Propagation rule

The first criterion depends on the propagation rules and the way to calculate the effective input starting from external inputs and weights: it can be a linear combination (Linear nodes), a polynomial combination (Sigma-pi nodes) or a nonlinear boolean function (Cubic nodes).

2.3.2 Activation function

The second classification can be traced on the basis of different types of activation function. Basically they can be divided in two great categories: the step-like functions and the radial basis functions. The former ones have been the first to be used and are composed by generalized forms of the step-function. The most important functions of this class are:

- step function: it's very easy to be implemented, but it's not invertible and it's not up to approximate smooth functions;

- linear ramp: it's simple, but has a linear zone that can imitate continuous functions. It's not invertible;

- sigmoid: it's invertible and continuously differentiable. It can both approximate functions and take fuzzy decisions, but it's more expensive in terms of computational cost and its outputs are limited to positive values;

- hyperbolic tangent: has the same advantages of the sigmoid and can also produce both positive and negative outputs. It's difficult to be implemented.

All of these function are very important because show a behavior very close to the biological neurons' one. In subsection 2.3.4 is shown a very useful applications of such a type of functions.

The radial basis functions are inspired to the biological neurons of the visual cortex: they produce a significant response only close to a specific point in the space of inputs called center and the amplitude can be modulated by a scale factor. A typical example of these functions are the gaussian, the multiquadric, the polyharmonic spline and the thin plate spline.

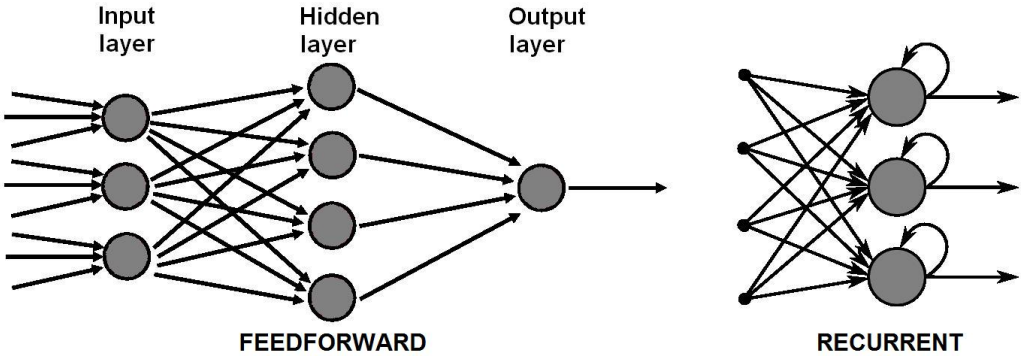


Figure 2.1: Two typical topologies for an ANN.

They are mainly used to approximate given functions particularly in time series prediction and control of nonlinear systems with a sufficiently simple chaotic behavior and also in 3D reconstruction in computer graphics.

A last activation function has to be considered, the linear activation function. In this case the neural state is the same as the effective input, which allows to approximate non limited behaviors too.

2.3.3 Topology

Another classification criterion focuses on the pattern of connections between the units and the propagation of data. As for this pattern of connections, the main distinction that can be made is between feed-forward networks and recurrent networks.

Feed-forward neural networks were the first and arguably the simplest ANNs devised. In these networks, the information moves strictly in one direction only, that is forward, from the input nodes, to the output nodes. The data processing can extend over multiple layers of units, but no cycles or loops are present. An example of a three layers feed-forward neural network is shown on the left side of figure 2.1.

A Recurrent Neural Network (RNN) is a kind of ANN in which connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. RNNs must be approached in a way different from feed-forward networks, typically using dynamical systems theory to model and analyze them. An example of a two layers RNN is shown on the right side of figure 2.1.

Classical examples of feed-forward networks are the Perceptron and the Adaline. Examples of recurrent networks are Elman networks, Hopfield networks and nonlinear autoregressive networks and the Continous Time Recurrent Neural Network (CTRNN).

2.3.4 Static and dynamic networks

In the present work the most important distinction among ANNs is based on their capability to model time-dependant phenomena. From this point of view static ANNs and dynamic ANNs can be identified.

A static network is an ANN that produces a specific output depending only on the inputs given in the current instant. Systems showing this behavior are called reactive agents, because they produce a direct, time-independent reaction.

For this reason to obtain the desired network is sufficient to know a set of inputs and the corresponding desired set of outputs. Starting from these sample data, by means of a training process, is possible to calculate the correct set of parameters defining the network univocally. The adaptability of a such a model allows, if the sample has been correctly chosen, to produce acceptable outputs also when inputs are different from ones thought of in the initial design.

Almost all the static networks are composed by neurons that, with little variations, are versions of the first neural unit proposed by McCulloch and Pitts in 1943 [10]: the perceptron. A typical perceptron-like neuron uses a type of composition called nonlinear weighted sum, that can be mathematically summarized as:

$$y_i = \Phi(S_i) \tag{2.1}$$

where S_i, called the net sum, is defined as follows:

$$S_i = \sum_{j=1}^{N} w_{ij} y_j + \theta_i \tag{2.2}$$

The scheme that represents such a structure is illustrated in figure 2.2. This formulation shows that in a static network, after that the structure has been chosen selecting the number of neuron and the combination rule, the only sizing parameters are the synaptic weights and the activation offsets.

A dynamic network not only deals with nonlinear multivariate reactive behavior, but can also include time-dependent features such as various transient phenomena and delay effects. Its neurons have an internal state depending not only on the inputs received at a given time but also on the states evaluated in the previous instants. For this reason they are also called pro-reactive agents. The state of the single neuron can be calculated by a differential equation of the following form generally resumed in:

$$\dot{y}_i = f(\tau_i, w_{ij}, y_j, \theta_i) \qquad j = 1 : N \tag{2.3}$$

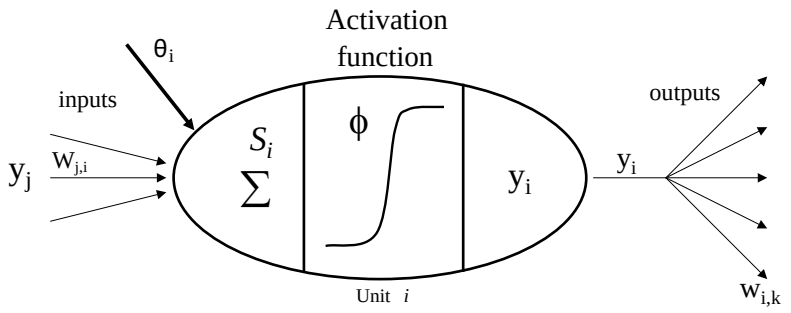


Figure 2.2: An example of neural unit.

where f is the generic nonlinear combination between the weighted inputs, and the additional parameter τ_i is the time constant of the single neuron.

Typical examples of these networks are: the Dynamic Neural Network (DNN), the Continuous Time Recurrent Neural Network CTRNN and the Time-Delayed Recurrent Neural Network (TDRNN). All of these systems can be selected with an evolutionary approach, that is particularly able to reproduce a biological behavior (see [11] for more detail).

2.4 Choice criteria

In the present work ANNs have been used for two different purposes:

- approximate a function;

- reproduce time-dependent behavior.

In the former case the function could be a known analytical formulation, such as an angle versus position conversion, or a unknown relationship between input and output where the only given data are statistical samples. A useful result to understand what kind of network can best fulfill these tasks is the universal approximation theorem for feed-forward neural networks formulated by Hornik et al. [12]. They found that:

- a two layer ANN with sigmoid activation functions on all its units can represent every boolean finite function;

- every real function $\mathbb{R}^n \Rightarrow \mathbb{R}$ limitated and continue can be approximated with an arbitrarily little error with a two layers network with sigmoids on the hidden layer and linear functions on the outputs one.

- every function can be approximated with a three layers network having the output layer made of linear units.

It means that a quite simple ANN can be used instead of a complex nonlinear analytical function reaching the same results. It's important to underline that this drives to a reduction of the computational cost of the operation, but increases also the capability to manage noisy or incomplete data.

In this work all the functions to approximate are of the second class, so the networks to choose are two-layers feed-forward ANNs with sigmoids on the first layer and linear functions on the second one. The simplest way to calculate the parameters of this kind of networks is to use an existing tool like the Neural Network Fitting Tool comprised in MATLAB that uses hyperbolic tangent sigmoid activation function.

For the latter case, it's clear that the only way to simulate a dynamic system pass through a dynamic network, but lots of ANNs, although showing dynamics behavior, cannot model an arbitrarly complex time-dependent system. To understand which type of network is better to use, another approximation theorem involving CTRNNs exists (see Funahashi and Nakamura [13]). It states that for any finite interval of time, they can approximate the trajectories of any smooth dynamical system on a compact subset of $\mathbb{R}^n$ arbitrarily well. This means that, despite their simplicity, they are universal dynamics approximators. For this reason they has been adopted in this work to reproduce dynamic behavior. In the section below a deeper analisys of this type of ANNs will be developed.

2.5 An overview on CTRNNs

A CTRNN can be identified by a set of differential equations of the following general form:

$$\dot{y}_i = \frac{1}{\tau_i}\left(-y_i + \sum_{i=1}^{N} w_{ij}\Phi(y_j + \theta_j) + \sum_{k=1}^{N_i} W_{ik}I_k\right) \tag{2.4}$$

where y_i is the state of each neuron, τ_i is its time constant (τ_i>0), w_{ji} is the strength of the connection from the j-th to the i-th neuron, θ_j is a bias term, Φ is the activation function, I_k represents a constant external input and W_{ik} is the weigth of the k-th input in the i-th neuron. The activation function usually adopted is the sigmoid one, but for the particular applications of this project it is better to use the hyperbolic tangent as was shown in previous works [14][15].

With N neurons and N_i inputs, the network can be completely identified by giving a total number of parameters equal to:

$$N_{tot} = N(N + N_i + 2) \tag{2.5}$$

It has already been said that a CTRNN can approximate every dynamic model, but the problem is how many units have to be used to obtain a sufficiently accurate approximation. A high number of neurons produces optimal results but requires a very long and computationally expensive training process to obtain the correct set of parameters, so it can be useful to investigate small networks and their behavior.

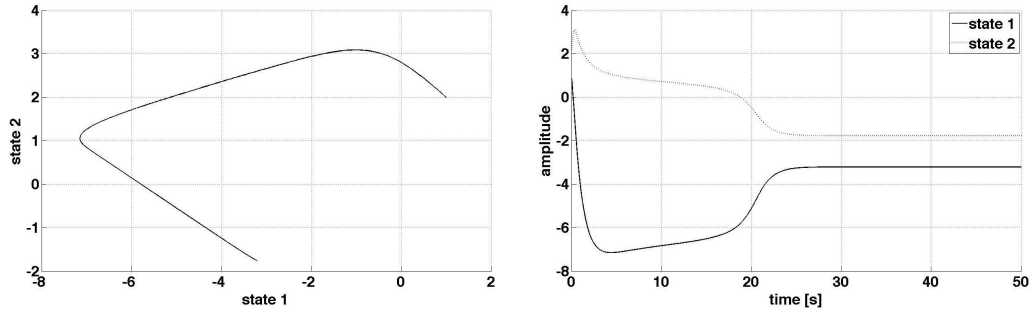
These networks show some typical aspects of nonlinear systems:

- they do not follow the principle of superposition (linearity and homogeneity);
- they may have multiple isolated equilibrium points;
- they may exhibit particular behaviors such as limit-cycle or bifurcation.

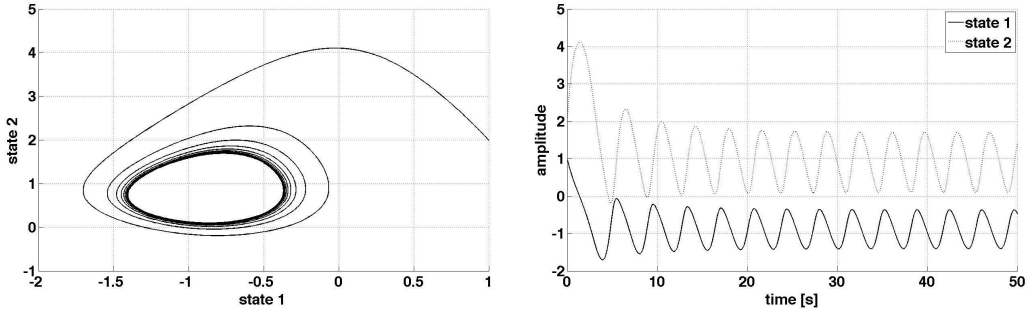
A limit cycle is a closed trajectory in the phase space, producing periodic behaviour in the time domain and it has the property that each trajectory sufficiently close to the limit cycle tends to it asimptotically. A bifurcation occurs when a small smooth change made in the parameter values causes a sudden change in its behaviour, thus changing equilibria from stable to unstable ones but also turning an equilibrium point into a limit cycle.

The first property can be challenging because it makes it impossible to analyze the system within the classic LTI theory. So to evaluate a CTRNN it is necessary to use another instrument, the phase plane method, that allows to investigate the equilibrium points, the bifurcations and the limit cycles.

The analysis of the CTRNN state-space produces a very interesting result: the simplest CTRNN with only two neurons shows up to nine equilibrium points, stable and unstable, and a limit cycle depending on the choice of the parameters (see Beer [16]). This means that also a network with a few units can model lots of different and quite complex behaviours including periodic trajectories in the time domain and attractors in the state domain, both fixed-points and limit cycles. Moreover: the chararacteristics of the attractors can be varied acting on external input I. An example of such a behaviors are shown in figure 2.3. Since the goal of this work is to design a controller, at this point the problem is not only to model a



2.3.1: Attraction point.



2.3.2: Limit cycle.

Figure 2.3: Limit cycle and attraction point in a two-neurons CTRNN.

dynamic system, but it's also necessary to achieve all the characteristics of robusteness required by the control theory (see Friedland [17]). From this point of view CTRNNs are powerful instruments because:

- they react very well to disturbance, filtering noise on inputs, even without any specification during training;
- they respond with an acceptable output even when presented with inputs that they have never seen before;
- if correctly trained, they produce good results even when parameters change in an important measure.

In conclusion a CTRNN can be used like a controller each time that it's necessary to simulate a periodic behaviour or to reach a given target point.

The simplest way to implement a neural controller consists in using measured or observed variables as states and, as control signal, the states derivatives. If applicable, this method allows to avoid every integration, thus producing a very efficient and robust control system also from a numerical point of view.

2.6 Network training

The identification of the set of parameters that defines a ANN is called learinig or training process. Given a specific task to solve, learning means using a set of observations to find the one which solves the task in some optimal sense. The optimum is obtained defining a cost (or fitness) function that evaluates how far away a particular solution is from an optimal solution. The cost function is dependent on the task of approximate a model and on a priori assumptions related to the implicit properties of the model, its parameters and the observed variables.

The learning can be classified from two different points of view: the learning paradigm and the learning algorithm. The learning paradigm is related to the model of the environment where the ANN works. The learning algorithm consists in a set of learning rules each of them used to modify the value of the network parameters. The criterion to follow to modify the parameters is the minimization of the cost function, inserted in a iterative process.

There are three major learning paradigms, each corresponding to a particular abstract learning task. In supervised learning, there is a given set of sample couples of input/output and the aim is the one of finding a function that matches those data. The cost function is related to the mismatch between network mapping and the data. In unsupervised learning there are some given inputs and the cost function to be minimized, that can be any function of the inputs, and the network's output.

Classic learning algorithms, such as Hebbian rule, Back-Propagation and Forward-Propagation, are based on some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. There is another interesting method based on an evolutionary approach: genetic algorithms, widely adopted in previous works.

Since classic learning applied to CTRNN is very slow, a feasible alternative for weight optimization is aGenetic Algorithm (GA). From a mathematical standpoint, genetic algorithms are fundamentally stochastic methods based on the casual generation of solutions, called individuals, each of them identified by a set of parameters, called chromosomes. There are many individuals that make up the population. Once fixed the network's strucure the method works by the an iterative process that evolves through the following steps:

1. definition of an existance field for each parameter (maximum and minumum permitted values);

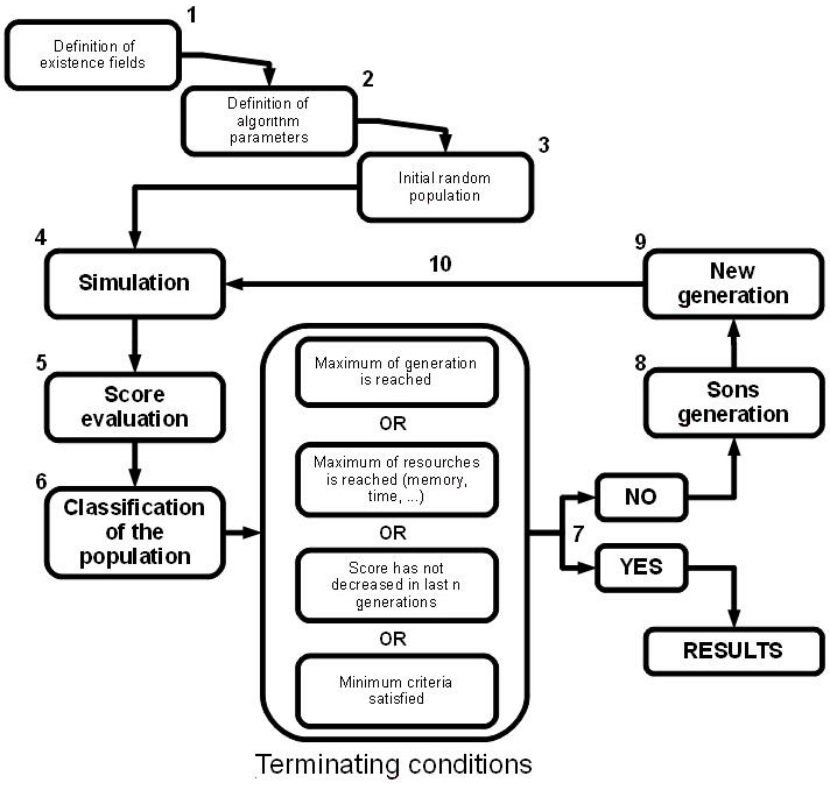


Figure 2.4: Flow chart of the genetic algorithm.

2. definition of algorithm parameters: total individuals number, generations number, crossover fraction, mutation percentage, percentage of parents in every generation;

3. creation of a random population of individuals compatible with the fixed limits in the domain;

4. simulation of the behavior of each individual in the current generation;

5. evaluation of the cost function for each individuals;

6. classification of all the individuals on the basis of their performances and choice of the parents;

7. evaluation of the terminating conditions;

8. generation of the children through crossover and mutation;

9. creation of the new generation composed by the best parents and the best children;

10. go to step 4.

The flow chart has been summarized in figure 2.4. In the inizialization phase (step 1 to 3) the preliminar definitions are performed. After that starts the iterative process that consists in the evaluation of the present generation and the creation of the next one (steps 4 to 10). The algorithm stops only if one of the terminating criteria (step 7) has been satisfied. The algorith terminates when one of the following conditions has been verified:

- a solution that satisfies minimum criteria is found;
- a fixed number of generations reached;
- the allocated budget (computation time/money) has been reached;
- the highest ranking solution's fitness is reaching or has reached a plateau such that further iterations no longer produce better results.

Therefore, many different neural networks must be evolved until a stopping criterion is satisfied. This allows to evaluate all the space of the solutions, avoiding to lock the calculation into local minimum, in order to obtain, if possible, several different networks that satisfy the convergence condition.

The instrument used to apply the genetic algorithm in this work is a tool developed by the Sheffield University and incorporated in MATLAB: the Genetic Algorith Toolbox (GAT). See [18] for more details. All the fitness functions used in the present work have the form:

$$\Psi = \Psi_0(N - \sum_{i=1}^{N} e^{(-k_i(A_i/\bar{A}_i-1)^2)}) \tag{2.6}$$

where Ψ_0 is an amplification coefficient, N is the total number of evaluation parameters, A_i the value of the i-th parameters and $\bar{A}$ its desired value. The coefficient k can be modulated to obtain a certain $\bar{\Psi}$ when the error on the parameter reach a given value:

$$\lim_{A\to\bar{A}} \Psi = 1 - (1 - k(A_i/\bar{A}_i - 1) = k(A_i/\bar{A}_i - 1)^2 \tag{2.7}$$

$$\Psi < \bar{\Psi} \quad \Rightarrow k < \frac{\bar{\Psi}}{(A_i/\bar{A}_i - 1)^2} \tag{2.8}$$

For a network used to approximate functions, a simpler approach has been adopted: as already said in section 2.4, an existing tool has been used. The `nftool` function of MATLAB uses the Levenberg-Marquardt backpropagation algorithm to train the network.

Chapter 3

Control strategy - The problem of walking

The aim of this chapter is to illustrate the main ideas that are the foundation of the NEMeSys control system. At first the global aspects of the walking behavior will be studied and then a particular solution of this problem will be discussed. In the last sections of the chapter all the parts of this solution will be explained in detail.

3.1 General features

Walk can be defined, in the most general way, similarly to a method of terrestrial locomotion that uses limbs. The main differences with other kinds of movement are the presence of a non-continuous contact with a substrate and the usage of more than one multiple-DoFs appendage.

Walking is one of many behaviors where machines still lag notably behind the performance of animals, so it is natural to examine walking in animals to look for hints to improve the performance of machines. From a cognitive standpoint, walking appears to be a fairly automatic behavior. Nevertheless, it's also immediate arguing that walking in a natural environment requires considerable capabilities from the controller, involving different features, and it's all but a trivial behavior.

The most important global aspects of the walking problem can be summarized as follows:

- redundancy. In systems concerned with walking, the number of degrees of freedom is normally larger than the one necessary to perform the task. Thus, there may be a manifold of leg postures for a given kinematic boundary condition;

- autonomy. The redundancy requires the system to select among different alternatives according to some, often context-dependent, optimization criteria, which means that the system usually has to adopt some choices without external command;

- embodiment of the controller. To maintain the problem at a level as simple as possible, the controller needs the ability to exploit the physical properties of the body

- situatedness of the body in its environment. Each walking system is a physical systems situated in complex and often unpredictable

environments, which means that any movement may be modified by the physics of the system and the environment.

The best solution to fulfill all these tasks, as described in section 1.1, is a decentralized control structure that consists of a number of distinct modules each one solving particular subtasks. The division into modules and the choice of the subtasks have been accomplished starting from biomimetics findings: the followed approach consist in assembling logical components to model the system properties of the behaving animal.

Although many animals have body appendages that can be used for walking, only a few species have been investigated in sufficient detail. Among them, the stick insect is the more suitable for the purposes of this work thanks to its capability to walk on very complex and irregular substrates.

3.2 The walking system

Before starting the discussion about the structure of the controller it's useful to evaluate the main characteristics of the system to be controlled. It is a simplified model of a stick insect, with a rigid body supported by six legs: two legs are called *ipsilateral* if situated on the same side, *contralateral* if situated on the opposite side.

Each leg has been modeled as a manipulator with three joints and divided into three movable sections: the *coxa*, closest to the body, the *trochanter-femur*, or just femur as they are fused and the *tibia*. A fourth part, the tarsus, provides and holds the ground contact in real stick insects, but it's not strictly necessary to perform a walking behavior and this is why it won't be considered in this model.

The Body-Coxa (BC) joint is best described as a socket joint, whereas the Coxa-Trochanter (CT) joint and the Femur-Tibia (FT) joint are hinge joints. The CT and the FT axis are parallel, hence, the femur and tibia lie on the same plane.

Orientation and motion of the coxa segment out of the body can be described by three orthogonal rotations, one of which is dominant. On the left hand side of Figure 3.1 there are shown the two orientations ψ' and ψ'' that change the least, thus both are considered fixed. However, although variability during the walk is small, the angle ψ' in the stick insect is large. In this model, instead, both these angles have been taken equal to zero, which drives the third axis to be perpendicular to the body plane and reducing the BC joint to a hinge joint like the other ones. In this way all the DoFs can be treated almost in the same manner.

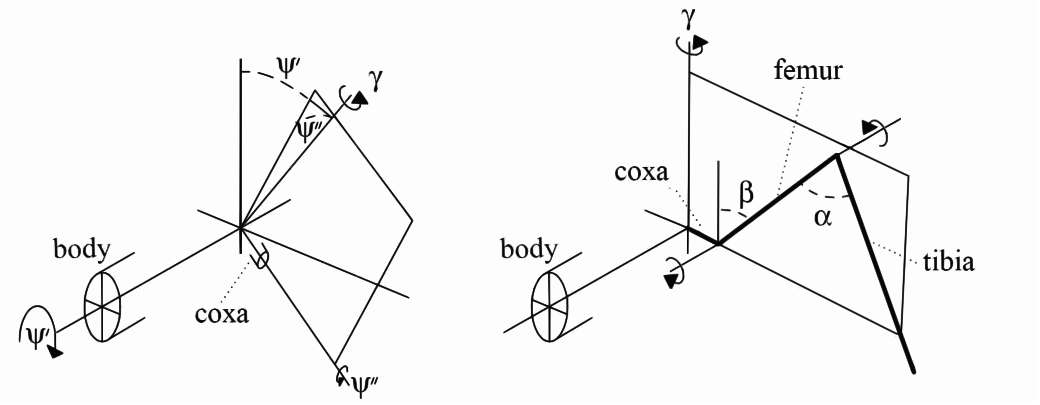


Figure 3.1: Leg orientation and angular variables.

On the right side of Figure 3.1, the geometric parameter defining the leg are illustrated. The leg can be described as a 3-joint manipulator. Starting from the body it's possible to identify: the Body-Coxa (BC) joint identify by the angle γ, the Coxa-Trochanter (CT) joint identify by the angle β and the Femur-Tibia (FT) joint identify by the angle α. The BC joint can be modeled as a socket joint where two orientation (ψ' and ψ'') vary only a little during the motion and can be considered as fixed. In this work their values have been taken equal to zero. The third orientation is the only significant DoF of the joint, the γ angle, that determines mostly how much near to the front or to the back the leg plane it is positioned. The β angle specifies how high or low the knee is. Finally, the α angle determines the lateral proximity of the tarsus to the body. Note that all the joints influence motion in all directions, i.e. joint variables are non-linear with respect to Cartesian coordinates. Joints and the variables used to label their angular position are used arbitrarily to name the joint.

Rotation of leg segments is usually accomplished by groups of antagonistic muscles: each one of them can pull in either direction. Muscle groups controlling the BC, CT and FT joints are known as protractor-retractor, levator-depressor and extensor-flexor respectively. To simplify the matter, the control problem in terms of angles evaluating the effect of actuators will be treated later on (see sections 6.1 and 7.3.1).

3.3 Problem definition

The modular approach adopted in this work allows the simplification of the problem of walking control in a large measure, but needs a good distinction among all the tasks.

Biological findings about various animals (e.g. Wendler [19]) show that the movements of individual legs are managed by fairly independent control systems. On the other hand each leg has an influence on the other ones and it's the combination of all these influences that allows to generate an efficient walk. This means that the global motion problem can be solved by accomplishing two macro-tasks:

- control the motion of each single leg;

- coordinate all the legs;

The control of the single leg it's fundamentally the problem of generate the step cycle. During the walking, the individual legs typically move cyclically and, in order to facilitate the analysis, the motion of a leg is often partitioned into phases: the control of each phase and transition among phases are called subtasks of the walking problem. Other features regarding the single leg control are related to maintaining a fixed leg and to reacting to external disturbances. In section 3.5 all these arguments will be discussed deeply.

The coordination among legs is essential to regulate global parameters such as advancing speed, stability margin or body attitude and position. The first task consists in maintaining a steady motion and it can be obtained introducing local weighted influences among neighboring legs. To fulfill more complex task like regulating body attitude, local rules aren't enough and a global planning is required. The whole problem will be treated in section 3.6.

In this work a bottom-to-top approach has been adopted to create an incrementally realistic motion generator. Starting from the design of efficient controllers for every phase of the step cycle, a selector regulating the transition among phases has been developed producing a complete single leg controller. At this level mutual influences among neighboring legs have been introduced and only in the last phase of the work global coordination parameters have been taken into account accomplishing the global task of walking control.

3.4 Global controller architecture

The approach to the problem of walking control is not unique: it heavy depends on external parameters such as substrate properties and global required speed. Walking in predictable environments and fast running, to a large degree, rely on leg mechanical properties. Conversely, slow walking in unpredictable terrain, e.g. climbing in rugged structures, has to

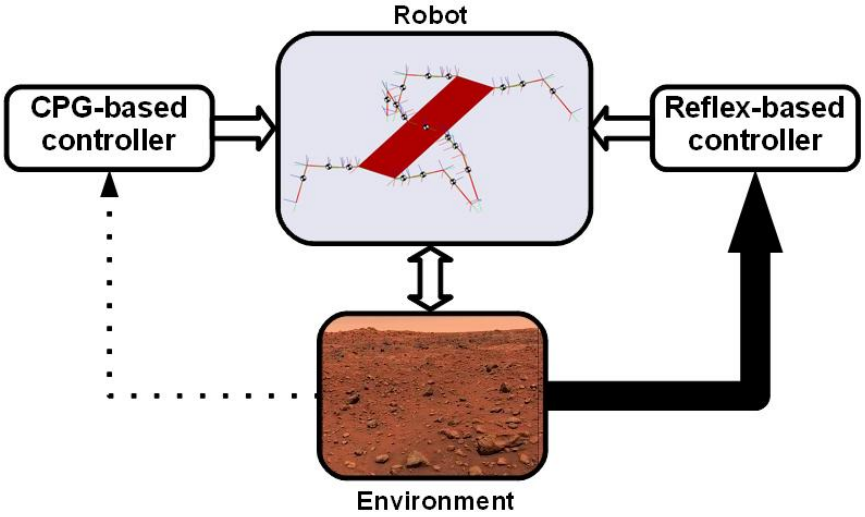


Figure 3.2: CPG vs. reflex-based approach.

rely on feedback systems that monitor and react to specific properties of the environment. From these considerations the two different methods descend to structure a walking controller: central pattern generator and reflexes-based systems.

A Central Pattern Generator (CPG) may be defined as a system that can endogenously, i.e. without sensory feedback or central input, generates periodic motor commands for rhythmic movements such as locomotion. To be classified as a rhythmic generator, a CPG requires: two or more processes that interact such that each process increases and decreases sequentially, and that, as a result of this interaction, the system returns repeatedly to its starting condition. Typically a CPG controls each single leg and the coordination is obtained by a temporal synchronization of all the CPGs. This system allows to reach very high advancing speeds and a regular motion also with quite a simple control architecture. On the other hand they require legs with particular reactive characteristics and powerful motors and have very low crossing and climbing capabilities. The best-known CPG-based walking behavior can be observed in cockroaches.

On the other hand, a reflex-based controller can be viewed as a closed loop control system with fixed input/output characteristics that produce an output only in relation with an environment depending input. The coordination is typically based on rules that mediate the reflexes of every single leg. The reflex driven approach seemed to provide a simple way to stabilize the walking patterns, also in very difficult conditions, by providing a set of fixed situation-reaction rules to external disturbances and as a way to regulate leg coordination among multiple independent legs. The

problem of this system is the complexity required to be sufficiently flexible: to be able to face lots of situation, a high number of reflex has to be taken into account. The most relevant example of such a behavior can be found in stick insects.

In insects only one of this structure isn't able to explain all the observed behavior as underlined, among the other, by Porcino [20]. For a space-oriented project, like this one, it's clear that, at this level of development, a very fast CPG-based robot has only little utility. In an unknown environment, like planetary surfaces, the first priority for a robot is to ensure a steady motion in any moment. A walking robot, besides, must be able to reach locations out of the range of wheeled ones, balancing its complexity with its higher capabilities. Furthermore a slow locomotion allows to reduce energy consumption and to avoid shocks to the scientific payload. All these considerations favored the choice on a reflex-based controller.

3.5 The control of a single leg

The problem of controlling a single leg can be simplified partitioning the step cycle into the following two phases:

- during the *stance* phase, the leg maintains ground contact and is retracted to propel the body forward, while supporting the weight of the robot. The terms *power stroke* and *support* (or *ground*) phase are also used in the literature to denominate this phase;

- during the *swing* phase, the leg is lifted off the ground and moved in the direction of walking, to touchdown at the location where the next stance should begin. The terms *return stroke* or *transfer* (or *aerial*) phase are also used in the literature to denominate this phase.

This division is not the only one we may use, but it permits to reduce the complexity of the problem, just producing the same results of the other ones.

The phases are mutually exclusive behaviors: a leg cannot be in swing and in stance at the same time. Therefore, the control structure must include a mechanism for deciding the transition between swing and stance, creating the complete step cycle. This criterion can be identified on the basis of spatial variable:

- at the Anterior Extreme Position (AEP) the transition from swing to stance occurs;

- at the Posterior Extreme Position (PEP) the transition from stance to swing occurs.

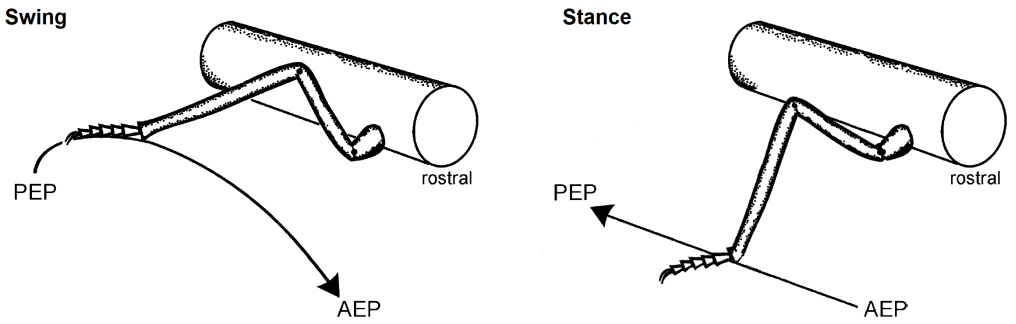


Figure 3.3: Swing and stance phases.

This means that to produce a stereotypical step cycle, the leg controller can be composed by only three modules each of them fulfilling a subtask:

- a swing module that controls the leg during the transfer phase;
- a stance module that controls the leg during the support phase:
- a selector that works at an higher level and switches between using the outputs of either the stance or the swing module.

To guarantee a robust and efficient motion on irregular surfaces a stereotypical step cycle is not sufficient. For this reason a limited number of standard reactions to typical unpredictable disturbances, called reflexes, has been added to the standard controller.

Another interesting aspect of the controller's general structure is the choice of control variables: some advantages can be obtained by choosing joint velocity commands rather than joint position commands in contrast to traditional robot controllers. Biology's use of velocity based control is quite sensible, since the main objective of locomotion is to keep the body moving forward at a desired rate of speed. Velocity control is also generally simpler and more stable than position control, since velocity control in the presence of inertial dynamics generally involves only a single integration from actuator force.

The scheme of the complete single-leg controller is illustrated in figure 3.4 with particular attention to the input/output relation. The swing and stance nets receive, as outputs set, the joint angular velocities of the three leg joints (BC, CT and FT). They are controlled by the selector net in a mutually exclusive manner. The state of the selector net depends on the sensory input from the own leg and weighted information corresponding to the coordination rules. The reference values for the swing net are calculated by a target net, that receives, as input, the position of the anterior ipsilateral leg. The height net controls the body clearance during the

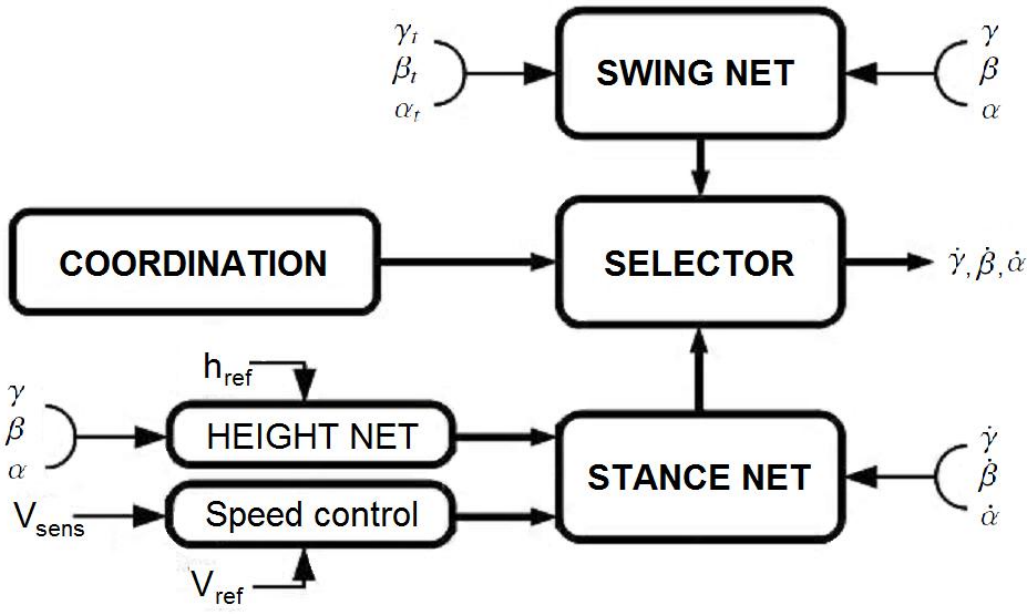


Figure 3.4: Scheme for the controller of a single leg.

stance phase. Except for the input to the target net and the coordinating influences, all sensory inputs are local proprioreceptive feedback of the leg itself. The stance net receives two global commands: controlling body advancing V_{ref}, and yaw $\dot{\theta}_{ref}$ velocity.

3.5.1 Swing phase

Controlling a swing movement is easier than controlling a stance movement, because a leg in swing is mechanically uncoupled from the environment and, owing to its small mass, essentially uncoupled from the movement of the other legs. Therefore, whatever a leg does during a swing movement, it has virtually no impact on the movements of the other legs.

According to figure 3.1, each stick insect leg can be modeled as a manipulator with 3 DoFs of rotation. As physiological experiments on insects have shown that each one of these DoFs may have only a weak coupling to the other two, the controller must have three motor outputs at least, one for each leg joint.

Since the stick insect regulates swing movements and compensate for external perturbations, the control system must receive proprioreceptive sensory feedback to account for a closed-loop control. Moreover, because swing movements are known to be targeted towards a given location, the controller must also receive information about a desired posture.

Detailed analysis show how the best choice is to take the three angular

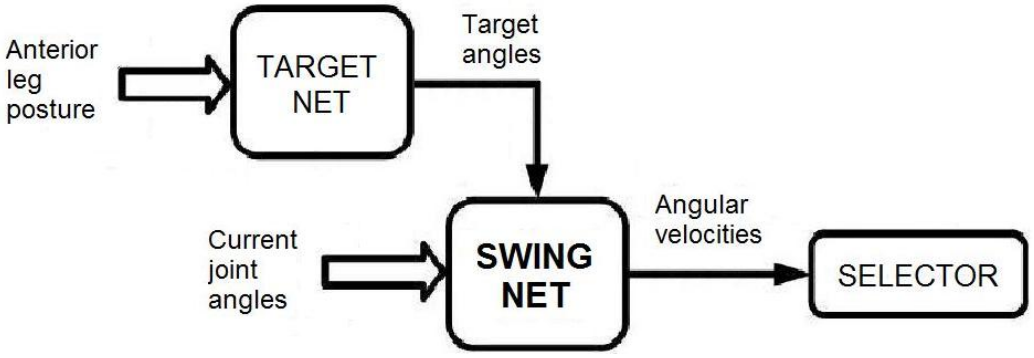


Figure 3.5: Swing control module.

coordinates defining the actual leg configuration and the three defining the target as inputs and the three joint angular velocities as outputs. All these considerations can be summarized in the scheme illustrated in figure 3.5, that shows the correlation existing among inputs and outputs: it receives, as inputs, the current measured joint angles and a set of target angles that define the AEP. Its outputs are the three desired joint angular velocities. The target net calculates the target angles starting from the current joint angles of the anterior ipsilateral leg. It is not present in the front legs controller where the AEP has a fixed value.

To understand which behavior the controller should reproduce it is interesting to evaluate a general swing trajectory as shown by Cruse and Bartling [21]. In figure 3.6 it's possible to see how the swing phase can be defined by three points: the posterior extreme position (PEP), where the leg lifts off the ground to start swing, and the anterior extreme position (AEP) where the leg ends swing by touching the ground. The third point is the Superior Extreme Position (SEP), the dorsal extreme position of the swing movement. The swing height is defined as the distance in the x-z plane between the SEP and the midpoint between AEP and PEP. To fulfill the task, the controller must show two different behaviors: the leg has to lift off and the foot has to move to the target position (AEP).

The first micro behavior permits to avoid all the obstacles below the trajectory so, with an high SEP, the walk will be possible also on rough terrain. On the other hand, to reach these positions, the motion should be very fast thus becoming very energetically ineffective. Numerical relationship giving the optimal SEP depending on PEP coordinates has been found starting from behavioral experiments on insects [22]. In this work a different approach has been used: a minimum value for the SEP has been taken and the controller has been designed to produce trajectory with a SEP always above the limit value. This behavior involves only the CT

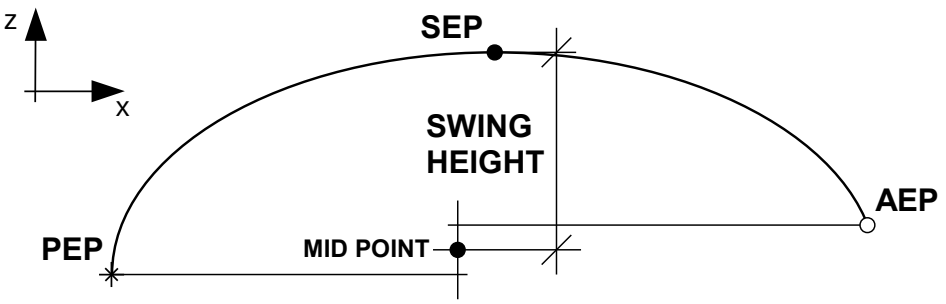


Figure 3.6: Swing trajectory geometric parameters.

joint angle that has to be dependent on the TC angle to follow the correct up-down movement.

The second micro behavior means that this point must be an attractor for the controlled system, but only in the two dimensions of the ground plane. In the direction perpendicular to the support surface, the attraction point must be placed below the ground reference position thus prohibiting the foot to move too much slowly immediately before the ground contact and allowing it to detect the ground also if it's beyond its reference position. This behavior involves all the three joint angles but mainly the TC one.

3.5.2 Stance phase

In a complex environment, the task of controlling the stance movements of all the legs on the ground poses several major problems. It is not sufficient to assign the motion of each leg on its own, because the mechanical coupling through the substrate implies that efficient locomotion requires the coordinated action of all the joints that connect the body to the ground. Thus, the action of up to 18 joints may need to be coordinated at any moment in time. However, the number and combination of mechanically coupled joints varies, depending on the current pattern. Accordingly, the task to be solved changes continuously. Besides, the control of a kinematic chain composed by three hinge joints is a nonlinear task and further complexity is introduced by joints and actuators properties, such as friction or flexibility.

In classic approach to walking machine, the so-called force distribution problem, which occurs whenever parallel kinematic chains are mechanically coupled via the ground, can be solved only by computationally costly algorithms that take into account the complete dynamics of all the legs in stance and try to optimize some additional criterion. The nature of the mechanical interactions and the search for a global optimum requires a single

central control system instead of a distributed processing. In technical solutions, such centralized system makes real-time control difficult, even in simple cases.

Although the task to coordinate many joints appears complex, insects master it with their simple nervous system. This means that they use, instead of a centralized controller, a decentralized approach able to exploit the dynamics of the body-environment interaction without calculating all the required information explicitly, e.g. the mutual influences among the joints.

To solve this particular problem in a very easy way, Cruse et al. [23][24] propose to replace a central controller with distributed control in the form of local positive feedback. For more detail on positive feedback in biological systems see De Angelis [25]. The positive feedback occurs at the level of the single joint, feeding back the position signal to control the motor output of the same joint, without direct relationship with the rest of the kinematic chain.

To understand how positive displacement feedback works, one must consider, a standing legged system that begins to move one joint, while keeping all the feet on the ground. Owing to the mechanical connections, all other joints of the moved leg, and even joints of the other legs, passively adjust to the active joint movement. Thus, the movement direction and speed of each joint do not have to be computed because this information is already provided by the physics of the whole system.

This example is related to postural control, a problem ruled by resistance reflexes that maintain the position of each leg with local negative feedback circuits. In the control of walking a resistance reflex may be replaced by an assistance reflex: it can be modeled by a positive feedback that transforms this passive movement into an active one.

There are, however, several problems to be solved. The first is that positive feedback using the raw position signal would lead to unpredictable changes in movement speed and not necessarily to the nearly constant walking speed which is usually desired. This problem can be solved by introducing a kind of band-pass filter into the feedback loop. The effect is to make the feedback proportional to the angular velocity of the joint movement and not proportional the angular position. This strategy is also known as Local Positive Velocity Feedback (LPVF).

The second problem is that using positive feedback for all three leg joints leads to unpredictable changes in body height, even in a computer simulation excluding gravity. To avoid these effects the height control has been decoupled to forward movement. According to experiments on stick insect, LPVF has been applied only on γ and α angles, that are deputy

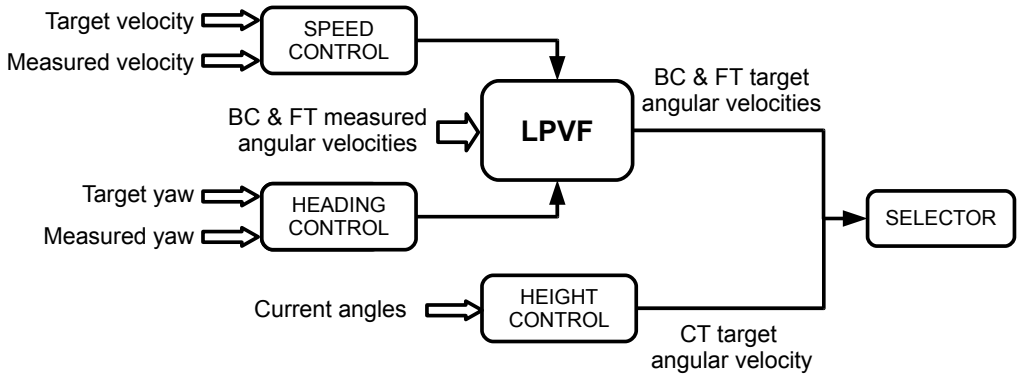


Figure 3.7: Stance control module.

to propel the robot, but β is controlled by a negative feedback circuit that controls the body height only.

A third problem related to using positive feedback is the following. When a stationary walking robot is pulled backward by gravity or by a brief disturbance, with LPVF, it should continue to walk backwards even after the initial pull ends. This has never been observed in actual walking animal. Therefore, in this work it has been assumed that a supervisory system exists which is not only responsible for switching on and off the entire walking system, but also specifies walking direction. This influence is represented by applying a small, positive input value which replaces the sensory signal if it is larger than the latter.

Positive velocity feedback could produce an irregular motion because each single leg influences the global motion. To avoid this behavior all the legs initiate forward-bound propulsive forces at the start, as it is the case in insects, and a central yaw-control system maintains the desired course.

Finally, it's necessary to choose a method to regulate the speed in such a controller. Assuming a central value which represents the desired reference walking speed, it is compared with the actual speed and the error signal is subject to a nonlinear transformation. The output is a gain, then multiplied with the signals providing the positive feedback for each joints.

The complete structure for the stance control module or *stance net* is summarized in figure 3.7. The stance net consists in two output units that set the angular velocity of the BC and FT joint during stance. As the CT joint mainly affects height of the body relative to its feet, it is controlled by a separate module (height control). The stance net receives sensory input from proprioreceptors signaling angular velocities, i.e. closing a local positive velocity positive feedback loop (LPVF). Also, it receives central commands that determine the velocity of forward translation and yaw ro-

tation. Both of these signals are subtracted from a corresponding sensory input signal, thus closing a negative feedback loops.

3.5.3 Phase transition

The problem of transition between phases seems to be complex, but truth to say it's simple and depends on two parameters only identified by numerous authors, e.g. Cruse [26]. They are the following:

- the Ground Contact (GC) input excites the stance phase, at the same time as it inhibits the swing phase;
- the Posterior Extreme Position (PEP) input indicates that it is time for the transition to the swing phase.

This process can be controlled by a single module, called *selector*, that switches the outputs calculation from a phase to the other one. Even if the two motion phases are mutually exclusive, the output units receive self-excitation (i.e. positive feedback) to fix into a state. This was found to be less sensitive than the use of mutual inhibition from the output units.

3.5.4 Reflexes

A reflex is a type of behavior that can be roughly defined as a rapid, automatic involuntary response triggered by external stimuli. The response persists for the duration of the stimulus and its intensity is correlated with the stimulus's strength. Reflexes are very useful during locomotion, making the animal able to react to any unpredictable situation rapidly.

The control system described in this work for the generation of a step cycle in each leg of the robot, is based on reflexes as it is illustrated in section 3.4. This approach allows an higher flexibility than for CPG-based systems, but the reflexes introduced in that model are only a part of the total reactive behaviors shown by a stick insect: an assistance reflex to propel the body and a step reflex to lift-off the leg.

In real animals other reflexes exist, that allow them not only to generate a regular walking pattern, but also to produce the correct behavior in reaction to a large number of unpredictable changes in the environment. From this point of view a reflex may be considered as a stereotypical response to a standard changing in the external inputs. Now the problem is how many reflexes must be implemented in the controller to solve the problem of walking correctly. Recent studies of neurophysiologists identified three more different typologies of reflex in insects (Beer et al., 1997):

- *stepping reflex*: when a leg slides or assumes a configuration poorly stable, performs a limited aerial phase to return in a posture able to ensure a sufficient stability;

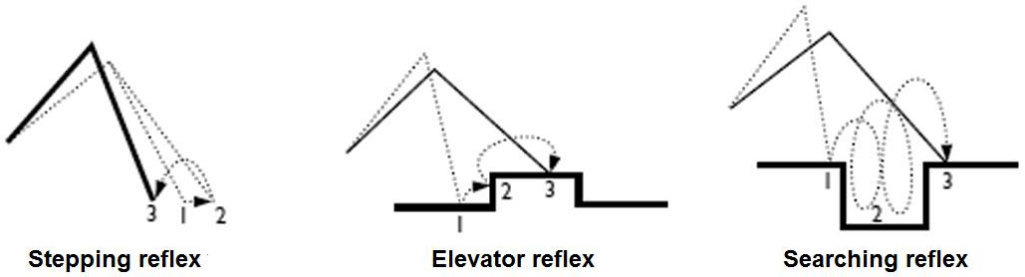


Figure 3.8: Overview on reflexes.

- *avoidance reflex*: when a leg touches a obstacle, it moves away from it and changes its elevation to produce a trajectory able to avoid it;

- *searching reflex*: when a leg doesn't find the ground at the end of its swing phase, starts a rhythmical motion in an increasing region of space until a supporting surface is found.

In figure 3.8 such behaviors are shown. They will be discussed in detail in section 4.4.

3.6 Legs coordination

The second macro behavior required in walking control is the coordination among single legs movements. The problem must consider two things at least. First, the timing of lift-off and touchdown in each leg must not impair the stability of the whole body. Second, the number of legs in stance determines the upper bound of propulsive force. Therefore, efficient timing and coordination of power and stance direction of all legs in stance must be controlled to ensure their synergistic action. This two features will be developed in section 3.6.1 and 3.6.2 respectively.

These two first requirements can be fulfilled without a central planning. To reach a satisfying solution it is sufficient to insert local influences that link each limb only with its neighbors. This method, shown in subsection 3.6.3, allows to simplify the problem considerably, reducing computational costs.

Some other problems need a global supervision on all the legs to be correctly solved: the main of them is the control of the body's height and attitude. To manage these parameters, it's necessary both to measure and control variables of all the legs and to perform global calculation. The other global feature, treated in subsection 3.6.4, involves the generation

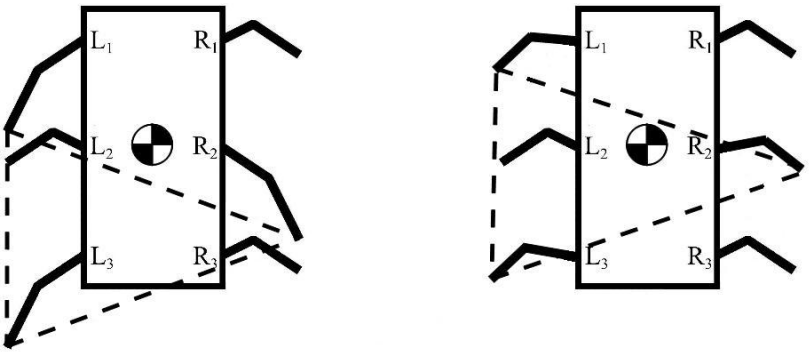


Figure 3.9: Static stability.

of curve trajectories in function of the body velocity and of the turning radius.

3.6.1 Stability

The main high-level requirement for a walking controller is to ensure the stability of the robot and to keep it during the walk. The work on stability analysis was based on the Point of Center of Gravity (PCG) of the robot. In this study, PCG denotes the two-dimensional point obtained by projecting the Center of Gravity (CoG) onto a horizontal plane.

An ideal legged locomotion machine is statically stable at time t, if all the legs in contact with the support plane at the given time remain in contact with that plane, when all the legs of the machine are fixed at their locations at time t and the translational and rotational velocities of the resulting rigid body are simultaneously reduced to zero. It can be shown that this definition is equivalent to the condition PCG $\in A_{sup}$ where A_{sup} is the area delimitated by the supporting polygon as shown in figure 3.9. The supporting polygon (dotted lines) is the convex boundary obtained by linking all the feet of the supporting legs. In a statically stable posture (left side), the projection on the ground surface of the robot's centre of gravity (PCG), lies inside the supporting polygon. On the opposite in a statically unstable configuration (right side) the PCG is placed outside this pattern, causing the fall of the robot. From this condition, it's possible to define the static stability margin, as the shortest distance from PCM to the support polygon's boundary.

In this work a walk will be said to be statically stable if the static stability margin is positive at all times during the locomotion, i.e. the condition PCG(t) $\in A_{sup}(t)$ $\forall$ t is satisfied. Furthermore, the terms static balance and statically balanced walk will be used instead of static stability. To maintain a correct posture in order to avoid falling over, it can be useful to impose

the static stability condition during the entire walk.

With a static balance requirement, at least three legs have to touch the ground at the same time for locomotion if an ideal legged locomotion machine is assumed. This means that biped and quadruped walker require more complex dynamical balancing to reach good performances, but it's not a problem for hexapod: previous studies on insects show that the walk patterns allowing to reach the highest speeds for six-legged walking platform are all statically balanced (see Wilson [27] and Song and Waldron [28]).

Another consideration can be done supporting the choice of limiting the stability analysis to statically stable gaits: the concept of a safe walk can be introduced, to be a walk where, if all the joints are suddenly frozen, the system still ends up in a statically stable equilibrium. This concept does not imply a statically balanced gait, since falling is allowed as long as the system ends in a safe configuration.

3.6.2 Gaits

A gait can be generally defined as a manner of moving the legs in walking or running or, more properly, as a pattern of locomotion characteristic for a limited range of speeds described by a number of quantities, one or more of which change discontinuously at the moment of transition to other gaits. The concept of gait is strictly related to the stride concept, that is a period of locomotion defined by the complete cycle of a reference limb.

A gait can be numerically quantified by four parameters:

- the stride duration, the duration of one stride;
- the stride length, the distance the trunk translates during one stride;
- the duty factor (typically denoted β), the temporal fraction of a step cycle during which the foot is on the ground;
- the relative phase of leg l (typically denoted ϕ_l), described as the leg's phase with respect to a reference leg.

The simplest way to define a gait mathematically is a sequence of binary vectors that indicates the phase (swing or stance) of each leg and the phase duration. The gait is thus defined by the sequence in which the legs change phase. This analysis can be summarized in a gait diagram showing the phases of the different legs as a function of time.

In six-legged walkers the gaits that ensure the best performances can be chosen from biological findings. In his reference work on hexapod locomotion Wilson [27] shows that all the typical gaits observed in stick insects are based on five concepts:

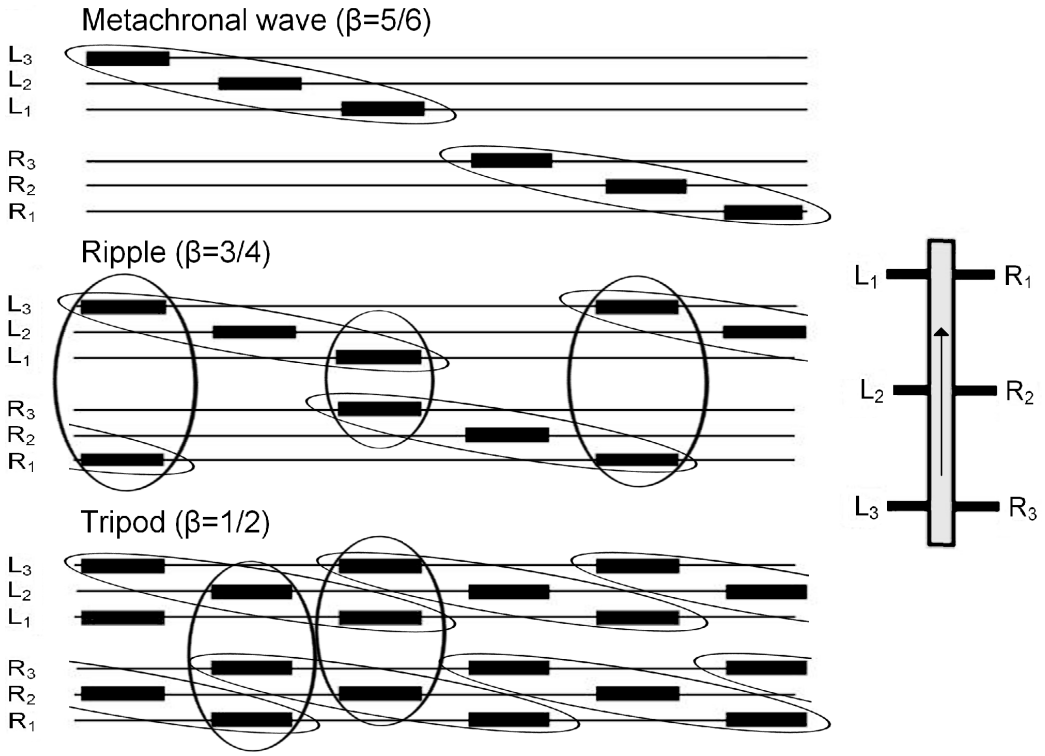


Figure 3.10: Gait diagrams of the most significant wave gaits.

- a wave of protractions runs from posterior to anterior;
- contralateral legs of the same segment alternate in phase;
- protraction time is constant;
- retraction time decreases as walking frequency increases;
- the intervals between steps of ipsilateral adjacent legs are constant, while the interval between the foreleg and hind leg steps varies inversely with frequency.

Gaits of this family are called *wave-gaits* and show a periodic behavior with a $\beta \geq 0.5$ equal for all the legs. The most important characteristic of these patterns is their capability to ensure a statically balanced motion on a wide speed range. Another important feature is the possibility to adapt the gait to speed variations continuously. In figure 3.10 the gait diagrams of the most important wave-gaits have been summarized. An overview of the most significant wave-gait is shown in figure 3.10. The leg are numbered as in the drawing. The horizontal axis represents the time. The solid

bar indicates protraction (swing phase). Continuous enclosures indicates fixed basic sequences of steps of the legs on one side, i.e. metachronal wave, or simultaneous protractions. Contralateral legs on the same segment, e.g. hind legs (R3-L3), show an 180-degrees phasing. An increasing duty factor β drives to a faster, but less stable gait.

On really rough terrain, cyclic gaits are not suitable and the *free gait*, a gait that can produce both regular or irregular pattern depending on coordination influences, is used instead. This means that a correct coordination among legs must be able not only to change gait continuously when the required velocity varies, but also to produce a non-periodic free gait when disturbances are overwhelming.

3.6.3 Coordination rules

The coordination rules that regulate the gait in a six-legged walker have been identified by Cruse starting from his experiment on stick insects and can be summarized as six influences among both contralateral and ipsilateral legs. The influence of inter-leg coordination mechanisms is stronger between ipsilateral legs, than between contralateral legs. No direct influences between the diagonal legs have been found. The influences are as follows:

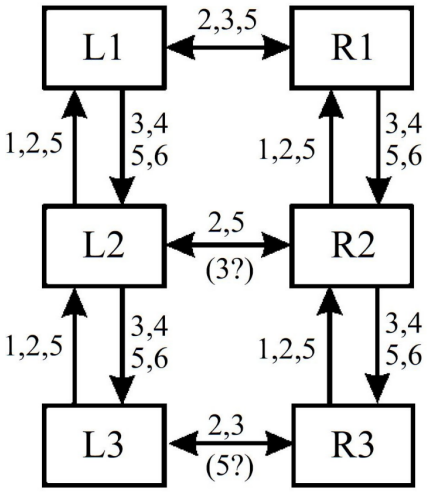
1. the start of a transfer phase is inhibited if the ipsilateral posterior leg is transferring (and up to 100 ms after footfall). This can cause a prolonged support phase;

2. the start of a transfer phase is excited if the ipsilateral posterior leg or the contralateral leg has just entered the support phase. This can cause a shortened support phase;

3. the start of a transfer phase is more strongly excited, the further the leg is to the rear of a supporting ipsilateral anterior leg or contralateral leg. This causes the leg to start its transfer phase before the anterior leg;

4. the start of a support phase is targeted to occur next to the (supporting) ipsilateral anterior leg. This causes a follow-the-leader type of gait to emerge;

5. a) the force of a supporting leg increases if an adjacent leg encounters increased resistance;
   b) The support phase is prolonged, if the load of an adjacent leg increases;



Rule	Action / Goal
1	Suppress lift-off / Avoid static instability
2	Facilitate early protraction / Favour temporal coherence
3	Enforce late protraction / Maintain temporal coherence
4	Aim touch-down location / Exploit prior foothold
5	Distribute propulsive force / Share load efficiently
6	Enforce correction step / Avoid stumbling

Figure 3.11: Coordination rules that couple step cycles of neighboring legs.

6. if a foot is placed on the top of another foot, the placed foot is repositioned slightly to the rear (to avoid stumbling).

A scheme of how this coordinating system works is shown in figure 3.11. Each of the six legs (boxes labeled L1 to L3 for the left front, the middle and the hind legs, respectively, and R1 to R3 for the corresponding right legs) signals information about its current state to its neighbors (arrows). The known coordination rules are numbered according to the list to the right. Numbers next to the arrows denote the rules known to be present in a given signaling pathway. The actions/goals correspondence that can be associated with the experimental evidence, is listed aside.

These rules have been implemented on lots of numerical control models, some of them applied on real walking platforms. Applications show that mechanisms from 1 to 3 are sufficient to produce all the hexapod statically stable gaits, ensuring a correct control of the walk over a wide range of velocity.

The main problem of this approach is the calculation of the weights that modulate the mutual influences among legs. It's a very expensive process because it requires a simulation of the complete multi-legged model that involves a large number of parameters. Moreover, if the weights aren't correctly calculated, the said rules don't guarantee a stable gait for the whole operative range of velocity.

A different approach has been introduced by Porta and Celaya: it allows to use Boolean criteria only to coordinate the legs, and its parameters

can be easily hand-tuned. The main disadvantage of this method is its incapability to produce an efficient gait at low velocities.

In this work a mixed approach has been adopted: starting from the simplest set of rules producing an always stable gait, then corrected with some of the six rules developed by Cruse to obtain a more efficient gait.

3.6.4 Curve walking

The production of curve trajectories is another important aspect of the legs coordination. The course control is a very complex problem that requires a specific action on each limb changing both the control law for the single leg and the gait pattern adopted for the straight walk case.

Basically, the behaviors observed in insect depend on one parameter only: the turning radius, defined as the radius of the arc of circumference to be followed at a given moment. The decreasing of the value of this variable produces five behaviors identified by Zolotov for the *Apis Mellifera* [29]. They are the following:

1. at the inner flank: gait unchanged, decrease of the stride length, change of the step direction; at the outer flank: increase of the stride length;

2. at the inner flank: decrease of frequency and velocity of metachronal waves; at the outer flank: increase frequency and velocity of the metachronal waves;

3. at the inner flank: anchoring of the hind leg, lateral steps in the fore and middle legs;

4. at the outer flank: anchoring of the hind leg;

5. at the inner flank: backward marching.

This means that to produce a large curve trajectory (case 1) it is sufficient to change the stance velocity of the legs on the outer and the inner side and to vary the calculation of the AEP for the inner side, without any alteration in gait structure. These results can be easily obtained with two updates only on the original controller. The cases of smaller radii haven't be considered in this work.

Some examples of curve gait, observed by Zollikofer [30], are shown in figure 3.12. Solid-line triangles indicates tripods R1-L2-R3 and dashed-line triangles indicates tripods L1-R2-L3. The longitudinal axis of the body is indicated from head position (dot) to centre of mass (end of line). Walking direction is from left to right. With high turning radii (10÷20 units), the

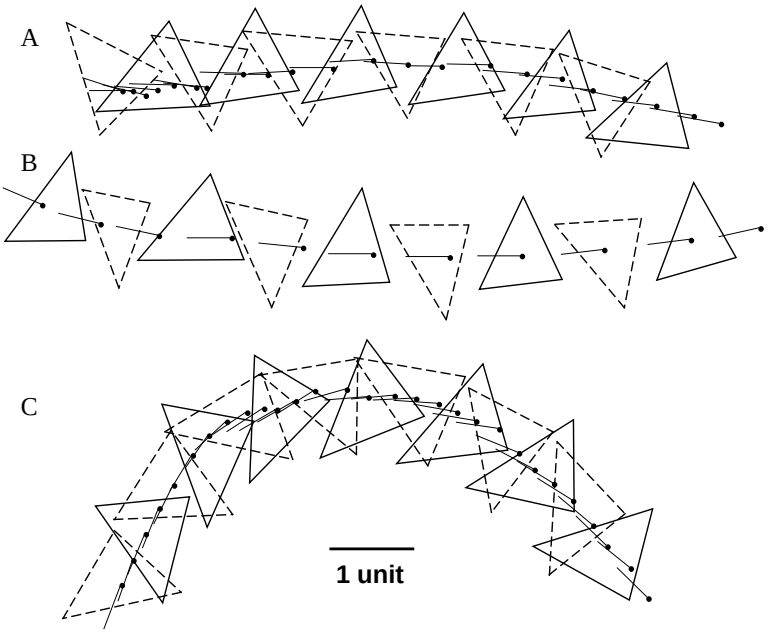


Figure 3.12: Gait changes during a curve walking.

gait is almost unchanged from the normal forward walking (A,B). Only little differences in the stride length on the two sides. No remarkable differences between high (A) or low (B) speeds. With a smaller radius of curvature (about 3 unit in C) it's possible to see how the gait change, especially how the AEP gets closer to the body, but also a little frequency variation between the two sides.

3.7 Final control structure

The most significant shortcoming of the biological-based approach proposed by Cruse et al. is that their model did not consider any gravitational effects, inertial dynamics, or ground contact reactions. In the case of a stick insect, it's possible to ignore such inertial dynamics. At the scale of a typical six-legged robot, however, such effects are meaningful, and change in a significant manner the stability of the closed-loop system.

On the basis of these issues, Wait and Goldfarb [31] proposed a modified version of this type of control system that is based on its biological paradigm, but that can provide stable locomotion also in the presence of dynamics, while still enabling the significant benefits (i.e., self-selecting, robust, emergent behavior) of the behavioral-based approach. A block diagram of the proposed approach is shown in figure 3.13.

The controller can fundamentally divided into two part. The high-level

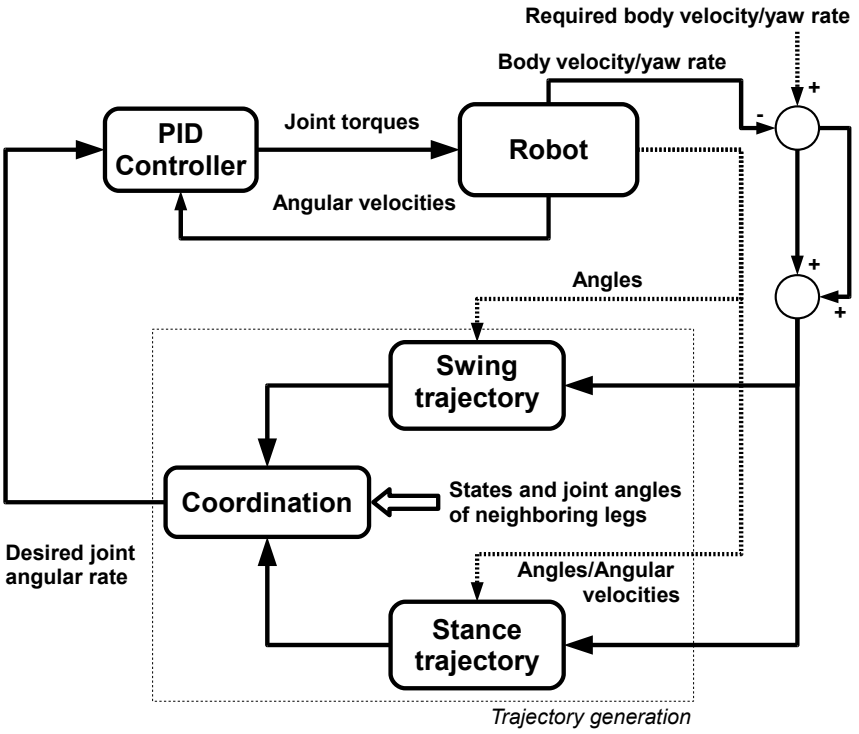


Figure 3.13: Block diagram of the complete control structure.

module called *Trajectory generation* gives as output the desired motion in term of required angular joint velocities. Its inputs are a mix of measured parameters (ground contact, joint angles) and calculated ones (body, yaw and angular velocities). The structure of this subsystem is based largely on that of the Cruse system in that there are independent blocks that generate the swing and the stance trajectories and a third block that chooses which of the two trajectories to use, as shown in the previous sections.

The main difference consists in a low level controller that manages the angular velocities calculated by the higher levels, giving the correct command for the actuators. The low-level module is a simple Proportional-Integral-Derivative (PID) controller that acts on the single joint. It receives as input the error between the measured and the required angular velocity giving to motors the correct torque. This solution increases also the flexibility of the walking system as a whole, robot and controller, allowing to change only the block of trajectory generation to validate other higher-level control strategies.

Chapter 4

Single leg controller

In the present chapter the controller for a single leg will be described in detail. To understand the whole structure, the architecture and the results for each single module will be illustrated. For a simulation of the complete leg motion see section 6.2.

4.1 Swing control

The control of the swing phase can be achieved with a single module that generates the desired trajectory during this phase. This trajectory, as previously described in section 3.5.1, must show two different behaviors: the lift off from the ground and the targeting on the AEP. In the present work the control module consists in an artificial neural network called *swing net*.

The variables of the problem are the angular positions and the three target angle that define the AEP univocally. The current angles can be measured directly by angular displacement sensors while the AEP angles are calculated apart and reflect a spatial coordination between adjacent legs (see section 5.4). On the basis of these two kinds of inputs it's possible to calculate the desired joints angular velocities for each time step.

The training process of each ANN requires the definition of a fitness function (see 2.6). In the present case the network must simulate two different behaviors but only one of them requires to be scored in the fitness function. Every quantitative evaluation of the lift-off properties, needs a knowledge of the shape of the whole trajectory that isn't required to correctly design a swing phase. For this reason, instead of a curve-fitting criterion, a Boolean one has been adopted: the solution is considered suitable only if the trajectory shows a SEP higher than a reference value. The score given by the fitness function evaluates the targeting on AEP only in terms of minimum distance of the foot from the desired position.

The fitness is evaluated in a purely kinematics simulation of the leg motion. To stop this simulation a temporal criterion has been used: when the time reaches a given value the simulation stops and the evaluation of the fitness is done on the best result obtained, i.e. the closest to the target. To speed up the swing phase, when a sufficiently precise target has been reached, one more training must be done to maximize the mean angular velocities. The maximum allowed velocity depends on actuators characteristics and it's equal to 91 $\mathrm{deg\,s^{-1}}$ (see subsection 7.3.1).

The other requirement in the project of a ANN is the structure of the network, meaning with this definition the type of neuron, the number of units and the connection among them. Two distinct cases have been investigated: static ANN (see subsection 4.1.1) and dynamic ANN (see subsection 4.1.2), by evaluating if the latter more complex solution might perform better than the simpler former one.

4.1.1 Static network

The usage of a static network to control the swing phase is the solution proposed by Cruse et al. [32] for their Walknet controller. The reference paper for this application has been written by Linder [33]. In that work he suggests that the simplest possible network was sufficient to solve the problem. This ANN is a single-layer feed-forward network consisting of three neural unit, the outputs of which are the angular velocity of the BC ($\dot{\gamma}$), CT ($\dot{\beta}$) and FT joint ($\dot{\alpha}$) respectively . Each unit receives a weighted input from external sensors that signal the current joint angles (γ, β, α). Further inputs are the target angles (γ_t, β_t, α_t) from the target net.

In a detailed study using genetic algorithms, Linder shown that seven non-zero weights out of a total of 18 are sufficient to simulate typical swing movements. The structure of the network is illustrated in figure 4.1. This solution consists essentially of three negative feedback controllers, one for each of the three leg joints. The control loop is closed via the leg itself, as movement changes the posture and, thus, the sensory input. Moreover, the controller of the BC joint (protraction) and the CT joints (elevation) are coupled via a seventh weight that is responsible for the lift-off/touch-down behavior. Its variation allows to modulate the maximum height reached by the trajectory (SEP). A sensitivity analysis on this parameter shows how an increase in the SEP produces a decrease in the accuracy of the targeting behavior. From a mathematical point of view the network structure can be summarized by the equations set as follows:

$$\dot{\gamma} = w_{\gamma\gamma}\,\gamma + w_{\gamma\gamma_t}\,\gamma_t \tag{4.1a}$$

$$\dot{\beta} = w_{\beta\gamma}\,\gamma + w_{\beta\gamma_t}\,\gamma_t + w_{\beta\beta}\,\beta + w_{\beta\beta_t}\,\beta_t \tag{4.1b}$$

$$\dot{\alpha} = w_{\alpha\alpha}\,\alpha + w_{\alpha\alpha_t}\,\alpha_t \tag{4.1c}$$

Starting from these considerations the weights for this network adapted to the present geometry have been calculated. The parameters for the training algorithm are shown in table 4.1. Remembering the characteristics of the behavior to simulate, it's possible to fix some constraints on the weights:

- the maximum permitted magnitude is 2;

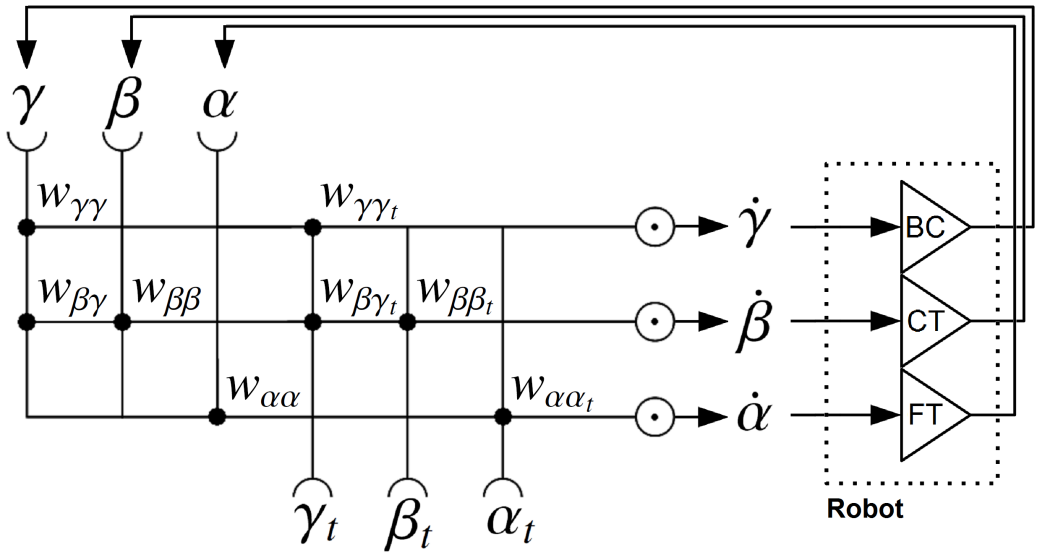


Figure 4.1: Static swing net.

- the auto feedback weights on measured angles are all negative;
- the auto feedback weights on target angles are all positive;
- the BC influence on CT has a positive weight on the measure and a negative one on the target;
- the two weight that calculate $\dot{\alpha}$ have the same magnitude but opposite sign.

The fitness function evaluates when the minimum distance from the AEP is reached during the simulation and then, for this time, calculates the score using the error between each angle and its desired value. This process occurs only if the SEP is beyond the limit value, otherwise a standard value, equal to three times the maximum value that the fitness can assume, is given to the score.

The training process must take into account the changes either in starting or in targeting position. For this reason each evaluation uses three different values for the x and the z of the two positions: the reference one, an upper one and a lower one; one value only has been considered for the y because it shows minor variations in the real environment. This produces a total of 81 simulations to be evaluated for each individual, accepting as total score, the worst one obtained.

The angular displacement and velocities are shown in figure 4.2 for the reference case and some resulting trajectories in figure 4.3. It's possible to

N° of variables	7
Population size	500
Generation number	50
Elite count	10
Mutation	Adaptive
Crossover fraction	0.8
Threshold score ($\bar{\Psi}$)	5
Error level on angles [%]	5
Minimum SEP [mm]	z_{AEP}+30

Table 4.1: GA parameters for the static swing net training.

underline how this very simple ANN is able to produce precise trajectories on a wide range of PEPs and AEPs. This behavior shows also a great robustness since it reacts in a very effective way to external disturbances as shown in figure 4.4. This robustness can be explained looking at the velocity field (see figure 4.5): the reference target is an attraction point for the foot movement on the x-y plane. Along the z direction the attraction point is placed below the plane of the target position in order to ensure the contact with the ground. In figure 4.6 there is illustrated the trend on error due to variation in the starting position (left side) and in the targeting position (right side); in both the evaluations, the other position is the reference one. It's possible to see how the minimum error is reached very close to the reference position and shows more sensitivity to a variation of the target than to a variation of the starting position.

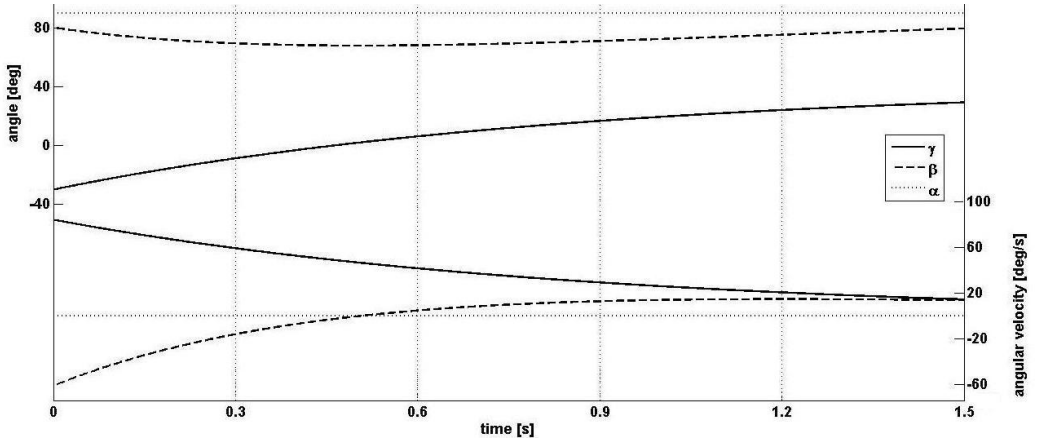


Figure 4.2: On the top angular displacement of a swing trajectory in the reference case. On the bottom the correspondent angular velocities.

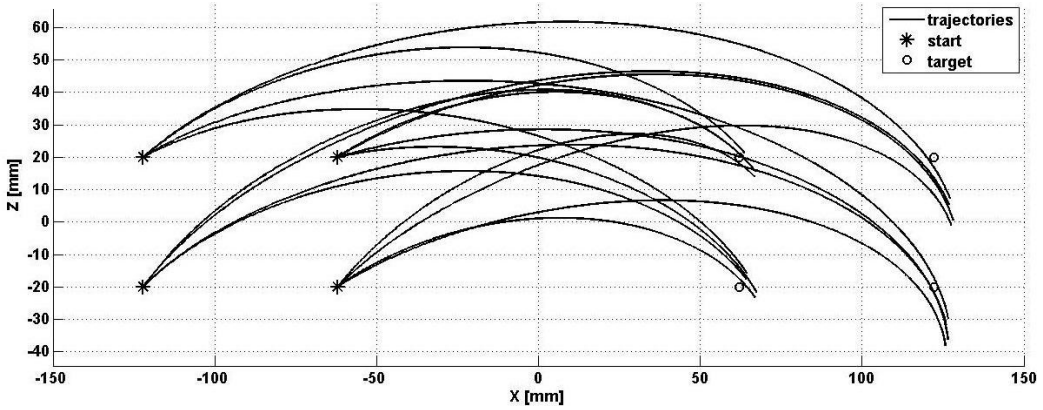


Figure 4.3: Swing trajectories with different start and target positions.

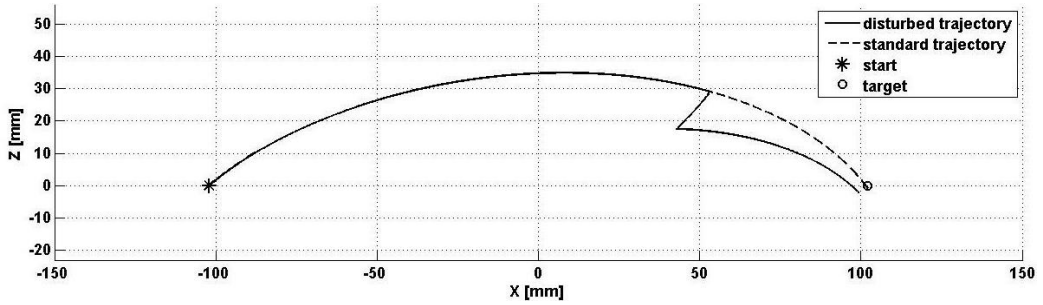


Figure 4.4: Response of the swing controller to an external disturbance.

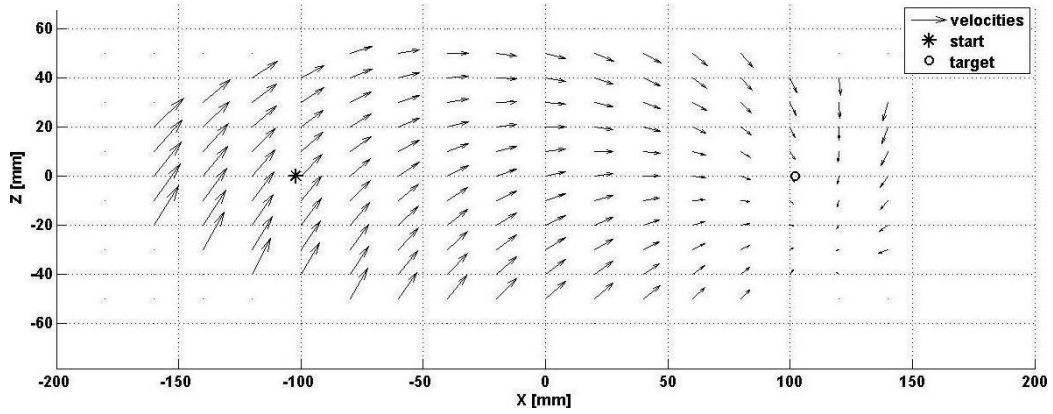


Figure 4.5: Field of velocities.

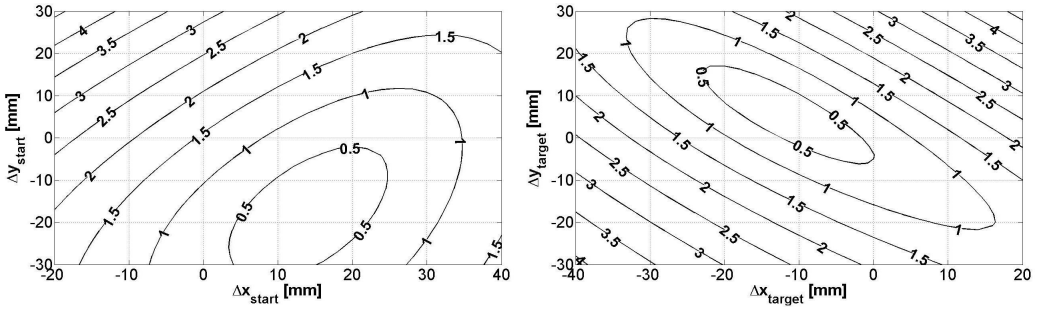


Figure 4.6: Error on target as function of trajectory parameters: on the left starting position, on the right reference target.

4.1.2 Dynamic network

The choice of a dynamic neural network is based on its superior capability to simulate a time-dependent behavior. CTRNNs will be investigated for the reasons illustrated in section 2.4. In such a network it's necessary to understand which are the states and which are the inputs: in this case it is immediate to choose the current angles as state and the target ones as inputs. This choice allows to update the state with the measured angular displacements and to use directly the state derivative calculated by the network without any integration, reducing numerical problems in both cases.

The work on CTRNNs starts from the results obtained with the static swing net. For this reason the first trained network consisted of three uncoupled neurons, with a connection only from γ to β unit. This structure did not allow to produce a swing motion because it was not able to simulate both the required behaviors. Training with completely interconnected versions of this network has not produced significantly better results. The solution of this problem can be found by adding a neuron that is completely interconnected with the β unit and that receives the same kind of influence from the γ one. The mathematical model of the dynamic recurrent ANN can be summarized by the equations set as follows:

$$\tau_\gamma \dot{\gamma} = \left(w_{\gamma\gamma} \ \tanh(S_\gamma) + w_{\gamma\gamma_t} \ \gamma_t \right) \tag{4.2a}$$

$$\tau_\beta \dot{\beta} = \left(w_{\beta\gamma} \ \tanh(S_\gamma) + w_{\beta\beta} \ \tanh(S_\beta) + w_{\beta h} \ \tanh(S_h) + w_{\beta\gamma_t} \ \gamma_t + w_{\beta\beta_t} \ \beta_t \right) \tag{4.2b}$$

$$\tau_h \dot{h} = \left(w_{h\gamma} \ \tanh(S_\gamma) + w_{h\beta} \ \tanh(S_\beta) + w_{hh} \ \tanh(S_h) + w_{\beta\gamma_t} \ \gamma_t + w_{\beta\beta_t} \ \beta_t \right) \tag{4.2c}$$

$$\tau_\alpha \dot{\alpha} = \left(w_{\alpha\alpha} \ \tanh(S_\alpha) + w_{\alpha\alpha_t} \ \alpha_t \right) \tag{4.2d}$$

remembering that the variable S_i are the weighted sum defined as follows:

$$S_i = y_i + \theta_i \tag{4.3}$$

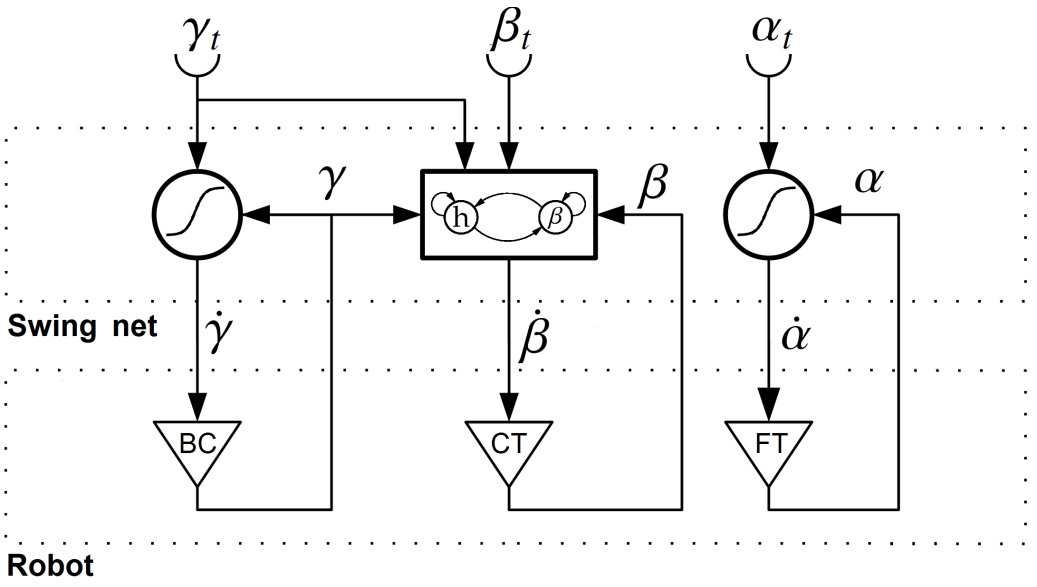


Figure 4.7: Dynamic swing net.

where y_i is a general state. The state h, also called hidden state, has been added to correctly recontruct the behavior of the CT joint angle. A scheme of the network is shown in figure 4.7. The recurrence of the network is guarantee by the leg that acts as an integrator on the three angular velocities produced by the swing net and fed back. The hidden state is integrated numerically. With these links the total number of weights decreases from 36 in the full connected network to 22, but it is much higher than in the static case. Moreover no boundaries on the weight can be introduced because the nonlinear properties of the model doesn't allow an easy interpretation of each single contribution.

To improve the efficiency of the whole process the best solution is to uncouple the training of FT joint angle from the other two. This is possible because there isn't any functional dependence between α and the other angles, but there is a temporal synchronization only. The parameters for these training algorithms are shown in table 4.2. In both these cases too, as in the static one, if the SEP is below its limit value an high score is assigned to it and the variation on starting and targeting positions have been taken into account.

The first training is on γ and β: it involves 18 weights and considers an α that linearly moves from its starting to its targeting value. The fitness evaluates the error on target angles at the time that it minimizes the distance between current and target position. In the CTRNN training the distance is measured in the angular displacement space instead of in the

	BC+CT	FT
N° of variables	18	4
Population size	10000	100
Generation number	50	50
Elite count	40	10
Mutation type	Gaussian	Gaussian
Mutation scale	20	20
Mutation shrink	1	1
Crossover fraction	0.8	0.8
Threshold score ($\bar{\Psi}$)	5	5
Error level on angles [%]	5	1
Minimum SEP [mm]	$(z_{PEP}+z_{AEP})/2+30$	$(z_{PEP}+z_{AEP})/2+30$

Table 4.2: GA parameters for the dynamic swing net training.

Cartesian coordinates one, in order to speed up the convergence of the algorithm. This training is very computationally expensive because only a few trajectories have the correct shape.

After that, the simpler training on α is performed. For the part of the network that regulates the motion of the other angles, the weights obtained at the previous step have been used. The error between the FT angle and its target to evaluate the score has been taken. In this training also the variations on y of PEP and AEP have been considered, since the α motion heavily depends on them. The synchronization is automatically reached because the error is calculated for the position that minimizes the distance from the target in the Cartesian space.

Note how the first case requires a number of simulation two order of magnitude greater than the second one: the higher number of parameter and the much more complex behavior to simulate, justify this difference. To accelerate the convergence, the error level for the first training has been relaxed. In the other one, a greater precision can be obtained without any problem.

The obtained trajectories are shown in figure 4.9. Trajectories are less precise than in the case of a static swing net, but the error is sufficiently low in the standard operative range. Moreover the incorrect targeting follows reasonable behavior, tending to minimize the distance from the reference target. The lower accuracy is compensated by an improved robustness. The velocity fields (see figure 4.11) shows an attraction point similar to its static counterpart but the behavior is much stronger because is able to compensate an higher level of disturbances.

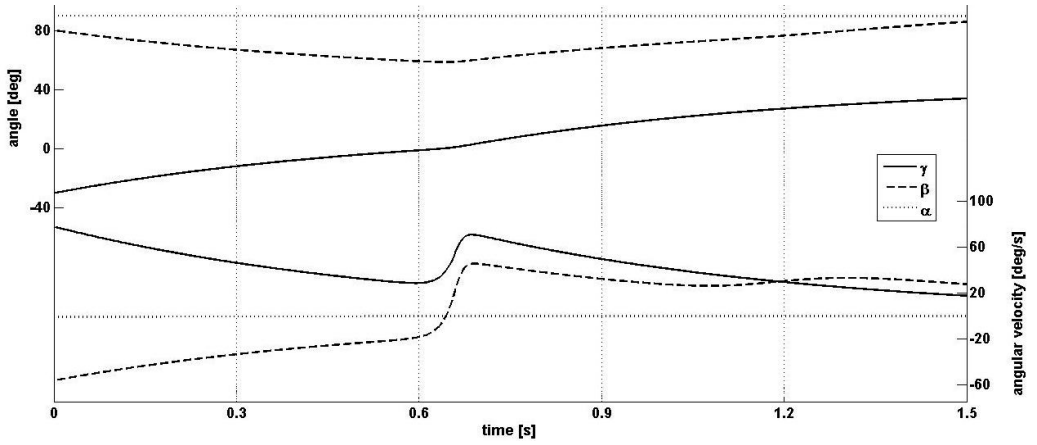


Figure 4.8: On the top angular displacement of a swing trajectory in the reference case. On the bottom the correspondent angular velocities.

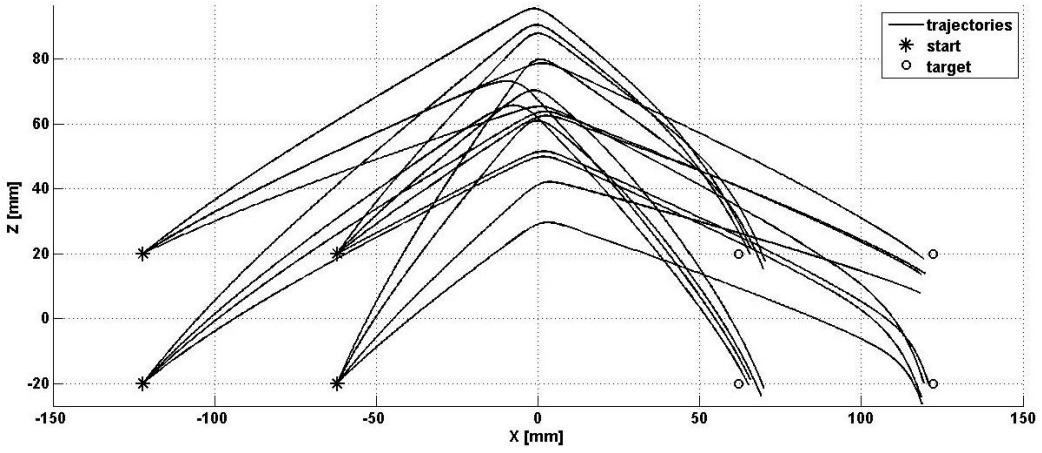


Figure 4.9: Swing trajectories with different start and target positions.

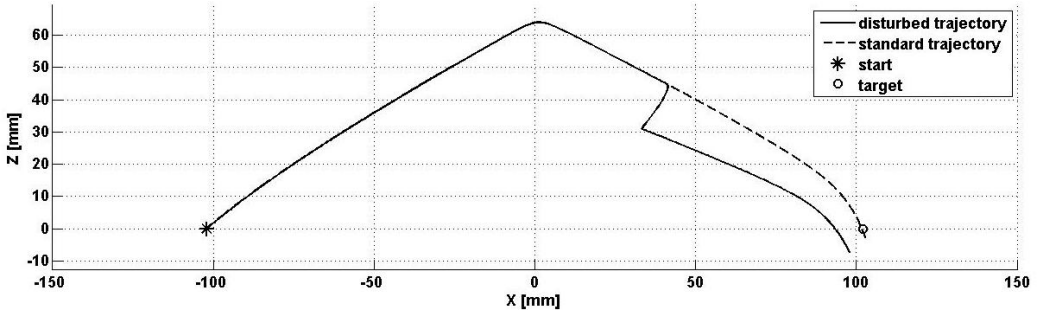


Figure 4.10: Response of the swing controller to an external disturbance: the perturbed trajectory return autonomously to the reference target.

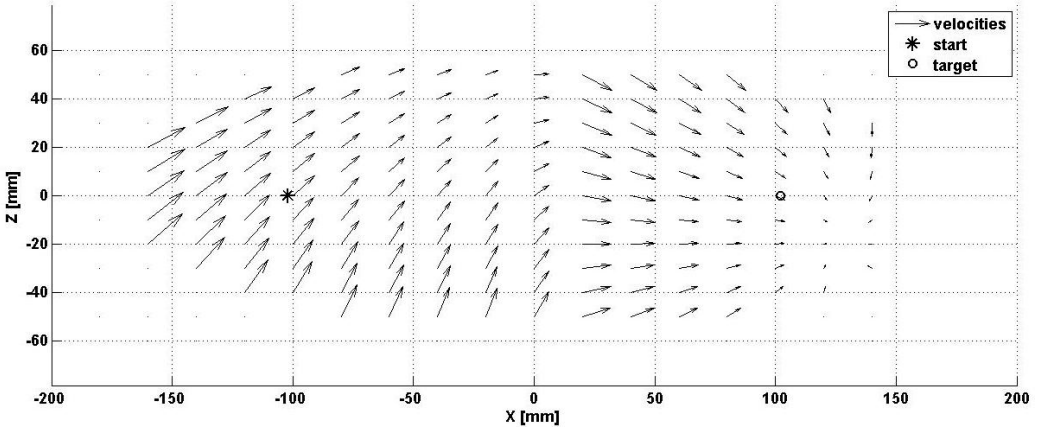


Figure 4.11: Field of velocities.

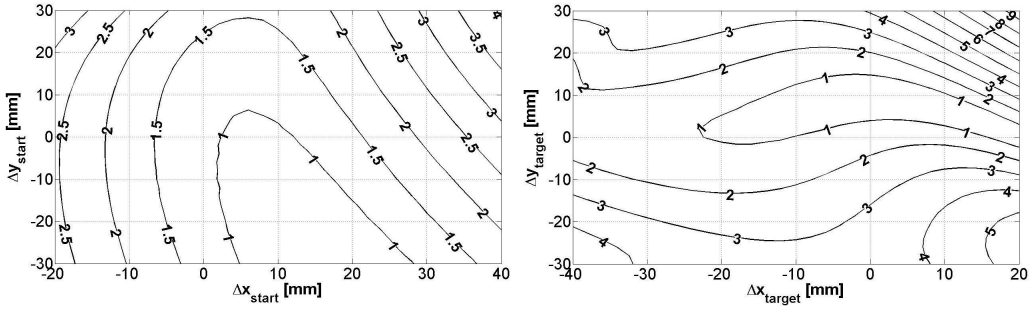


Figure 4.12: Error on target as function of trajectory parameters: on the left starting position, on the right reference target.

In figure 4.12 there is illustrated the trend on error due to the variation in the starting position (left side) and in the targeting position (right side); in both the evaluation, the other position is the reference one. It's possible to see how the minimum error, in this case, isn't reached for the reference position, but is less sensible to variation in starting y and more sensible to variations in targeting y, than the static swing net and the opposite behavior can be observed for the x coordinate (see figure 4.6).

4.2 Stance controller

In subsection 3.5.2 the main features of stance control have been already explained. At first the LPVF controller on the BC and CT joint angles will be implemented. After that, it will be necessary to introduce the other aspects that involve the power stroke, especially the height, velocity and direction control. In the last part of the section a more comprehensive ap-

proach to the problem will be discussed briefly. Since the stance controller has been implemented in a purely kinematical form, but can be evaluated only in a dynamical contest, all the results of the present section have been derived from the dynamic simulation (see chapter 6).

4.2.1 Local positive velocity feedback

The Local Positive Velocity Feedback (LPVF) solution allows one to solve the problem of controlling the BC and FT joints in a very simple form maintaining a decentralized architecture. The problem is how to effectively implement this solution. The LPVF, as shown in figure 4.13, is fundamentally a closed-loop system with an High Pass Filter (HPF) on the feedback path. The scheme of this system is very simple with a first order HPF on the feedback path and a unitary gain on the forward path. A typical first order high-pass filter has a transfer function that can be expressed as follows:

$$G(s) = \frac{\tau s}{1 + \tau s} \tag{4.4}$$

where τ is the time constant of the HPF, the inverse of the cutoff frequency. The closed-loop transfer function between input and output becomes:

$$y(s) = (1 + \tau s)x(s) \tag{4.5}$$

This means that the output of such a system corresponds to the sum of the input and the derivative of the input. In the present case the input is the angular displacement and its derivative is the angular velocity. This structure allows one to solve the main problem of the normal simple positive feedback. Feeding directly back the joint angle, the correspondent angular velocity could continuously increase in an unpredictable way. The introduction of the HPF avoids this behavior by maintaining the speed almost constant as required to a stance control system. In this case, for example, a short velocity impulse given at the input, after the pulse, leads to an almost continuously constant velocity output value.

Considering as input the measured current joint angle and as output the required one, it's possible to formulate the problem in the discrete time domain, where the current value is referred to step k and the required to step $k + 1$. With this assumption, the LPVF controller can be implemented, for the angle α, as follows:

$$\alpha(k + 1) = \alpha(k) + \tau\dot{\alpha}(k) \tag{4.6}$$

In the discrete time domain the angular velocity can be calculated by a first order backward finite difference formula as:

$$\dot{\alpha}(k + 1) = \frac{\alpha(k + 1) - \alpha(k)}{\Delta t} \tag{4.7}$$

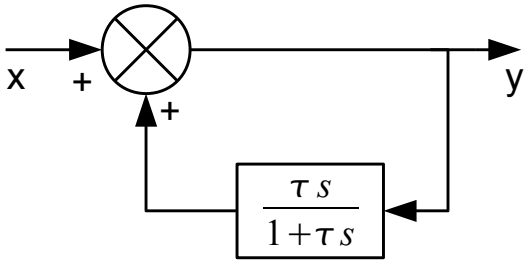


Figure 4.13: High pass filtered positive feedback.

Taking $\alpha(k+1)$ from the 4.7 and substituting it in the 4.6 it's possible to alternatively formulate the problem as follows:

$$\dot{\alpha}(k+1)\Delta t + \alpha(k) = \alpha(k) + \tau\dot{\alpha}(k) \tag{4.8}$$

that allows to found the required relation among the angular velocities as:

$$\dot{\alpha}(k+1) = \frac{\tau}{\Delta t}\dot{\alpha}(k) \tag{4.9}$$

The formulation is the same for the γ angle. This means that the LPVF can be completely implemented in terms of angular velocities by simply multiplying the current value for a given gain. The gain depends on the integration time step but this results can be easily justified remembering that a positive feedback system is stable only when is verified the following condition:

$$\frac{\tau}{\Delta t} < 1 \tag{4.10}$$

Since the time constant τ of the HPF, to correctly manage the problem, should depend on the sampling frequency that is the inverse of the time step Δt, the total gain K that multiply the angular velocity can be taken as constant independently to the time parameters. To correctly evaluate them a brief hand tuning has been performed. The chosen values are K_γ=0.0086 and K_α=0.0057. Lower values produce a stance phase with a SEP too low, higher value cause the saturation of the angular velocity and lead to instability. In order to avoid backward walking induced by directional biases, e.g. when gravity opposes the walking direction on a steep slope, a supervisory system has been introduced. It specifies the correct walking direction, but allows also to stop or start the entire system. This is also necessary to avoid any influences of the previous swing phase on the succeeding stance phase. The central bias in walking direction is caused

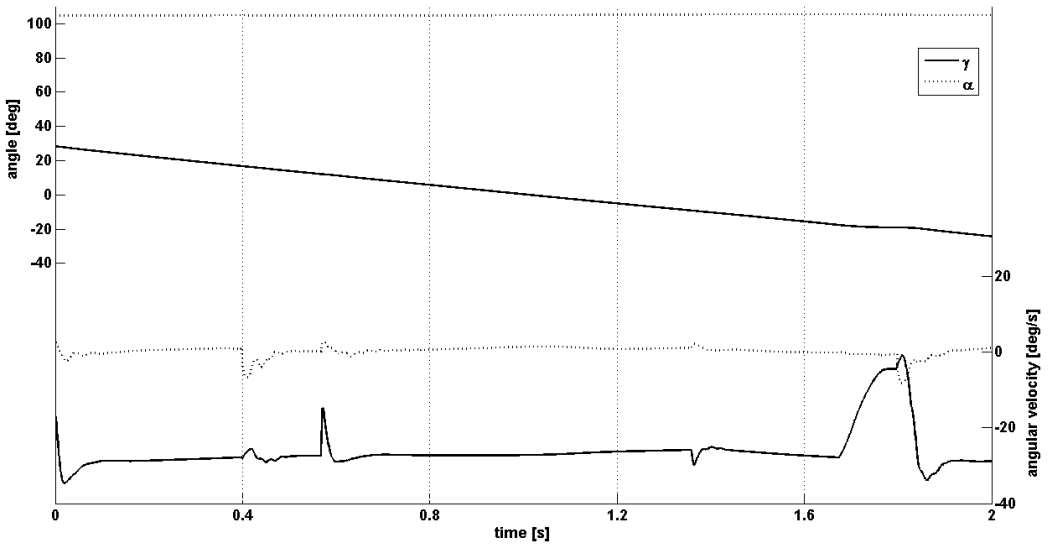


Figure 4.14: Angles and angular velocities of a reference stance trajectory in a dynamic simulation environment.

by a constant, small offset value, which replaces the sensory information of the BC joint angular velocity $\dot{\gamma}$ if it is larger than the latter. This added contribution can also be seen as a pulse that allow to the LPVF to reach a non null constant angular velocity. The offset value has always been taken equal to 5 deg/s, less than 6 % of the maximum permitted value.

This alternative formulation allows to give directly an angular velocity command to the actuator as required in section 3.5. The problem is that the sensors mounted on the joints measure angular displacements only. To obtain an information about the angular velocity is necessary to numerically calculate the derivative of the measure. In the present work a second order backward finite difference formula has been used.

The stance phase, although it's implemented by a kinematic rule, could be evaluated only in a taking into account the interactions with the environment. For this reason the result shown in figure 4.14 have been obtained by a dynamic simulation (see chapter 6). During a reference stance phase the most part of the motion depends on the γ angle that move the leg backward, propelling the body forward. The angular velocity shows the behavior previously predicted for a system with LPVF, remaining almost constant after a starting initial perturbation, given by the walking direction offset. The angle α shows small variations only, due to numerical error, but also to the motion of the other legs. This behavior can justify a different control strategy, e.g. a closed-loop negative feedback to increase lateral stability.

4.2.2 Height control

The control of the body clearance depends completely on the CT joint in order to maintain the modularity approach adopted so far: the γ and α angles are responsible for ground parallel movements and thus they provide the thrust by using LPVF, whereas the β angle doesn't show any reversal reflex, but controls the movement in the direction perpendicular to the ground. This action is performed by the simplest control system possible. The required angular velocity is calculated by a classical closed-loop regulator that uses negative feedback. The actual body height, evaluated at the leg insertion point, is compared to a reference value. The resulting error is transformed into a desired angular velocity, equivalent to a desired change in the angular position in the discrete time domain, by a linear characteristic. The choice of using a simple proportional controller has its biological counterpart in the findings on height control in stick insects that show an individual proportional element acting as a servomechanism on each leg (see Cruse et al. [34]). The control law can be summarized as follows:

$$\dot{\beta} = K_\beta(h_{sens} - h_{ref}) \tag{4.11}$$

The proportional gain K_β has been set equal to $2.5\,\mathrm{deg\,s^{-1}\,mm^{-1}}$. The body height in the reference configuration has been considered as reference body height h_{ref}. The results obtained with this controller are illustrated in figure 4.15. If the ground has a step (bold line), when the leg passes from swing (dashed line) to stance (solid line), it sees a change in the local body height. The controller tries to maintain the reference body height (dash-dotted line), by feeding-back a $\dot{\beta}$ command, proportional to the error on height. The system reacts by reaching the regime value in less than 0.5 s. This results can be justified remembering that the linearized model of the closed-loop system can be represented by a first-order transfer function with an integrator on the forward path, that transform $\dot{\beta}$ in h, and a unitary gain on the feedback path. The gain of the integrator is K_βG where G is the derivative of h(β) evaluated in the reference position that assume the value of $2.65\,\mathrm{mm\,deg^{-1}}$. The resulting closed-loop transfer function is:

$$H(s) = \frac{K_\beta G \frac{1}{s}}{1 + K_\beta G \frac{1}{s}} = \frac{1}{1 + \frac{1}{K_\beta G} s} \tag{4.12}$$

This means that the time constant can be estimated as $\tau \approx 1/(K_\beta G)$. The obtained value of τ=0.15 s means that the regime condition should be reached in about 4τ=0.6 s that is almost exactly the data given by the simulation.

The value of the current body height h_{sens} isn't directly measured by any sensor but can be easily deduced by the measure of the joint angles.

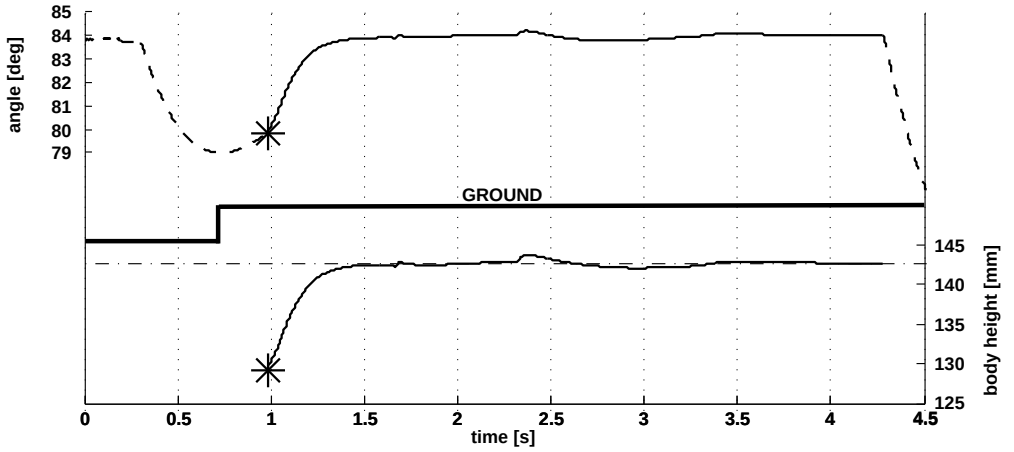


Figure 4.15: Body height control.

The distance of the body from the ground corresponds to the z coordinate in a Cartesian system centered in the BC joint. The value can be calculated by the analytical expression as follows:

$$h_{sens} = L_t \cos(\alpha - \beta) - L_f \cos(\beta) \tag{4.13}$$

where L_t and L_f are the length of the tibia and femur sections respectively. The complete analytical formulation uses twice a trigonometric function: this increase a lot the computational cost of this function. The best way to reduce the simulation time is to approximate the function by an ANN as previously shown in section 2.4. In the present case the simplest network possible has been used, with two outputs, one input and only one neural unit on the hidden layer. The calculation time decrease of almost one order of magnitude from the 5.03×10^{-5} s of the analytical case to the 9.1×10^{-6} s of the neural one. Comparison between the standard and the approximate results are shown in figure 4.16 where a mapping of the approximation error due to the ANN as function of the CT and FT joint angles (left side) and of the foot position on the y-z plane (right side) is illustrated. Since the network has been trained to best-fit the reference position, the error is smaller, when the foot is closer to this one. The error level is always lower than the 5 ‰ of the reference height.

In the present work the desired body clearance h_{ref} has been taken as fixed. A further development might be the introduction of a higher-level controller that vary the reference height of each leg in the same way, in order to regulate the total body height or in a different way, to change its attitude. These results can be obtained without any variation on the single leg control structure, but only by adding an additional module.

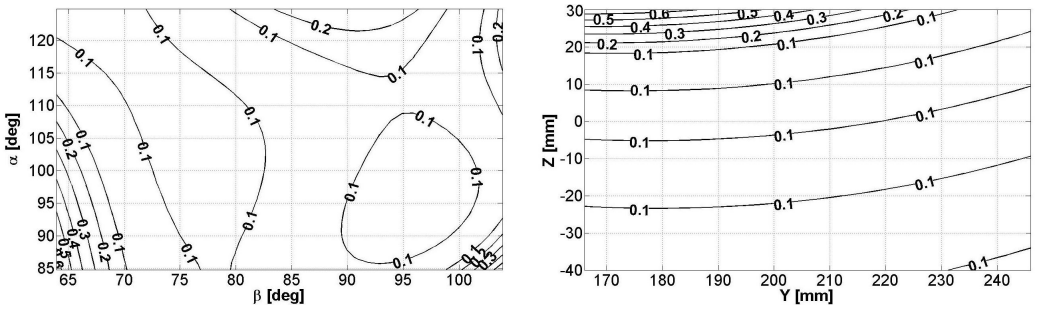


Figure 4.16: Error on body height calculated by the ANN.

4.2.3 Velocity control

To maintain a constant advancing speed it is necessary to correct the values of the angular velocities previously calculated. Since the movements on the plane parallel to the ground mainly depend on the γ and α angles, the correction has been applied only on them. The velocity correction works as a classic negative feedback supervisor that modifies the amplification of the positive feedback control signal by a gain K_v. The required angular velocities can be calculated as follows:

$$\dot{\gamma}(k+1) = K_\gamma \, K_v \, \dot{\gamma}(k) \tag{4.14a}$$

$$\dot{\alpha}(k+1) = K_\alpha \, K_v \, \dot{\alpha}(k) \tag{4.14b}$$

where K_γ and K_α are the fixed static gains of the LPVF controller. To evaluate the gain K_v it is necessary to define the error e as the difference between the required velocity V_{ref} and the measured velocity V_{sens}. The required speed descends from a higher control level, e.g. a human operator or a pattern optimizer, and can be varied over a normalized range from 0 to 1 where 0 means a null speed and 1 indicates the maximum allowed value (see section 5.5). The current advancing speed of the single leg can be obtained indirectly from the angular variables as follows:

$$\begin{aligned} V_{sens} = &- ((L_c + L_f \sin\beta + L_t \sin(\alpha - \beta)) \cos\gamma\dot{\gamma} + \\ &+ (L_f \cos\beta\dot{\beta} + L_t \cos(\alpha - \beta)(\dot{\alpha} - \dot{\beta})) \sin\gamma) \end{aligned} \tag{4.15}$$

The correction depending on advancing speed can be calculated as follows:

$$K_v = \begin{cases} 1 + e & : \mathrm{e} \geq 0 \\ \frac{1}{1-e} & : \mathrm{e} < 0 \end{cases} \tag{4.16}$$

These nonlinear characteristics, proposed by Schmitz et. al [35], ensures values either bigger or smaller than 1 at the output when walking too

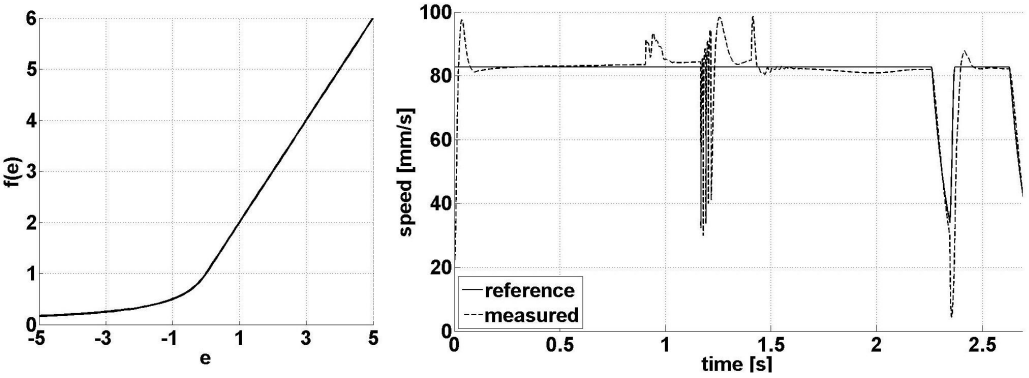


Figure 4.17: Control of the advancing velocity.

slowly or too fast respectively. Its trend in function of the error on the advancing velocity is shown in figure 4.17 (left side). As the global output of the controller is based on global (V_{ref}) and local (V_{sens}) data, it is different for each leg in stance phase: this amplifies or weakens the control signal to compensate its own error. This can produce little variations in the direction, but it's in accordance with the decentralized structure approach followed thus far. The control of the global velocity accords with the biological findings on stick insects reported by Cruse [36], who found a velocity control rather than a position control during leg retraction.

Like each other stance parameter, the advancing velocity too needs to be evaluated in a dynamic environment (see chapter 6). The results for a typical ripple gait are shown in figure 4.17 (right side). The actual trend fits very well the reference value except for some little variation and a sudden oscillation that happens at 1.2 s. The variations can depend on phase transitions of the leg (initial time), variations in the reference signal that the leg doesn't follow correctly or adjustments in the body attitude due to the motion of other legs. The oscillations depend on numerical errors that may occur when a neighboring leg touches the ground. They are not present in motion of a real robot.

4.2.4 Direction control

The direction control, in the same way as height and velocity one, relies on a negative feedback strategy. Owing to its closed-loop nature, this system can compensate for unbalanced coupling factors or other inequalities between right and left legs. The yaw-turning velocity is assumed to be known, i.e. rotation around the body vertical axis, the deviation between the desired and the actual headings is determined. This error signal is subtracted from the BC joint angles of all the legs, with opposite signs for

left and right side. If the yaw reference is set to zero, the system moves straight with small, side-to-side oscillations in heading like all that can be observed in walking insects (see Kindermann [37]). To simulate curve walking, a small positive or negative bias is added to the reference value in order to determine the curvature direction and magnitude. This model makes it possible robust course control and performs very good also with very small turning radii, up to 1.5 times the total body length.

The actual yaw angular velocity can be estimated on the basis of the current joint angular variable of the supporting legs, but the analytical formulation of this problem is quite complex. Moreover in a real-world controller it can amplify every error on the measures. To avoid this problem it is necessary to introduce a angular rate sensor that hasn't been mounted yet on the robot. The best choice in this preliminary phase is to evaluate the direction control in the multibody simulation where a correct information on yaw rate is available. Moreover this behavior can be evaluated on a complete model of the robot only. For more detail see chapter 6.

4.3 State selector

The selector is the module deputy to govern the transition between swing and stance phase and thereby to control the rhythm of a leg's motion. In line with the conclusions of section 3.4 an oscillator based on external inputs has been used to control the rhythmic alternation of swing and stance phases. Cruse et al. [6] suggests that this requirement can be fulfilled in the easiest way by a module driven by two sensory input and the information about the activity of both the states that can be modeled as a simplified four-unit ANN.

In this network, the motor outputs that are active during return stroke and during power stroke are each represented by a single artificial unit (RS and PS in figure 4.18). The activity of a state is indicated by a Boolean truth value that works as switch for the velocities calculation: if the RS is true and the PS is false the angular velocities are given by the swing net, in the opposite case by the stance net; in the simulation the real leg is modeled as a triple integrator. Other situations cannot happen because the states are mutually exclusive. The sensory inputs are given by two distinct units: a ground contact unit (GC in figure 4.18), that signals the onset of foot contact with the substrate, and a PEP unit that observes whether the leg has reached a reference posterior extreme position (PEP in figure 4.18) or not. The GC unit simply consists in an evaluation of the force measured by the sensor mounted on the foot: if the value is higher than a threshold, chosen equal to 0.1 N, the GC becomes 1, otherwise 0. In figure 4.18 (right side)

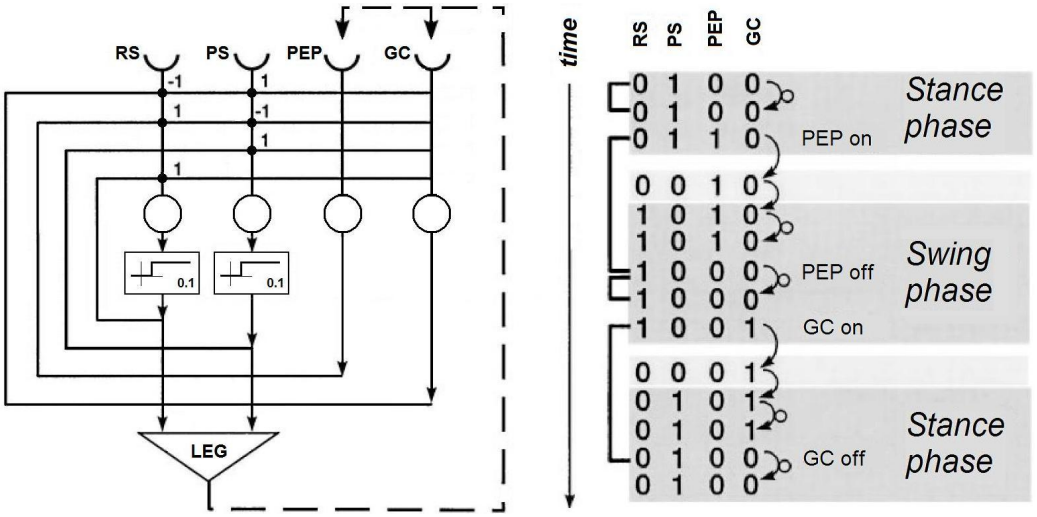


Figure 4.18: The selector network.

has also been shown an example of pattern of state transitions needed to produce a proper step rhythm. A true value indicates the activity of the state.

For the PEP unit a more complex structure is needed as shown in subsection 4.3.1. In this work the sensory inputs have been summarized as logical information, but is possible to manage directly the analogous values of forces and displacements if disposable. This four-unit network is made internally recurrent by allowing the outputs of each unit to feed back to its own input and to that of all other units. These connections can be found in lot of classic neural oscillators such as the CTRNN central pattern generator shown in section 2.5. The work done on Walknet shows how the training process always led to an unexpected result: each motor unit developed positive feedback to itself (see figure 4.18). Hence, state changes occur only when the simulated leg movement reaches a position to trigger a change in sensory input. In other words, the trained model relies on positive feedback to maintain the current step phase until a sensory unit signals that the leg has reached an appropriate endpoint. This positive feedback was unexpected because typical model to generate an alternative motion, such as neural CPGs, incorporates mutual inhibition between antagonistic motor pools. However, such a mutual inhibition, when implemented in the controlled system proved to be less stable in the sense that it was more easily disrupted by noise added to the activity of the units. Therefore, the positive feedback version was selected for the final control model.

The step activation function on the output units has a threshold value:

this simply means that the state can be considered active only when the correspondent output has a value larger than zero. The value for the threshold has been arbitrarily taken equal to 0.1 but any other value comprised in the interval (0,1). This is a good choice in the case of non Boolean sensory inputs.

This architecture, unlike a purely central neural oscillator or a mixed model, does make the model entirely dependent on the reliability of its sensory input. In the insect, the large number of units contributing to measure leg parameters should guarantee sufficient reliability, but in this model it isn't so. This implies that the robot will be less robust to each sensor failure and that with this kind of controller, the only way to improve the robustness is to mount redundant sensors. This is the main disadvantage of a completely reflex-based control strategy.

4.3.1 PEP net

The selector requires a Boolean information that indicates if the foot has passed or not its posterior extreme position. This evaluation can be done only by knowing both the PEP value and the current position of the foot. In order to simplify the problem the comparison is done in terms of Cartesian coordinate taking into account the x one only, because is the most significant. The PEP can be calculated starting from a reference value of 119 mm and adding all the coordinating influences (see chapter 5 for more details). The current x coordinate of the foot requires a more complex procedure. Since it's not directly measured, in opposition with the GC parameter, it needs to be calculated starting from the current joint angles as follows:

$$x = L_c + L_f \sin\beta \; + L_t \sin(\alpha - \beta)) \sin\gamma \tag{4.17}$$

This analytical expression, as the body height one, involves trigonometric functions that produces an increase in the computational cost. The solution is the same: replace the exact formulation with an approximating ANN. The structure requires three inputs, one output and a minimum number of hidden unit of three. With such a network the calculation time becomes one order of magnitude smaller than the original case: 6.95×10^{-5} s versus 7.35×10^{-6} s. A mapping of the approximation error due to the ANN as function of foot position on the x-y plane (left side) and of the foot position on the x-z plane (right side) is illustrated in figure 4.19. The error tends to be smaller closer to the reference position in the x-z plane, but not in the y direction due to the nonlinearity of the behavior. It increases faster after the reference PEP because it is a limit position for the motion. The error level is always lower than the 1 % of the reference length.

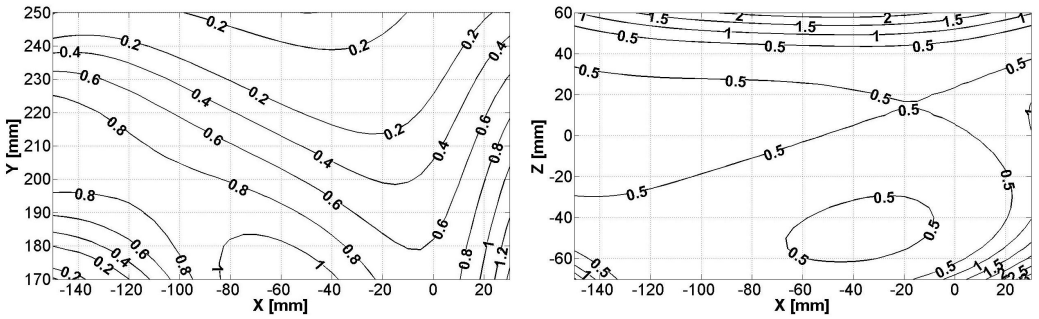


Figure 4.19: Error on PEP calculated by the ANN.

The complete PEP net is composed by three units: a sum node that translate the PEP from its reference value in dependency with the coordinating influences, an ANN that takes as inputs the current joint angles and calculate the current x coordinate of the foot and a switch node that evaluate if the foot is placed behind the PEP or not.

4.4 Reflexes

A reflex is a behavior that happens only when a sudden change appears in the environment. This means that the control of reflexes must be active only for a little fraction of the whole motion. In order to manage these situations it is necessary to create an independent module to control each reflex. When the change occurs, the module is activated and the calculation of the angular velocity for that leg skips from the current standard module (swing or stance) to the required reflex one. This strategy has been successfully adopted by the Case Western University team (see Espenschied et al. [38]) to control their Robot II (see figure 1.3). Another method to solve the reflex problem has been used in the latest versions of Walknet (see Cruse [1]). In this approach all the reflexes have been implemented as extensions of the swing net: this seems to be closer to the biology than the first one, but it requires a much more complex training process on the swing module. To maintain the modularity of the controller, the first approach is preferable and has been adopted in the present work.

4.4.1 Avoiding reflex

When an insect leg strikes an obstacle during its transfer phase, it initially attempts to avoid it by retracting and elevating it for a short time and then renewing its forward motion from this more favorable position. This behavior is called elevator reflex and has been adopted in lots of controller. A deeper analysis shows that there is not only a single elevator but at least

Reflex	Time [s]	$\Delta\gamma$ [deg]	$\Delta\beta$ [deg]	$\Delta\alpha$ [deg]
r1	0.1	–2	–5	—
r2	0.1	–4	–8	—
r3	0.1	—	–5	–5
r4	0.1	–2	+5	—

Table 4.3: Angles and duration time required for the avoiding reflexes.

four avoidance reflexes exist and are activated by the sensory feedback given by an obstacle:

- *r1*: a frontal stimulation of the tibia produces a levation and a retraction;

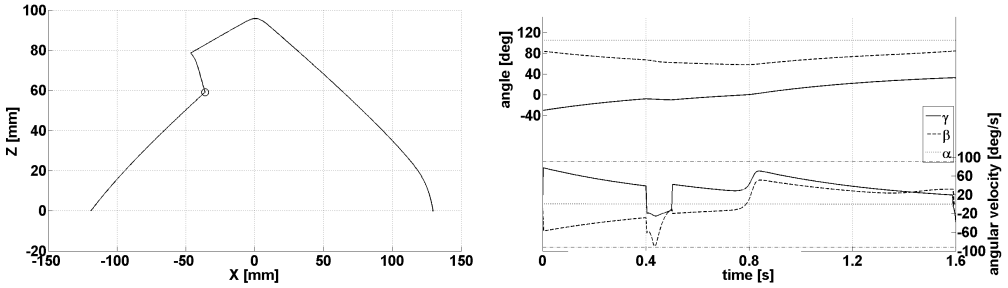
- *r2*: a frontal stimulation of the femur produces a levation and a retraction;

- *r3*: a lateral stimulation of the tibia produces a levation and a flession;

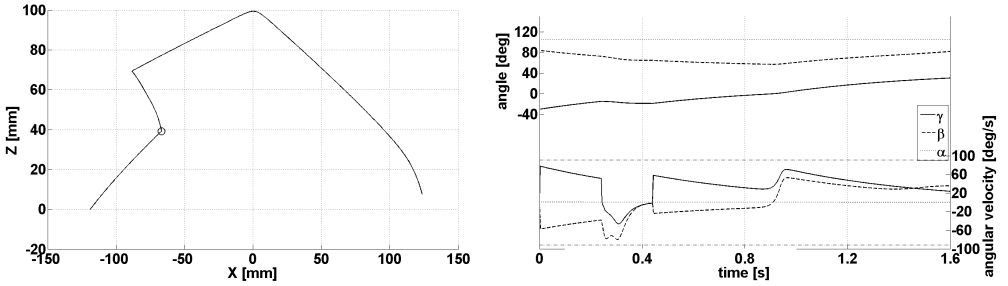
- *r4*: a dorsal stimulation of the femur produces a depression and a retraction.

It's very simple to reproduce an avoidance behavior, with a number of different strategies: in this work the avoiding reflexes have been implemented by four CTRNN all of them made by two completely interconnected units. Three parameters must be fixed to train the networks: the duration time and the two angular displacements imposed by the reflex. The value of each parameter has been hand-tuned by a trade-off between the duration of the reflex and the limits on the maximum permitted angular velocity. In Table 4.3 these data have been resumed. The complete training process is really fast due both to the little number of variables, that reduces the dimensions of the solution space, and the simplicity of the required behaviors. The parameters chosen for the GA, shown in table 4.4, have been the same used to train all the four avoiding reflexes.

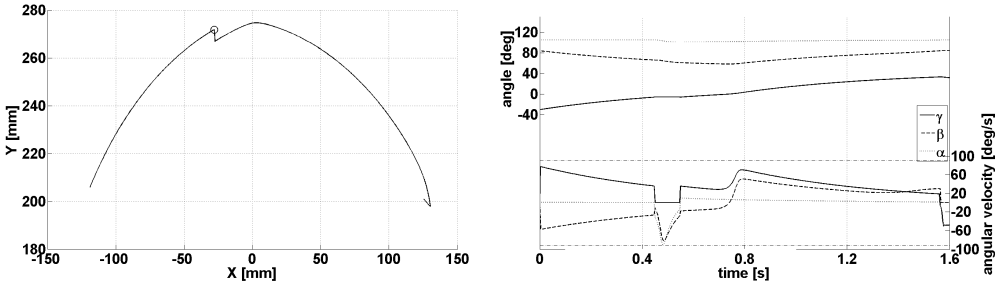
It might be interesting to understand how effectively the controller works: when the obstacle is detected, a sudden variation in the error on the measured angular velocity occurs. At that instant the current joint angles are memorized and a counter starts. The differences between the current joint angles and their values are taken as state variables for the neural network, when the external stimulation had been detected. The reflex is stopped when the counter reaches the assigned duration time.



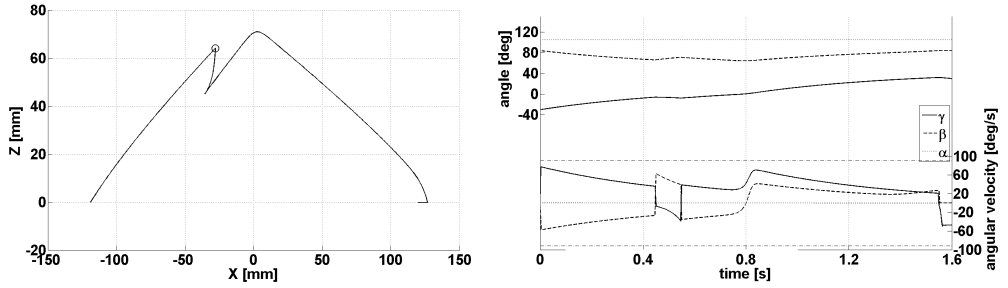
4.20.1: Reflex r1.



4.20.2: Reflex r2.



4.20.3: Reflex r3.



4.20.4: Reflex r4.

Figure 4.20: Angles and angular velocities during the avoiding reflexes.

N° of variables	8
Population size	100
Generation number	50
Elite count	4
Mutation type	Gaussian
Mutation scale	20
Mutation shrink	1
Crossover fraction	0.8
Threshold score ($\bar{\Psi}$)	5
Error level on angles [%]	5

Table 4.4: GA parameters for the training process of avoiding reflexes.

With the addition of the avoiding behaviors the motion of the leg becomes able to maintain its capabilities also when it faces obstacles of various nature. The effects on swing trajectories produced by these reflexes are shown in figure 4.20.

4.4.2 Searching reflex

On a natural irregular terrain, supporting surface may be missing, e.g. if there is a hole, or may be lost, e.g., if part of the terrain slides away from under a leg. Insects can successfully negotiate terrain with large gaps and then a walking robot must be able to do the same.

Stereotypical searching reflex in stick insect has been studied as a variation in the swing motion (see [39][40]). In this work a different approach has been adopted, following the one proposed by Pearson and Franklin [41] and applied by Espenschied et al. [38] on Robot II. This strategy consists in moving the leg rhythmically in an increasing region of space to search for a supporting surface. To fulfill this requirement efficiently, the controller must show three different micro behaviors:

- oscillating motion describing a circle in the leg movement sagittal plane (x-z);
- increasing size of the oscillation with each cycle;
- advancing motion in the direction of robot movement (x) and almost constant in the other two direction, especially z.

This behavior is much more complex that the other ones that have been shown up to now. The only way to simplify the design of this module is to realize a sub-module for each micro behavior.

Angle	Frequency [Hz]	Phase [deg]
γ	1.41	0
β	1.41	80.4
α	1.41	57.5

Table 4.5: Frequency and phase parameters for a searching reflex.

The oscillating movement can be obtained in the easiest way with a CTRNN like a CPG, as shown in section 2.5, with a unit for each angle. The training of a rhythmical generator can be done by using a fitness function based on the harmonic behavior of the neural states, as widely investigated in the previous works on NEMeSys (see [14] [15]). In this case two distinct purpose drive the choice of the fitness: the three angles must have the same frequency and must show a relative phase. The frequency and phase values have been calculated, once fixed the desired trajectories in the Cartesian coordinate space, by the analytical expression of the joint angles as function of x, y and z. The reference values, that allow to maximize the angular velocities, are shown in table 4.5. To reduce the size of the problem, two solutions have been adopted: since the frequency can be scaled only by changing the time constants, their value have been taken equal to 1 and hand-tuned later; the training of the α angle has been done only after γ and β and every influences of the FT joint on the other ones hasn't been considered. This strategy allows to reduce the total number of parameters from 12 to 10, but up to 6 for the single training process that has to investigate a much smaller space of solutions. To increase the size of the oscillations a gain with a linear time-dependency has been applied directly to the angular velocities given by the oscillators. The gain's parameters have been hand-tuned.

The advancing behavior is very complex too, because depends on a condition in the Cartesian coordinates space, but the problem is expressed in terms of joint angles. To obtain satisfying results, three CTRNNs, each of them composed by two completely connected units, have been used. These networks have been trained with the purpose to fit, with one of their two states, the ideal trajectory required for a joint angle and with an input corresponding to the target angular value that can be reached in the furthest allowed position. The score to minimize is the maximum error on the reference trajectory. Because each network is completely uncoupled to the other, except for the timing, three different training, with the same GA settings, have been done to reduce the time for each simulation.

The problems emerging from this structure are fundamentally two. The former is the use of two different networks to calculate the required

	Oscillating		Advancing (3×)
	γ+β	α	
N° of variables	6	4	8
Population size	2000	100	1000
Generation number	50	30	50
Elite count	10	2	2
Mutation type	Gaussian	Gaussian	Gaussian
Mutation scale	20	20	20
Mutation shrink	1	1	1
Crossover fraction	0.8	0.8	0.8
Threshold score ($\bar{\Psi}$)	5	5	5
Error level on angles [%]	—	—	5
Error level on frequency [%]	1	1	—
Error level on phase [rad]	0.1	0.1	—

Table 4.6: GA parameters for the training process of a searching reflex. Both the oscillating and the advancing behavior are present.

angular velocities, obtaining the outputs simply summing the two results. This means that it isn't immediate to insert the proprioreceptive feedbacks coming from the sensors. Same as in avoiding reflex module the idea is to use as states, not directly the angles, but values obtained subtracting the joint angles at the start of the reflex, to the measured angular displacement, but also, for each of the two networks, the ideal angles obtained when the other network only was active, calculated by a numerical integration. The latter problem is the need of a nonlinear offset to add to the β target angle to avoid excessive variations in the z coordinate.

The searching behavior starts when the difference between the target z_{AEP} and the current foot height becomes greater than a given threshold. The choose threshold value has been taken equal to 10 mm. The reflex stops when the foot touches the ground. If this event doesn't happen, the foot returns to its starting position with a stepping reflex and the motion of the entire robot must be stopped.

In figure 4.21 the results for the searching reflex controller are shown in term of joint angles and angular velocities. This module produces a sequence of oscillatory movements biased in the direction of motion that grow in amplitude with each cycle, as required by rules. This effects cause a constant growth in the angular velocities and drive to a saturation for the α angle in the last part of the reflex. The correspondent trajectories are illustrated in figures 4.22 and 4.23. The mean foot height remains almost

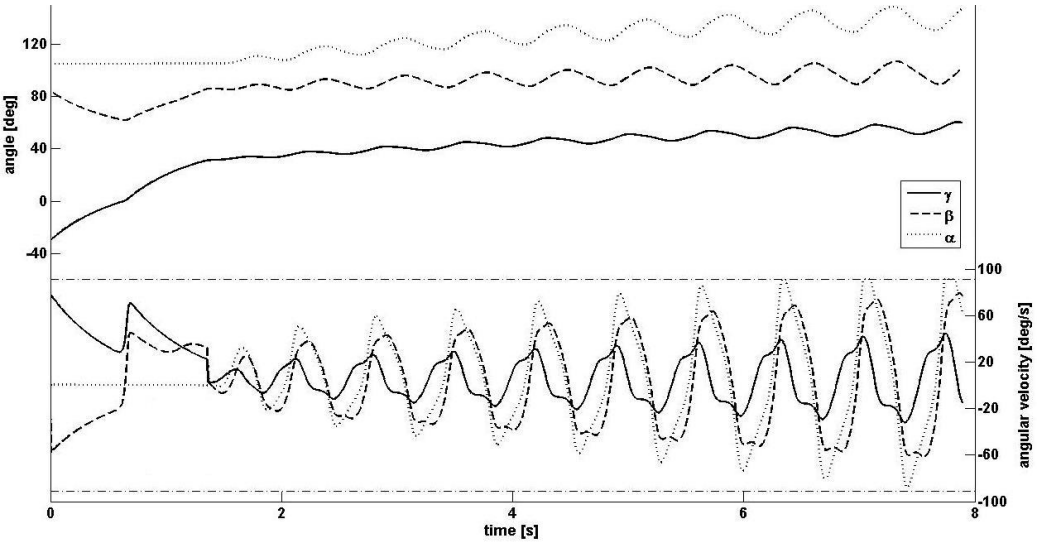


Figure 4.21: Angles and angular velocities during a searching reflex.

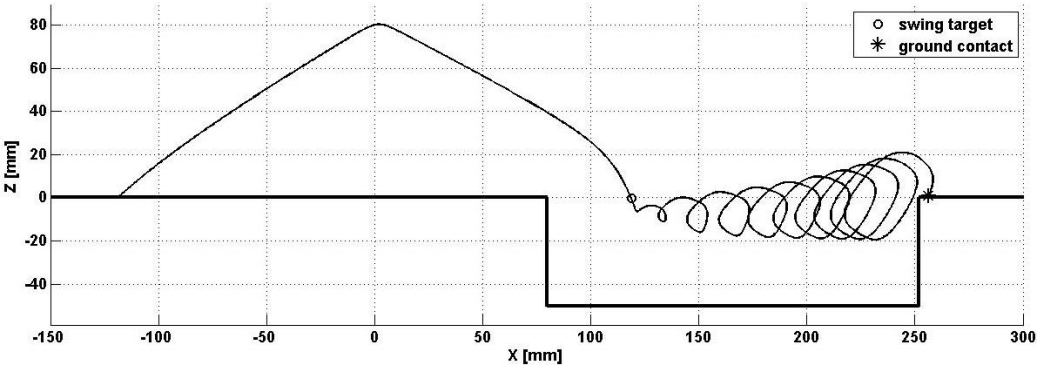


Figure 4.22: Foot trajectory in the x-z plane during a searching reflex.

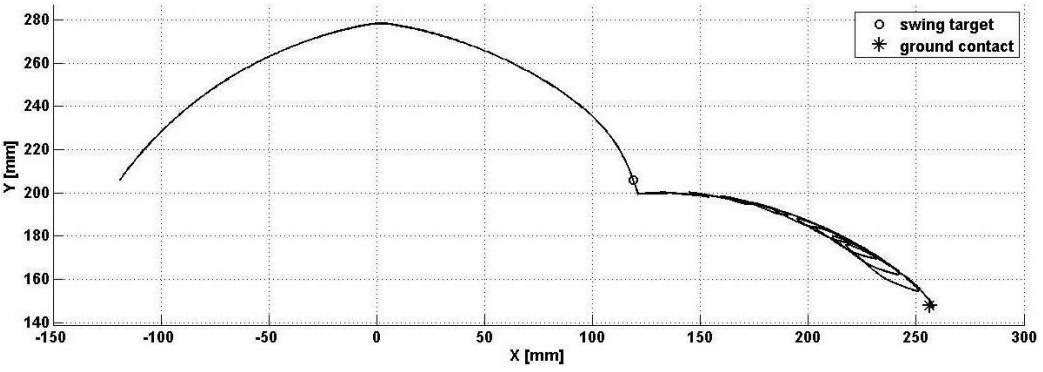


Figure 4.23: Foot trajectory in the x-y plane during a searching reflex.

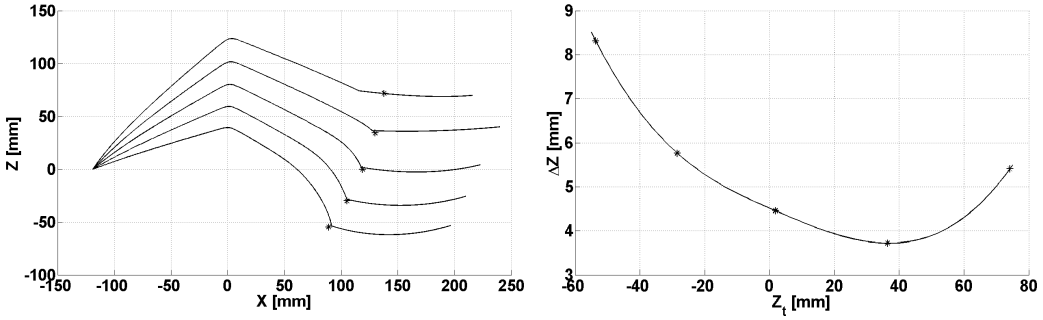


Figure 4.24: Error on advancing movement depending on starting height.

constant and the lateral distance from the body symmetry plane decreases in order to maximize the explored area in the direction of forward motion.

In figure 4.24 the error on advancing movement is illustrated. A change in the starting height produces a variation in the advancing behavior that loose regularity, especially when the required AEP is placed below its reference value (left side). In such a case, the maximum error on the trajectory, compared to the desired one, can reach 1 cm. The error show a minimum when th AEP is placed 4 cm above its reference value.

4.4.3 Stepping reflex

Locomotion on rough terrains requires a legged platform to adjust itself rapidly to mechanical perturbations of the single legs. Insects do this using two strategies: in response to small perturbations, they activate muscles to oppose the change in angle of a perturbed joint, producing the so-called swaying reflex in the robot as shown in section 4.2. To re-establish a stable posture in response to larger displacements, this approach isn't sufficient: the robot's leg has to lift and move towards the center of its range of movement, reducing the loads acting on its joints and improving the stability of the posture. This behavior is known as stepping reflex (see [38]).

The identification of the starting instant for the stepping reflex is done by space parameters: when a single joint angle goes beyond its upper or lower limit, the reflex module is activated targeting the movement of the whole leg to the reference values of the three joint angles (see table 4.7). This happens when the leg is in its stance phase so the reflex induces a phase transition that does not depend on the selector. This means that the implementation must by-pass the selector module to perform this behavior. The limit values represent the value beyond which the torques required by the actuators to maintain a given posture become greater than the maximum allowed values or the stability can be guarantee by the coordination system. In opposition, the reference values represent the sets of

Angle	Reference	Upper limit	Lower limit
γ	0/±30	35	–35
β	80/83.9	120	65
α	90/104.8	130	73

Table 4.7: Limit and reference values (in degrees) for the stepping reflex.

joint angles that allow to maintain static stability and minimize the torque level. Three different sets have been considered, corresponding to the reference PEP, AEP and resting position. The controller chooses, as target, the set closest to the reached limit position. The reflex ends when the ground contact occurs, exactly as for a standard aerial phase.

To avoid a simultaneous execution of too many stepping reflexes, that produces a statically unstable posture, the priority of each leg follows the same coordination criteria applied during the normal motion (see chapter 5). Another problem typical of the stepping reflex is the one that it's detected during the stance phase, while the other ones start during a swing phase, but its execution consist in an aerial phase. These two features require a more complex implementation that involves both the two states and the transition criterion too.

To reproduce the stepping behavior a static ANN has been used, instead of a CTRNN. This choice can be justified remembering that a reflex has a very little duration time during which disturbances exert a small influence on the motion and that the initial and target conditions vary much more than in the swing control case: in the CTRNN case, this drives to a training process too much complex when compared to the importance of this module.

In this case, as in the other reflexes, the stepping variable γ_s, β_s and α_s aren't directly the measured joint angles, but these ones decreased by the correspondent target values: in this way all the targets of the network become zero. This allows to reduce the total number of synaptic weights from nine to six, because it is useless to size a weight that has to be multiplied for zero. The network consists in three neural units with a functional dependency that can be summarized in the following equations:

$$\dot{\gamma}_s = w_{\gamma\gamma}\,\gamma_s \tag{4.18a}$$

$$\dot{\beta}_s = w_{\beta\gamma}\;|\gamma_s| + w_{\beta\beta}\,\beta_s + w_{\beta\alpha}\;|\alpha_s| \tag{4.18b}$$

$$\dot{\alpha}_s = \begin{cases} w_{\alpha\beta}\;|\beta_s| + w_{\alpha\alpha}\,\alpha_s & : \beta_s > 0 \\ w_{\alpha\alpha}\,\alpha_s & : \beta_s \leq 0 \end{cases} \tag{4.18c}$$

This structure shows the same auto feedback loops seen in the swing net

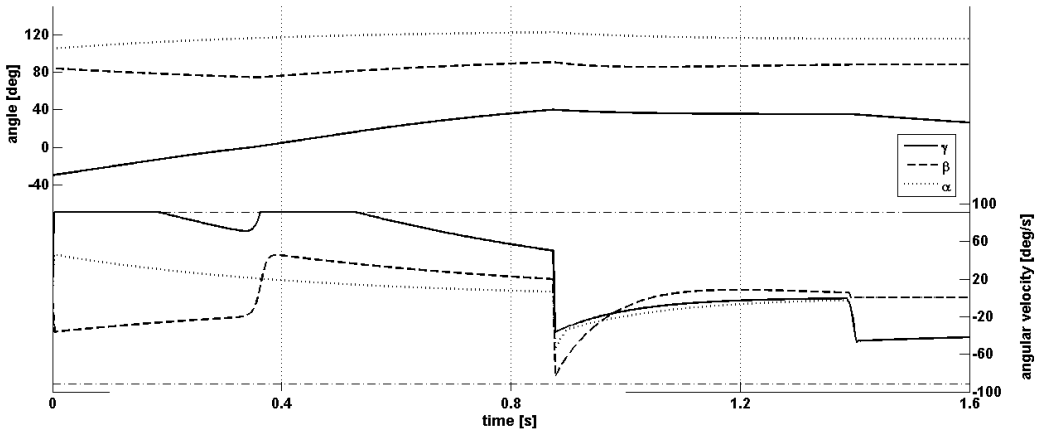


Figure 4.25: Angles and angular velocities during a stepping reflex.

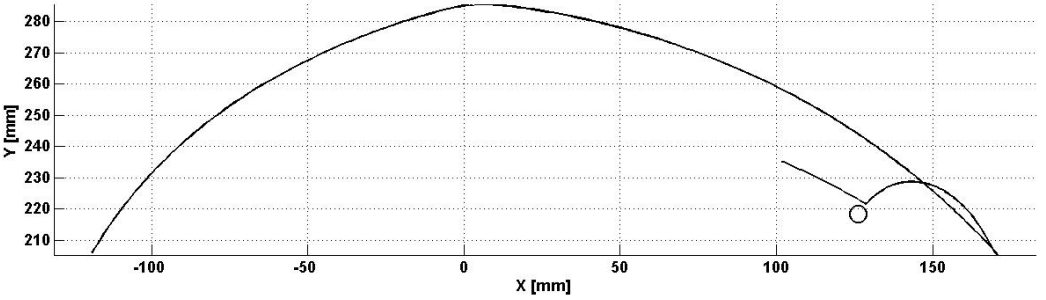


Figure 4.26: Trajectories in the x-y plane during a stepping reflex.

and a nonlinear coupling among different joints. In this case the coupling involves not γ and β only, but also α. These further links and the nonlinear dependencies ensure the lift-off behavior for every configuration. This complication of the original static swing net is necessary because the stepping reflex can occurs with all the possible combination of signs that the network variables can assume, a situation much more complex that a standard swing motion that involves only a limited range of angles.

By this simple set of rules it is possible to reproduce all the stepping reflexes that can occur during the motion. An example of these trajectories in the x-y and x-z plane is illustrated in figures 4.27 and 4.26 respectively. In figure 4.25 the correspondent angles and angular velocities are shown. It's possible to see how, when the stepping reflex occurs it produces a sudden variation in all the angular velocities. This is necessary to perform the motion as fast as possible as required to a reactive behavior. Although its rapidity, this reflex causes the velocity to saturate only for little configurations because it involves limited displacement, in opposition with the searching behavior that shows wide saturations since it produces

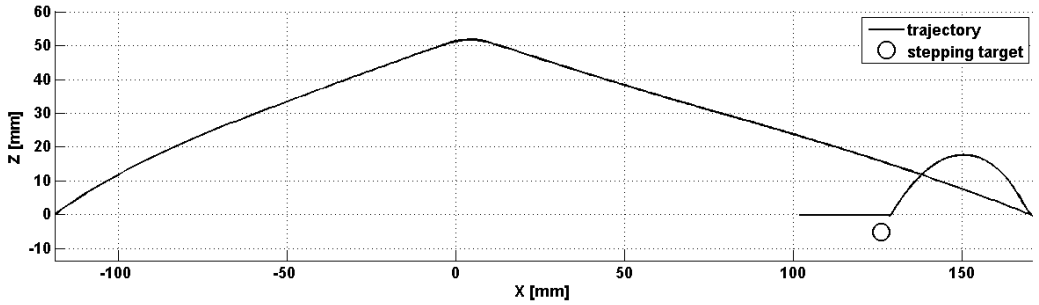


Figure 4.27: Trajectories in the x-z plane during a stepping reflex.

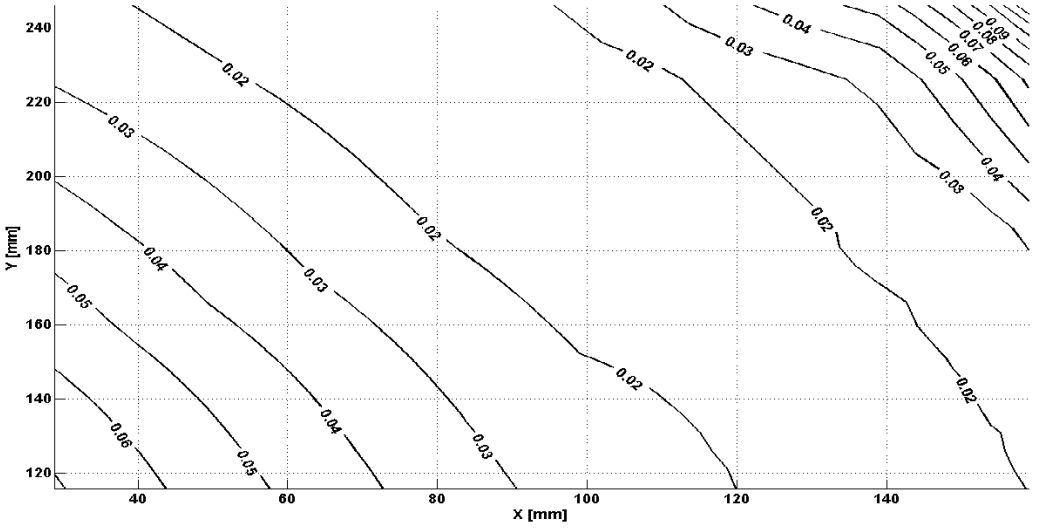


Figure 4.28: Error on advancing movement depending on starting height.

larger ones. In figure 4.28 the error on the target in function of the starting position has been plotted. The error level is almost ignorable for all the evaluated configurations, since it remains always below 0.1 mm

Chapter 5

Coordination

The strategy of the present work is to design a free-gait controller able to adapt the coordination among legs to produce the most efficient gait depending on required advancing velocity and external environment. This problem can be solved by the application of local influence that varies the motion of each limb in dependency of its neighbors. Advantages of local rules are their simplicity and the resulting simpler implementation. However, the global behavior of a system of local rules can only be predicted with difficulty. Therefore, it is also hard to set the design process to meet certain criteria, e.g. body stability. The best tested system of local coordination rules is the one based on stick insects behavior, elaborated by Cruse [42]. Another very interesting model of this type has been proposed by Porta and Celaya [43]. These two systems will be discussed in sections 5.1 and 5.2 respectively. The main idea of the this chapter is to merge the structures of the two controllers, taking features of both of them, in order to create a coordination model that had all the advantages of the two systems, but also reduced the problems of each one. This process will be described in section 5.3.

5.1 Cruse's coordination rules

The behaviors of the legs affected by coordinating influences have been already illustrated in subsection 3.6.3. In this section a quantitative formulation of those rules will be introduced. Cruse's model implemented in its simplest version introduces only three of the coordination rules (1-3) plus a fourth rule (4) that involves the targeting of the swing phase: it will be discussed apart (see section 5.4). The model for rules 1 to 3 uses default positions for the PEP of each leg. The local coordination rules shift these default PEPs along the x-axis. These relationships can be summarized qualitatively as follows:

1. the PEP of the affected leg is shifted backwards along the x-axis if the ipsilateral posterior leg is protracting. This prolongs retraction in the affected leg;

2. the PEP of the affected leg is temporarily shifted forward if the ipsilateral posterior leg or the contralateral leg has just changed from protraction to retraction. This shortens retraction in the affected leg;

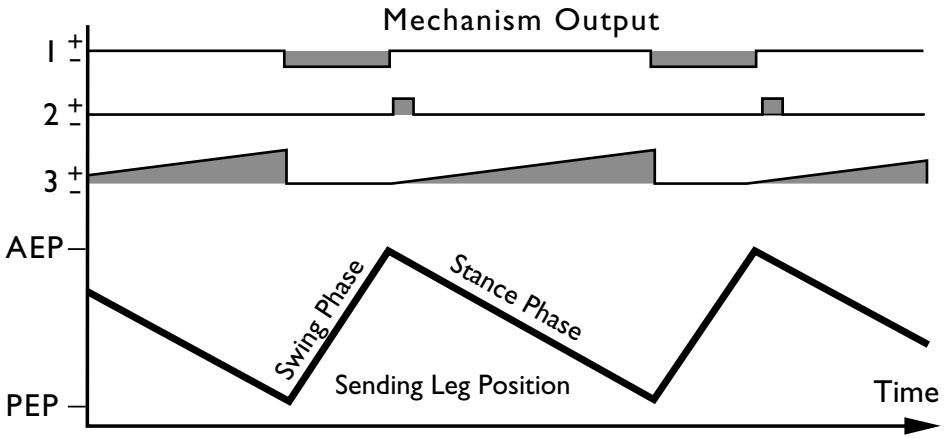


Figure 5.1: Coordinating influences.

3. while the ipsilateral anterior or the contralateral leg is retracting, the PEP of the affected leg is shifted forward proportionally. This shortens the retraction in the affected leg.

The rules listed above produce variations in the motion of the affected leg as it is illustrated in figure 5.1. Each rule can be mathematically expressed as a mechanism that can be summarized in the following form:

$$\Delta PEP_1 = -m_1 \tag{5.1a}$$

$$\Delta PEP_2 = m_2 \quad : \; t < \tau \tag{5.1b}$$

$$\Delta PEP_3 = m_3(t/T) \tag{5.1c}$$

where the time t is taken 0 at the start of the stance of the sending leg and T is the duration time of this phase. The interval τ is very little and often taken equal to 60 ms. This means that each rules is defined by only one numerical parameter. Reference values for PEP displacements compared to other geometrical parameters are illustrated in table 5.1. The values for each parameter are expressed in normalized units as given by each reference work. The values used in the NEMeSys case are the ones that give the more acceptable results.

The influences required in the simplified case (see figure 5.2) can be summarized in three groups: rule 1 and 2 acting forward and rule 3 acting backward on ipsilater legs, rule 2 and 3 acting between couple of correspondent controlateral legs. All the rules are necessary because each of them shows a specific effects on the receiving leg. Rule 1 avoid lift-off to avoid static instabilities. Rule 2 facilitates early protraction to favor temporal coherence. Rule 3 enforces late protraction to maintain temporal

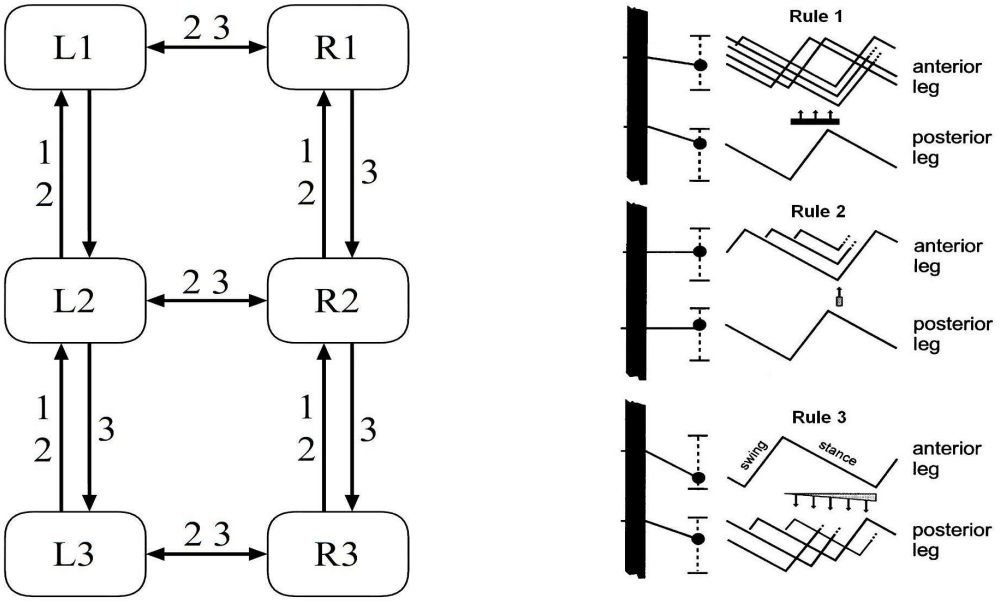


Figure 5.2: Simplified Cruse's coordination model.

coherence. Taking the start from these considerations, a total of 18 rules work in a six-legged walker. This means that to correctly size a controller for the coordination among leg based on this set of local influences, up to 18 parameters are needed. The usage of natural symmetries makes it possible to reduce this number, but, also in this case, the space of the solutions is quite wide. Another important problem of the training process involves the evaluation of the fitness function: to calculate a score that allows to obtain the correct weights, a very high number of simulations of the complete model is necessary, e.g. with all the six legs active and connected by the body, because either the variations in the advancing velocity or the variability of the external environment must be taken into account. These two features mean that the whole process is very cost expensive and gives results only in a long time.

The complexity of the weights calculation is not the only trouble related on Cruse's model. This coordination scheme maintain the stability delaying or anticipating the transition from stance to swing, avoiding to have too many legs in swing at the same time. The problem is how this timing is obtained: supposing that the protraction velocity is almost constant, as shown by experiments on stick insects (see Wendler [19]), Cruse establishes that the duration of the stance phase can be varied by changing the position of the PEP, the geometric parameter that manages the transition. This method has an important limitation: if the affected limb has to

Parameter	Espenschied	Roggendorf	NEMeSys
Step size	2	20	1
Mechanism 1	0.59	15÷20	0.7
Mechanism 2	1	3.5	0.5
Mechanism 3	0.58	6	0.2÷0.25

Table 5.1: Reference parameters for Cruse's coordination rules.

remain in contact with the ground for a long time, e.g. the middle leg, when the hind leg performs a searching reflex, the PEP should be shifted far behind. This motion is not possible because the maximum allowed posterior position is limited either by the geometry of the leg, e.g. segments length and limit angular displacements, or by the physical contact with the body. In such a situation becomes it is impossible to maintain the static stability of the robot by simply using the above said rules; other rules or global criteria have to be introduced, losing the advantages of a decentralized controller.

In its comparative analysis on some significant coordination mechanism for a free-gait controller, Roggendorf [44] found the same difficulties in the calculation of the parameters of the Cruse's model and identified other problems of this method: it performed reliably only when walking slowly on even ground while its performance worsened with higher walking speeds; during the simulation of an obstacle crossing, the model shows instabilities while ascending the obstacle and while descending it; during a curve walking it shows a high probability to become statically unstable.

5.2 Porta and Celaya model

In order to describe the model of the gait generator proposed by Porta and Celaya [43], the term of neighboring legs must be defined. According to the Porta and Celaya model, the legs connected by arrows in figure 5.2 are neighboring, with the exception of the middle legs, which are not considered neighbors. Thus each leg has exactly two neighbors.

To ensure stability, it must be granted that a sufficient set of legs stay on the ground supporting the body at any time. In the case of most six-legged robots, this requirement can be satisfied by observing the following rule:

Rule 1. *Never have neighboring legs raised from the ground at the same time.*

In their work Porta and Celaya assume that Rule 1 is a sufficient condition to guarantee statically stable gaits. The fulfillment of this rule as-

sures that, at any time, the robot will be supported by at least three non-neighboring legs, forming a triangular support polygon (see subsection 3.6.1) for which it is possible to assume it will always contain the vertical projection of the center of gravity (PCG). This assumption might not hold in robots with very particular leg configurations or when the robot is too much tilt. In any case, the violation of Rule 1, however, will result in a situation in which two neighboring legs are out off the ground at the same time, most probably leaving the robot in an unstable situation.

According to Rule 1, a leg can be raised to make a step only while its two neighboring legs are in contact with the ground. However, Rule 1 by itself is not enough to determine a gait. It leaves undetermined which one of a pair of neighboring legs should actually perform a step when Rule 1 would allow both of them to be raised. To deal with these situations, it's possible to assign priorities to the legs introducing another rule:

Rule 2. *A leg should perform a step when this is allowed by* Rule 1 *and its two neighboring legs have stepped more recently than it has.*

According to the Porta and Celaya model, a leg is lifted if it has a higher lifting priority than both the neighboring legs. Lifting priority is defined, taking the start from Rule 2, as follows:

- protracting legs have the highest lifting priority. This choice implicitly means that neighboring legs cannot protract simultaneously as required by the Rule 1;
- retracting legs have a lifting priority negatively proportional to the legs' distance from their physical PEP: the closer a leg is to its physical PEP, the higher its lifting priority is. The physical PEP is determined by leg geometry mainly on the basis of leg segments' length. It can be defined as the hindmost point a leg can reach in a normal walking position.

Porta and Celaya [43] proposed to determine the lifting priority by temporal parameters. In that paper they also describe how they determine the reference velocity v_{ref}. Their method is, however, closely linked to their method for the retraction trajectory control, that can be explained by a physical analogy as illustrated in figure 5.3 (right side). An optimal posture of the legs is defined (dotted lines) as the posture that minimize its distance from the reference one without changing the stability characteristics. Each retracting leg tries to minimize the distance from its actual position (solid lines) to its position in the optimal posture. The minimization follows a gradient (springs). The resulting translational and rotational vectors are averaged to determine the global movement vector for

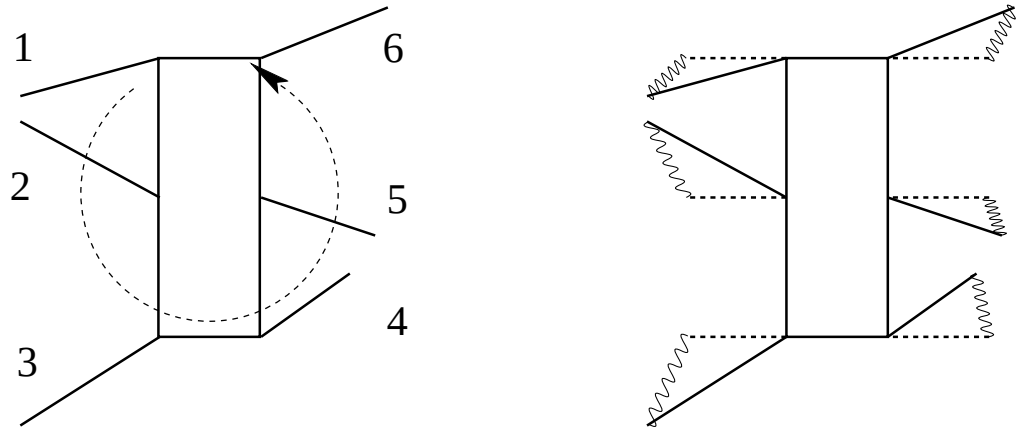


Figure 5.3: Legs configuration in the Porta and Celaya model.

all legs. Retraction in a given leg is achieved by protracting other legs to new positions: legs that step (protract) will shift their positions forward (in front of their optimum), dragging the body with them by minimizing their distances to their corresponding optimal positions once they reach the ground. Thus, retraction velocity is determined by the protraction trajectory length. Curved walking can be achieved by making protraction trajectories that are longer on one side of the body than on the other. This method hasn't been reproduced in the present work, but is useful to understand how it's possible to manage the retracting velocities in the correct way to enhance the stability of the whole system.

To reach a better comprehension of how effectively this gait generator works, another concept has to be introduced: the *gait state*. For two adjacent legs a and b, we will denote as:

$$a < b$$

when leg b has more priority to step than leg a. Taking the start from this notation, the gait state of a six-legged robot at a given time is defined as the list of the six relationships as the ones above that can be established between neighboring legs. The gait state can be represented as a row of six symbols < or >, corresponding to the relationships between priorities of neighbors. The legs are numbered counterclockwise and number 1 is the front leg of the left side as shown in figure 5.3. The resulting order is: 1, 2, 3, 4, 5, 6. Thus, for example, the state:

<><><>

represents the following relationships between leg priorities:

$$1 < 2 > 3 < 4 > 5 < 6 > 1$$

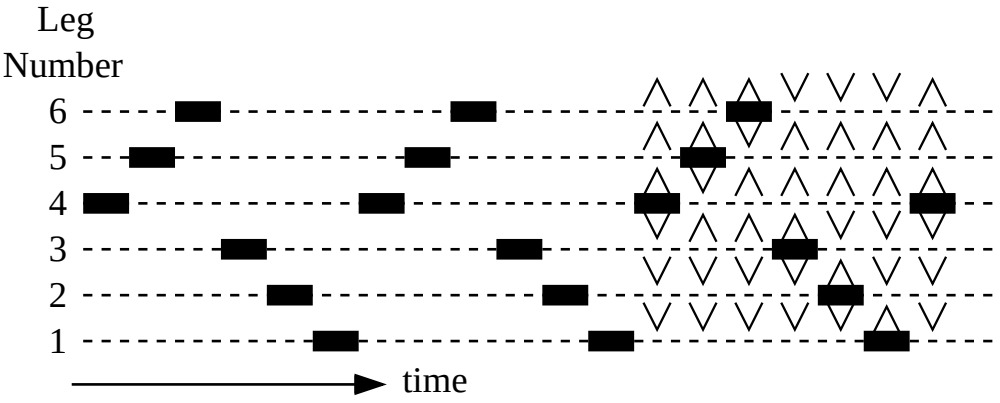


Figure 5.4: Wave gait emerging from the Porta and Celaya model.

The gait state is important because it determines in a very immediate way the number of legs that can start a protraction at a given time: if a leg appears between a <> pair is allowed to protract, otherwise it must remain in support phase. The description of a gait pattern by the gait state notation, is a powerful instrument, because it's able to characterize all the hexapod gait in a very useful way. An example is the description of a generic wave gait, the most stable and efficient kind of walk on a flat surface, is shown in figure 5.4.

Porta and Celaya [45] propose an optimized version of that model: if a>b>c>d change priorities to a>b<c>d, thus leg c can be lifted despite the original rule. The converse also holds: if a<b<c<d change priorities to a<b>c<d, thus leg b can be lifted. Since protracting legs always have higher lifting priorities than both neighbors, the rule not to lift neighboring legs simultaneously is not violated. However, since more distant neighborhood relations have to be taken into account, the model is less local when using this optimization procedure. Without the optimization only immediate neighborhood relations are required for coordination. This optimization was used in the controller presented in the section below.

The main problem of the Porta and Celaya's model is its incapability to produce autonomously the most efficient wave gait when the reference advancing velocity becomes slower. At the maximum speed permitted by the static stability requirement, the gait converge to the tripod one, but with smaller velocities the coordination criterion above, starting from a general position, aren't able to drive to the gait more appropriate to its range of speed, as done by the Cruse model (see [46]). This behavior can be explained with the motivation supported by Roggendorf in his work [44]. He considers that this model tends to lift legs as soon as possible with

the purpose of maximize the advancing velocity. This requirement drives the walking pattern to the tripod gait that is the fastest possible, but, at the same time produce a very poor performance with low retraction velocities. The generation of gaits at the lower velocities becomes a problem because this controller tends to concentrate the swing phases in almost the same time slice, producing a situation in which the legs are all close to their PEP and the robot is too inclined. This is the kind of posture in which Rule 1 doesn't ensure static stability and an unstable gait can occurs.

5.3 Coordination model

In the present section the coordination model adopted for the gait generation of the NEMeSys robot will be discussed. The system consists in a hybrid version of the two models summarized above that tries to take into account advantages and disadvantages of both of them. The purpose is the design of the simplest controller possible able to fulfill, at the same time, the two following requirements:

1. ensure static stability in all the situations. The criteria on which is based the generation of the gait pattern, must ensure a statically stable motion not only during a typical walk, but also on a wider range of possible movement, i.e. during a searching reflex. Thus the controller must be able to decrease the advancing velocity of the whole robot or to stop it if necessary;

2. produce an efficient gait. The fulfillment of the stability requirement shall not reduce the efficiency of the gait. Thus, during the walking on flat surface the free-gait has to naturally evolve through a stereotypical wave-gait. This behavior should ensure to obtain the most efficient gait at every speed, in the entire range of velocities.

The first requirement implies that another criterion, based on the retraction velocity, has to be considered. This criterion must be global, to manage the advancing velocity and has to be as simple as possible to avoid any decrease in the computational efficiency of the whole system. The second requirement implies that the rules that produce a certain wave-gait must be found. These rules must be obviously local and have to be implemented in their simplest form. Moreover they should be the same used for the stability control, or at least don't have to be in conflict with them.

The comparison done by Roggendorf shows how the most effective system for the generator of gaits, among the tested ones, is represented

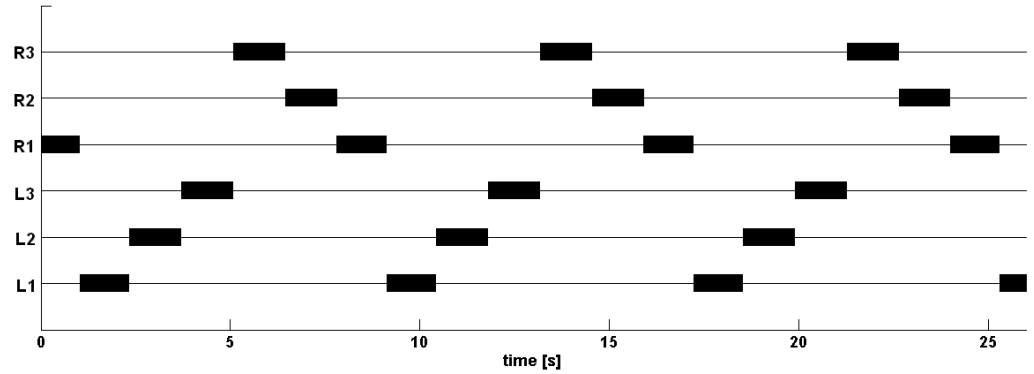


Figure 5.5: Slow gait with an unmodified Porta and Celaya model.

by the model developed by Porta and Celaya. It fulfils per se the first requirement of the problem: the stability margin is always maintained above zero during the entire walks. This results descends automatically from the structure of the controller, because Rule 1 doesn't allow to reach an unstable posture. Moreover, this model outperforms not only the simple Cruse's method, but also more complex, even globally controlled ones. The better results are obtained in terms of higher stability margin, especially during complex behavior such as crossing an obstacle or curve walking. Another important advantage of this model is that it reached the lowest computational cost among all the principal tested gait generators. These considerations show how a good choice to design an efficient gait controller is to start from the Porta and Celaya model and to introduce some modification to it in order to improve its capabilities. To solve some of the troubles that afflict this system, Roggendorf was the first to introduce some modification to the model.

The first important updating was a different method to calculate the reference velocity: as it was shown in the previous section the original version uses a quite complex method based on minimization criteria that involves the calculation of lots of global parameters. Instead, an even simpler approach can be adopted: as legs get close to their physical PEPs, the retraction velocity v_r for all legs is lowered. In this way the retraction velocity can reach zero as needs by the first control requirement. This approach is somewhat similar to the one discussed by Porta and Celaya [43]. In the Porta model, one leg that approaches its physical extreme will 'stretch its spring', exerting a backward force on the body, eventually halting retraction altogether; in this one the same behavior is obtained with a much simpler kinematical criterion.

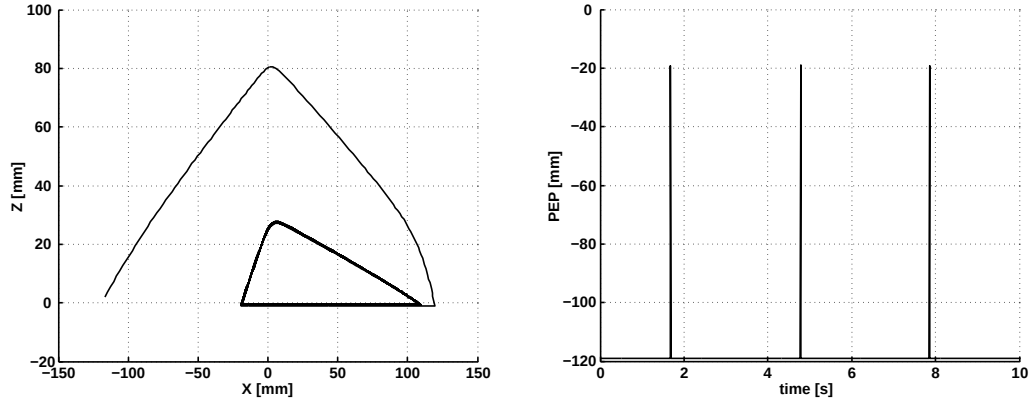
Another problem of the Porta and Celaya model is its tendency to lift off the leg as soon as possible. This allows to reach the most effective tripod gait earlier than in the other models, but with low retraction velocity, the PEPs are shifted very far forward, eventually to positions shortly behind the AEPs, to maintain the correct timing. This can lead to dangerous instabilities because the support polygon assume a very stretched configuration. An example of this behavior is illustrated in figure 5.5. To avoid it a second modification of the Porta and Celaya model has been proposed. In this version, the lifting rules were applied only if the distance D of a given leg from its physical PEP was smaller than a preset threshold T. In his work the threshold is set to 5.0 length units in the simulation, compared to mean step size of 20 units. These parameters are used to calculate the actual retraction velocity v_r. Each retracting leg proposes a desired global retraction velocity accorded by equations 5.2.

$$\begin{cases} v_r = \frac{D}{T} v_{rref} & : \text{D<T} \\ v_r = v_{rref} & : \text{D}\geq \text{T} \end{cases} \tag{5.2}$$

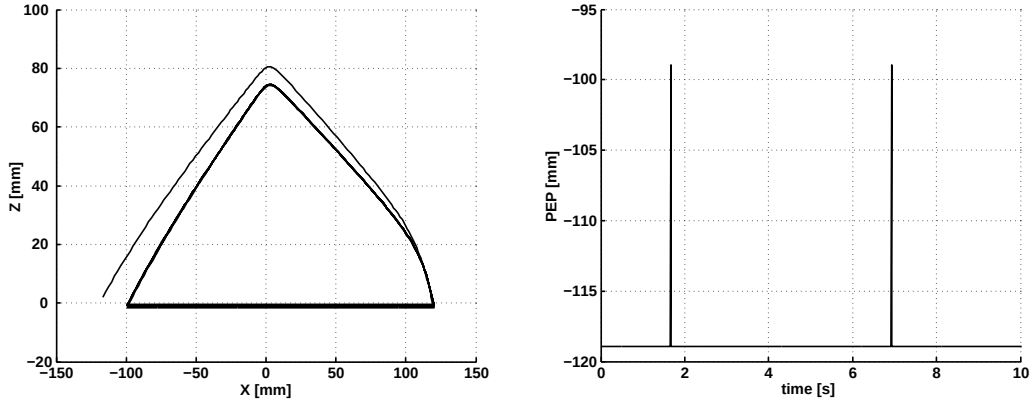
This method to calculate the global reference velocity implies that the leg with the smallest proposal determines the actual v_{ref}. The calculated reference velocity will obviously be a fraction of the required one (v_{rref}) and in the ideal case of perfectly coordinated legs movement, it will assume exactly the required value. This coordination rule requires a global planning to be performed: all the reference velocities proposed by the six legs have to be taken into account to calculate the actual one. The introduction of a global rule seems to make the method lose the properties depending to its decentralization. This situation, however, doesn't produce any decrease in the performances, because the formulation of the rule is very simple and involve only parameters that have been already calculated.

Roggendorf proposed another modification to the Porta and Celaya model. This control system made use of the threshold T for calculation of v_r only, but it calculated the lifting priorities with a different approach. It was checked whether the walker would become unstable if that leg were lifted: a leg was lifted only if the walker retained static stability. The problem is evaluating the stability a global parameter as the stability margin must be calculated. This means that this extension implies a much more advanced planning than the previous ones and drives to a greater increase in the computational cost. This is the reason why it will not be employed in this work.

The only parameter involved in the sizing process of this modified Porta and Celaya model is the value of the threshold T. The reference value



5.6.1: High threshold.



5.6.2: Reference threshold

Figure 5.6: Effects of a too much high threshold T on the coordination.

found by Roggendorf is exactly 1/4 of the mean step size. To evaluate the possibility to change it, a sensitivity analysis about its effects on the quality of the gait pattern has been performed showing the behaviors as follows:

- the increase of the threshold value causes the PEP to be placed too near to the AEP, especially at low velocities. This is the same problem shown by the unmodified model that has a threshold virtually equal to the step size. An advantage of an high threshold is to have a quick transition from a free-gait to the more effective tripod one. The other important advantage in increasing the threshold is to improve the regularity of the gait: with a sufficiently high T, the leg doesn't have to slow down to enhance the static stability of the gait, maintaining a proposed v_r always equal to v_{rref};

- the decrease of the threshold value drives to an irregular motion.

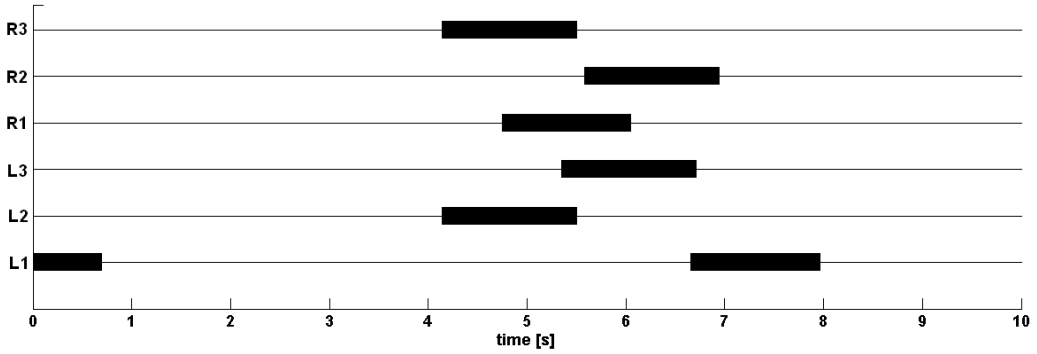


Figure 5.7: Effects of a too much small threshold T on the coordination.

> With a too much little T, the leg hasn't enough time to correctly decrease its velocity, and this causes to a loss of coordination. During a poorly coordinated walk, the controller tries to keep a leg on the ground as longer as possible stopping its motion; this is the reason why the advancing velocity becomes irregular. The main advantage of this behavior is the possibility of having the PEP closer to its reference value, producing a legs' motion that allows to reach a globally optimal posture.

The irregularity effects occurs only for threshold values very close to zero, so the sizing criterion is to minimize the threshold, but, at same time, to maintain a regular motion. The limit value has been found equal to 1/12 of the mean step size. Smaller values produce sudden decrease in the advancing velocities followed by returns to the reference ones, when the global configuration allows it. This behavior can be identified especially at the fastest gaits.

The version of the Porta and Celaya model modified by Roggendorf and adopted in this work fulfils the stability requirement completely, but doesn't allows to solve the incapability of this model to reproduce a wave-gait for low velocities. On the other hand, as it has already been said, the same behavior autonomously emerges from the Cruse's set of rules. The solution might be to understand which of the rules are responsible of this particular behavior and to insert them into the modified version of the Porta and Celaya model, paying attention not to lose the useful capabilities that it shows per se.

The evaluation of a decentralized control system, as the one adopted to manage the coordination, is very difficult to obtain with classic techniques. The way used in related works, as in the present one, is the Kindermann's ethological approach to systems analysis that consists of an evaluation and

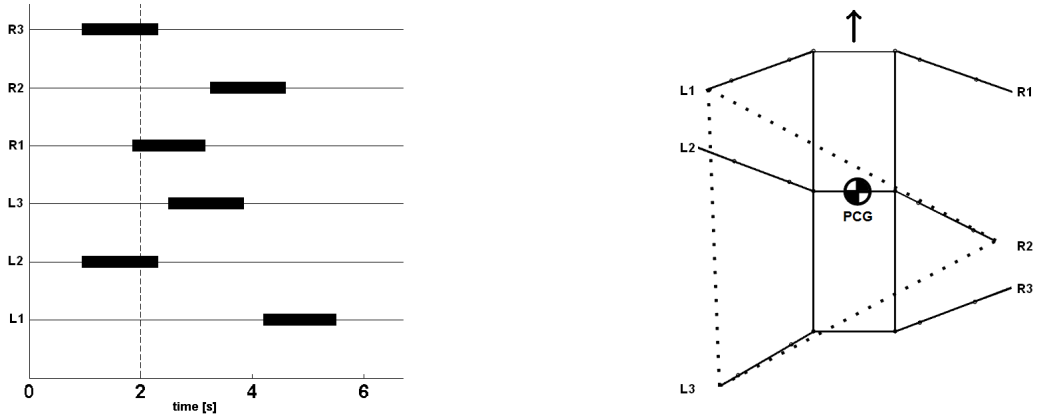


Figure 5.8: Effects due to the absence of coordinating mechanism 2.

a quantification of the behavior of the complete, rather complex models. The systems are run in a global simulation, and the general behaviors are observed building a statistical description of their main aspects rather than a complete formal analysis. This means that an evaluation of the actual influence of each single rule is not available. To understand reasonably how a rule works, the most interesting results are the ones obtained in lesion studies (e.g. Espenschied et al. [46]). In these works that aimed to evaluate the robustness of the Cruse's coordination model, entire mechanisms have been removed. Without a given mechanism, the coordination loses one or more of the behaviors identified by Wilson (see subsection 3.6.2) making it possible to establish some relations between mechanisms and behaviors as follows:

- mechanism 2 promotes normal back-to-front metachronal wave on the ipsilateral legs. This mechanism exerts an excitatory influence on anterior legs. Upon touchdown of a leg, it facilitates the lift-off of the next anterior leg. Thus starting from a random legs configuration it tends to put the swing phase of each leg juts after the protraction of the posterior one producing a metachronal wave on each side. In figure 5.8 the effects due to the absence of this rule are illustrated. Without it the legs, instead of reproduce a back-to-front wave, try to reach in every case a tripod gait. At low velocities, this behavior concentrate the start of swing phases all at the same time (left side). In this configuration more than one leg is in transfer phase and legs on the ground are close to their PEP. This produces a very tilt posture, reducing the stability margin (right side);

- mechanism 3 promotes 180-degrees phasing between cross-body leg

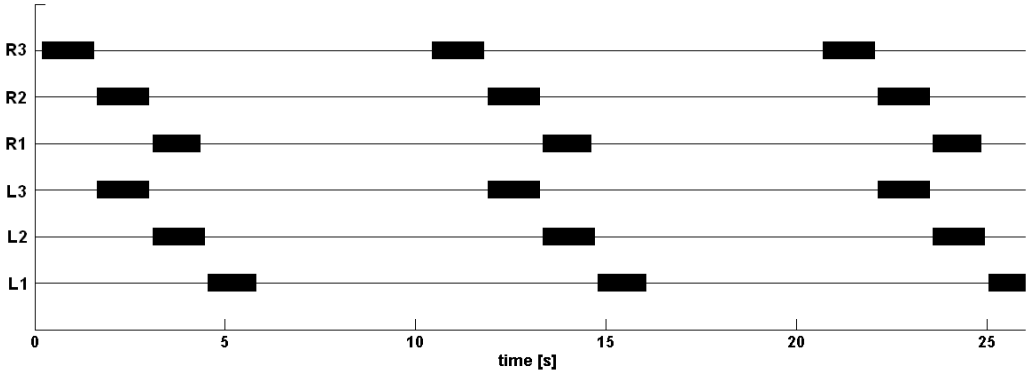


Figure 5.9: Effects due to the absence of coordinating mechanism 3.

pairs. This mechanism exerts an excitatory influence on the neighboring limbs during late stance. The closer a leg gets to its normal lift-off position, the stronger the facilitatory influence is on the other legs to undergo a stance-swing transition. This mutual influence among two contralateral legs causes the identified phasing. In figure 5.9 the effects due to the absence of this rule are illustrated. Without it the robot tends to concentrate all the swing at the same time. This produces an inefficient gait at low velocities, with a peak in the power consumption too much beyond the requirements for a slow motion.

It's clear how this two mechanisms are sufficient to solve the main problem of the modified Porta and Celaya model: mechanism 2 produces the backward sequences of swing phases on each side and mechanism 3 phase them to produce the correct wave-gait.

The first step is the introduction of mechanism 2 alone. The action required for this mechanism is the generation of a back-to-front metachronal wave on each of the two sides of the robot. This means that the rules need to be implemented only between two ipsilateral legs. Since the rule moves toward the front of the robot the couple of hind legs isn't affected, so, two rules only for each side have to be taken into account, for a total of four local influences. To size the parameters correctly for this mechanism a consideration has to be done: in the full Cruse model, rule 2 acts synergically with rule 1 to couple a back-to-front sequence of swing movements, first by suppressing, then by facilitating a stance-swing transition in the anterior leg. In the present model the situation is quite different: instead of the rule 1, the Porta and Celaya criterion based on the lifting priority has been used. It exerts an inhibitory influence on anterior legs as rule 1, but also on the other neighboring leg, thus the influence required by

rule 2 is different from the one needed in the reference case. The actual values for the parameters could be far different from the reference ones as shown in table 5.1, especially for one reason: the Porta and Celaya model tends to put the swing phase of each strictly before or after the protraction of its posterior neighbor. Since the wave-gait can be obtained in the first case only, in the second one the influences have to be very high to counterbalance. To calculate them an hand-tuning process has been performed starting from the standard values and evaluating the sensitivity of the gait properties depending on each parameters. This process allows to classify three important behaviors:

- the duration of the mechanism has to be extended up to the whole first half of the power stroke of the sending leg. This is an important change when compared to the classic formulation in which this duration was very small: about 60 ms by temporal criteria, less than the 5% of the power stroke duration for a tripod walk. The effect of this behavior is to accelerate the transition from a free-gait to a wave-gait, by locking the swing of a leg as soon as possible just after the protraction of its posterior neighbor. With a smaller duration time, the wave-gait emerges only after a high number of steps, especially at lower velocities, while with greater ones, the motion becomes unstable during the faster gaits;

- the influence of the mechanism depends on advancing velocity. At lower velocities the duration of the stance phase is much higher than with the swing one. This means that a value of influence comparable with the step length has to be applied to 'unlock' a swing phase placed just before the protraction of its posterior neighbor, instead of after. This happens because, in this case, the swing is performed while the sending leg is at the end of its power stroke, so the difference between the current location of the return stroke of the target leg and the required one is comparable with the length of the power stroke itself. At higher velocities the problem is the same, but the duration of the stance phase is very close to the swing one and the required effect can be obtained also with a smaller displacement of the PEP, compared with an half of the step. At these speeds another problem occurs: a much higher value of displacements produce an unstable gait, so its value must be reduced;

- the regularity of the gait depends on the strength of the influence. A strong influence, like a high duration time, is able to produce a fast transition from a general free-gait, to a wave one. With a lower influence some legs have to wait the other ones, that are still in stance,

decreasing their protracting velocities. Since the global advancing velocity is the minimum of these ones, the whole robot has to walk with a slower speed, until the configuration allows to start with a new step sequence. This behavior generates an irregular advancing velocity that causes an inefficient motion. The problem is that it isn't possible to increase the strength of the influence arbitrarily, because this could produce an instable gait, as discussed in the previous point;

In this work the influences of both the mechanisms have been calculated by spatial, instead of temporal, parameter. This choice has been preferred because the controller structure allows to have almost exact values of all the geometric parameters starting from the measured joint angles, while the characteristic times of the motion could be only roughly estimated. The implementation of mechanism 2 is the same as show in section 5.1, but the threshold T_2 is a displacement, instead of a time:

$$PEP_1 = PEP_{1-ref} + m_2 \qquad : (AEP_2 - x_2) < T_2 \tag{5.3a}$$
$$PEP_2 = PEP_{2-ref} + m_2 \qquad : (AEP_3 - x_3) < T_2 \tag{5.3b}$$
$$PEP_5 = PEP_{5-ref} + m_2 \qquad : (AEP_4 - x_4) < T_2 \tag{5.3c}$$
$$PEP_6 = PEP_{6-ref} + m_2 \qquad : (AEP_5 - x_5) < T_2 \tag{5.3d}$$

The mechanism 2 is only able to produce statically stable gaits that show a metachronal wave propagating among the legs of each side, but to introduce correct relationship from a side to the other, is necessary to introduce another rule. The second step consists of the usage of the mechanism 3 to obtain the correct phasing between the two sides. Before starting with the sizing of the parameters of this mechanism, an important consideration must be done. The back-to-front wave of protracting has been obtained with four rules acting from a leg toward its anterior neighbor: this means that on the couple of hind legs no more influences have been added to the Porta and Celaya model. But, more important, it also implies that the coordination of the three legs attached on each side descends from the motion of the correspondent hind leg. Taking the start from this consideration it is possible to argue that the complete coordination can be achieved by applying the mechanism 3 only between the two contralateral hind-legs. This simplification could seems quite brutal, but the results obtained by the simulation show all its effectiveness. Moreover it allows to reduce the number of influences from six to only two. The only parameter that manages this mechanism is the maximum displacement reached at the end of the sending leg's power stroke. The formulation for

the mechanism 3 can be summarized as follows:

$$PEP_3 = PEP_{3-ref} + m_3 \left(1 - \frac{x_4 - PEP_4}{AEP_4 - PEP_{4-ref}}\right) \tag{5.4a}$$

$$PEP_4 = PEP_{4-ref} + m_3 \left(1 - \frac{x_3 - PEP_3}{AEP_3 - PEP_{3-ref}}\right) \tag{5.4b}$$

where m_3 is the weight of the influence, PEP_{ref} the reference posterior extreme position. Obviously, as widely shown before, this coordinating influences acts only if the sending leg is performing a power stroke, otherwise is equal to zero. The calculation of the strength of this coordinating mechanism has been performed the same way as the parameters of mechanism 2: by an hand-tuning process that evaluates the sensitivity of the gait properties depending on its only parameter. The reference value of m_3 taken from table 5.1, is of about 25% of the normal step length. Variations from this value show the behaviors as follows:

- at low velocities, an increase in the strength of the mechanism accelerate the phasing. In the same way of mechanism 2 an high influence reduces the time to pass from a free-gait to a wave one. The problem is that higher strength corresponds to a higher phasing and this changes the gait starting a side too much late after the other one. This behavior can be identified only when a sufficient number of stride have been performed. This is the reason why the simulation time depends on the advancing velocity;

- at high velocities the required influence seems to vanish. This happens because the Porta and Celaya model converges to a tripod gait at the highest velocities without any add-on. With just a little lower advancing speeds the obtained phasing is far from being correct. This means that the mechanism must be maintained active for all the range of velocities.

When the value is altered with respect to the reference one, all the shown behaviors cause decreases in the performances, without introducing any advantage. This means that maintaining the reference value appears to be the best choice in order to size the weight of this mechanism in the correct way.

5.4 AEP targeting

In this investigation upon the mechanisms that regulate the targeting behavior shown by the legs during the swing phase, Cruse [47] obtained an

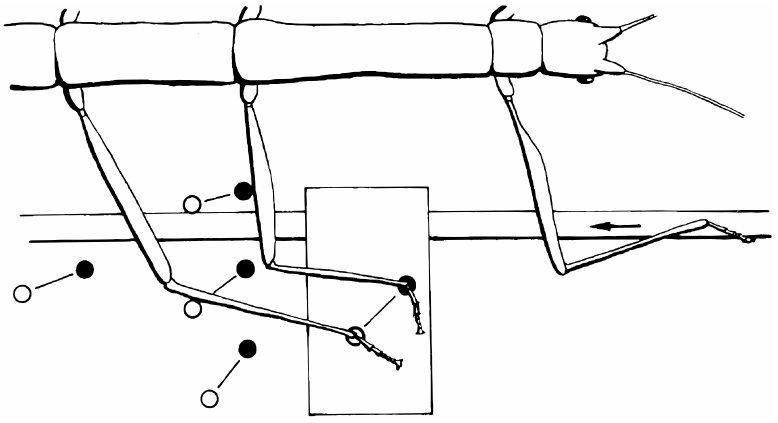


Figure 5.10: Targeting on AEP in a stick insect.

unexpected result. He found that the anterior extreme position of a leg is not fixed within a coordinate system relative to the body, but it's controlled by the position of its ipsilateral anterior leg. This means that the behavior can be considered as a form of coordination among legs and, for this reason, he treated it in this way, inserting this mechanism in his biological-based coordination model as rule number four (see subsection 3.6.3). This rule is also called the follow-the-leader placement strategy.

The calculation of the leg's target position during the swing phase can be treated as a problem apart from the rest of the coordination, because it involves the module that controls the return stroke, instead of the selector. The idea is to create a separate module able to give, as outputs, the three joint angles used by the swing net as targeting inputs. This additional part of the controller has been introduced for the first time in Walknet and called target net (see Dean [48]). In the present section a simplified version of this module will be discussed.

The first consideration is that for the two front legs, the AEP doesn't depend on any parameter of other legs. In this case the value of the AEP is always the same and it's taken equal to the reference anterior extreme position. The choice to maintain a constant AEP for the front legs could seems a problem especially when they face an obstacle or a gap, but it isn't so. When a leg finds a gap, a searching reflex is activated to cross it, as explained in subsection 4.4.2. A general obstacle can be climbed if the leg is sufficiently elevated during the swing phase or can be completely avoided when its trajectory passes above it. Should the gap be larger or the obstacle higher than the capabilities of the control system, a correction of the pattern at an higher level would be necessary (see section 1.1). For the other legs, the module basically consists in a function able to take, the cur-

rent joint angles of the correspondent anterior ipsilateral leg as inputs and gives the set of joint angles that represents the required AEP as outputs . This function can be expressed in its simplest form as follows:

$$x_{AEP} = x_{ant} - \Delta_{AEP} \tag{5.5a}$$

$$y_{AEP} = y_{ant} \tag{5.5b}$$

$$z_{AEP} = z_{ant} \tag{5.5c}$$

where the Cartesian coordinates are expressed in the absolute reference frame and the value for Δ_{AEP} has been taken equal to 60 mm. The reason for this choice can be justified remembering that the model of the robot uses monodimensional beams to simulate the limb's segments, but the real legs has a square section with a side of 25 mm (see chapter 7). In the worst case the contact between two feet may occur when the distance from a footprint to the other becomes lower than 35 mm about. Taking a margin equal to the side of the section the value of 60 mm is obtained. The required behavior is shown in figure 5.11 (left side). The requirement is the same, independently from the current position of the target footholds.

In his work on leg targeting in stick insects, Cruse found that the actual behavior is quite more complex, because the AEP shows a displacement not only in the x direction, but also in the y one. Moreover, the value of these displacements changes as a function of the position of the sending leg and it assumes different values when the target leg is a middle or a hind one. This complexity seems to be useless in the control of NEMeSys because it has a geometry far different from the one on which Cruse performed his test: the stick insect shows a differentiation among front, middle and hind legs, while the robot have six legs with the same geometric characteristics. Other differences can be found in the position of the legs on the body and in the ratios among characteristic lengths, such as the distance among legs or the mean step length. The superior symmetry of the robot allows to use only the x-component of the required displacement obtaining the correct results.

The first idea could be to use the Cartesian coordinate directly to solve the problem, but, since the model is completely controlled in terms of angular variables, this isn't the best choice. Obviously the calculation of the AEP can be expressed in a very simple form, but this procedure requires a double conversion, first from angles to displacements and then from displacements back to angle. The process involves nonlinear trigonometric functions that increase the computational cost, so, a better solution is to formulate the problem directly in the joint angles space.

This means that the aim is to fulfill a task in the Cartesian space only by using calculation in the joint space with angular variables. As shown

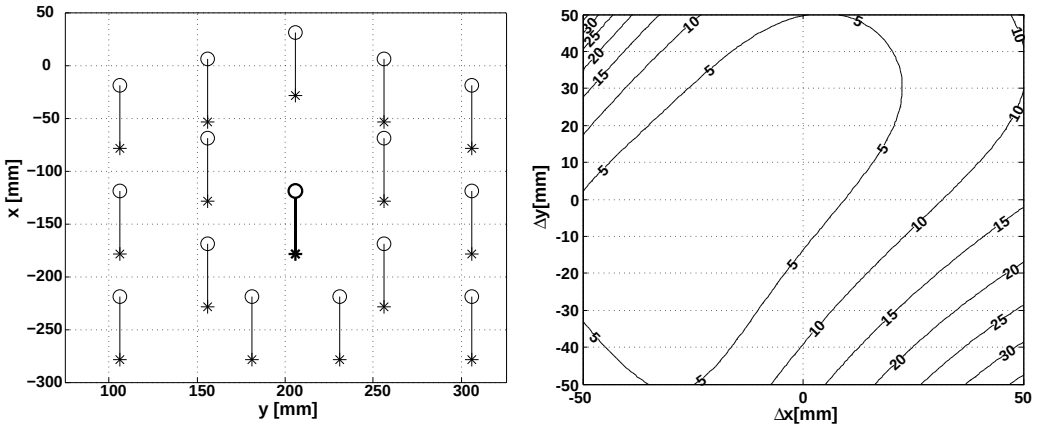


Figure 5.11: Targeting on AEP with the simple form of the target net.

in section 2.4, a simple method to solve such a problem might be to approximate the function with a properly trained static ANN. To train the network it's necessary to start from a standard set of inputs and outputs to fit. The method to obtain these values is quite simple: a great amount of sets of joint angles has been selected inside the limit working space of each joint; for each of these sets the analytical, cost expansive calculation described above has been performed obtaining the correspondent sets of target angles; taking the former values as inputs and the latter ones as outputs, it's possible to start the training process. To obtain sufficiently accurate results in all the range of joint angles is necessary to use at least three neurons on the hidden layer of the network. The results obtained with this configuration are shown in figure 5.11 (right side). The error between the desired position and the one actually given by the ANN has been mapped as a function of the displacements from the reference PEP in the x-y plain. The values are sufficiently low in a wide range of positions, but increases suddenly in certain areas. This can be justified because this region are close to the limit values of the position used during the training process.

The main disadvantage of this solution is its poor flexibility. If the position of the sending leg is far from its reference value, the target leg will receive the input to assume a posture that can reduce the stability considerably or increase the stress levels too much. To solve this problem it's possible to start from Cruse's findings, by using a 2-dimensional displacement that can be expressed as a function of the position of the sending leg. Instead of using the values found for the stick insects, an optimization criterion has been adopted: the required target position will be the one that finds a good compromise between the follow-the-leader requirement and

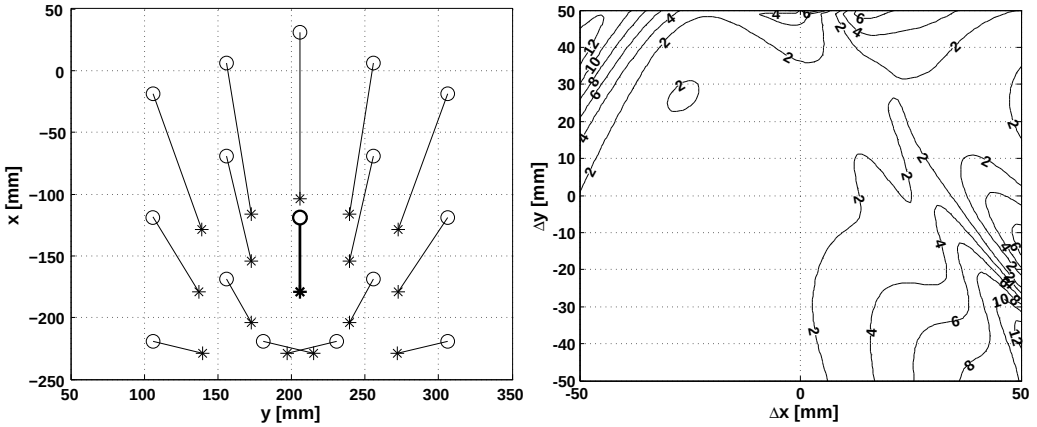


Figure 5.12: Targeting on AEP with the complete form of the target net.

the displacement from the reference AEP value. Moreover the distance between the two footprints will always be sufficiently high to avoid every contact between the sending and the receiving leg. The simplest method to obtain these results is to start from a standard displacement and to correct it with a function of the distance of the sending leg from its reference position, with the constraint that the distance between the two legs can't be smaller than a limit value δ=40 mm. The functional dependency can be summarized as follows:

$$x_{AEP} = x_{ant} - \Delta_x + f(x_{ant} - x_{ref}) \tag{5.6a}$$

$$y_{AEP} = y_{ant} - \Delta_y + g(y_{ant} - y_{ref}) \tag{5.6b}$$

$$z_{AEP} = z_{ant} \tag{5.6c}$$

$$\sqrt{(x_{AEP} - x_{ant})^2 + (x_{AEP} - x_{ant})^2 + (x_{AEP} - x_{ant})^2} > \delta \tag{5.6d}$$

The functions f and g have a linear behavior. The relative weights between the two influences have been found by an hand-tuning process. The correspondent behavior is shown in figure 5.12 (left side). When these relationships are known it's possible to produce sets of inputs and correspondent sets of outputs and to proceed with the same training process as described above. In this case to obtain sufficiently correct results it's necessary to use higher number of hidden units: a good compromise between precision and network complexity is given by an hidden layer with twelve neurons. The results obtained with this configuration are shown in figure 5.12 (right side). The calculation error of the network on the x-y plane is smaller than the one in the simplified design. This happens because the number of hidden units is much greater than in the previous case.

5.5 Results

In this section the results obtained for different gait pattern will be illustrated. A single input to the controller, the gain on advancing velocity G_v, varies the robot speed from zero to the maximum value which is approximatevely of 160 mm/s. As the speed is varied, a continuous range of statically stable gaits is produced, as shown in subsection 5.5.1. In subsection 5.5.2 the critical case of leg failure is illustrated. For each gait, the first figure represents the gait diagram, the second one the advancing speed proposed by the gait controller (left side) and the longitudinal x position in the body reference frame for each leg as function of the time (right side).

The robot begins to walk always in the same reference configuration, but the results have been shown when the regime condition has been reached. This is necessary to do a correct comparison among different gaits, because the slower is the gait, the longer is its transient.

5.5.1 Wave gaits

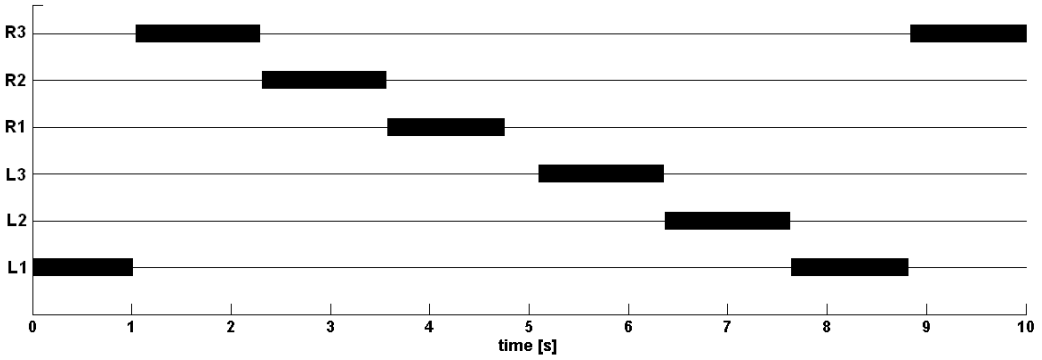


Figure 5.13: Gait diagram of a slow wave gait (β=5/6).

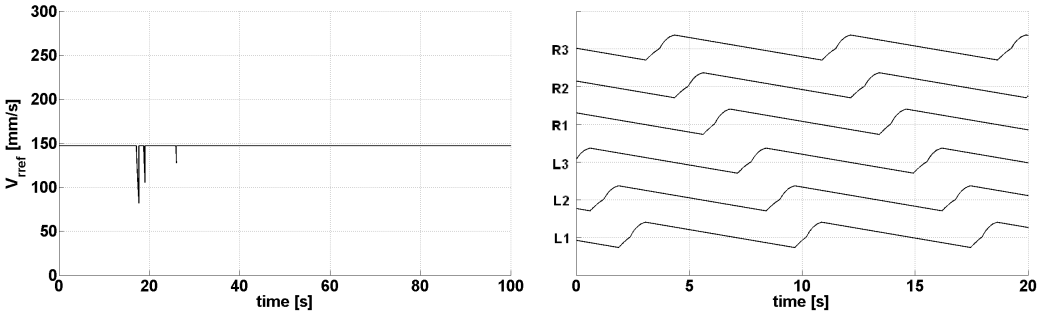


Figure 5.14: Proposed advancing speed and stepping patterns of each leg of a slow wave gait (β=5/6).

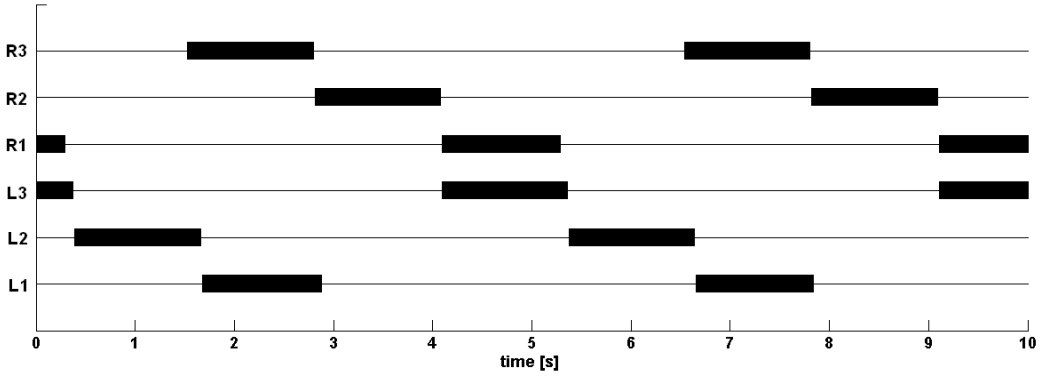


Figure 5.15: Gait diagram of a slow ripple gait (β=3/4).

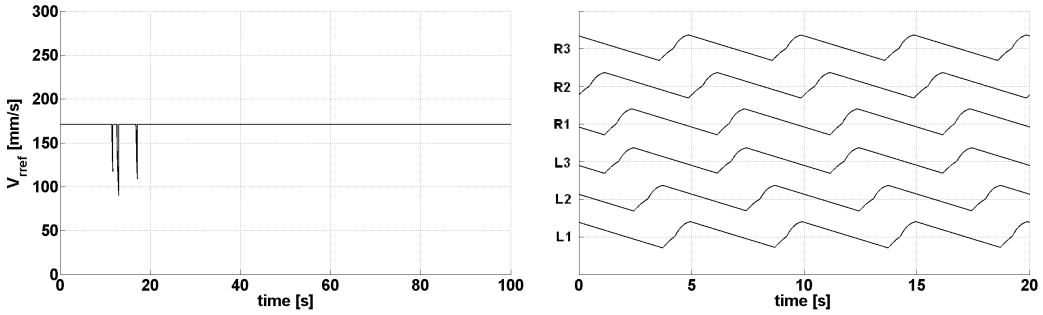


Figure 5.16: Proposed advancing speed and stepping patterns of each leg of a slow ripple gait (β=3/4).

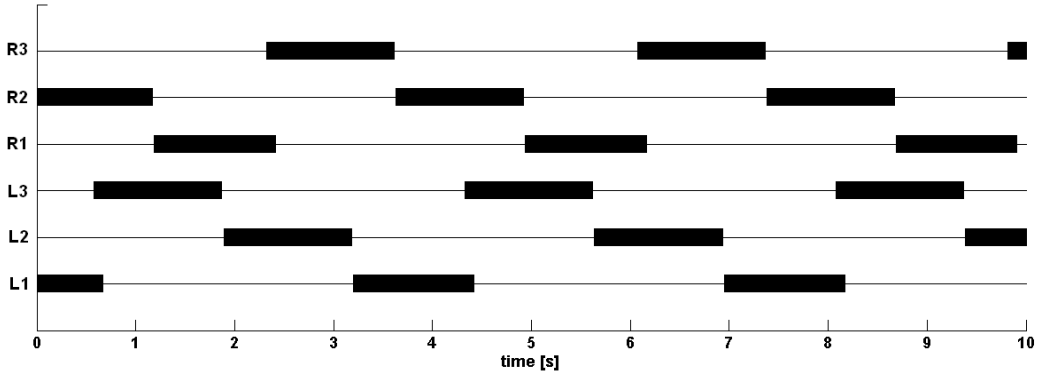


Figure 5.17: Gait diagram of a fast ripple gait (β=2/3).

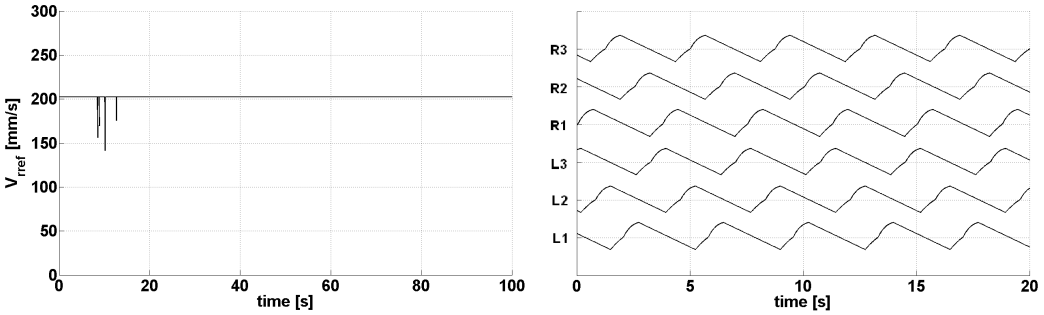


Figure 5.18: Proposed advancing speed and stepping patterns of each leg of a fast ripple gait (β=2/3).

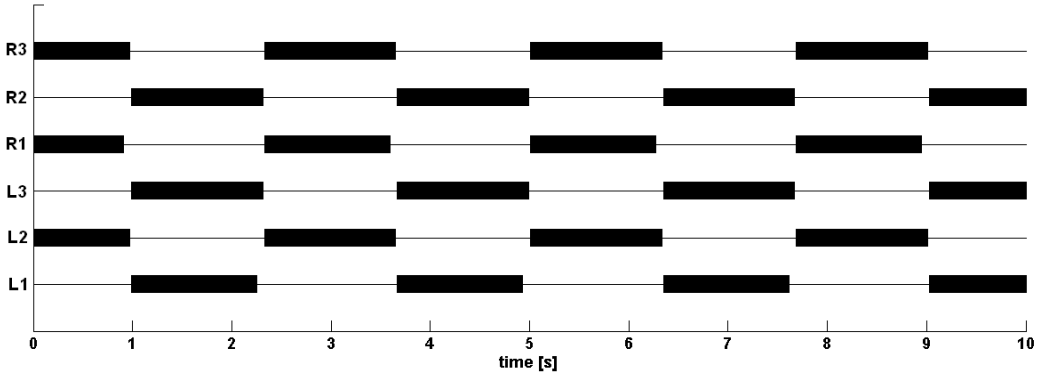


Figure 5.19: Gait diagram of a tripod gait (β=1/2).

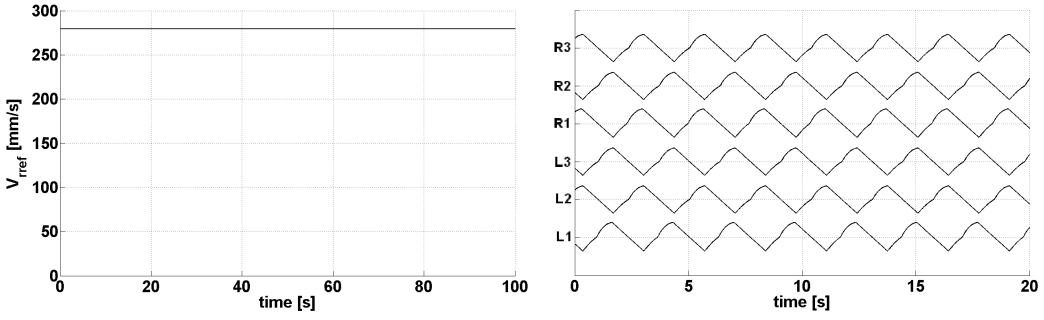


Figure 5.20: Proposed advancing speed and stepping patterns of each leg of a tripod gait (β=1/2).

The gait diagrams previously illustrated, show how the coordination system fulfils all of the requirement: the gaits range from the slow wave gait to the tripod gait, all of which display a metachronal back-to-front wave. More important, the gait controller avoids to any couple of neighboring leg to be in swing phase at the same time, automatically ensuring a statically stable motion. These results show also an almost perfect accordance with the ones obtained by Wilson [27] in its work on insects (see figure 3.10). The main difference is that, in an actual insect gait, between the swing phases of a couple of neighboring legs there is always a period during which both of them are in stance. This is a further strategy to improve the robustness of the gait that hasn't been considered for the NEMeSys controller.

The trend for the proposed velocity is the same for every gait: after a brief transient phase, during which its value can repeatedly decrease returning to the reference value in less than a second, the gait reaches its regime condition and the proposed velocity remains constant ensuring the regularity of the motion. This means that the main disadvantage of this controller, the possibility to produce an irregular advancing motion, doesn't occur in the entire range of the wave gaits.

The behavior of the x-position of the foot allows to identify two characteristics of a stereotypical wave-gait. The former is that the duration of a swing phase remains constant, independently from the global advancing velocity that is regulated by the speed of the protracting legs only. The latter is the 180-degree phasing that exist between two contralateral legs. Both of these two behaviors are correctly simulated by the gait controller in all the evaluated gait pattern.

It may be interesting to evaluate quantitatively the effect of the reference velocity command G_v on the two most important gait parameter: the duty factor β and the actual advancing speed. In both the case the functions $\beta(G_v)$ and $V(G_v)$ monotonically decrease as shown in figure 5.21. It's possible to obtain analytical expression of these functions by a least mean square approximation. It gives the following results for the duty factor β:

$$\beta = 0.7853 \times G_v^2 - 1.8847 \times G_v + 1.6047 \tag{5.7}$$

and for the global advancing speed V:

$$V = 278.68 \times G_v - 113.13 \tag{5.8}$$

These two functions can be very useful to command the actual robot in a real-time application. By inverting them it's possible to assign directly, as command, the advancing speed or the gait form, expressed in terms of duty factor.

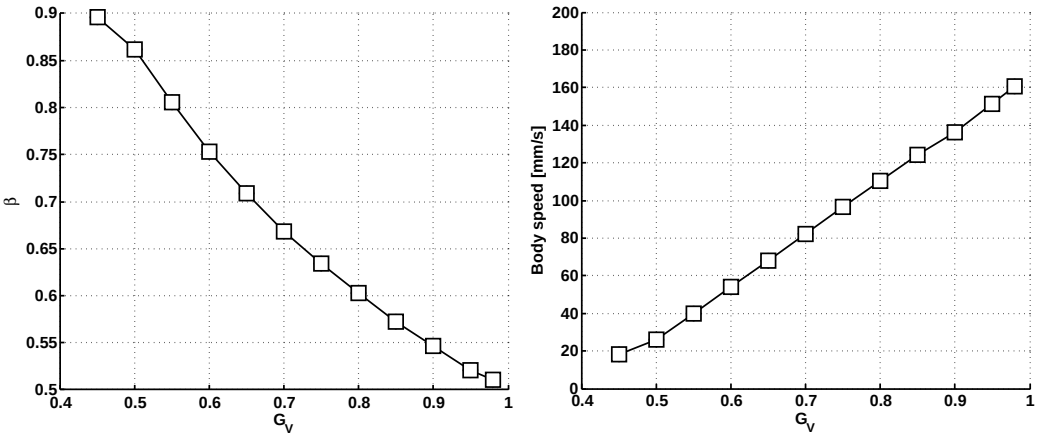


Figure 5.21: Effects of changes in the velocity command G_v on duty factor β and on body advancing speed.

It's important to note that to a given value of G_v corresponds a regime value of advancing speed different from the relative proposed one V_{rref}. The ratio between the two is constant and equal to 0.415. This is not a problem, because the trend of V_{rref} interest only from a qualitative point of view. To obtain the same values is sufficient to normalize it by adding another gain to the stance net.

A last consideration: the two analytical relation shown above works only if G_v>0.5 and G_v<1. Below this range (G_v=0.45 in the figure) the behavior changes and becomes less regular. This is not a problem because the advancing velocities in this range of gains are too much low to be effectively employable during the motion. Above the limit, the global advancing speeds doesn't change because the gait controller stops the legs that walk too much faster in order to maintain a tripod gait. This behavior is necessary to ensure a statically stable motion with always at least three non-neighboring legs that touch the ground. Moreover, although the lack of regularity, the behaviors maintain their trends since the two curve are monotonically decreasing in the entire range of permitted G_v values.

5.5.2 Leg failure

A major advantage of a hexapod robot on quadruped and biped one is its capability to maintain a statically stable posture also when one, or even two legs, are in failure. This means that the leg cannot correctly follows the trajectory given by the controller because one or more sensors or actuators don't work or because the structure is broken. For the stability this implies that the leg isn't able to support its part of the body weight or from a more formal point of view, that it can't be a vertex of the supporting polygon

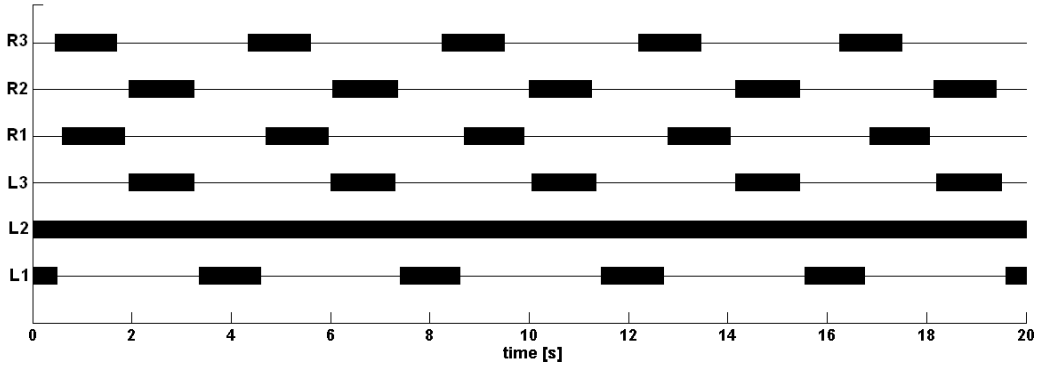


Figure 5.22: Gait diagram with a leg failure on L2 (G_v=0.65).

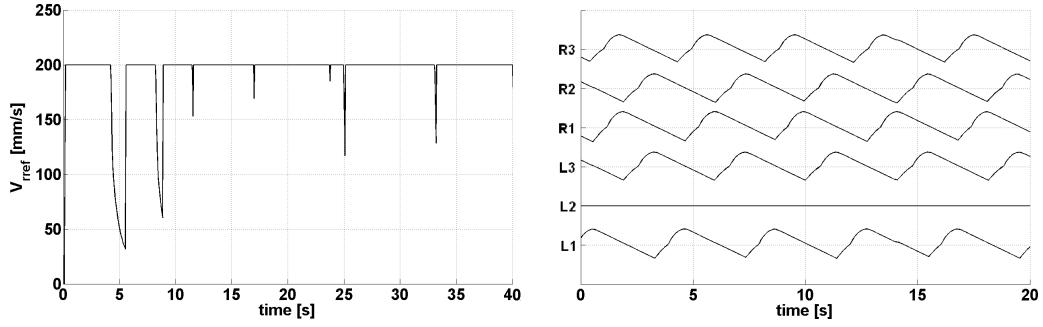


Figure 5.23: Proposed advancing speed and stepping patterns of each leg with a leg failure on L2.

like it was always in swing phase (see subsection 3.6.1).

A coordination method must allows a stable motion also when the failure of at least one leg occurs. This can be obtained in a easy way with a method based on local rules like the one adopted in this work. It is sufficient to switch-off the broken leg and to bypass the rules by applying them directly between the other limbs. It's obvious that a high-level fault-detection system is necessary in order to understand when switch-off a leg and bypass it.

In this section a failure in the middle left leg (L2) has been considered: in this case the hind and front legs become neighbors and the coordination has been applied between them. In figures 5.22 and 5.23 the resulting motion is shown. The main difference from a wave gait is the loss of regularity in the global advancing velocity. Another degradation in the performance is the loss of the metachronal wave in the gait pattern. This can be easily justified by remembering that the system has been designed in order to work optimally with six legs. The static stability is maintained with a velocity command G_v<0.7, but above this value it isn't ensured anymore.

Chapter 6

Numerical results

This chapter describes the derivation of a dynamic model of NEMeSys. The complete model is too complex for formal derivation, so a numerical approach based on a multibody software was used. The first section discusses aspects of modeling walking robots, gives general and specific assumptions to NEMeSys. Then section 6.2 describes the results obtained by an application of the controller developed in the previous chapters.

6.1 Model

In order to simulate the robot's behavior, some kind of model of a legged system is necessary, as underlined by Riddenström [49]. This virtual environment is also very useful for analysis and control design. A relatively complete simulation of a walking robot requires several kinds of models:

- kinematic differential equations and dynamic differential equations of the robot's rigid body model;
- actuator and sensor models;
- models of the environment, such as a model of the ground and the interaction between foot and ground, including ground geometry and characteristics or a model of external and internal disturbances;
- model of control implementation and communication system.

To test the controller it is important the introduction of a dynamic model of the robot and of dynamics effects in the components and in the interaction with the environment. Without this complexity it would be impossible to understand how the system effectively reacts to the gravity or if it's really able to walk upon the terrain. For the design phase, on the other hand, it might be enough just a partial kinematic description of the robot as shown in the previous chapters.

The usage of an analytical description of the dynamics and the kinematics could be interesting with a traditional control system design that requires a complete knowledge of the equations that control the motion. In this case a partial definition of the kinematic equation that describes the robot has been sufficient to design the controller completely, so, to

simulate the dynamics, a simpler method can be used. For example, this approach could be the usage of a multibody model that allows to describe the whole robot with a very immediate procedure. A multibody model is typically used to describe the dynamic behavior of interconnected rigid or flexible bodies, each of which may undergo large translational and rotational displacements. In the present case the small load levels and the low frequencies allows to approximate each component of the robot as a rigid body.

Note that there is, of course, a tradeoff between using detailed models and the speed with which the simulation runs. Therefore, models of sensors, control implementation and communication have been included and used in special cases only. On the other hand, the rigid body model and ground model are always used, whereas different types of actuator models have been switched between frequently.

6.1.1 Simulation environment

To simulate the behavior of a multibody model a great number of available softwares exist, such as MSC ADAMS or MBDyn. The main problem of these programs is that it's quite difficult to interface them with software developed in different languages. In the present case the choice of the software mainly depends on two different requirements:

- to interface the simulation environment with the control system that has been designed in MATLAB and converted in a C code;
- to find a user-friendly multibody software as simple as possible but able to post process the simulation's results.

The best solution to this problem is given by MATLAB itself. MATLAB is an integrated environment that combines numeric computation with graphics and a high-level programming language. Additional MATLAB tools are called toolboxes. Simulink/SimMechanics is a MATLAB toolbox that provides a simulation and prototyping environment for modeling, simulating and analyzing multibody systems [50]. As any other Simulink application, it has a graphical interface for creating and working with block diagrams.

6.1.2 Multibody model

The rigid body model can be described by the following general assumptions:

- the six articulated leg are attached to a main body, i.e. the trunk or simply body;

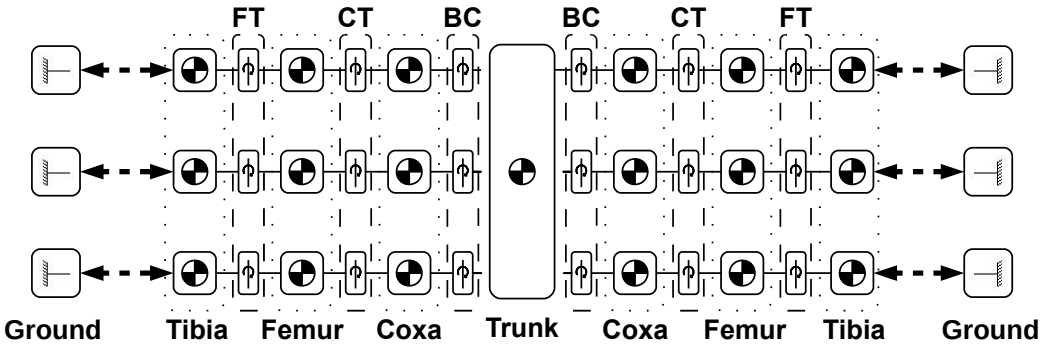


Figure 6.1: Conceptual scheme of the complete multibody model.

- the force or torque applied at each joint is the output of an actuator model, or zero for an non actuated joint. Friction, backlash and elasticity in the power transmission are modeled in the actuator model. Similarly, the actuator model also includes models of the physical joint limits;

- the interaction with the environment is completely described by external forces, i.e. outputs of a ground model or baricentric gravitational effects.

The last assumption implies that the rigid body model is modeled as an open mechanical loop regardless of the number of feet in contact with the ground. A different approach could have been to assume that a foot in contact with the ground creates a closed mechanical loop, with a different set of differential equations since the number of degrees of freedom has changed. However, the open loop alternative was chosen since it makes it easier to model slipping and a wide variety of ground characteristics by simply changing a little set of parameters.

Each leg is structurally identical and consists of three rigid bodies that correspond to the three sections of the leg, as described in section 3.2, with rotational joints between each couple of bodies. This assumption implies that some mechanical details of NEMeSys have been ignored:

- the mechanisms that physically composed each joint such as bevel gears and shafts aren't modeled as separate bodies, but lumped with the section on which are connected;

- the foot is not modeled as a separate rigid body. Instead, the foot's mass and inertia is lumped with the tibia ones. Moreover, one may neglect, in a complete measure, the motion of the additional DoFs

Part	N°	Length [mm]	Mass [g]
Trunk	1	574	2626
Coxa	6	53	27.9
Femur	6	126	264
Tibia	6	167	258
TOTAL	19	574	5925

Table 6.1: Multibody model parameters.

introduced by the mechanism necessary to permit the foot's force sensor's correct work.

A conceptual scheme of the complete mutibody model used in the dynamic simulation is show in figure 6.1. The Simulink/SimMechanic model allows to describe each body only by a reduced set of parameters and the joint only by its axis. This means that it's possible to adapt the model to every change or correction in the configuration of the actual walking platform. The reference values used for the simulation are the final data directly measured on the real robot once it has built on, except for the inertia that have been obtained, starting from the Computer-Aided Design (CAD) models of each single part, by a specific function offered by the modeling program. The measured data have been summarized in table 6.1.

In SimMechanics each body is modeled by the definition of points, placed on the body itself, identified by a local coordinate system with an orientation that can be defined in respect to another body at which this one is jointed or in the global system. In the same way a joint between two bodies represent an active or fixed DoF that can be defined in global axis or relative to one of the bodies that links. As global reference has been chosen the centre of gravity of the trunk of the robot, but with a nonzero height equal to distance from the ground in the reference posture. The central body that models the trunk is simply defined by the six reference systems by which the legs are attached to it.

6.1.3 Actuators model

The robot mounts 18 brushed DC electric motors, each of them acting on the DoF of each single joint. The choice among different actuators models has been done by a trade-off that takes into account the simulation speed against the accordance of the results with the experimental data. The models range from the ideal torque source up to and including the DC motor's electrical and mechanical dynamics, viscous damping and linear spring/damping characteristics. The most effective model of the electrical

dynamics is the classic RL scheme that can be mathematically summarized as follows:

$$\begin{cases} V_a = R_a i + L_a \dfrac{di}{dt} + E_g \\ E_g = K_e \omega \\ J\dot{\omega} = C - K_t i \end{cases} \tag{6.1}$$

where V_a is the input voltage, i the electrical current, ω the rotational velocity of the motor shaft and E_g the electromotive induced force. The values K_t and K_e are the torque constant and the electric constant of the actuator respectively. Moreover, J is the rotor inertia, L_a the motor inductance and C the output torque given by the motor. The value of L_a is typically sufficiently small to ignore the derivative term: this is equivalent to state that the transient time of the electric dynamic of the motor is much shorter than the other time constants of the problem. The value of J can be neglected too, thus ignoring the internal mechanical dynamics. These two simplifications reduce the equations to the following form:

$$\begin{cases} V_a = R_a i + K_e \omega \\ C = K_t i \end{cases} \tag{6.2}$$

The mechanical effects introduced in the model don't change the simulated behavior of the robot. The stiffness of the motor is sufficient high to neglect it and no significant damping effects have been found. The only mechanical problem is the backlash given by the planetary gears of the transmission. It has been modeled like an uncertainty on the angular value at each joint. Unmodeled problems, including wire pulley dynamics and slippage as well as the motor's temperature dependence might occur during the experimentations on the actual robot, but only with minor effects. Experience has shown that the motor parameters are really temperature dependent, but if the motor works for a little time and at low speed, these effects can be neglected.

6.1.4 Sensors model

The robot mounts only two type of sensors: linear potentiometers to measure angular displacement and force sensors to identify the ground contact. The potentiometer is fundamentally a three terminal resistor with a sliding contact that forms a variable voltage divider. A simplified version of the electric scheme of a loaded potentiometer is illustrated in figure 6.2. From this structure descends a relation between the voltages and the resistances:

$$\frac{V_L}{V_S} = \frac{R_2 R_L}{R_1 R_L + R_2 R_L + R_1 R_2} \tag{6.3}$$

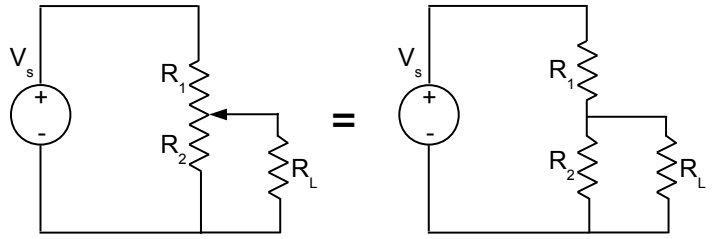


Figure 6.2: Electric scheme of a potentiometer with a resistive load.

This means that if the resistance is linearly dependent on the angular displacement the angle measured by the potentiometer can then be directly related to the output voltage with a simple linear relation as follows:

$$\theta = \frac{\theta_{MAX}}{2}\left(V_{out} - \frac{V_{MAX}}{2}\right) \tag{6.4}$$

where θ_{MAX} is the amplitude of the field of measure V_{out} is the measured tension and V_{MAX} is full scale of the instrument. This formulation shows how the potentiometer doesn't have any dynamic behavior and its effects can be simply modeled as a static module. For more detail on sensors modeling see Doebelin [51]. The only addition to the model given by these sensors is an electric noise that changes the measured signals. To correctly simulate the behavior of the potentiometers it's necessary to evaluate this noise by an identification procedure that can be summarized as follows:

- measure a signal with a null input, equivalent to measure only noise;
- calculate the Power Spectral Density (PSD) of this signal;
- verify that the signal is ergodic by evaluating whether the statistical parameters are stationary.

The process allows to found that the PSD of the measurement noise is almost constant for all the frequencies in the spectrum. This means that the noise can be modeled as a white noise with a variance calculated starting from the spectral analysis of the measured data and equal to 5.12×10^{-5} V^2.

The problem is less sensitive to the characteristics of the force sensors' model because in this work they have been used only as a contact sensors: the force value is compared to a threshold and if the former is greater than the latter, the ground contact is detected. In each case these sensors show only an electrical dynamics that is sufficiently fast to be completely neglected. The only mechanical behavior that can be found is a little backlash in the mechanism that transmit the force to the sensor. It hasn't been

considered for two reasons: it is sufficiently little to become negligible compared to the other effects and its introduction complicates too much the whole model, by introducing an additional DoF in each leg. In the same way as for the potentiometers, also for the force sensors, the only behavior that must be model is an electrical noise that could be evaluated with the same method. In the actual model this contribution hasn't been inserted because a sufficiently high value of threshold for the ground contact level makes it useless.

All the sensors give an analog information, so the errors depending on discretization and quantization had to be taken into account. In the present simulation this hasn't been done because no sampling has been modeled, considering the sampling and the quantization processes sufficiently accurate to avoid any significant degradation phenomena.

6.1.5 Ground contact model

Terrain geometry is described using plane surfaces. In this work it has been assumed that the ground applies forces only and no torques, and that this action is limited to the foot contact point. This simplification can be justified by the small footpad radius (about 25 mm).

The ground force is calculated as a function of the penetration depth and the velocity with some typical parameters illustrated in table 6.2. A linear spring/damper model is easy to use, but can result in discontinuous impact forces, due to damping and nonzero impact velocities. For the force perpendicular to the surface, a nonlinear spring-damper model to avoid these effects is typically used. In the present work the formulation proposed by Lankarani and Nikravesh [52] has been used, that can be summarized as follows:

$$F_z = -k_z \delta_z^n - d_z \delta_z \dot{\delta}_z \tag{6.5}$$

where k_z and d_z are the stiffness and damping coefficients, and δ_z is the penetration depth. For the x and y-components, a smoothed viscous friction with a maximum saturation based on the vertical force is used as by . They can be summarized as follows:

$$F_x = -\gamma F_z \frac{2}{\pi} \arctan\left(d_h \frac{\dot{x}}{F_z} \frac{\pi}{2}\right) \tag{6.6a}$$

$$F_y = -\gamma F_z \frac{2}{\pi} \arctan\left(d_h \frac{\dot{y}}{F_z} \frac{\pi}{2}\right) \tag{6.6b}$$

where γ is the friction coefficient and $\dot{x}$ and $\dot{y}$ are the velocities in the x and y direction respectively.

Type of terrain	k_z $[\frac{N}{m}]$	d_z $[\frac{Ns}{m^2}]$	γ	d_h $[\frac{Ns}{m}]$	n
Weakly damped	70'000	2'000	0.7	2'000	1
Heavily damped	50'000	100'000	0.7	2'000	1

Table 6.2: Ground contact parameters for two different terrains.

6.1.6 Control model implementation

In order to simulate a limited control frequency, instead of using different simulation times for the dynamic model of the robot and the controller, a filtering of the sensory inputs and actuator commands have been performed. Similarly, signals are delayed to reflect the fact that the control is done over a bus, but the corresponding effects, e.g. irregularity in the motion, haven't been modeled. However, the simulation runs substantially slower when including this model, but the results don't change in a significant way, so usually it's not used. Moreover a single step delay has been introduced on some signals to avoid algebraic loops in the model.

Another problem that can be found during the implementation of the controller, is the definition of geometric conditions that control the some parts of motion, e.g. the transition from stance to swing. The introduction of dynamical effects produce the risk of oscillating behavior close to the cross of the condition because the solver identifies a sudden change in the parameter of the problem and tries to reduce the time step. Further problems occur when a time delay has been introduced. To avoid this kind of problems it is sufficient to introduce a threshold suitably low to not change the structure of the controller.

6.1.7 Simulation parameters

The last important consideration that should be done involves the parameters of the dynamic simulation. At first it's necessary to identify an integration method. In this case a variable step solver has been used to avoid the troubles depending on the variability of the problem due to the ground contact. Each time a leg lifts off or touches the ground, the geometry of the kinematics chains that composed the model changes and this requires a different level of accuracy. On the other hand a fixed step method had to use a time step with the same order of magnitude of the minimum time constant of the problem that in this case can be estimated as the one of the spring-damper that models the foot-ground interaction. With such a nonlinear behavior, the time step should become too much small to allow the simulation to be completed in a reasonable time.

The other main problem is the identification of which method effec-

tively use. The present model shows a quite stiff behavior, because it incorporates very fast dynamics such as the foot-ground interaction and the electric model of the DC-motor. For this reason a multistep solver for stiff problem has been used: it is a variable order integration method based on the numerical differentiation formulas (NDFs). Optionally, it uses the backward differentiation formulas (BDFs, also known as Gear's method) that are usually less efficient.

It is also requires the choice of suitable initial values and/or a method to start the simulation. In the present work all the simulations start with the six legs placed at a little distance (about 2 mm) above the ground: this method allows to avoid the generation of unreasonably high forces or oscillating effects due to an incorrect interaction with the ground.

6.2 Dynamic model results

In this section the results given by the complete simulation will be discussed. The most important difference between the kinematic evaluation used during the design process and the analysis of a model that takes into account dynamic effects is the possibility to estimate the torque level required by the actuators to produce the trajectories defined by the gait controller.

To better understand the behavior of the system the simulation has been performed in three different conditions: a slow forward walking (see subsection 6.2.1), a fast forward walking (see subsection 6.2.2) and a curve walking (see subsection 6.2.3). For each condition all the joint parameters (angles, angular velocities and torques) of the most loaded leg will be shown. For a regular wave gait, this one is a middle leg, alternatively the left one and the right one, depending on the time slice. In this case the middle left leg has been chosen (L2 in figure 5.2 or 2 in figure 5.3). Other global data such as the gait diagram and the body parameters (coordinates and velocities) will be illustrated.

6.2.1 Slow wave gait

To obtain a slow wave gait a velocity command G_v equal to 0.5 has been assigned. According to the gait analysis performed in chapter 5, equation 5.7 gives a correspondent value of duty factor β=0.859. The mean duty factor obtained by the simulation is β=0.843 with an error on the actual value that corresponds to the 1.9% of the ideal one. This is a demonstration of the effectiveness of the control approach proposed in section 5.5.

The angles and the angular velocities for the left middle leg are shown in figure 6.3 and 6.4 respectively. The behavior is very regular with a per-

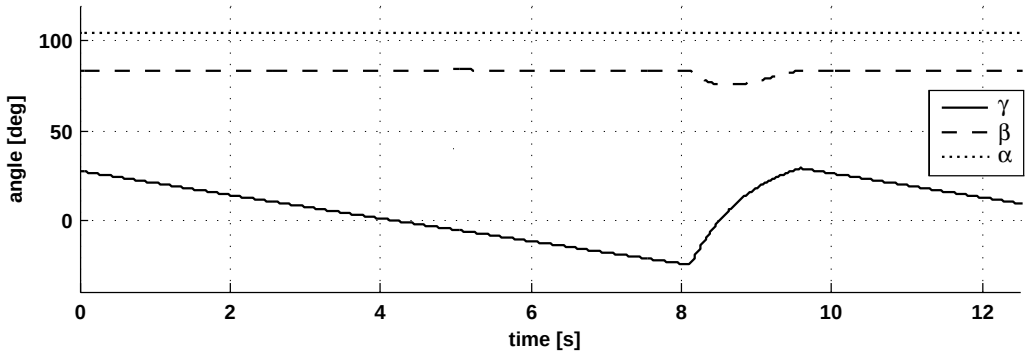


Figure 6.3: Angles of the middle left leg during a slow wave gait (G_v=0.5).

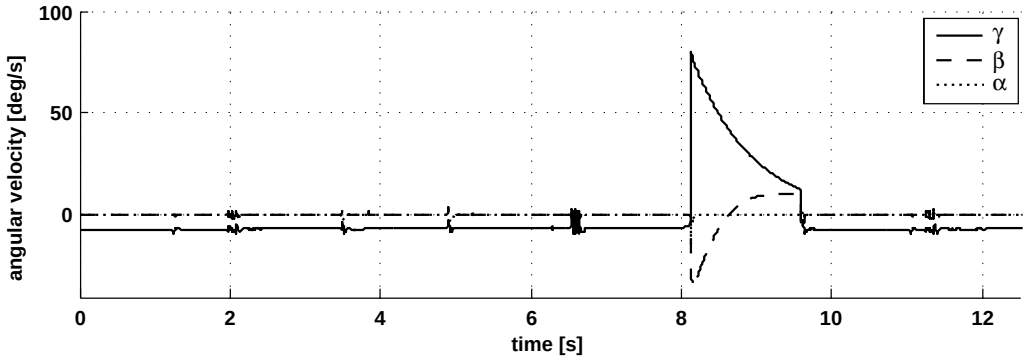


Figure 6.4: Angular velocities of the middle left leg during a slow wave gait (G_v=0.5).

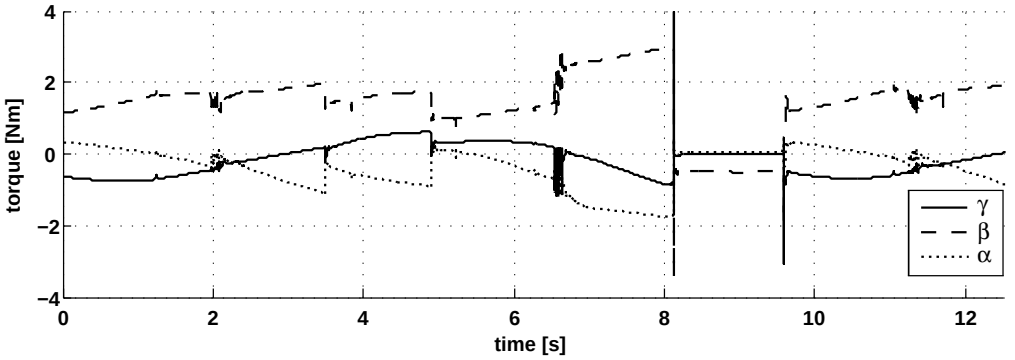


Figure 6.5: Joint torques of the middle left leg during a slow wave gait (G_v=0.5).

fect accordance with the results of the kinematic analysis, especially for the swing phase. Only small variations in the joint velocities can be observed due to the coupling with the other legs by the trunk: the entire motion of each single leg is almost independent for the other ones, as previewed

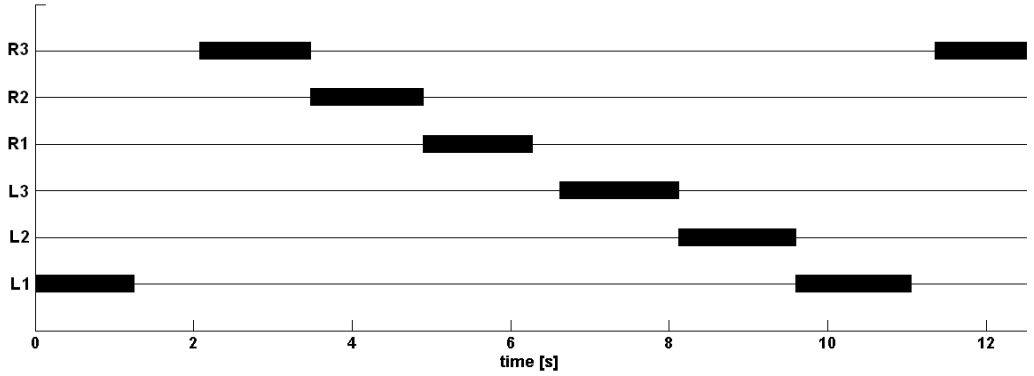


Figure 6.6: Gait diagram of a slow wave gait (G_v=0.5).

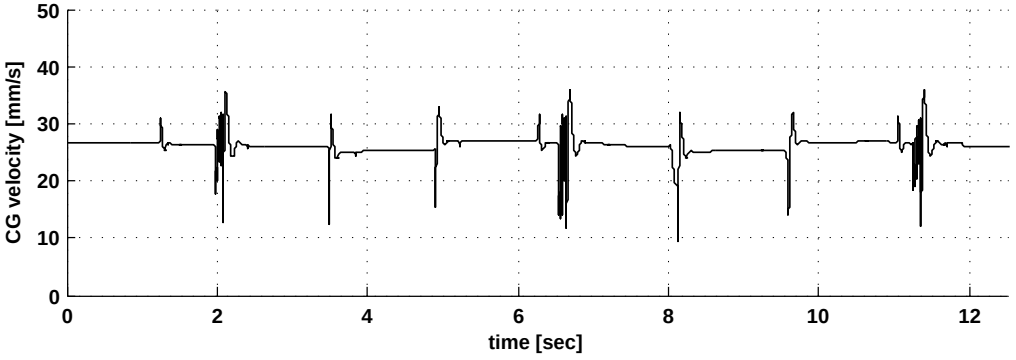


Figure 6.7: Global advancing velocity for a slow wave gait (G_v = 0.5)

by the decentralized approach. The sudden variation that occurs in the CT joint angular velocity during a stance-to-swing transition can be easily managed by the actuators.

In figure 6.5 the torques required for each joint are illustrated. With a slow motion, the torque level remains always below the limit value of 2 Nm except for the last part of the stance phase during which the CT joint as to increase its torque in order to maintain the required body height. It's obvious that it's the most stressed joint because it has to counterbalance the main external force: the gravitational one. For the same reason the BC joint is the less stressed because its action is perpendicular to this kind of loads. Two spikes appears in correspondence of phase transitions, but they are due to numerical effects.

The gait diagram in figure 6.6 shows almost the same behavior given by the kinematic analysis: one leg in aerial phase only for every instant and a metachronal wave that moves from back to front at each side. No important difference can be identified.

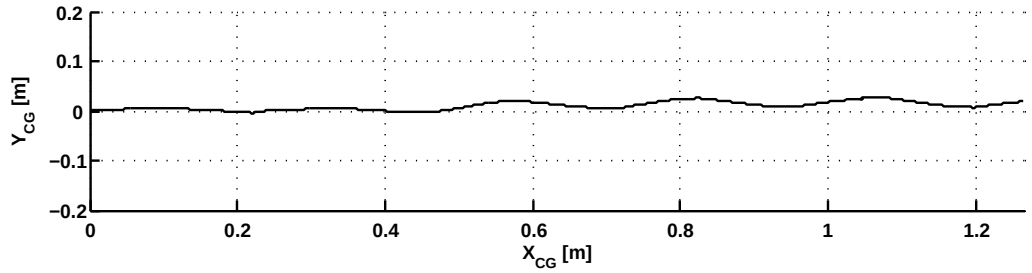


Figure 6.8: Trajectory in the x-y plane for a slow wave gait (G_v=0.5).

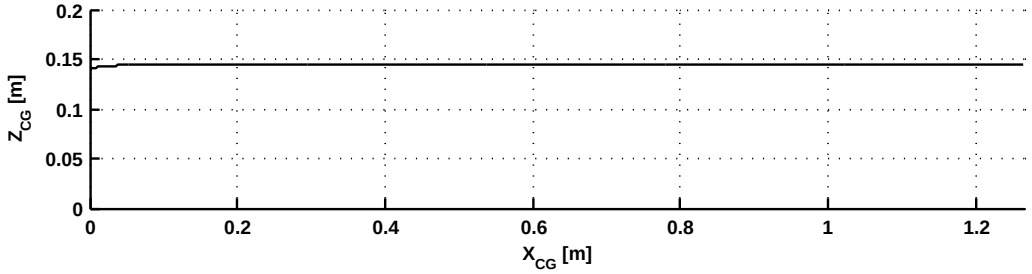


Figure 6.9: Trajectory in the x-z plane for a slow wave gait (G_v=0.5).

The global advancing velocity has been taken equal to the body CoG speed in a global reference frame. Its behavior has been illustrated in figure 6.7. With a velocity command G_v=0.5, equation 5.8 gives a value of V=26 mm/s. The actual mean speed is V=25.2 mm/s with an error of about the 3% on the prediction. Although the reference speed is the same during the entire simulation, the actual speed varies every time a phase transition occurs, with spikes across the transition itself. This happens because the nonlinearity of the problem changes the action of the controller in dependency on the current legs' configuration.

The trends of the y and z coordinates of the body's CoG have been shown in figures 6.8 and 6.9 respectively. The z coordinate is maintained perfectly constant by the feedback control system acting on the CT joint of each leg. No significant variation can be seen. The y coordinate shows a oscillating behavior, the same identified in walking insects by Kindermann [37], but also a tendency to turn on the left side. This behavior is very weak compared to the advancing one and can be easily justified remembering that no closed-loop control has been applied to this component of the motion.

6.2.2 Tripod gait

The tripod gait has been obtained in the same way as the slow wave one, by fixing a value of velocity command G_v=0.95. This value has been cho-

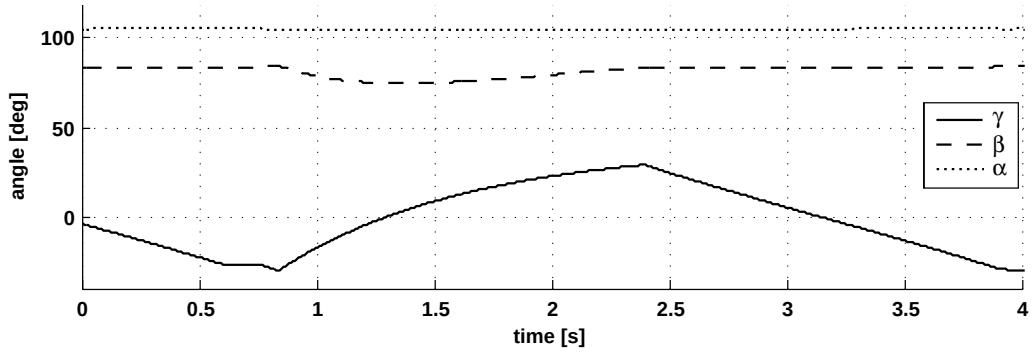


Figure 6.10: Angles of the middle left leg during a tripod gait (G_v=0.95).

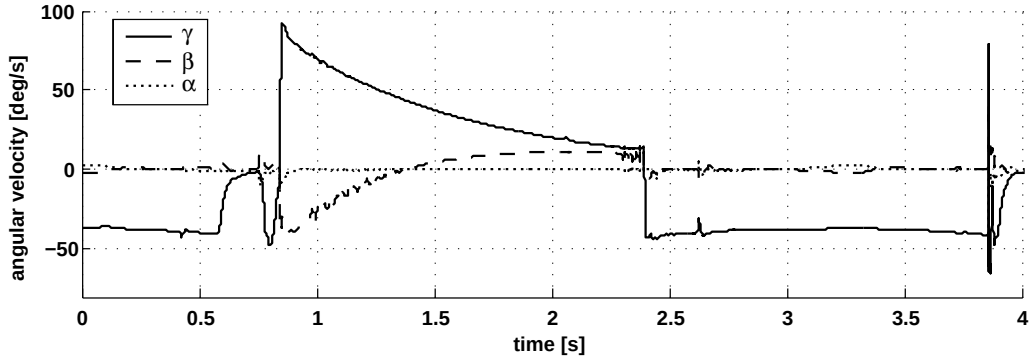


Figure 6.11: Angular velocities of the middle left leg during a tripod gait (G_v=0.95).

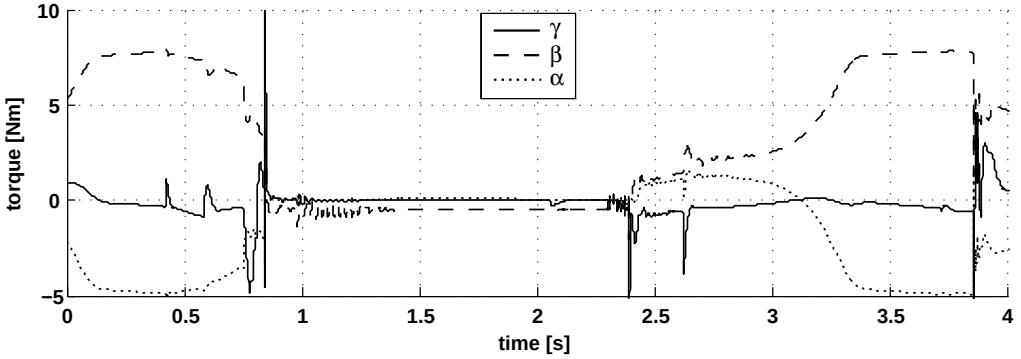


Figure 6.12: Joint torques of the middle left leg during a tripod gait (G_v=0.95).

sen in order to obtain a tripod gait that show a proposed velocities that tends to zero. This has been done to evaluate the effects of irregularity introduced by the gait controller. Equation 5.7 gives a value of duty factor β=0.521 that compared to the duty factor β=0.543 obtained by the simulation means a error of 4.2%. It's greater than the one in the slow case

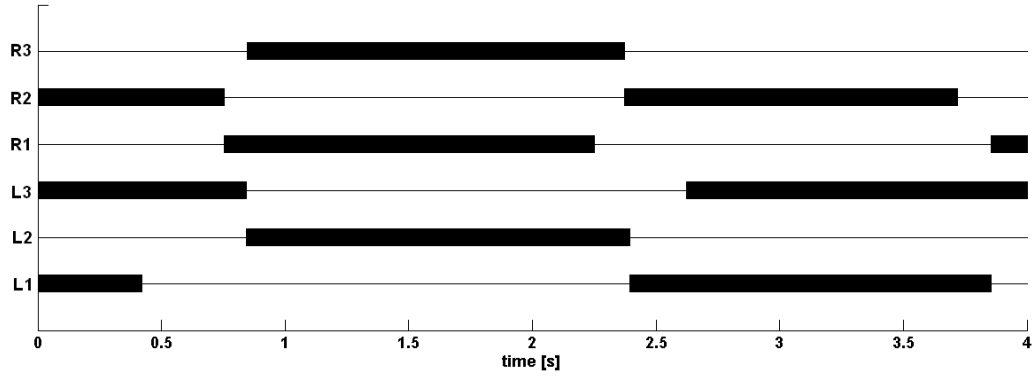


Figure 6.13: Gait diagram of a tripod gait (G_v=0.95).

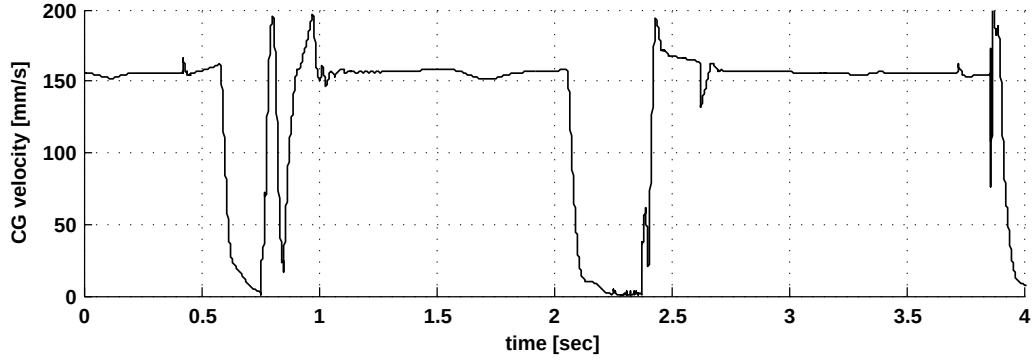


Figure 6.14: Global advancing velocity for a tripod gait (G_v = 0.95).

because the motion is already become irregular.

Angles and angular velocities of the middle leg of the left side are shown in figures 6.10 and 6.11 respectively. There are no major difference with the behaviors identified in the slow case except the fact that the motion is faster. The most interesting feature involves γ and $\dot{\gamma}$ in the instants that proceed the lift-off (t≈0.7 s). At this time as required, by the coordination system, the leg slows down in order to maintain a statically stable posture. This happens because the reference velocity is too much elevated.

The torques are illustrated in figure 6.12. During the swing phase, their activation levels are quite low: the torques required by BC and the FT joints are almost zero and the CT one only shows a significant activation to produce the lift-off of the leg. In the stance phase the situation changes: in order to propel the body at the required velocity, all the joints show an high level of activation especially during the last phase of the stance. The CT and the FT joints reach a continuous requirement of 7 Nm and of 5 Nm respectively with a non-neglectable peak of 7.5 Nm.

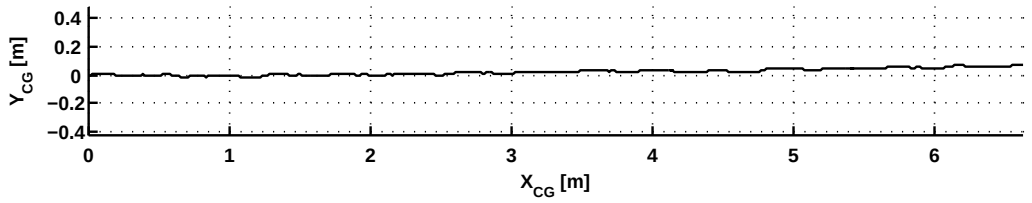


Figure 6.15: Trajectory in the x-y plane for a tripod gait (G_v=0.95).

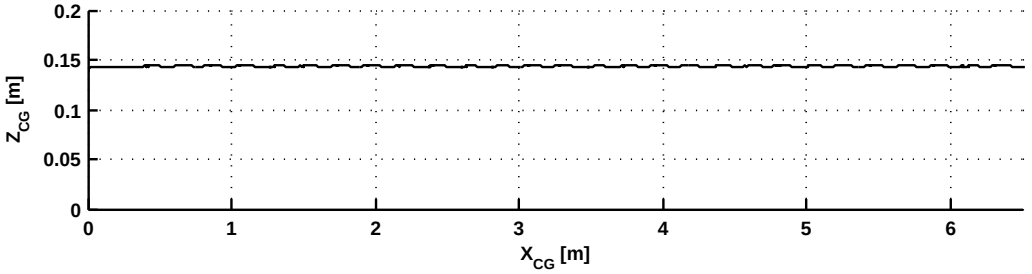


Figure 6.16: Trajectory in the x-z plane for a tripod gait (G_v=0.95).

The gait diagram (figure 6.13) and the correspondent body advancing speed (figure 6.14) illustrate the effect of a too much high velocity command on the global motion. The gait pattern becomes less regular in respect of the ideal tripod gait (see figure 5.19), by losing the 180-degrees phasing and the back-to-front wave behaviors. Nonetheless it maintained for the entire motion the static stability. This happens because the advancing velocity is reduced even up to stop the motion, as happens between 2 s and 2.5 s, in order to avoid statically unstable postures, as required by the gait controller.

For the motion in the x-y and x-z plane (figures 6.15 and 6.16) the same considerations of the slow motion can be done: the z coordinate is more regular thanks a closed-loop control system on β, the y one, although not directly controlled, shows a ignorable side-to-side heading a little tendency to turn left only. Both the trajectories are less regular than the correspondent ones in the slow case, because the global motion looses regularity in order to ensure its stability.

6.2.3 Curve walking

The curve walking can be obtained by giving to the stance controller of each leg a non null reference yaw velocity as shown in subsection 4.2.4. For this simulation a Yaw_{ref}=8 deg/s has been used, with a velocity command G_v=0.65. The correspondent curve gait generated by the control system produces a differentiation in the motion between left and right side. To underline this behavior, the joint parameters of the both middle legs have

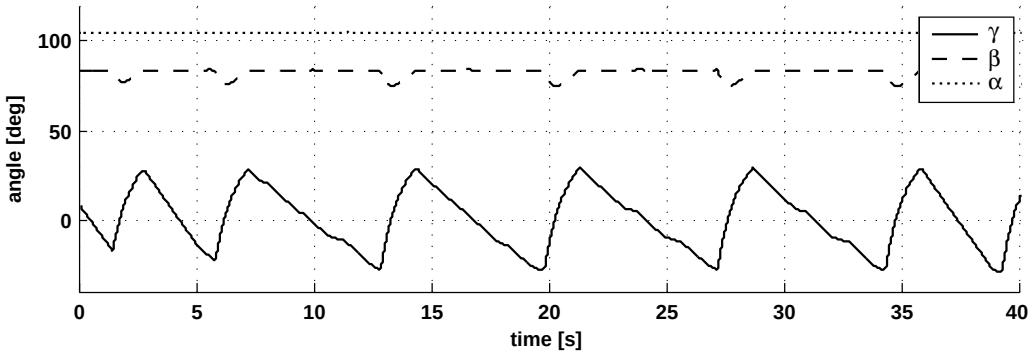


Figure 6.17: Angles of the middle left leg during a curve gait.

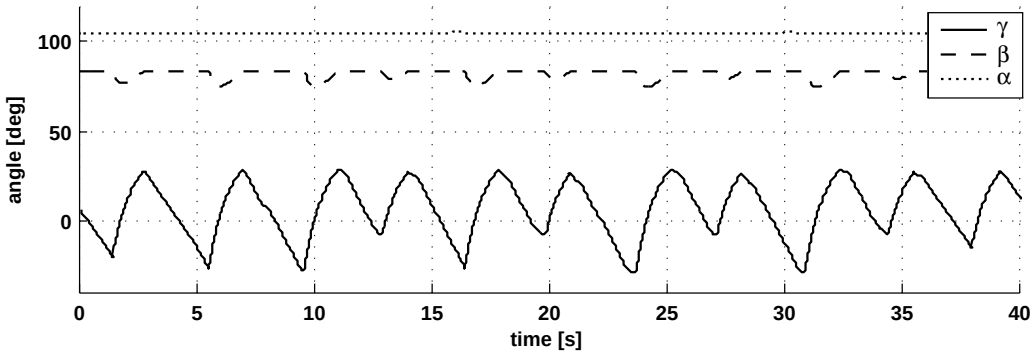


Figure 6.18: Angles of the middle right leg during a curve gait.

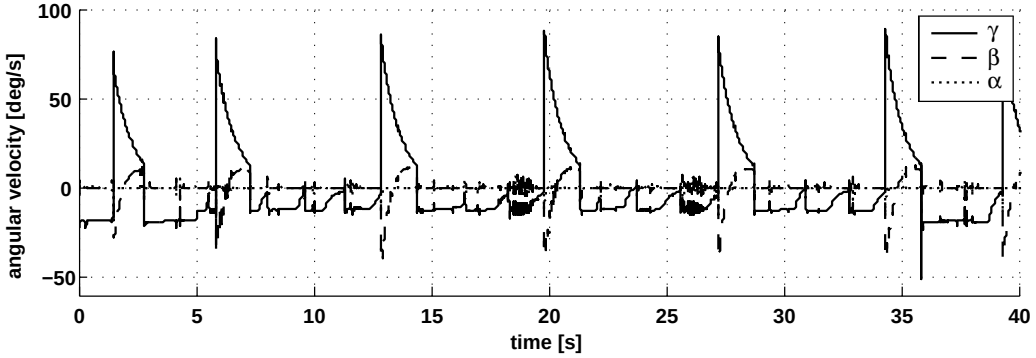


Figure 6.19: Angular velocities of the middle left leg during a curve gait.

been illustrated.

The joint angles are illustrated in figures 6.17 and 6.18, for the left and the right middle leg respectively. The correspondent angular velocities are shown in figures 6.19 and 6.20. The differentiation can be easily identified as an increase in the frequency of the right (outer) flank and a correspondent decrease in the frequency of the left (inner) one. This is one of the

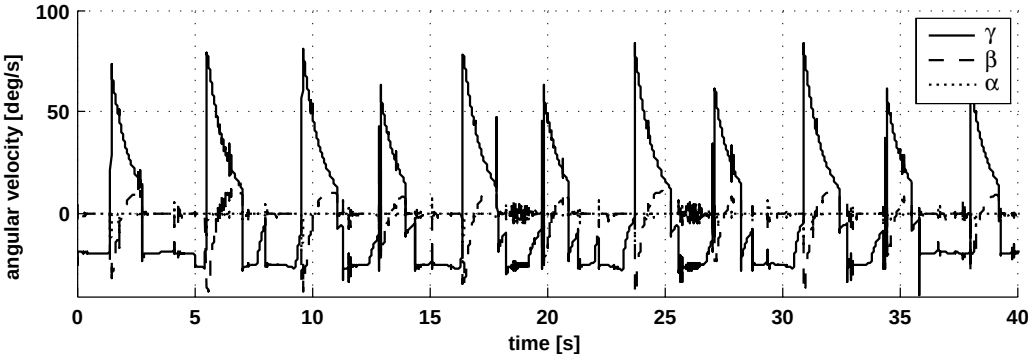


Figure 6.20: Angular velocities of the middle right leg during a curve gait.

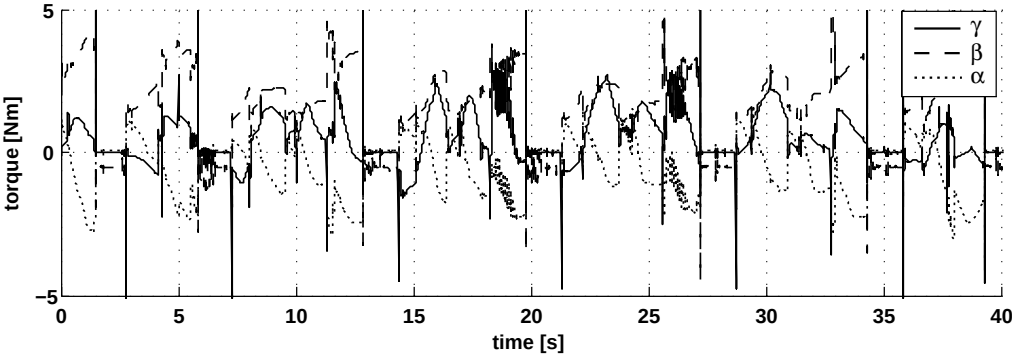


Figure 6.21: Joint torques of the middle left leg during a curve gait.

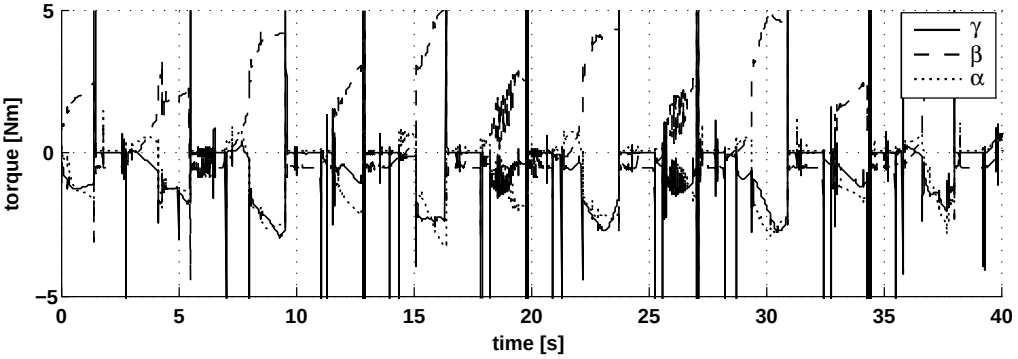


Figure 6.22: Joint torques of the middle right leg during a curve gait.

insect-like behaviors described in subsection 3.6.4. Moreover the leg on the outer flank varies its stride length.

The torques have the same qualitative behavior shown in the previous cases, but the different motion on the two side produce higher torques on the right side (figure 6.22) than on the left one (figure 6.21). The outer

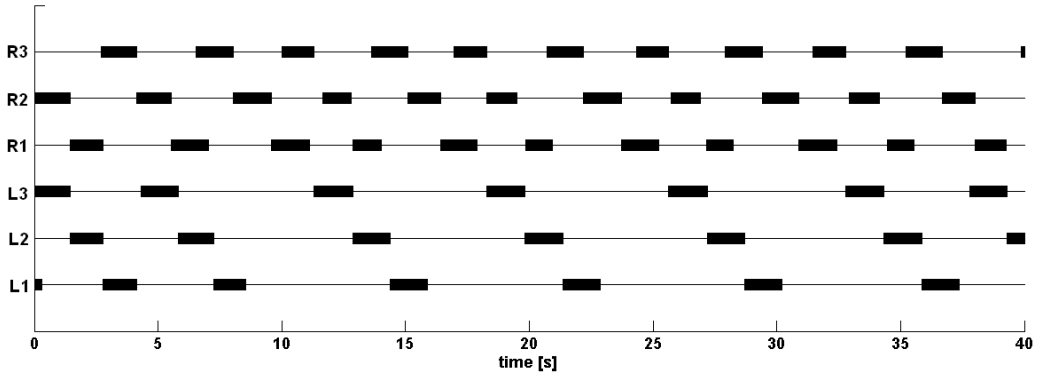


Figure 6.23: Gait diagram of a curve gait.

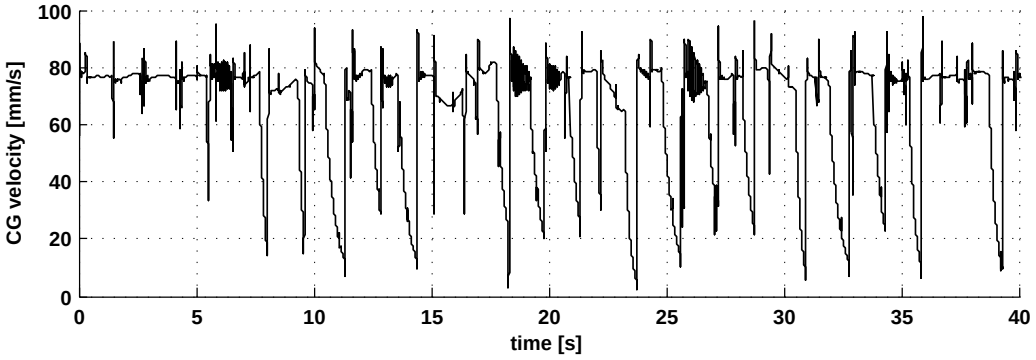


Figure 6.24: Global advancing velocity for a curve gait.

flank shows also a less regular trend with more spikes during the phase transition. This happens for its faster motion but also because its step length isn't constant.

The gait diagram in figure 6.23 shows the differentiation for all the six legs in terms of step frequency: it's smaller on the inner side than on the outer one. Nonetheless, the static stability is still ensured for the entire simulation. It's interesting to notice how the curve walking doesn't affect the metachronal wave among ipsilateral legs that is maintained in both the sides, but causes the loss of the 180-degrees phasing among contralateral legs.

The mean value of advancing velocity during the curve walking is V=58.7 mm/s. This value has been obtained by evaluating instantaneously the component of the CoG velocity tangent to the trajectory as shown in figure 6.24. If compared to the correspondent velocity V=68 mm/s calculated for the straight gait with the same G_v, it shows how the curve gait is only little slower than the standard one. The main problem is the greater irregularity of the speed that reaches almost null values very often. The

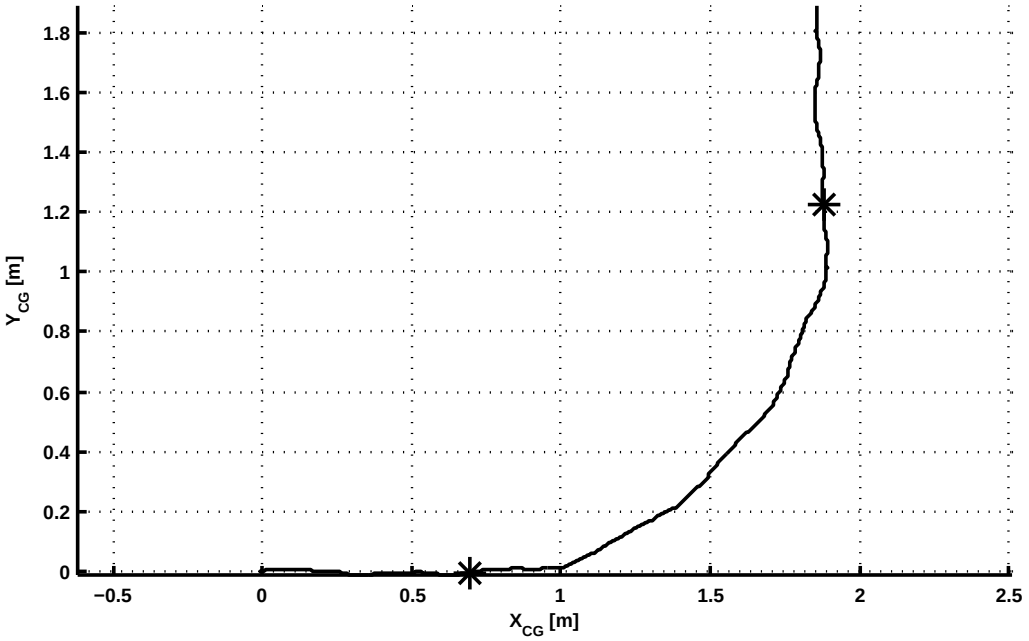


Figure 6.25: Trajectory in the x-y plane for a curve gait.

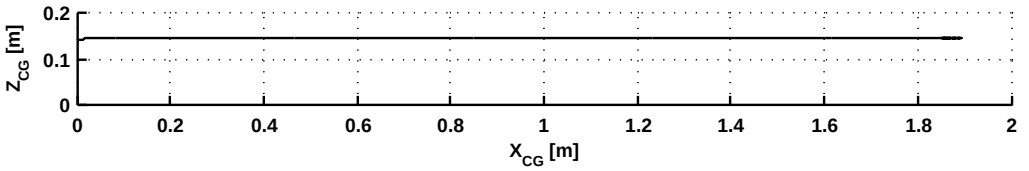


Figure 6.26: Trajectory in the x-z plane for a curve gait.

incapability to ensure a regular motion during a regular curve walking is the main problem of this kind of gait control.

The trajectories in the x-y plane (figure 6.25) illustrates in the best way the turning of the body. Asterisks indicated the start and the stop of the non null yaw reference. The robot react to this command without any remarkable delays. The z coordinates (figure 6.26) is maintained constant for all the simulation, independently from the value of the yaw command, thanks to the feedback controller.

Finally a sensitivity analysis on the turning behavior has been performed, by changing two global parameter: the yaw reference velocity Yaw_{ref} and the velocity command G_v. In figures 6.27 and 6.28 are illustrated the changes in the x-y trajectories and in the mean turning radius R respectively. The mean turning radius is the radius of the osculating circle (figure 6.29) obtained as solution of a least mean square problem. The trends show by the diagrams indicated that R doesn't vary signifi-

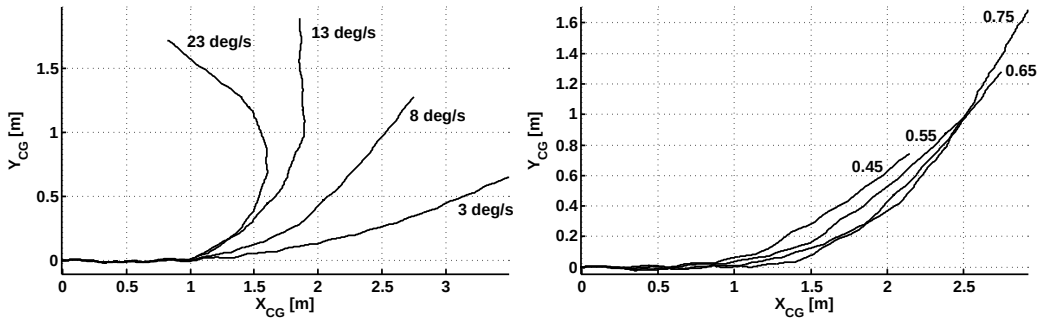


Figure 6.27: Effects of variations in turning reference velocity Yaw_{ref} and in velocity command G_v on the trajectory in x-y plane.

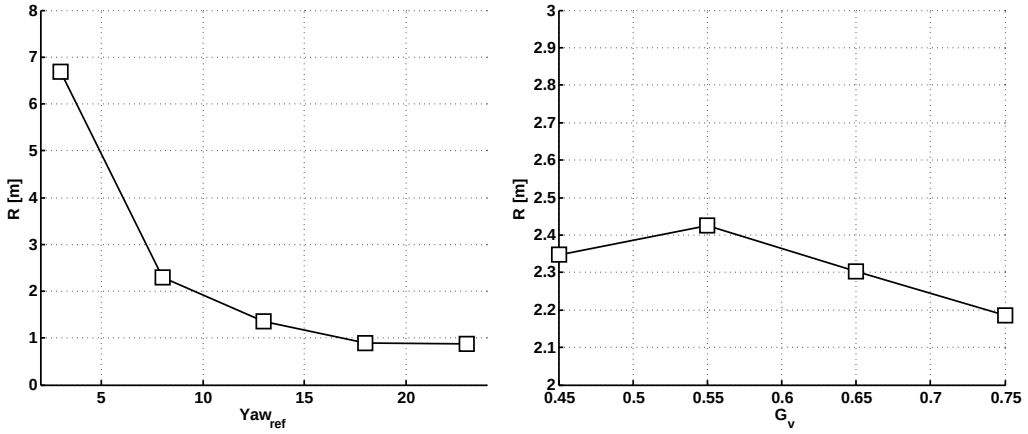


Figure 6.28: Effects of changings in turning reference velocity Yaw_{ref} and in velocity command G_v on turning radius R.

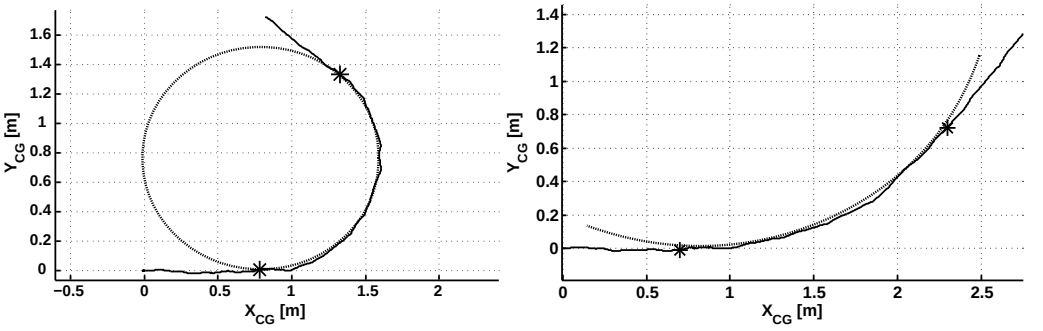


Figure 6.29: Osculating circle definition for two different curve trajectories.

cantly if the advancing speed changes, but it's inversely proportional to the yaw reference velocity. It can be easily understood remembering that if Yaw_{ref}=0 the walk becomes straight, a situation that corresponds to $R = \infty$.

Chapter 7

Walking platform

The purpose of this section is giving an overview of the robot on which the control system shown in previous chapters will be tested. First of all the design of the new mechanical platform will be presented, than the discussion will be on the changes of the electrical parts necessary to adapt the old components to the new ones (especially the contact sensors).

7.1 Design requirements

NEMeSys was initially thought as a robot for space exploration, but, after the evaluation of what this means in term of construction techniques, the target was translated to a platform to test new control structures. This choice was made to match the fixed limits on the costs and the complexity of the robot.

In this project the same guideline is followed, and it tries to recover the initial idea of space application whenever possible. Starting from these considerations some important targets have been fixed:

- reduction of the weight. This is important both for a space exploration use of NEMeSys and for a correct motion of the robot. The cost of the transfer to another planet is dependent on the weight of the payload heavily, so, the higher the weight is the higher will be the cost of the launch. Furthermore the test on the old version of the robot showed that the CP requested torque was higher than the maximum available from the propulsive system, so that a reduction of the weight is mandatory;

- reduction of the costs. To minimize the total cost is necessary to use, in the largest possible measure, components from the old robot. Particularly the motors and the planetary gear adaptors must be the old ones, so that some parameters, e.g. the dimensions of propulsive system but also the maximum torque available, are fixed. For the same reason, where possible, the use of common materials and profiles for the design of the structure is to prefer;

- reduction of the dimensions. The robot must be as small as possible, with respect of the size of the electronic boards and of the motors, taken from the old robot, but hasn't to limit the movement of the

legs, within the range set during the design of the controller (see table 4.7). Moreover, thinking about the stowage of the robot during the transfer to a planet, if possible it should be able to reach a configuration that employs the least volume possible;

- some parts must be removed without difficulties to permit changes or reparations. This requirement is very important because this robot is going to be a platform for future tests also with controllers completely different from the one presented in this work, so that changes in structure, or more probably in sensors and actuators must be possible and quite easy. This necessity is very common in the sphere of space exploration because the number of production is very small, so that is usual to update a lot of time the same object rather than building a new one.

In order to comply with this requirement the choice made is to completely rebuild the structure, to study a new way for the transmission of the torque and to modify the electronic system of the old version of NEMeSys in the least possible percentage.

7.2 Mechanical system

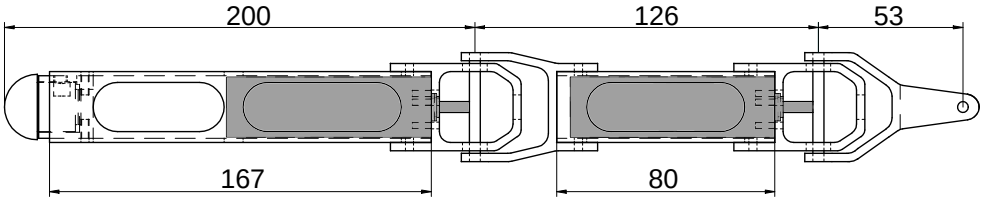
At first the new mechanical system has been designed beginning from the general layout and than focusing on the most demanding parts, especially the construction of the joints. At least some structural analysis have been made in order to calculate the distribution of internal forces and to check validate the design.

7.2.1 General layout

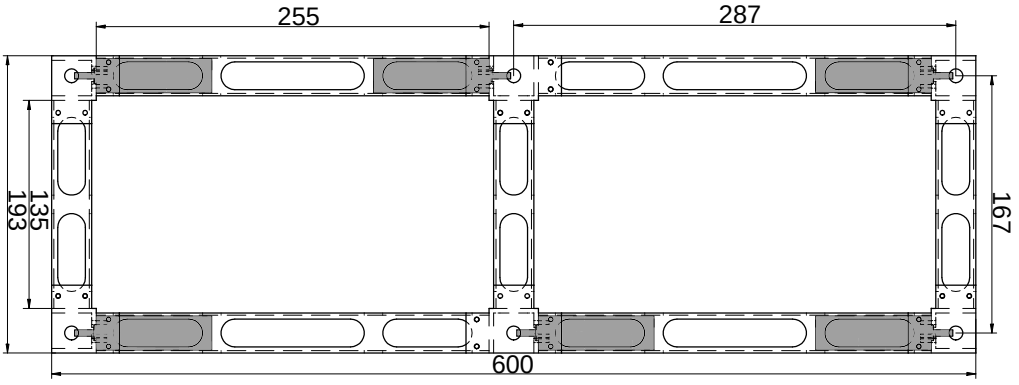
Starting from the considerations described in the previous section, a structure based on a mix of standard profiles with a square section and milled from solid parts, both in aluminum, has been designed.

This solution permits a high level of modularity because the dimensions of the body and of the legs are dependent from the length of the standard profile, so that the geometry can be easily and cheaply varied by changing the profiles themselves. On the contrary the milled from solid parts form a rigid and robust box for the transmission of the torques and, if needed, can be re-used if the shape of the robot will be changed.

In order to ensure the possibility of disassembling the robot, most parts of the junctions between the pieces are made with screws and nuts. For the shaft that connect the parts in mutual motion, a shape-based connection has been used.



7.1.1: Leg draft.



7.1.2: Body draft.

Figure 7.1: General layout of NEMeSys.

To save space the motors are placed into the profiles and the torque is transmitted to the joint by miter gear as well as the angular position sensors of the CT and FT joints. The disposition of the eighteen motors is shown in figure 7.1. On the top is represented the leg; on the bottom is shown the draft of the body. The gray parts are the motors-adaptors set. The profiles used for the robot are all of the same type, with square section of 25 mm of side dimension and a thickness of 1.3 mm. This dimension is the one that best matches the need to have the thinnest possible leg and the capability to store the motors inside the profile. As the motor has a diameter nearly to 22 mm, only a 0.5 mm free space remains between profiles and motors. In order to have a better cooling of the propulsive system, but also to reduce the weight, holes on the lateral faces of the profiles are necessary. The angles of the profile can be used to house the wires for the power supply of the motors and for the sensors.

The thickness of the milled parts has been reduced to 3 mm, that is the best trade-off found between robustness, lightness and ease of manufacturing. Also the usage of bearings between rolling parts imposes a minimum limit to the thickness of the wall.

The size of a walking robot and length/width or length/height ratios are limited by different and sometimes conflicting considerations:

- ensure the correct motion. The joints must be able to reach adequate angles and two ipsilateral legs must not to collide when the posterior one is at the AEP and the anterior one at the PEP. In this configuration a gap between the feet is desirable, so that a further motion e.g. a searching of the posterior one may be possible. This requirement fixes a lower limit to the longitudinal distance from two BC joints, and consequently to the body length;

- limit the joint torques. In section 7.3.1 the maximum torques available from the motor-adaptor set are shown. An analysis of the typical shape and posture of an hexapod robots (but also of insect like *Carausius Morosus*) display that the most stressed joint is the CT one. The torque on this joint depends (once that the weight of the robot and the gait are fixed) on the horizontal distance between CT and FT joint, function of CT-FT joints distance and of β angle. The lower is the gap between the two joints, the lower the torque required will be;

- size of the components and of the payload. In order to have compact robot, the motors have been placed into the profiles that form the structure. This poses a minimum limit to the length of the profiles. At the moment no payload is installed on the robot, but in the future a set of optical or attitude sensors are expected;

- capability of overtake obstacles. This requirement affects the height of the body from the ground. This parameter is a function of the length of the femur and of the tibia and of the value of α and β angles. Ones that the standard (at rest) height is fixed, is usual in robot tests to fix the highest obstacle that the robot has to overtake, as a percentage of the body height (usually 50-60%).

Taking the start from these consideration the best trade-off has been found with the configuration shown in figure 7.1.

At first the dimensions of the leg have been set. The femur has the minimum length necessary to carry the motor-adaptor group, so that the CT-FT joint distance is 126 mm. The reference for β has been set at 80°, giving the robot a little lateral stability. For the length of the tibia a value of 167 mm has been chosen, so that, using a reference for α of 90° in resting position, the robot should be able to overtake obstacles with a height of 70 mm. The coxa is 52 mm long, to allow a correct range of γ.

	Minimum γ [deg]	Maximum γ [deg]
Front legs	–38	+128
Middle legs	–38	+38
Posterior legs	–128	+38

Table 7.1: Mechanical limits for angle γ.

The body has a length of 604 mm, a width of 194 mm and a thickness of 32 mm. This size is the one that most matches the necessity of storing the electronic boards and the correct distance between ipsilateral legs. Once that the boards dimensions will be reduced, the width of the body could be decreased until a minimum of approximately 100 mm only by changing three profiles.

The choice of the body's length of is also a result of the setting of the resting position of the legs. In order to use the same identical controller for all the legs, the resting configuration chosen is the same for all the legs, with γ=30°, β=80° and α=90°. The range of γ around the resting position during normal walking is set to a maximum of ±30°. Is mechanical limits are the ones shown in table 7.1. Choosing the length of 600 mm for the body a minimum longitudinal distance of about 60 mm between feet of consecutive legs is reached during normal walking.

7.2.2 Joint design

In order create the eighteen degrees of freedom (three for each leg) necessary for the motion of the robot, the same number of revolute joints must be created. The joints are composed by a shaft fixed to one of the two part in mutual motion and free to rotate with respect to the other one. A couple of bearings are useful to avoid friction and early degradation of the aluminum parts.

After the creation of the joint, it is necessary to find a way to transmit

Producer	Maedler
Thansmission ratio	1:1
No. of teeth	20
B [mm]	4
d [mm]	12
ND [mm]	8
Admissible MD [Ncm]	1.8

Table 7.2: Bevel gear specifications.

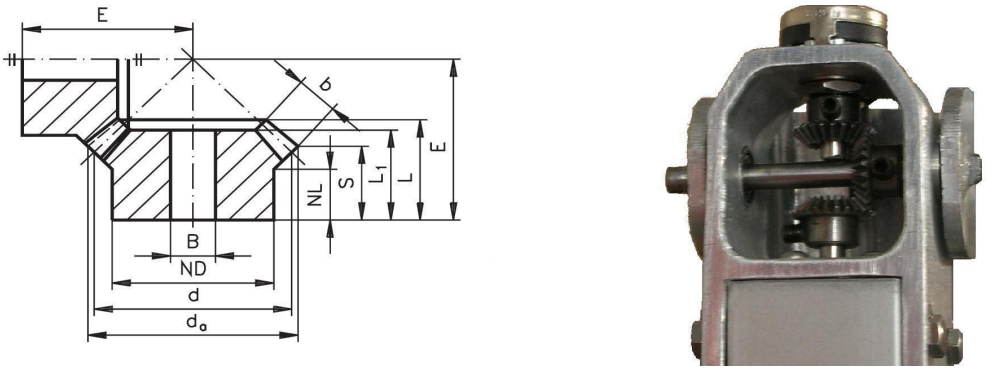


Figure 7.2: Meter gear dimension and joint example.

the torque to the shaft. Seeing that to save space the motors are oriented perpendicularly to the corresponding joint, the most simple one is to use miter gears. These components are very useful because they allow the change of the direction of the motion, but must be mounted with the utmost precision and a way to fix them to the shafts must be found. In this work this last task is performed by threaded pins screwed on the lateral side of the gears. In order to have best friction the shafts have been flattened in correspondence of the pins. In table 7.2 are presented the main data of the gear used for this work. The measures are referred to the left side of figure 7.2. In order to save space, the angular position sensors are operated by miter gears too. On the right side of figure 7.2 is shown an example of the joint between femur and tibia of NEMeSys.

7.2.3 Structural analysis

In order to confirm the correct sizing of the robot frame, some structural analysis have been performed. These analysis have been used only as control, not for the design, for which design only the data given from the profile producer have been used.

The material used for all the structural parts of Finite Element Method (FEM) models is aluminum 6061-O, that has low mechanical properties shown in table 7.3. This is because of the uncertainty about the exact quality of the material effectively used during the construction.

The first important information is the knowledge of the internal forces. For their evaluation a FEM model developed with *Femap/Nastran* software has been used. In order to obtain the correct results, this model needs the real distribution of the mass, so that inertial loads can be correctly evaluated. The distribution of stiffness can be far from the real one, causing a false deformation of the structure, but this is not important for this first calculation whose target is only to evaluate the internal forces.

Yeld stress [MPa]	55
Tensile stress [MPa]	125
Young module [GPa]	69
Density [Kg/m^3]	2700

Table 7.3: Mechanical properties of 6061-O aluminum alloy.

Because of this all the structure has been modeled by `BEAM` elements. The section of the element is a square tube with a variable thickness that ensures a distribution of mass next to the real one. The non-structural elements, e.g. the motors and the electronic boards, have been modeled as lumped mass, `MASS` element using Nastran language, connected to the beam elements with `RBE2` rigid elements. The total number of element is 340, 278 of which of beam type, 40 of mass type and 22 of rigid type and the number of nodes is 299.

The analysis made on this model are all of the static type with different postures of the legs. No dynamic analysis have been performed because the frequencies of interest are very low, so that the most important load on the robot is his own weight. The simulation of the gravitational field has been performed by applying a body load.

The postures analyzed are the ones corresponding to typical gaits. The most stressful for the structure is the tripod one, particularly on the side where only one leg is in contact with the ground, just after that the stance phase begins. In table 7.4 are reported the maximum of each internal force and the zone where are reached, with reference to the left side of figure 7.3. The positions corresponding to the numbers of the table are shown on the right side of the same figure. The most stressed part is the leg number 2, particularly the coxa and the femur for bending moment and shear force and the tibia for axial force. The latter is in the compression direction. The *Plane 1 moment* shown in table 7.4 corresponds to the maximum torque requested to the CT motor during the tripod walk and can be compared with the one obtained during the simulation shown in section 6.2. The two results present a good accordance, taking in account that the FEM analysis have been made at the beginning of the work. Moreover while the Simulink analysis is a dynamic calculation, the FEM one is only static, so that differences due to inertial loads are present.

The second step of the structural analysis is the calculation of the stress into each component of the robot. In this case the model must consider the real geometry of the parts, so that the equivalent stiffness are the correct ones. The target, in this case, is both to know the values of the stresses and to calculate the deformation of the structure. For the analysis the single

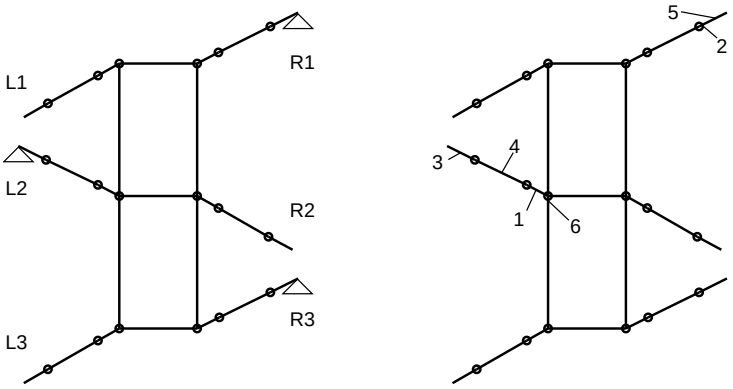


Figure 7.3: Beam model constraints and most loaded sections.

	Value	Position
Plane 1 moment [Nmm]	5999	1
Plane 2 moment [Nmm]	145.8	2
Axial force [N]	27.11	3
Plane 1 shear force [N]	25.62	4
Plane 2 shear force [N]	0.88	5
Torque force [Nmm]	1455	6

Table 7.4: Maximum of internal forces in the beam model.

component, or small group of components, have been considered, in order to have a lower number of nodes and elements. The loads applied to the parts are the maximum of the ones resulting from the beam analysis just described.

The profiles have been modeled using `PLATE` element with the thickness equal to the real one. For the milled parts two solution are possible: the first one is to model also these components with plate element; the second one is the use of `SOLID` elements. The former way allow to have a low number of nodes, reducing the computation time; the latter permits to know the distribution of the stress along the thickness of the wall. Looking to the dimensions of the milled parts it is possible to affirm that a plate modeling would describe with good accuracy the distribution of stresses as the ratio between thickness and section dimensions are around 1/10. However both the two approaches have been used.

The results of the analysis show that the most stressed profile is, as expected, the one corresponding to the femur during a tripod walk on the side where only foot is in contact with the ground. The value of the maximum Von Mises stress is 41.9 MPa.

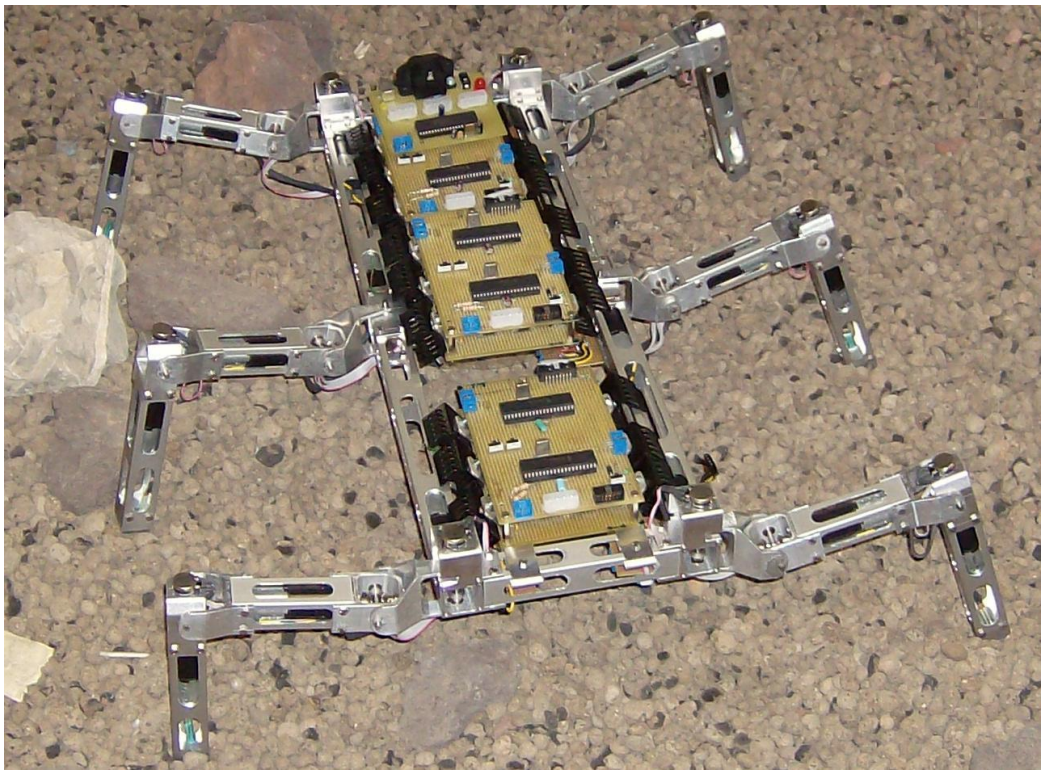


Figure 7.4: The NEMeSys robot in a simulated martian environment.

The milled parts present a significant value of stress only next to the bearings and the shafts.

The last consideration is about the use of solid or plate element to model the milled parts. The difference between the results of the two different calculation are small on condition that the nodes of the plate model are sufficiently thick.

7.3 Electrical design

The electrical system of NEMeSys is composed by three main parts:

- actuation system. To this part is entrusted the motion of the joints.

- sensor system. This subsystem is primary for a control structure like the one used in this work (see section 3.4).

- communication and connection system. This subsystem allows the data exchange between the boards and between boards and PC.

Some parts of these system are existing from the old versions of NEMeSys; some others have been updated or completely replaced by new ones.

7.3.1 DC electric motors

The actuation of the eighteen joints is assigned to as many DC electrical motors. A planetary gear set to reduce the speed and increase the torque must be introduced between the motors and the joints. These two components have been taken from the old robot, so no changes with respect of previous work have been made (see [14] for more details). In tables 7.5 and 7.6 the principal data of the components are summarized.

The data show that the maximum continuous torque generated from the motor-adapter is of 5.12 Nm, and a peak of 15 Nm can be reached. In section 6.2 the torque diagrams show that for the most of the time the necessary torque is below the maximum continuous and reaches a maximum below the permitted continuous torque for all the gaits except for the tripod one. During this last gait a peak of 7.5 Nm is achieved. This value is low in relation to the one allowed from the motor, but is high with respect to the maximum torque of the adapter. Only the wave gait respects the torque limit of the adaptors, with a maximum of 2.7 Nm. In the test made with old version of NEMeSys the exceeding of the torque didn't cause any failure, so that also in this work these components are used.

The command to the motor is given by using a Pulse Width Modulation (PWM) technique. This is a very efficient way when the is necessary an intermediate amount of electrical power. The basic idea is to use a rectangular pulse wave whose pulse width is modulated resulting in the variation of the average value of the waveform. The electrical circuit necessary for the use of PWM has been developed during the first two works on NEMeSys. For more detail see [14] and [53].

Producer	Maxon
Product code	A-max 22
Input voltage [V]	12
Maximum power [W]	6
Maximum continuous torque [mNm]	7.425
Stall torque [mNm]	22
Maximum speed [RPM]	10500
Torque constant [mNm/A]	10.9
Speed constant [RPM/V]	875

Table 7.5: Motor specifications.

Producer	Maxon
Product code	GP-22 C
Reduction ratio	690
Maximum continuous torque [Nm]	2
Maximum peak torque [Nm]	2.7

Table 7.6: Adapter specifications.

7.3.2 Angular position sensors

For angular position measure single-turn potentiometers have been used. This type of sensor is suitable for this usage because is the one that best matches the requests of littleness, low cost end quite high precision of measure. In order to reduce to the minimum the space taken from the sensors and to avoid parts protruding from the legs, the CT and FT potentiometers' axis are turned by ninety degrees with respect to the shafts of the joints and the motion is transmitted using miter gears. This solution is also used for the BC joint of central legs. For anterior and posterior leg, instead, the BC angular sensors have been installed coaxial to the shafts in order to ensure an easier assembly. The main data of the potentiometer used on the robot are shown in table 7.7. The signal from potentiometer is sent directly to an analogical input of the microcontroller of the corresponding leg and, from here, using a particular communication standard (see section 7.3.4), to the device that exchange the data with a PC through a Universal Serial Bus (USB) standard. The power supply to the potentiometer is at 5 V.

Producer	Piher
Product code	T-16SHM04N102A
Mechanical rotation angle [deg]	300±5
Electrical rotation angle [deg]	280±20
Resistance [KΩ]	1±20%

Table 7.7: Potentiometer specifications.

7.3.3 Contact force sensors

To detect the contact force between the foot and the ground a Force Sensing Resistor (FSR) has been used (see [54]). They are a polymer thick film (PTF) device which exhibits a decrease in resistance with an increase in the force applied to the active surface. This is the principal updating in the electronic part with respect to the old versions of NEMeSys. In NEMeSys

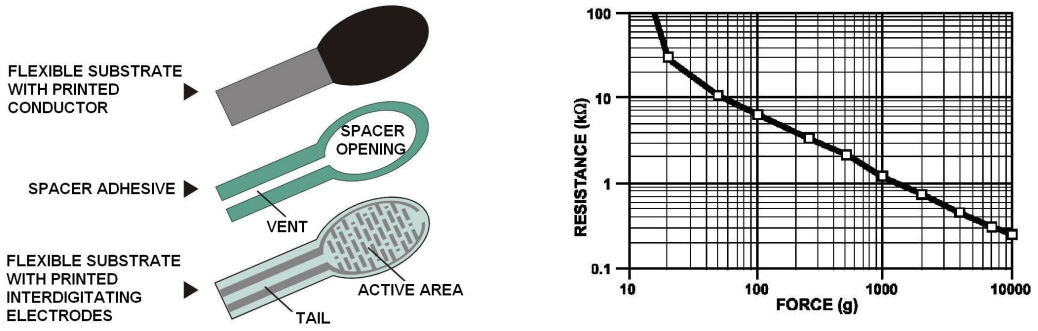


Figure 7.5: Force sensor construction and characteristic curve.

Mk I and II the contact sensor was simply a switch commanded by a movable part of the leg.

The major advantage of the contact sensor used in NEMeSys Mk III is that allows the measure of the reaction force, not only the detection of the contact. The information of the load is important to improve the control structure, especially the coordination model. Having the reaction force the coordination rule no. 5 of Cruse model could be implemented, permitting a better control of the stability. In this work only the contact detection has been used, living to further development the possibility to use the load information.

Other leads of the resistive contact sensors are the lightness (only 1 g for a sensor), the smallness (particularly the thickness is only 0.3 mm), the ease of montage (the sensor is pasted on an aluminum base), and the long lifetime (more than 10 million actuations). In table 7.8 are shown the main features of the adopted model of sensor. On the left side of figure 7.5 are shown the parts that compose the FSR. On the right side a typical force-resistance characteristic is illustrated. At low-force side a switch like response is evident. This turn-on threshold, or break force, that swings the resistance from greater than 100 kΩ to about 10 kΩ (the beginning of the dynamic range that follows a power-law) is determined by the sub-

Producer	Interlink Electronics
Product code	400
Active area diameter [mm]	5
Force resolution	better than 0.5% full scale
Turn-on force [g]	20 to 100 (depending on mechanics)
Unloaded resistance [Ω]	> 1M

Table 7.8: Force sensor specifications.

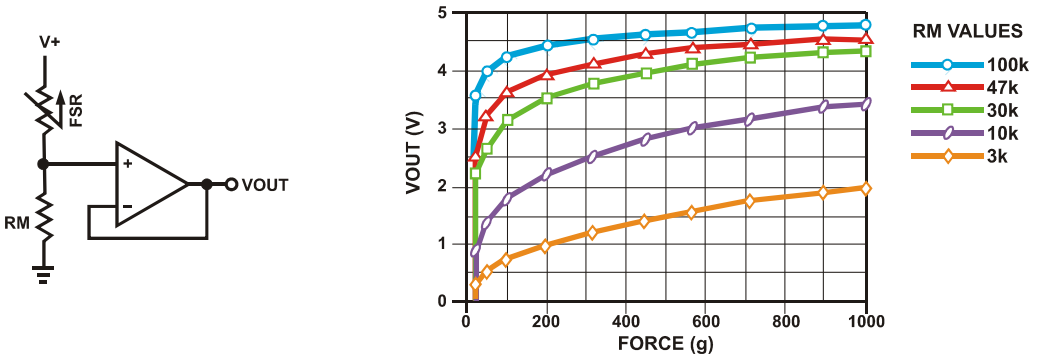


Figure 7.6: Electrical interface for force sensor

strate and overlay thickness and flexibility, size and shape of the actuator, and spacer-adhesive thickness (the gap between the facing conductive elements). Break force increases with increasing substrate and overlay rigidity, actuator size, and spacer adhesive thickness. Eliminating the adhesive, or keeping it well away from the area where the force is being applied, such as the center of a large FSR device, will give it a lower rest resistance. At the high force end of the dynamic range, the response deviates from the power-law behavior, and eventually saturates to a point where increases in force yield little or no decrease in resistance. The saturation point is more a function of pressure than force. The saturation pressure of a typical FSR is on the order of 100 to 200 psi. Forces higher than the saturation force can be measured by spreading the force over a greater area.

The FSR output is a resistance, but controller input must be a tension, so a little electrical interface must be built. The most simple way to have a tension as output with a variable resistance in input is to build up a voltage divider by using an operational amplifier. On the left side of figure 7.6 is shown the electrical circuit used for this work. The graph on the right side illustrates the trend of the output tension function of the force for different values of RM. The operational amplifier necessary to obtain a voltage divider is of the class *rail to rail*, with a range of output from 0 to 5 V.

The selection of the value of the resistance RM illustrated in figure 7.6 depends on the range of measure of interest: the lower is the maximum foreseen load, the higher must be the value of RM to have a good resolution. Looking to the typical loads expected on the leg, a value of 10 kΩ has been chosen. Using this resistance in the low force part the resolution is very high and measures until 10 N are possible. The first aspect is important in this work, because, as said before, although a force measure is available, the controller needs only to know if the foot is in contact with the

terrain or not, so that is necessary to establish a reference value in Newton over which the foot is considered on the ground. The good resolution at low load permits to choose a lower threshold value.

7.3.4 Communication and connection system

The changes made on the connections with respect to the old versions of NEMeSys are minor and concern the reduction of the length and of the weight of the cables and the optimization of wire path. For this reason, for low current connections, ribbon cables have been used. This is possible for the sensors and for the communication between the boards. For the power supply to the motors, instead, single wires have been used. About communication and processing of data, three devices are involved: a PC, and two different types of Programmable Interface Controller (PIC).

In figure 7.7 is shown a scheme of the communication system used for this work. For each leg the information about angles and contact force detected from the sensors are sent to analogical inputs of a PIC18F4431 microcontroller that converts the data in convenient units (degrees and Newtons), calculates the actual angular velocities using the pseudo-derivative described in section 7.5 and turns all the information to a PIC18F4550 (the same for all the legs) using the Inter Integrated Circuit (I^2C) communication protocol. This last device communicates these information using the USB standard to a PC, where the high-level controller described in previous sections calculates the next step angular velocities. These are the reference values that are sent back to the 4550 microcontroller and from here to the correct PIC4431. The software of this device computes the proper value of current for each motor necessary for the correct motion. The electronic boards that manage the data flow and the motors actuation have been developed in previous works over NEMeSys. Only minor changes have been made. For more details see [53] and [55]. The most important innovation is the construction of six new little boards (one for each leg) installed on the in the vicinity of the BC joint. All the wires coming from the

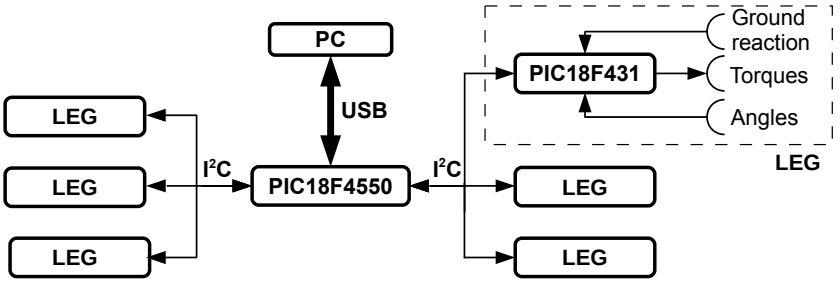


Figure 7.7: Electrical connection scheme.

legs are connected to these boards so that the disassembly of the legs from the body if faster and easier. Also the operational amplifier needed for the conversion of the signal coming from the contact sensors is positioned on these cards.

7.4 Weight evaluation

As before said the reaching of the lighter configuration is one of the most important target.

It must be taken in account that a huge part of the total weight is due to not structural parts, particularly propulsive system and electronics. These components have been taken with only minor changes from the old robot. The graph in figure 7.8 shows the contribution of each type of component to the total amount.

The weight of the structure is about the 23% of the total. The major contribution is from motors and adapters, that represent the 41% of the total weight. The boards give an important contribution, so that a miniaturization in the future is desirable.

Another significant partition of the weight is the one related with the location. The table 7.9 summarize the weights of the main subparts of the robot. The data are the sum of the single components, estimated by the use of CAD and then verified on real model, and include screws, nuts, wire, motors, adaptors and all is necessary for the transmission of the torque. The electronic boards, although fixed to the body, are per se in this list, because of their weight importance. These values are the ones used in the

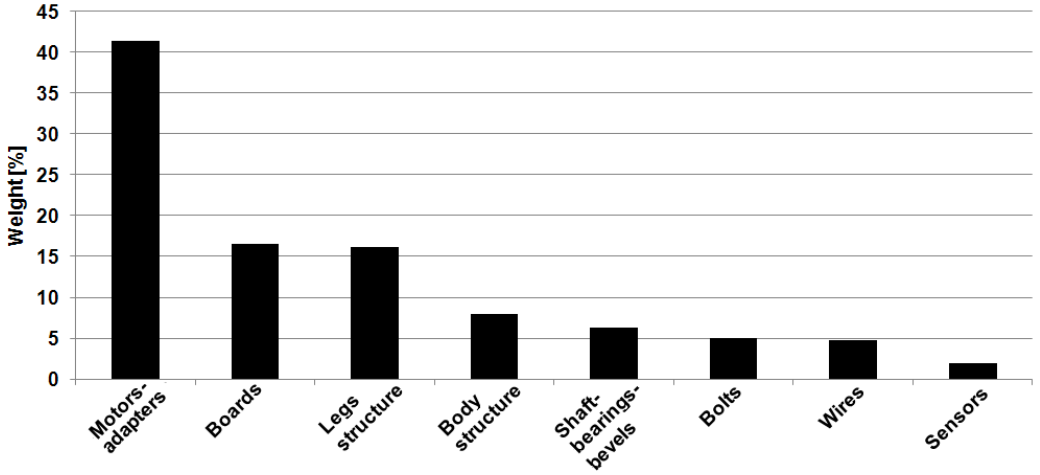


Figure 7.8: Partition of the weights between type of components.

Part	Number	CAD weight [g]	Real weight	Error [%]
Coxa	6	28	28	0
Femur	6	264	277	+4.9
Tibia	6	258	257	–0.4
Body	1	1574	1538	–2.3
Boards	4	1050	1030	–1.9
Wires	—	300	210	–30
TOTAL	—	6224	6150	–1.2

Table 7.9: Weight of the principal subparts of NEMeSys.

Model	Year	Weight [g]
Mk I	2005	≈7500
Mk II	2008	≈8300
Mk III	2010	6150

Table 7.10: Weight comparison between versions of NEMeSys.

multibody model presented in section 6.1.

The main difference found between CAD and real weight is in the femur. The cause of this is the high quantity of wires in this part (CT and FT motor supply, potentiometer supply and signal and contact sensor wires). The estimation in CAD supposed a uniform distribution of the wires. The reduction of the weight of the boards is due to the remaking of the supply and communication one, that in new version is a half of the old one for dimensions. The item wires includes only the ones between the boards.

At last, a comparison between the weights of the three versions of NEMeSys can be made. The table 7.10 shows the three values. The weight of the first version, NEMeSys Mk I, is the one reported in [53]; for the other two versions the values are the ones measured.

The first version adopted a way similar to the one of this work for the transmission of the motion, but without bearings, so that an abnormal backlash appeared after a short time. The body was made by an aluminum plate and the legs by standard profiles. In NEMeSys Mk II the body was re-made using a truss structure and motors become the joints themselves, trying to avoid the first version problems. The results were poor and the weight increased a lot. The third version (NEMeSys Mk III) is the current one. The weight reduction from Mk 2 to Mk 3 models is considerable, about the 25%.

7.5 Software

The software used for the control of NEMeSys con be divided into three main blocks:

- generation of the trajectory. The next-step angular velocity is generated by the PC. This is the controller described in the previous sections;
- follow the reference. Once that the reference value of angular velocity has been generated, it is necessary a low level controller that takes as input the error between current angular velocity and reference one and exits the correct value of current to give to the motor;
- management of the communication. This part comprises all the functions and the libraries necessary to have a correct I^2C and USB communication. These software are at most available from the *Microchip*, the PIC producer, and only minor changes are necessary.

The high level controller is written using C language because of the good trade-off between ease of programming, the presence of a lot of functions in the free libraries and the computational power needed that this language offers.

The low level controller, instead, has to be programmed using Assembler. This because has to work over the 4431 microcontrollers that have very low computational power, so that the software must be optimized as more as possible. Another advantage of the Assembler language is the possibility of setting the system register to adapt the configuration of the device according to the requests. All the register settings are described on the data sheets of the PIC devices [56][57]. The USB communication software is written using C; the I^2C one is programmed in Assembler. For more details see [55].

7.5.1 Low level controller

For the low level controller three are the main requirements:

- low computational cost;
- high adaptability to changes of the high level controller;
- ease of calibration of the parameters.

For these reasons a PID (Proportional-Integral-Derivative) controller has been chosen. This is a very common and used class of feedback controllers, particularly when some variables of the problem are unknown or

not completely modeled. Furthermore the calibration process of these controller can be made easily and only looking to the response of the system. A typical expression of a PID controller is shown in equation 7.1.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{7.1}$$

where K_p, K_i and K_d are proportional, integrative and derivative gains respectively, $e(t)$ the error between reference and measure at current time and $u(t)$ the control action.

Referring now to this work, $e(t)$ is the error between the angular velocity requested from the high level controller and the one calculated from the measure of the potentiometer. To obtain the angular velocity from the angle measure has been implemented a second order formulation for the derivative numerical calculation (see 7.2).

$$y'(i) \approx \frac{1}{2} y(i-2) - 2y(i-1) + \frac{3}{2} y(i) \tag{7.2}$$

Obviously a backward formulation has been used, as the calculation is real-time performed. This forces to save in the PIC memory the value of the angular position for the last two steps, but allows to have a better accuracy on the derivative calculation.

In order to estimate the integral of the error the trapezoidal rule (equation 7.3) has been used.

$$\int_a^b f(t)dt \approx (b-a)\frac{(f(a) + f(b)}{2} \tag{7.3}$$

In the present work a is the previous time step and b is the current one. $f(a)$ and $f(b)$ are the corresponding values of the angular velocities.

The values of the PID constants have been hand-tuned looking to the response of the multi-body model described in section 6.1. At least a proportional controller must be used. This type of controller, using an appropriate value for K_p, is able to stabilize the response. Increasing the value of K_p, beyond a certain value instability occurs. However, a proportional controller produces a high error on the angular position. To recover the appropriate accuracy on the position, an integral controller is necessary. The final choice for this controller is a PI. This system can be also seen as a proportional controller both on the angular position and on the angular velocity. The derivative control (equal to a proportional control of the error on angular acceleration), usually used to decrease overshoot, is not necessary in this case. The best result can be obtained with gains $K_p = 0.02$ and $K_i = 0.95$.

Chapter 8

Conclusion

In this section are summarized the main results reached during this work, both in the control system and in the construction of the walking platform. At the end will be listed some possible improvements or changes for the next work on NEMeSys.

8.1 Results of the work

The first result obtained is the control of the single leg motion. At first only the standard walking has been implemented, beginning from the two phases swing and stance. For the former one two possible model has been designed, one using a static network and one using a dynamic network. Both these two networks have been tested on a huge number of different start and target positions in order to evaluate advantages and disadvantages of each one. A positive feedback has been implemented in order to control the power stroke. A closed loop has been applied on height and body speed. Once that the single phases have been optimized, the design of the selector has permitted to reproduce a standard single leg motion. The next step has been the implementation of the reflexes by using ANN. These are the main expression of the interaction between the robot and the environment. Three types of reflexes have been identified looking to the behavior of stick insect. The single leg motion is so completely simulated.

The coordination between the legs has been the following target. Beginning from two different existing models of coordination, a new one has been designed, trying to keep the positive aspects of each one. The simulations of the motion using this new model have demonstrated to be able to ensure the static stability of the robot for a large range of speed, that can be varied with continuity from a value next to zero to the maximum permitted from the motors. The typical gaits of stick insect walking automatically emerge by changing the value of the reference velocity. On flat terrain wave gait from the rear to the front appears. It is demonstrated that this type of gait is the most efficient one. Furthermore also in case of failure of one leg, the coordination is ensured. Also curve trajectory can be performed only by giving a reference value of the angular speed of yaw, without any change in the control and coordination parameters. It is important to underline that these results have been obtained by using only kinematic models, without taking in account dynamic effects.

In order to proof the effectiveness of the control system designed, it has been tested on a multibody model. During these simulations a realistic model of the ground has been used in order to have the effective values of ground reactions. The output of these calculations have confirmed a good accordance with the kinematics ones demonstrating that the design can be made ignoring the dynamic effects. However these simulations have given the possibility to analyze particular behavior that can't emerge only considering the kinematics, especially during the stance phase, where the interaction with the environment is strong. The controller has also demonstrated a high level of robustness also against disturbance on the measures.

The use of a reflex-based controller has the main disadvantage of needing the signal of the sensors, so that in case of sensor failure the generation of a correct trajectory is impossible. In this case the only way is to switch off the corresponding leg and to continue with five legs, with a degradation of the performances, but maintaining the control and the coordination of the robot. This is also possible because of the high level of decentralization of the current controller, so that the trajectory of each leg can be generated using the data from the sensors of the leg itself.

Meanwhile a mechanical platform has been designed and built. The main result is a great reduction of the weight (about 20%) and of the size of the legs. The new type of junction allows to have more compact legs and body. Moreover the type of structure chosen for NEMeSys Mk III has a high level of modularity, allowing to change the shape rapidly and cheaply maintaining the current milled parts and replacing the profiles. The new contact sensors permit the measure of the ground reaction instead of the only contact detection of the last works.

8.2 Innovation

The main innovative feature developed during this work is the new coordination model shown in chapter 5. Compared to the other proposed decentralized coordination models, the present one matches the ability to ensure the static stability of the robot and to generate wave gates on flat ground. The static stability is maintained not only in case of standard walking on flat terrain, but also un presence of obstacles and when particular behaviors, e.g. reflexes, are activated. The emersion of wave gate, instead, ensures that the walking is performed by the most efficient way.

The second innovation is about the swing trajectory generation: in this work a dynamic ANN has been designed, trained and tested. The main advantage with respect to the static swing network is its robustness with

respect to the static one in front of disturbance.

In order to represent particular behaviors a complete set of reflexes have been implemented using ANN. These are an innovation with respect to the previous work because are completely uncoupled with respect to the normal walking trajectory generation. They only work if a particular input comes from the environment, so that, normally, they have no influence on the robot motion.

Moreover the modular approach improves the flexibility of the controller by allowing to introduce new behaviors in very simple way.

About the mechanical platform, the main innovation with respect to the old versions of NEMeSys is the joint architecture and the use of miter gears for the transmission of the motion.

8.3 Future developments

About the future development, the first thing to be done is the test of the controller presented in this work on the walking platform. The software necessary for this activity is at most already programmed by using C and Assembler. However it is necessary a check of the USB and I^2C communication and the calibration of the sensors. The first tests will be done with straight reference trajectory beginning from low speed gait. If these tests will conclude successfully high speed and curve walking could be tested.

With regard to the controller, starting from the actual one, a lot of development can be thought. The first one is the implementation of a system that allows to maintain the position when the robot is stopped in presence of disturbance. The LPVF acting during stance does not allow this behavior. It is necessary to introduce a stiffness on the joints somehow. The most simple way is the introduction of a negative feedback on the single joint every time that the current position has to be held.

A weakness of the controller described in this work is that no raise is possible if the robot falls. This because particular behavior are necessary in this case. The solution could be to implement a system that first of all detects the fall and computes a strategy for the lift. These strategy are stereotyped behaviors function of the position after the drop.

About the curve walking other behaviors can be applied: in this work has been implemented only the change of frequency of the gait to perform a curve trajectory, but other ones are possible. The simplest is the changing of the target position of the swing phase and of the PEP in relation to the yaw angular velocity reference. By this way it will be possible to reduce the radius of the trajectory, up to permit to the robot to turn on itself.

The presence of force sensors that measure the ground reaction on the

foot allows to design a more complex selector that takes into account the longitudinal and lateral position of the PEP and the amount of the reaction. The result of this will be a better control of the stability, that eventually will change the coordination model, adding to the current one the equivalent of the rule 5 of the Cruse model (see section 5.1).

For the high level control, attitude sensors can be installed on the robot. By this way will be possible to command the orientation of the body. This means that a better control of the reference height for each leg can be implemented, allowing, for example, to lift the forward part of the body if an obstacle is detected. For a better interaction with the environment a set of vision sensors could be used. By this way the detection of an irregularity of the ground can be made in advance, and an optimization of the trajectory and of the gait can be performed.

As previously said, with the current controller, in case of failure of failure even of one sensor, it is necessary to switch off all the legs, that has to be locked in swing position to avoid interference with the working legs. A strategy can be studied to try to use, perhaps with a degradation of the performance, the damaged leg too. An idea can be to insert a shadow CPG that, only in case of failure, using as input the signal of the remaining sensors of the leg, replaces the normal reflex-based controller. By this way the pattern generator is restored. Obviously this shadow controller must be synchronized through the coordination to the reflex-based ones working on the healthy legs.

A parallel activity on the robot could be the identification of the model from the data coming from the tests. By this work it could be possible to build a simple and efficient model for numerical simulation of any type of controller.

About the mechanical platform, the first thing to do could be the miniaturization of the electronic boards. This would mean also a great reduction of the weight (supposed about 10%), so that a payload or a on-board CPU could be installed.

List of Acronyms

AEP	Anterior Extreme Position
ANN	Artificial Neural Network
BC	Body-Coxa
CAD	Computer-Aided Design
CoG	Center of Gravity
CPG	Central Pattern Generator
CT	Coxa-Trochanter
CTRNN	Continous Time Recurrent Neural Network
DNN	Dynamic Neural Network
DoF	Degree of Freedom
FEM	Finite Element Method
FSR	Force Sensing Resistor
FT	Femur-Tibia
GA	Genetic Algorithm
GC	Ground Contact
HPF	High Pass Filter
I^2C	Inter Integrated Circuit
LPVF	Local Positive Velocity Feedback
NEMeSys	Neural Ento-Mechanic System
PCG	Point of Center of Gravity
PEP	Posterior Extreme Position
PIC	Programmable Interface Controller
PID	Proportional-Integral-Derivative

PSD Power Spectral Density

PWM Pulse Width Modulation

RNN Recurrent Neural Network

SEP Superior Extreme Position

TDRNN Time-Delayed Recurrent Neural Network

USB Universal Serial Bus

Bibliography

[1] H. Cruse, V. Dürr, and J. Schmitz. Insect walking is based on a decentralized architecture revealing a simple and robust controller. *Philosophical Transation of the Royal Society A*, 365:221–250, 2007.

[2] R. A. Brooks. A robot that walks; emergent nehavior from a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.

[3] C. Ferrel. Robust and adaptive locomotion of an autonomous hexapod. In *Proceedings From Perception to Action Conference*, pages 66–77, Lausanne, Switzerland, Sept. 1994.

[4] R.D. Beer, H.J. Chiel, R.D. Quinn, K. Espenschied, and P. Larsson. A distributed neural network architecture for hexapod robot locomotion. *Neural Computation*, 4(3):356–365, 1992.

[5] S. Colombano, F. Kirchner, D. Spenneberg, and J. Hanratty. Exploration of planetary terrains with a legged robot as a scout adjunct to a rover. In *AIAA Space 2004 Conference and Exhibit*, pages 28–30, San Diego (CA), USA, Sept. 2004.

[6] H. Cruse, T. Kindermann, M. Schumm, J. Dean, and J. T. Schmitz. Walknet - a biologically inspired network to control six-legged walking. *Neural Networks*, 11:1435–1447, 1998.

[7] H.J. Weidemann, F. Pjeifer, and J. Eltze. The six-legged tum walking robot. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS '94.*, volume 2, pages 1026–1033, Munich, Germany, Sept. 1994.

[8] A. Buschmann. Home of Tarry I & II: design of the walking machine Tarry II, March 2000. `http://www.tarry.de`.

[9] A. Buschmann. Home of Tarry I & II: frequently asked questions about Tarry, March 2000. `http://www.tarry.de`.

[10] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 7:115–133, 1943.

[11] S. Nolfi and D. Floreano. Learning and evolution. *Autonomous Robots*, 1:89–113, 1999.

[12] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[13] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[14] Alessandro Bursi and Marco Di Perna. Controllo di un robot a zampe per esplorazione planetaria con l'utilizzo di reti neurali dinamiche e algoritmi evolutivi. Master Thesis in Space Engeneerig, Politecnico di Milano, 2004.

[15] Gabriele Greco. Generazione e controllo della camminata di un robot esapode con reti neurali CTRNN e a mutua inibizione. Master Thesis in Space Engeneerig, Politecnico di Milano, 2009.

[16] R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[17] Bernard Friedland. *Control sytem design*. McGraw-Hill, New York, NY, 1986.

[18] The MathWorks, Inc., Natick, MA. *Genetic Algorithm and Direct Search Toolbox User's Guide*, January 2004. `http://www.mathworks.com`.

[19] G. Wendler. Laufen und stehen der stabheuschrecke: Sinnesborsten in den beingelenken als glieder von regelkreisen. *Zeitschrift fuer Vergleichende Physiologie*, 48:198–250, 1964.

[20] N. Porcino. Hexapod gait control by a neural network. In *International Joint Conference on Neural Networks*, volume I, pages 189–194, San Diego (CA), USA, June 1990.

[21] H. Cruse and C. Bartling. Movement of joint angles in the legs of a walking insect, *Carausius morosus*. *Journal of Insect Physiology*, 9(41):761–771, 1995.

[22] M. Schumm and H. Cruse. Control of swing movement: influences of differently shaped substrate. *Journal of Comparative Physiology*, 192:1147–1164, 2006.

[23] H. Cruse, C. Bartling, D.E. Brunn, J. Dean, M. Dreifert, T. Kindermann, and J. Schmitz. Walking: a complex behavior controlled by simple systems. *Adaptive Behavior*, 3:385–418, 1995.

[24] H. Cruse, C. Bartling, and T. Kindermann. High-pass filtered positive feedback for decentralized control of cooperation. In *Advances in artificial life*, volume 929 of *Lecture Notes in Computer Science*, pages 668–678. Springer Berlin / Heidelberg, 1995.

[25] D. DeAngelis, W. Post, and C. Travis. *Feedback in Natural Systems*. Springer, London, Berlin, Heidelberg, 1986.

[26] H. Cruse. Which parameters control the leg movement of a walking insect? ii. the start of the swing phase. *The Journal of Experimental Biology*, 116:357–362, 1985.

[27] D.M. Wilson. Insect walking. *Annual Reviews Entomology*, 11:103–122, 1966.

[28] S.M. Song and K.J. Waldron. An analytical approach for gait study and its application on wave gaits. *The International Journal of Robotics Research*, 6(2):60–71, 1987.

[29] V. Zolotov, L. Frantsevich, and E. M. Falk. Kinematik der phototaktischen drehung bei der honigbiene *Apis mellifera*. *Journal of Comparative Physiology*, 97:339–353, 1975.

[30] C.P.E. Zollikofer. Stepping patterns in ants. i. influence of speed an curvature. *The Journal of Experimental Biology*, 192:95–106, 1994.

[31] K.W. Wait and M. Goldfarb. A biologically inspired approach to the coordination of hexapedal gait. In *Proceedings of ICRA 2007*, pages 2655–2660, Rome, Italy, April 2007.

[32] H. Cruse, C. Bartling, G. Cymbalyuk, and M. Dreifert. A modular artificial neural net for controlling a six-legged walking system. *Biological Cybernetics*, 72:421–430, 1995.

[33] C.R. Linder. Self organisation in a simple task of motor control. In *Proceedings of the 7th international conference on simulation of adaptive behavior*, pages 185–194, Edinburgh, UK, Aug. 2002. MIT Press.

[34] H. Cruse, D. Riemenschneider, and W. Stammer. Control of body position of a stick insect standing on uneven surfaces. *Biological Cybernetics*, 61:71–77, 1989.

[35] J. Schmitz, T. Kindermann, M. Schumm, and H. Cruse. Simplifying the control of a walking hexapod by exploiting physical properties.

In *Proceedings of European Mechatronics Colloquium, Euromech 375 - Biology and Technology of Walking*, pages 296–303, Munich, Germany, March 1998.

[36] H. Cruse. Which parameters control the leg movement of a walking insect? i. velocity control during the stance phase. *The Journal of Experimental Biology*, 116:343–355, 1985.

[37] T. Kindermann. Behavior and adaptability of a six-legged walking system with highly distributed control. *Adaptive Behavior*, 9:16–41, 2002.

[38] K.S. Espenschied, R.D. Quinn, R.D. Beer, and H.J. Chiel. Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and Autonomous Systems*, 18:59–64, 1996.

[39] H. Cruse and B. Blaesing. Stick insect locomotion in a complex environment: climbing over large gaps. *The Journal of Experimental Biology*, 207:1273–1286, 2004.

[40] V. Dürr. Stereotypic leg searching-movements in the stick insect: kinematic analysis, behavioural context and simulation. *The Journal of Experimental Biology*, 204:1589–1604, 2001.

[41] K.G. Pearson and R. Franklin. Characteristics of leg movements and patterns of coordination in locust walking on rough terrain. *International Journal of Robotics Research*, 3(2):101–112, 1984.

[42] H. Cruse. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, 13:15–21, 1990.

[43] J.M. Porta and E. Celaya. A control structure for the locomotion of a legged robot on difficult terrain. *IEEE Robot & Automation Magazine*, 5(2):43–51, 1998.

[44] T. Roggendorf. Comparing different controllers for the coordination of a six-legged walker. *Biological Cybernetics*, 92:261–274, 2005.

[45] J.M. Porta and E. Celaya. Reactive free-gait generation to follow arbitrary trajectories with a hexapod robot. *Robotics and Autonomous Systems*, 47:187–201, 2004.

[46] K.S. Espenschied, R.D. Quinn, H.J. Chiel, and R.D. Beer. Leg coordination mechanisms in stick insect applied to hexapod robot locomotion. *Adaptive Behavior*, 1(4):455–468, 1993.

[47] H. Cruse. The control of the anterior extreme position of the hindleg of a walking insect, *Carausius morosus*. *Physiological Entomology*, 4:121–124, 1979.

[48] J. Dean. Coding proprioceptive information to control movement to a target: simulation with a simple neural network. *Biological Cybernetics*, 63:115–120, 1990.

[49] C. Ridderström. *Legged locomotion: balance, control and tools - from equation to action*. PhD Thesis, Royal Institute of Technology, Stockholm, Sweden, 2003.

[50] The MathWorks, Inc., Natick, MA. *Simscape™ 3 User's Guide*, March 2010. `http://www.mathworks.com`.

[51] Ernest O. Doebelin. *Measurement Systems: Application and Design*. McGraw-Hill, New York, NY, 5th edition, 2004.

[52] H.M. Lankarani and Nikravesh P.E. A contact force model with hysteresis damping for impact analysis of multibody systems. *Transaction of the ASME, Journal of Mechanical Design*, 112:369–376, 1990.

[53] Paolo Massioni and Stefano Nebuloni. Progetto e sperimentazione di sistemi di controllo per un prototipo di robot esapode per esplorazione planetaria. Master Thesis in Space Engeneerig, Politecnico di Milano, 2005.

[54] Interlink Electronics, Inc., Camarillo, CA. *FSR® Integration Guide & Evaluation Parts Catalog With Suggested Electrical Interfaces*. `www.interlinklelectronics.com`.

[55] Antonio Pisciotta. Progetto del sistema motorio di un robot a più arti. Master Thesis in Aeronautical Engeneerig, Politecnico di Milano, 2008.

[56] Microchip Technology Inc., Chandler, AZ. *PIC18F2331/2431/4331/4431 Data Sheet*. `http://ww1.microchip.com`.

[57] Microchip Technology Inc., Chandler, AZ. *PIC18F2455/2550 /4455/4550 Data Sheet*. `http://ww1.microchip.com`.