Chapter 5

Robot's walking

5.1 Introduction

The biped walking process or biped locomotion area has been studied for a long time, but it is only in the past years, thanks to the fast development of computers, that real robots started to walk on two legs. Since then the problem has been tackled from different directions.

First, there were robots that used static walking. The control criteria was to maintain the projection of the *center of gravity (COG)* on the ground, inside of the foot support area.

This approach was abandoned because only slow walking speeds could be achieved, and only on flat surfaces.

For the dynamic walking robots the *center of gravity* (or center of mass) can be outside of the *support area*, but the zero momentum point (ZMP), cannot [33]. The ZMP criteria has been broadly used to generate biped control algorithms [34], [35].

In this thesis, the walking problem is divided and focus on two themes: the *balance control* and the *walking sequence control.*

In *balance control*, a feedback-force system at each robot's foot was implemented to calculate the ZMP and then feed it in to the *incremental fuzzy PD controller* to decrease the ZMP error [14]. The controller's goal is to adjust the lateral robot's positions to maintain the ZMP point always inside of the *support region.*

The *walking sequence control* of a biped robot can be determined by con-

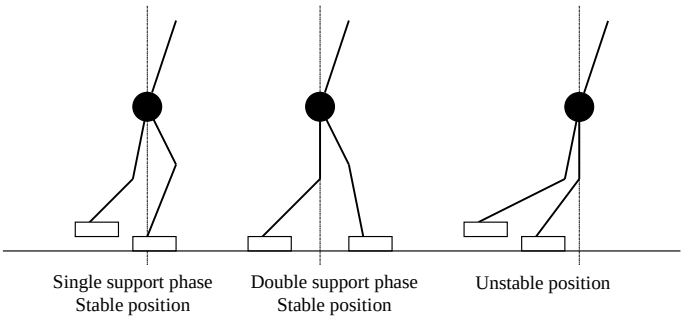

Figure 5.1: Static walking.

trolling the hip and foot trajectories [15]. To achieve stable dynamic walking, the change between *simple supports walking phase* and *double supports walking phase* should be smooth. In this thesis *cubic polynomials algorithms* are used to control the sagittal motion and guaranties a smooth change between the walking phases [6].

The robot's stability at dynamic walking was achieved applying the ZMP criteria in the *incremental fuzzy PD controller* to guaranties the balance control at walking [14] and the *cubic polynomials algorithms* to control the *walking sequence control.*

In the next sections, the *static* and *dynamic walking* are presented, the ZMP concept is widely explained, then the *balance control* and the *walking sequence control* are explained.

5.2 Static walk

Static walking assumes that the robot is statically stable. This mean that, at any time, if all motion is stooped the robot will stay indefinitely in a stable position. It is necessary that the projection of the center of gravity of the robot on the ground must be contained within the foot support area (Figure 5.1). The support area is either the foot surface in case of one supporting leg or the minimum convex area containing both foot surfaces in case both feet are on the ground.

These are referred to as single and double support phases, respectively . Also, walking speed must be low so that inertial forces are negligible [33][36].

This king of walking requires large feet, strong ankle joints and can achieve only slow walking speeds. It has been abandoned by most researchers for dy-

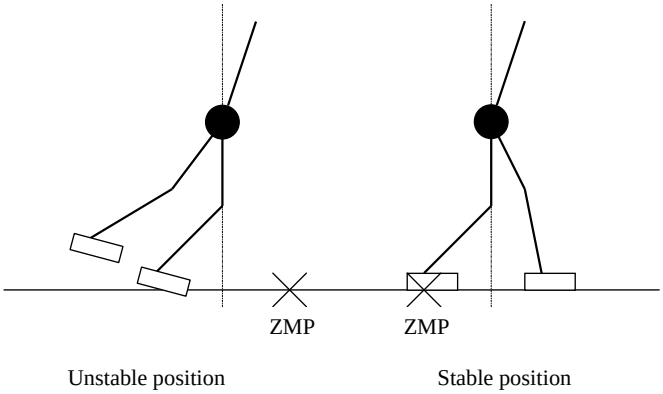

Figure 5.2: Dynamic walking.

namic walking, which provides more realistic and agile movements.

5.3 Dynamic walk

Biped dynamic walking allows the center of gravity to be outside of the support region for limited amounts of time. There is no absolute criteria that determines whether the dynamic walking is stable or not. Indeed a walker can be designed to recover from different kinds of instabilities. However, if the robot has active ankle joints and always keeps at least one foot flat on the ground then the ZMP can be used as a stability criteria. The ZMP is the point where the robot's total moment at the ground is zero. As long the ZMP is inside the support region the walking is considered dynamically stable because is the only case where the foot can control the robot's posture. It is clear that for robots that do not continuously keep at least one foot on the ground or that do not have active ankle joints (walking on stilts), the notion of support area does not exist, therefore the ZMP criterion cannot applied [37].

Dynamic walking is achieved by ensuring that the robot is always rotating around a point in the support region (Figure 5.2). If the robot rotates around a point outside the support region then this means that the supporting foot will tend to get off the ground or get presses against the ground. Both cases lead to instability. To draw an analogy with static walking, if all motion is stopped then the robot will tend to rotate around the ZMP [33].

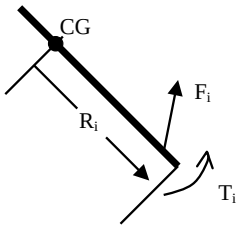

Figure 5.3: Force applied to a link.

5.4 Zero Moment Point (ZMP)

To achieve dynamic walking the center of gravity (or center of mass) can be outside of the support area of the robot, but the zero momentum point (ZMP), cannot.

The ZMP is the point where the total angular momentum is zero. The position of the ZMP is computed by finding the point $(X,\ Y,\ Z)$ where the total torque is zero. Since we are only interested in the ground plane we assume that $Z{=}0$. To avoid confusion, torque and moment mean in this thesis the same thing. The robot has n links; each link is subject to a total force f_i applied at a point determined by the vector r_i relative to the center of gravity of the link. T_i determines the total motor torque applied to the link. R_z is the ZMP vector and T is the robot's total torque [33]. An example of the forces applied to a link is represented in Figure 5.3.

The force, torque and position vectors have the following coordinates:

$$F_i : (F_{xi} + F_{yi} + F_{zi})$$

$$T_i : (T_{xi} + T_{yi} + T_{zi})$$

$$R_i : (x_i + y_i + z_i)$$

Then the total torque is computed as:

$$T = \sum_{i=1}^{n} (R_i + R_z) x F_i + \sum_{i=1}^{n} T_i = 0 \ [4.1]$$

Where x represents the cross product. [4.1] is then expanded as:

$$\sum_{i=1}^{n}(y_i+Y)F_{zi}-\sum_{i=1}^{n}(z_i+Z)F_{yi}+\sum_{i=1}^{n}T_{xi}=0$$

$$\sum_{i=1}^{n}(z_i+Z)F_{xi}-\sum_{i=1}^{n}(x_i+X)F_{zi}+\sum_{i=1}^{n}T_{yi}=0 \quad [4.2]$$

$$\sum_{i=1}^{n}(x_i+X)F_{yi}-\sum_{i=1}^{n}(y_i+Y)F_{xi}+\sum_{i=1}^{n}T_{zi}=0$$

Making *z=0* and solving these equations for x and y we obtain the ZMP coordinates:

$$X=\frac{\sum_{i=1}^{n}(z_iF_{xi}-x_iF_{zi})+\sum_{i=1}^{n}T_{yi}}{\sum_{i=1}^{n}F_{zi}} \quad [4.3]$$

$$Y=\frac{\sum_{i=1}^{n}(z_iF_{yi}-y_iF_{zi})+\sum_{i=1}^{n}T_{xi}}{\sum_{i=1}^{n}F_{zi}} \quad [4.4]$$

Considering the gravitational acceleration, mass and the inertia moment components, the ZMP can be also computed as follows:

$$x_{ZMP}=\frac{\sum_i m_i(\ddot{z}+g)x_i-\sum_i m_i\,\ddot{x}\,z_i-\sum_i I_{iy}\,\ddot{\theta}_{iy}}{\sum_i m_i(\ddot{z}+g)} \quad [4.5]$$

$$y_{ZMP}=\frac{\sum_i m_i(\ddot{z}+g)y_i-\sum_i m_i\,\ddot{x}\,z_i-\sum_i I_{ix}\,\ddot{\theta}_{ix}}{\sum_i m_i(\ddot{z}+g)} \quad [4.6]$$

Where x_{ZMP} and y_{ZMP} , are the ZMP coordinates, x_i, y_i, z_i are the coordinates of the mass center of the i link, m is the mass of the link *i,* g is the gravitational acceleration. I_{ix} and I_{iy} are the inertia moment components, θ_{iy} and θ_{ix} are the angular velocity around the x and y axes (take as a point from the mass center of the i link) .

In sagittal plane, the ZMP is directly calculated dividing the ankle torque by the force reaction at the ground:

$$x_{ZMP}=\frac{\tau_x}{\sum_i m_i(\ddot{z}_i+g)} \quad [4.7]$$

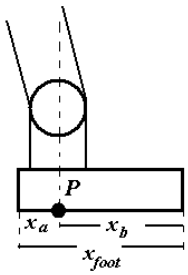

Figure 5.4: Stable area of the ZMP in the x axis.

5.4.1 ZMP as a control criteria

As mentioned, in dynamic walking, the center of gravity (or center of mass) can be outside of the support area, but the ZMP (the point where the total angular momentum is zero) cannot. The ZMP will be take as a control criteria (from now on this criteria will be mentioned as the *ZMP criteria*)

To maintain the balance in dynamic walking the ZMP point must be in the foot convex area, in contact with the floor as shown in figure 5.4.

The ZMP can be defined as [38],[14]:

$$x_{ZMP} \in \{S | S \in R, S \in (-x_a, x_b)\} \quad [4.8]$$

The negative sign of x_a means that point P is at the origin.

Since, the force sensors values are direct used to calculate the ZMP and for the lateral control is only necessary to know the ZMP value for one axis, the ZMP calculus is simplified as:

$$P_{ZMP} = \frac{\sum_{i=1}^{3} f_i r_i}{\sum_{i=1}^{3} f_i} \quad [4.9]$$

where f_i represents the force at the i sensor and r_i represent the distance between the coordinate origin and the point where the sensor is located. The figure 5.5, shows the sensors distribution (the tree circles) used for each robot's foot.

The total ZMP calculus will be obtained by the difference between each foot:

$$Total\_P_{ZMP} = P_{ZMP1} - P_{ZMP2} \quad [4.10]$$

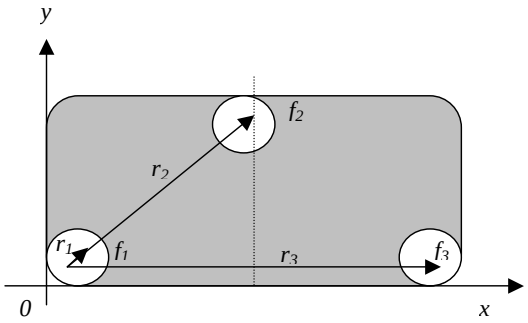

Figure 5.5: Sensor's distribution.

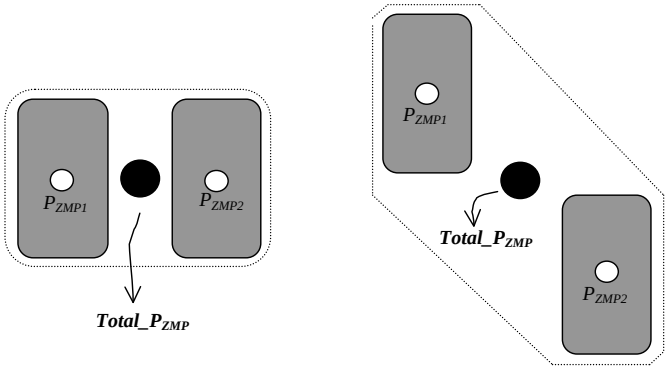

Figure 5.6: ZMP point (black point) in two cases, left) when the robot is stand, right) after the robot give a step.

Where P_{ZMP1} is the ZMP for one foot and P_{ZMP2} is the ZMP for the other foot. To maintain the robot's stability during walking, the Total ZMP must be always inside of the *support polygon*. The *support polygon*, is the polygon described by the robot's foots as shown in Figure 4.6 (pointed line).

Figure 5.6 shows the ZMP point (black point) in two cases, one with the robot standing (left), and other after give a step (right). The pointed line represent the *support polygon*.

5.5 Balance control

In traditional legged robots, stability is maintained by having at least three contact points with the ground surface at all time. With biped machines, only two points are in contact with the ground surface for that reason algorithms to

achieve balance most be implemented.

There are some techniques to implement a balance control for a biped robot, many of them are implemented using classic control techniques, but some others are implemented using soft computing or artificial intelligent techniques. In this thesis an *incremental fuzzy PD controller* to achieve balance in a biped robot is proposed.

An important control criteria to achieve dynamic walking (as already was mentioned) is to maintain the ZMP inside of support region (ZMP criteria). The use of this criteria has been broadly used to generate biped control algorithms [34], [35].

In order to implement the balance control in the "Dany walker", a feedback-force system at each foot was implemented to obtain the ZMP and feed it in to the *incremental fuzzy PD controller*. Then, it calculate the ZMP error and the ZMP rate. The controller goal is to adjust the lateral robot's positions to maintain always the ZMP inside of the support region.

The next section introduce the basis of the control system theory to a better understanding of the *fuzzy PD incremental controller* proposed in this thesis.

5.5.1 Control system theory

A control system is an arrangement of physical components designed to regulate, or to command, through a control action, another physical system so that it exhibits certain desired characteristics or behavior [39]. Control systems are typically of two types: open-loop control systems, in which the control action is *independent* of the physical system output, and closed-loop control systems (also known as *feed-back* control systems), in which the control action *depends* on the physical system output. Examples of open-loop control systems are a toaster, in which the amount of heat is set by a human, and an automatic washing machine, in which the controls for water temperature, spin-cycle time, and so on are preset by the human. In both these cases the control actions are not a function of the output of the toaster or the washing machine. Examples of feedback control are a room temperature thermo-stat with senses room temperature and activates a heating or cooling unit when a certain threshold temperature is reached, and an autopilot mechanism, which makes automatic course corrections to an airplane when heading or altitude deviations from certain preset values are sensed by the instruments in the plans cockpit.

In order to control any physical variable first, it must be measure. The sys-

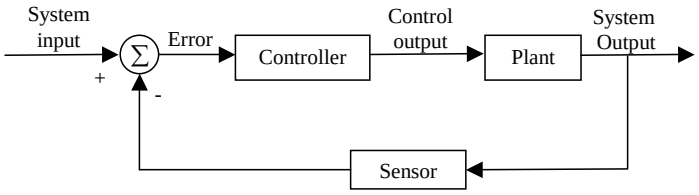

Figure 5.7: A closed-loop control system.

tem for measurement of the controlled signal is called a *sensor*. The physical system under control is called a *plant*. In a closed-loop control system, certain forcing signals of the system (called inputs) are determined by the responses of the system (called outputs). To obtain satisfactory responses and characteristics for the closed-loop control system, it is necessary to connect an additional system, known as a compensator, or a controller, into the loop. The general form of a closed-loop control system is illustrated in Figure 5.7 [40].

Control systems are sometimes divided into two classes. If the goal of the control system is to maintain a physical variable at some constant value in the presence of disturbances, the system is called a *regulatory type of control*, or a regulator.

The room temperature control and autopilot are examples of regulatory controllers. The second class of control systems are *tracking controllers*. In this scheme of control, a physical variable is required to follow or track some desired time function. An example of this type of system is an automatic aircraft landing system, in which the aircraft follows a ramp to the desired touchdown point.

The control problem is stated as follows [40]. The output, or response, of the physical system under control (i.e. the plant) is adjusted as required by the error signal. The error signal is the difference between the actual response of the plant, as measured by the sensor system, and the desired response, as specified by a reference input. In the following section we derive different forms of common mathematical models describing a closed-loop control system.

A mathematical model that describes a wide variety of physical systems is a *nth* order ordinary differential equation of the type [41].

$$\frac{d^n y^{(t)}}{dt^n} = w\left[t, y(t), \dot{y}(t), ..., \frac{d^{n-1} y^{(t)}}{dt^{n-1}}, u(t)\right] \quad [4.11]$$

Where t is the time parameter, $u(\ )$ is the input function, $w(\ )$ is a general non linear function, and $y(\ )$ is the system output or response function. If we

define the auxiliary functions.

$$\begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \dot{y}(t) \\ &\vdots \\ x_n(t) &= \frac{d^{n-1}y(t)}{dt^{n-1}} \end{aligned} \quad [4.12]$$

Then the single *nth*-order equation [4.11] can be equivalently expressed as a system of n first-order equations:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= x_3(t) \\ &\vdots \\ \dot{x}_{n-1}(t) &= x_n(t) \\ \dot{x}_n(t) &= w\left[t, x, (t), x_2(t), ..., x_n(t)u(t)\right] \end{aligned} \quad [4.13]$$

Finally, if we define n-vector- valued functions $x(\ )$ and $f(\ )$ by

$$x(t) = [x_1(t), x_2(t), ..., x_n(t)]^T \quad [4.14]$$

$$f(t, x, u) = [x_2, x_3, ..., x_n, w(t, x_1, ..., x_n, u)]^T [4.15]$$

Where $x(t)$ is the system state vector at time t (T is standard vector transpose), then the n first-order equations [4.13] can be combined into a first-order vector differential equation:

$$\dot{x}(t) = f\left[t, x(t), u(t)\right] \quad [4.16]$$

and the output $y(t)$ is given from Eq [4.12] as

$$y(t) = [1, 0, ..., 0]\, x(t) \quad [4.17]$$

Similarly, a system whit p inputs, m outputs, and n states will be described, in general, using the vector-valued functions $f(\ )$ and $g(\ )$ as:

$$\dot{x}(t) = f\left[t, x(t), u(t)\right] \quad [4.18]$$

$$y(t) = g\left[t, x(t), u(t)\right] \quad [4.19]$$

where $u(t)$ and $y(t)$ vectors are defined as:

$$u(t) = [u_1(t), u_2(t), ..., u_p(t)]^T \quad [4.20]$$

$$y(t) = [y_1(t), y_2(t), ..., y_m(t)]^T \quad [4.21]$$

And are an input vector and an output vector, respectively. The state variables vector *x(t)* is defined by Eq. [4.14]. Physical systems descriptions based on Eqs.[4.18] and [4.19] are known as state-space representations and $x_1, x_2, ..., x_n$ are known as state variables of the system. In the case of nonlinear time-invariant continuous-time systems, Eqs. [4.18] and [4.19] become:

$$\dot{x}(t) = f\,[x(t), u(t)] \quad [4.22]$$

$$y(t) = g\,[x(t), u(t)] \quad [4.23]$$

and for a linear time-invariant system (LTI) will reduce to the following form:

$$\begin{aligned} \dot{x}(t) &= A \cdot x(t) + B \cdot u(t) \\ y(t) &= C \cdot x(t) + D \cdot u(t) \end{aligned} \quad [4.24]$$

where constants *A, B, C,* and *D* are known as *system matrices.*

A first- order single-input-single-output nonlinear system is described using a discrete-time equation as:

$$x_{k+1} = f(x_k, u_k) \quad [4.25]$$

where x_{k+1}, x_k are the values of state at the *kth* and *(k + 1) th* time moments, and u_k is the input at the *kth* moment. a *nth*-order single- input-single-output system can be put in the following form:

$$y_{k+n} = f(y_k, y_{k+1}, ..., y_{k+n-1}, u_k) \quad [4.26]$$

and for a *nth*-order, multiple-input-single-output discrete system:

$$y_{k+n} = f(y_k, y_{k+1}, ..., y_{k+n-1}, u_{1(k)}, u_{2(k)}, ..., u_{p(k)}) \quad [4.27]$$

System Identification Problem

The general problem of identifying a physical system based on the measurements of the input, output, and state variables is defined as obtaining functions *f* and *g* in the case of a nonlinear system, and system matrices *A, B, C,* and *D* in the case of a linear system. There are algorithms that adaptively converge to these system parameters based on numerical data taken from input and output variables [42]. Fuzzy systems and artificial neutral network paradigms are two evolving disciplines for non linear system identification problems.

Control System Design Problem

The general problem of feedback control system design is defined as obtaining a generally non-linear vector-valued function *h( )*, defined as follows [43]:

$$u(t) = h\left[t, x(t), r(t)\right] \text{ [4.28]}$$

where *u(t)* is the input to the plant or process, *r(t)* is the reference input, and *x(t)* is the state vector. The feedback control law h is supposed to stabilize the feedback control system and result in a satisfactory performance.

In the case of a time-invariant system with a regulatory type of controllers are based on one of the general models given in Esq. [4.29] and [4.30], that is, either full state feedback or output feedback, as shown in the following:

$$u(t) = h\left[x(t)\right] \text{ [4.29]}$$

$$u(t) = h\left[y(t), \dot{y}, \int y dt\right] \text{ [4.30]}$$

In the case of a simple single-input-single-output system and a regulatory type of controller, the function h takes one of the following forms:

$$u(t) = K_P \cdot e(t) \text{ [4.31]}$$

for a proportional, or *P*, controller:

$$u(t) = K_P \cdot e(t) + K_I \cdot \int e(t)dt \text{ [4.32]}$$

for a proportional-plus-integral ,or *PI*, controller:

$$u(t) = K_P \cdot e(t) + K_D \cdot \dot{e}(t) \text{ [4.33]}$$

for a proportional-plus-derivative, or *PD*, controller:

$$u(t) = K_P \cdot e(t) + K_I \cdot \int e(t)dt + K_D \cdot \dot{e}(t) \text{ [4.34]}$$

for a proportional-plus-derivative-plus-integral, or *PID*, controller, where there are the output error, error derivative, and error integral, respectively, and for a full state-feedback controller:

$$u(t) = -\left[k_1 \cdot x_1(t) + k_2 \cdot x_2(t) + ... + k_n \cdot x_n(t)\right] \text{ [4.35]}$$

The problem of control system design is defined as obtaining the generally nonlinear function *h( )* in the case of nonlinear systems; coefficients K_P, K_I and K_D in the case of output-feedback systems; and coefficients $k_1, k_2, ..., k_n$ in the case of a full state-feedback control policy for linear systems. The function *h( )* in Eqs. [4.29] and [4.30] describes a general nonlinear surface that is known as a control, or decision, surface, discussed in the next section.

Control (Decision) Surface

In this section, the concept of a control surface, or decision surface, is defined. It is very important in fuzzy control systems [43]. The function h as defined in Eqs. [4.28], [4.29] and [4.30] is, in general, defining P non-linear hyper surfaces in a n-dimensional space. For the case of linear systems with output feedback or state feedback it generally is a hyper plane in a n-dimensional space. This surface is known as the control, or decision, surface shape.

The control surface describes the dynamic of the controller and is generally a time-varying non-linear surface. Owing to a model dynamic present in the design of any controller, techniques should exist for adaptively tuning and modifying the control surface shape.

Fuzzy logic rule-based expert systems use a collection of fuzzy conditional statements derived from an expert knowledge base to approximate and construct the control surface [44].

This paradigm of control system design is based on interpolate and approximate reasoning. Fuzzy logic rule-based controllers or system identifiers, are generally, model-free paradigms. Fuzzy logic rule-based expert systems are universal non-linear function approximators, and any non-linear function (e.g., control surface) of n independent variables and one dependent variable can be approximated to any desired precision.

Alternatively, artificial neural networks are based on analogical learning and try to learn the non-linear decision surface through adaptive and converging techniques, based on numerical data available from input-output, measurements of the system variables and some performance criteria.

Control System Design Stages

In order to obtain the control surface for a non-linear time-varying real-world complex dynamic system, there are a number of simplifying steps used in modeling a controller for the system. The seven basic steps in designing a controller for a complex physical system are as follows:

1.- Large-scale systems are decentralized and decomposed into a collection of decoupled subsystems.

2.- The temporal variations of plant dynamics are assumed to be slowly varying.

3.- The non-linear plant dynamics are locally linearized about a set of operating points.

4.- A set of state variables, control variables, or output features are made available.

5.- A simple, *P, PD, PID* (output-feedback), or state-feedback controller is designed for each decoupled system. The controllers are of regulatory type and are fast enough to perform satisfactorily under tracking control situations. Optimal controllers might also prove useful.

6.- In addition to uncertainties introduced in the first five steps, there are uncertainties due to external environment. The controller design should be made as close as possible to the optimal one based on the control engineers knowledge, in the form of input-output numerical observations data and analytic, linguistic, intuitive, and other kinds of information regarding the plant dynamics and the external environment.

7.- A supervisory control system, either automatic or a human expert operator, forms an additional feedback control loop to tune and adjust the controllers parameters, in order to compensate for the effects of variations caused by unmodel dynamics.

Assumptions in a Fuzzy Control System Design

A number of assumptions are implicit in a fuzzy control system design. Six basic assumptions are commonly made whenever a fuzzy logic-based control policy is selected.

1.- The plant is observable and controllable: State, input, and output variables are usually available for observation and measurement or computation.

2.- There exists a body of knowledge comprised of a set of expert production linguistic rules, engineering common sense, intuition, a set of input/output measurements data, or an analytic model that can be fuzzified and from which rules can be extracted.

3.- A solution exists.

4.- A good enough solution is been looking for, not necessarily the optimum one.

5.- A controller will be designed to the best of our available knowledge and within an acceptable range of precision.

6.- The problems of stability and optimality are still open problems in fuzzy controller design.

The following sections discuss the procedure for obtaining the control surface, *h( )*, from approximations based on a collection of fuzzy IF-THEN rules that

describe the dynamics of the controller. Fuzzy rule-based expert models can also be used to obtain acceptable approximations for the functions *f( )* and *g( )* in the case of a system identification problem. A fuzzy production rule system consists of four structures [45].

1.- A set of rules that represents the policies and heuristic strategies of the expert decision maker.

2.- A set of input data assessed immediately prior to the actual decision.

3.- A method for evaluating any proposed action in terms of its conformity to the expressed rules, given the available data.

4.- A method for generating promising actions and for determining when to stop searching for better ones.

The input data, rules, and output action, or consequence, are generally fuzzy sets expressed as membership functions defined on a proper space. The method used for the evaluation of rules is known as *approximate reasoning*, or *interpolate reasoning*, and is commonly represented by composition of fuzzy relations applied to a fuzzy relational equation.

The control surface, which relates the control action *u( )* to the measured state or output variables, is obtained using these four structures.

It is then sampled at a finite number of points, depending on the required resolutions, and a look-up table is constructed. The look-up table thus formed could be downloaded onto a read-only memory chip and would constitute a fixed controller for the plant.

5.5.2 Simple fuzzy logic controllers

First-generation (non-adaptive; i.e., the four structures above are fixed) simple fuzzy logic controllers can generally be depicted by a block diagram such as that shown in figure 5.8.

The knowledge-base module in Figure 5.8 contains knowledge about all the input and output fuzzy partitions. It will include the term set and the corresponding membership functions defining the input variables to the fuzzy rule-base system and the output variables, or control actions, to the plant under control.

The steps in designing a simple fuzzy logic control system are as follows:

1.- Identify the variables (inputs, states, and outputs) of the plant.

2.- Partition the universe of discourse or the interval spanned by each variable into a number of fuzzy subsets, assigning each a linguistic label (subsets include

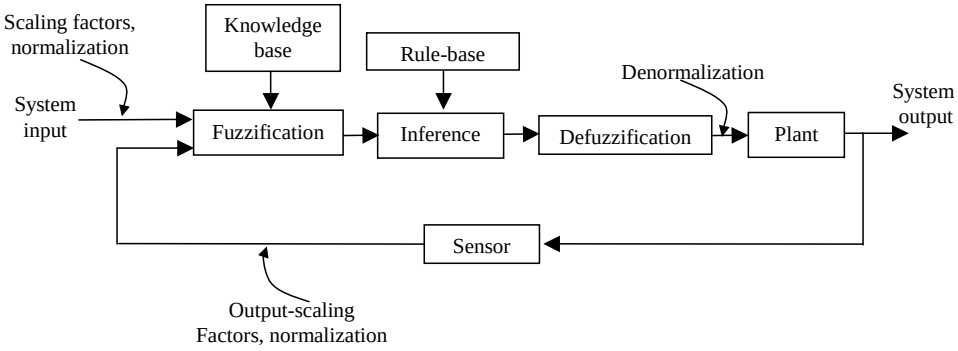

Figure 5.8: Simple fuzzy logic control system block diagram.

all the elements in the universe).

3.- Assign or determine a membership function for each fuzzy subset.

4.- Assign the fuzzy relationships between the inputs or states fuzzy subsets on the one hand and the outputs fuzzy subsets on the other hand, thus forming the rule-base.

5.- Choose appropriate scaling factors for the input and output variables in order to normalize the variable to the [0. 1] or the [- 1.1] interval.

6.- Fuzzify the inputs to the controller.

7.- Use fuzzy approximate reasoning to infer the output contributed from each rule.

8.- Aggregate the fuzzy outputs recommended by each rule.

9.- Apply defuzzification to form a crisp output.

In a non-adaptive simple fuzzy logic controller, the methodology used and the results of the nine steps mentioned above are fixed, whereas in an adaptive fuzzy logic controller, they are adaptively modified based on some adaptation law in order to optimize the controller.

A simple fuzzy logic control system has the following features:

1.- Fixed and uniform input-and output-scaling factors.

2.- Flat, single-partition rule-base with fixed and non-interactive rules. All the rules have the same degree of certainty and confidence, equal to unity.

3.- Fixed membership functions.

4.- Limited number of rules, which increases exponentially with the number of input variables.

5.- Fixed meta-knowledge including the methodology for approximate reasoning, rules aggregation, and output defuzzification.

6.- Low-level control and no hierarchical rule structure.

5.5.3 General fuzzy logic controllers

The principal design elements in a general fuzzy logic control system (i.e., non-simple) are as follows [46]:

1.- Fuzzification strategies and the interpretation of a fuzzification operator, or fuzzifier.

2.- Knowledge base:

a. Discretization/normalization of the universe of discourse.

b. Fuzzy partitions of the input and output spaces.

c. Completeness of the partitions.

d. Choice of the membership functions of a primary fuzzy set

3.- Rule-base:

a. Choice of process state (input) variables and control (output) variables.

b. Source of derivation of fuzzy control rules.

c. Types of fuzzy control rules

d.- Consistency, interactivity, and completeness of fuzzy control rules.

4.- Decision-making logic:

a. Definition of a fuzzy implication

b. Interpretation of the sentence connective *and*

c. Interpretation of the sentence connective *or*

d. Inference mechanism

5.- Defuzzification strategies and the interpretation of a defuzzification operator (defuzzifier)

Adaptation or change in any of the five design parameters above creates an adaptive fuzzy logic control system. If all five are fixed, the fuzzy logic control system is simple non-adaptive.

5.5.4 Incremental fuzzy PD algorithm

In this thesis, the *incremental fuzzy PD control algorithm* is proposed as a variant extension of the fuzzy PD controller [47] to implement biped's balance control. The *incremental fuzzy PD control algorithm* consists of only 4 rules and has the structure illustrated in figure 5.11.

The gains *Gu, Ge* and *Gr* are determined by tuning and they correspond respectively to the output gains, the *error* (ZMP error) and *error rate* (ZMP rate) gains.

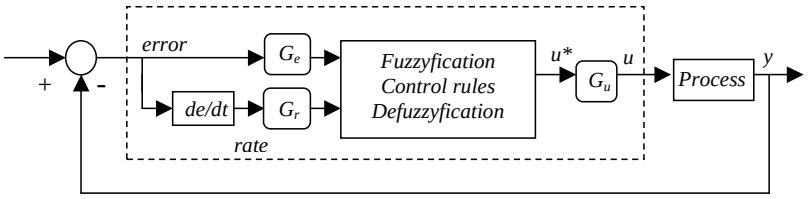

Figure 5.9: Fuzzy PD incremental algorithm structure.

The value u^* is the defuzzyficated output, or *crisp output*. The value u is defined by:

$$u = \left\{ \begin{array}{c} G_u * (u^*) \quad if\ |e| < \theta \quad for\ (t = 0,\ G_{inc} = 0) \\ G_{inc} * (u^*)\ \ if\ |e| > \theta \quad G_{inc} = G_{inc} + inc, until\ \ G_{inc} \geq G_u \end{array} \right\} \quad [4.36]$$

Where e is the *error* ($error^*Ge$), θ is a error boundary selected by tuning, and G_{inc} is the incremental gain obtained adding the increment *inc*.

Figure 5.10 shows the flow diagram for the incremental gain of u and Figure 5.11 shows the area where the absolute error is evaluated and the controller output is incremental (u=G_{inc}+inc).

Fuzzyfication

As is shown in figure 5.12, there are two inputs to the controller: *error* and *rate*. The error is defined as:

$$error = set\ point\text{-}y \quad [4.37]$$

Where *set point* is the point to be reached by the controller and y the controller output.

Rate is defined as follows:

$$rate = (ce\ -\ pe)\ /\ sp \quad [4.38]$$

Where *ce* is the current error, *pe* is the previous error and *sp* is the sampling period. Current and previous error, are referred to an error without gain. The fuzzy controller has a single incremental output, which is used to control the process.

The input an output membership functions for the fuzzy controller are shown in figure 5.12 and figure 5.13, respectively. H and L are two positive constants

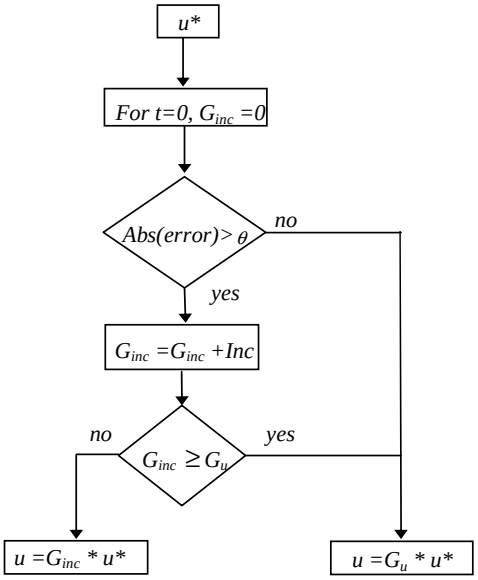

Figure 5.10: Flow diagram for the output gain.

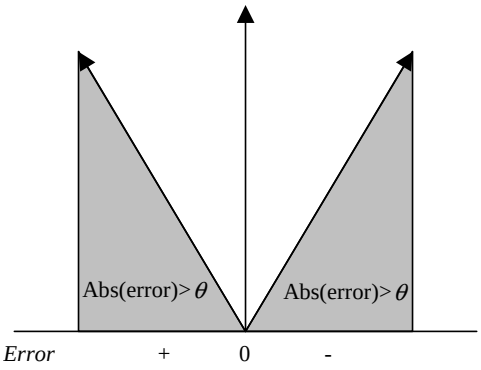

Figure 5.11: *Fuzzy PD incremental* absolute error area.

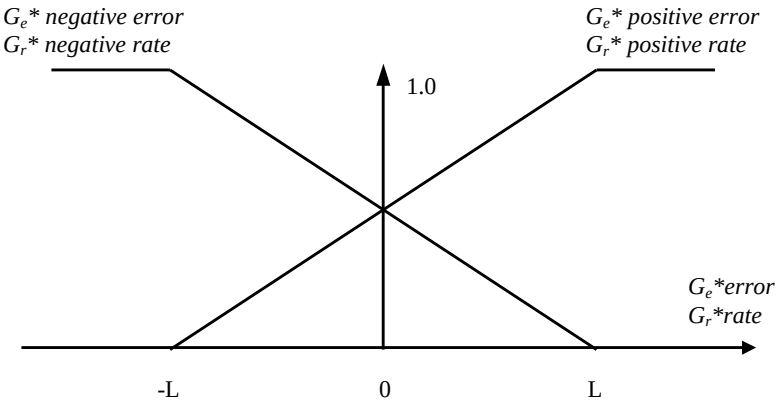

Figure 5.12: Input membership functions.

to be determined. For convenience we will take H=L to reduce the number of control parameters to be determined.

For the *error* input are: *Ge*negative error (en)* and *Ge*positive error (ep)* and for the *rate* input are: *Gr*negative rate (rn)* and *Gr*positive rate (rp)*, while the output fuzzy terms are shown in Figure 5.13 and they are: *Negative output (on), Zero output (oz)* and *Positive output (op).*

Figure 5.12 shows that each input has two linguistic terms.

As shown in figure 5.12, the same membership function is applied for *error* and *rate* but with different scaling factors or gains: *Ge* and *Gr* respectively.

The membership functions for the input variables, *error* and *rate*, are defined by equations 4.39.

$$\mu_{ep} = \frac{L + (Ge * error)}{2L} \qquad \mu_{en} = \frac{L - (Ge * error)}{2L}$$

$$\mu_{rp} = \frac{L + (Gr * rate)}{2L} \qquad \mu_{rn} = \frac{L - (Gr * rate)}{2L} \qquad [4.39]$$

Fuzzy rules

Exist four rules to be evaluated by the *fuzzy PD incremental controller*:

R1. If *error* is *ep* and *rate* is *rp* then output is *op*

R2. If *error* is *ep* and *rate* is *rn* then output is *oz*

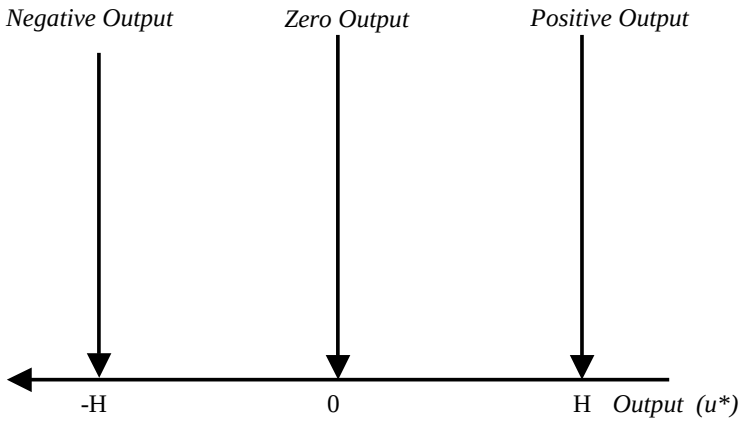

Figure 5.13: Output membership functions.

R3. If *error* is *en* and *rate* is *rp* then output is *oz*

R4. If *error* is *en* and *rate* is *rn* then output is *on*

In the those four rules, the *Mamdani* inference is used [48] that is:

$$min\ (a,b)\ [4.40]$$

being a the membership function of the antecedent and b the membership function of the control action.

Since the rules *R2* and *R3* generate the same output, the rule's union is calculated on those rules. The Lukasiewicz OR operator is applied [39] who is defined as:

$$min\ (a+b,\ 1)\ [4.41]$$

The *and* conjunction in each of the four rules corresponds to the *Zadeh* logical operation *AND*, defined by:

$$\mu_A(x) \wedge \mu_B(x) = \min\{\mu_A(x), \mu_B(x)\}\ [4.42]$$

Where $\mu_A(x)$ and $\mu_B(x)$ are the membership values of the fuzzy sets A and B in the x point, respectively.

Defuzzyfication

The defuzzyfication method used is the *gravity center*, in this case is represented by:

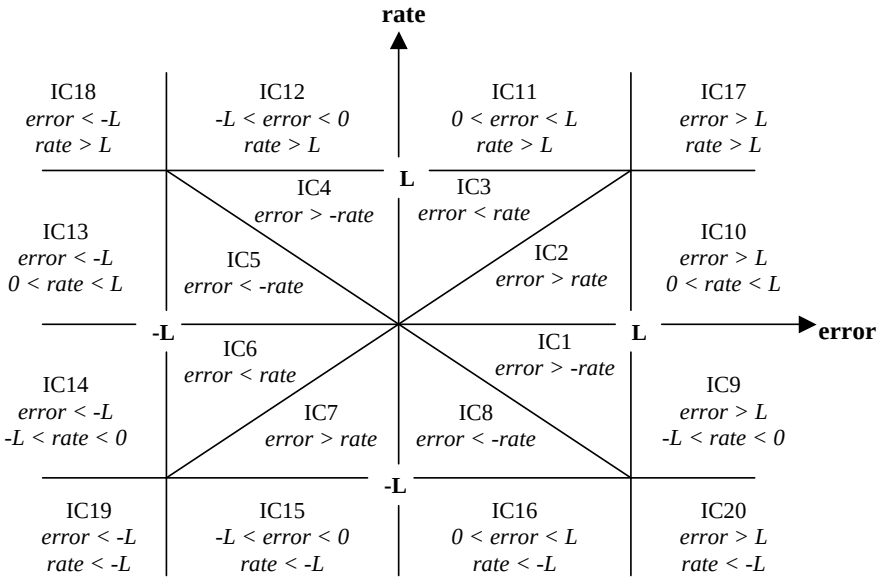

Figure 5.14: Input regions for *error* and *rate*.

$$u = \frac{-H(\mu_{R4(x)}) + 0(\mu_{R2(x)} + \mu_{R3(x)}) + H(\mu_{R1(x)})}{\mu_{R4(x)} + (\mu_{R2(x)} + \mu_{R3(x)}) + \mu_{R1(x)}} \quad [4.43]$$

The *error* and *rate* values ranges can be represented in 20 input regions *(IC)*, as is shown in figure 5.14.

If the membership functions and the 4 control rules are evaluated, H=L and the defuzzyfication is applied in each of the 20 inputs combinations, then 9 equations [6.34] can be obtained [49], which can determine the control signal u that should be applied, depending on the region in which the *error* and *rate* are.

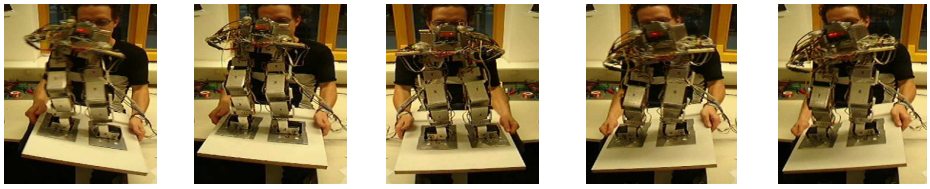

Figure 5.15: "Dany walker" balance during a positive and negative inclination angle.

$$u = \begin{cases} \frac{L}{2(2L-G_e|error|)}[G_e * error + G_r * rate] \; in \; IC1, \; IC2, \; IC5, \; IC6 \\ \frac{L}{2(2L-G_r|rate|)}[G_e * error + G_r * rate] \; in \; IC3, \; IC4, \; IC7, \; IC8 \\ \frac{1}{2}[L + G_r * rate] \; in \; IC9, \; IC10 \\ \frac{1}{2}[L + G_e * error] \; in \; IC11, \; IC12 \\ \frac{1}{2}[-L + G_r * rate] \; in \; IC13, \; IC14 \\ \frac{1}{2}[-L + G_e * error] \; in \; IC15, \; IC16 \\ L \quad in \; IC17 \\ -L \quad in \; IC19 \\ 0 \quad in \; IC18, \; IC20 \end{cases} \qquad [4.44]$$

Thus, to implement the *incremental fuzzy PD controller* algorithm in a compact way (less number of calculus), will be necessary only to know the region in which the inputs variables are and later evaluate the corresponding equation for this region. For example the first equation of [4.44] acts in regions *IC1, IC2, IC5, IC6.*

The *incremental fuzzy PD controller* algorithm was implemented for the "Dany walker" biped robot's balance in a microcontroller. Figure 5.15 shows the "Dany walker" in a real balance case trying to maintain the vertical during a positive and negative inclination angle.

5.6 Walking sequence control

As explained in 4.4, to implement the robot's balance in this thesis, the incremental fuzzy PD algorithm was proposed. But, walking sequence control was

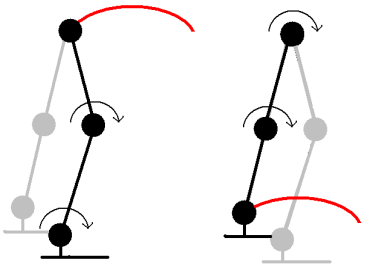

Figure 5.16: Motors movement preformed: *left*) by the leg in contact with the floor, *right*) by the leg rising in the air.

achieved using cubic polynomial interpolation. The *walking sequence control* of a biped robot can be determined by controlling the hip and foot trajectories. To achieve stable dynamic walking, the change between simple supports phase and double supports phase should be smooth. Usually in the beginning of the double supports phase, the foot impact against the floor (when it returns from the air) is very strong and obviously affects the walking balance. In this thesis cubic polynomials are used to control the sagittal motion and guaranties a smooth change between the walking phases [38].

The walking sequence control problem is divided in two parts, the first is represented by the leg in contact with the floor. The second part is represented by the leg rising in the air moving from backward to forward.

In the first case the hip motor don't changes its position. At the same time, the necessaries movements to perform the foot trajectory are made only by the knee and ankle motors as shown in figure 5.16 *left*. Here, the four balance control motors are omitted for the walking sequence analysis.

In the second case the ankle motor don't changes its position. At the same time, the necessaries movements to perform the foot trajectory are made only by the knee and hip motors as shows figure *5.16 right*.

5.6.1 Walking phases

A walking cycle can be divided into single support phase and double support phase. As shown in the figure 5.17, in the single support phase, one foot support the robot's weight while the other foot is moving on the air from backward to forward, at the same time the hip moves along a trajectory *Th* as shown in figure 5.18 (the others parameters will be defined later).

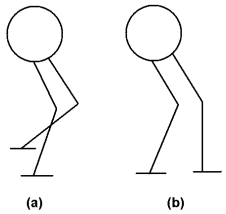

Figure 5.17: Walking phases, (a) single support, (b) double support.

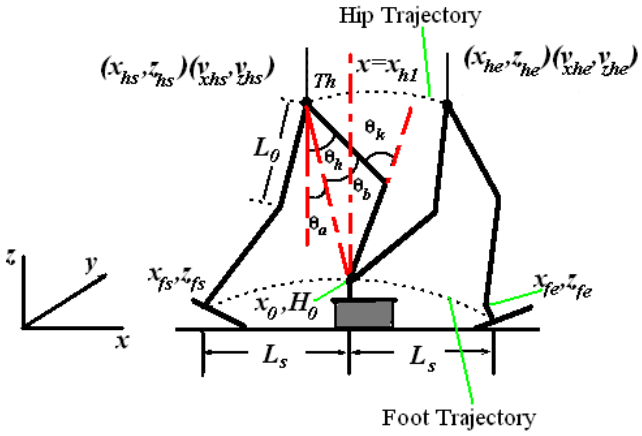

Figure 5.18: Hip and foot trajectory.

The simple support phase begins when the foot in movement leaves the floor and lift on the air and finishes when it returns to the floor. The double support phase begin when the foot in movement (at single support phase) touch the floor and ends when the foot at the floor (at single support phase) leaves the floor. To achieve dynamic walking, the change between simple supports phase and double supports phase should be smooth.

Usually in the beginning of the double supports phase, the foot impact against the floor (when it returns from the air) is very strong and obviously affects the walking balance. In this thesis cubic polynomials are used to control the walking sequence (sagittal motion) and guaranties a smooth change between the walking phases.

5.6.2 Cubic polynomial interpolation algorithm

The process of building a *f(x)* function, such verify, that in predetermined values of the independent variable *x0, x1, ... , xn* can takes values like *y0=f(x0), y1=f(x1), ..., yn=f(xn)* is known as interpolation and can be defined as a classic procedure for function approach [38].

Is very important to establish the type function such satisfy the interpolation condition *yi=f(xi)* with *i=0, ... , n* . Then exists an infinity number of functions that satisfy the data conditions to be interpolate.

A polynomial generic function can be generated by *n+1* coefficients with regard to a fixed base. The *n+1* conditions produces an equations system whose resolution generates the searched function.

Considering as a base

$$\Im = \left\{1, x, x^2 ..., x^n\right\} \text{ [4.45]}$$

a polynomial can be defined as:

$$P_n(x) = a_0 + a_1 x + a_3 x^2 + ... + a_n x^n \text{ [4.46]}$$

Considering an interpolation support of *x0 <x1 <, ... ,<xn* and their corresponding function values *y0, y1,... , yn* the equations system to generate the interpolation conditions are:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ .... & ... & ... & ... & ... \\ 1 & x_n & x_n^2 & ... & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ ... \\ y_n \end{bmatrix} \text{ [4.47]}$$

The Matrix of this system acquires a special structure denominated Vandermonde Matrix whose determinant can be calculated easily by:

$$\det(B) = \prod_{i>j} (x_i - x_j) \text{ [4.48]}$$

Because the interpolation nodes *x0 <x1 <, ... ,<xn* are different, is evident that $\det(B) \neq 0$ independently of the interpolation support. Thus, the interpolation problem always have an unique solution.

Cubic polynomial interpolation example

Consider the interpolation points (0,2), (1,1), (2,0) ,(3,5). The nodes support are *S={1,2,3}* that corresponds to *n=3*, then, the searched polynomial have to be of third degree or smaller.

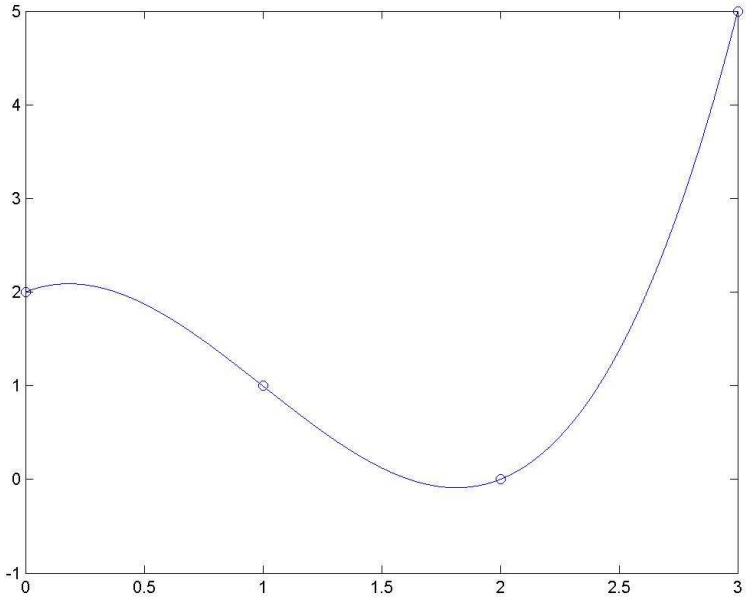

Figure 5.19: Polynomial interpolation.

$$P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \text{ [4.49]}$$

The equations system represented as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 5 \end{bmatrix} \text{ [4.50]}$$

Solving the equation system, we obtain *a0=2, a1=1, a2=-3* and *a3=1*, and the polynomial is represented as (Figure 5.19):

$$P_3(x) = 2 + x - 3x^2 + x^3 \text{ [4.51]}$$

Implementation

The *walking sequence control* algorithm is computed as follow:

1. For each step, the desired velocity (v_{xhe}, v_{zhe}) and the step length L_s are previously specified.

2. The desired angle is previously specified (figure 5.18).

3. The hip and foot trajectories are generated by different walking periods, then is chosen the trajectory that guarantees the ZMP criteria.

Hip trajectory for single support phase

The hip trajectory can be generated by cubic polynomials algorithms, if the initial and final state are known from single phase. In figure 5.18 the initial state is defined by [x_{hs},z_{hs}] and the final state by [x_{he},z_{he}]. The initial velocity [v_{xhs},v_{zhs}] (produced when the robot leaves the initial position) is also specified in the trajectory model. The same case is for the final velocity [v_{xhe},v_{zhe}] (when the robot arrives to his final position) [6].

The initial and final state positions for the cubic trajectory in z (the *zh(t)* direction) can be expressed as:

$$z_h(t) = \begin{cases} z_{hs} \quad if \quad t = kT \\ z_{he} \quad if \quad t = kT + T_s \end{cases} \quad [4.52]$$

where T is the period for the robot's step and T_s the period in single support phase.

$$z_h^{\bullet}(t) = \begin{cases} z_{zhs} \quad if \quad t = kT \\ z_{zhe} \quad if \quad t = kT + T_s \end{cases} \quad [4.53]$$

The cubic polynomial can be generalized by the following expression:

$$z_h(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \ [4.\ 54]$$

obtaining:

$$z_h(t) = z_{hs} + v_{zhs}(t - kT) + \frac{3(z_{he} - z_{hs}) - 2v_{zhs}T_s - v_{zhe}T_s}{T_s^2}(t - kT)^2$$

$$+\frac{2(z_{hs} - z_{he}) + (v_{zhs} + v_{zhe})T_s}{T_s^3}(t - kT)^3 \quad kT < t \leqslant kT + T_s \ [4.55]$$

$x_h(t)$ is divided in two parts: from $x_h(KT)$ to $x_h(KT + T_1)$ and from $x_h(KT + T_1)$ to $x_h(KT + T_p)$. The definition for $x_h(t)$ is shown in equation [4.56].

$$\begin{cases} x_h(t) = x_{hs} & t = kT \\ x_h(t) = x_{h1} & t = kT + T_1 \\ x_h(t) = x_{he} & t = kT + T_p \\ \dot{x}_h(t) = v_{xhs} & t = kT \\ \dot{x}_h(t^-) = \dot{x}_h(t^+) & t = kT + T_1 \\ \dot{x}_h(t) = v_{xhe} & t = kT + T_p \\ \ddot{x}_h(t) = a_0 & t = kT \end{cases} \quad [4.56]$$

where a_0 should be previously specified to satisfy the initial condition of acceleration. The cubic polynomial trajectory can be obtained using equation [4.57].

$$x_h(t) = \begin{cases} x_{hs} + v_{xhs}(t-kT) + \frac{1}{2}a_0(t-kT)^2 \\ \quad + \frac{(x_{h1} - x_{hs} - v_{xhs}T_1 - \frac{1}{2}a_0T_1^2)(t-kT)^3}{T_1^3} & kT < t \le kT + T_1 \\ x_{h1} + v_{xh1}(t-kT-T_1) \\ \quad + \frac{(3(x_{he} - x_{h1}) - 2v_{xh1}(T_p - T_1))(t-kT-T_1)^2}{(T_p - T_1)^2} \\ \quad + \frac{(2(x_{h1} - x_{he}) + (v_{xh1} + v_{x2})(T_p - T_1))(t-kT-T_1)^3}{(T_p - T_1)^3} & kT + T_1 < t \le kT + T_p \end{cases} \quad [4.57]$$

Swing and foot trajectory at single support phase

The cubic interpolation is used to generate the foot trajectory in single support phase. The initial and final foot position which represents the satisfied states and velocities are [6]:

$$x_f(t) = \begin{cases} x_f(t) = x_{fs} & t = kT \\ x_f(t) = x_{fe} & t = kT + T_s \\ \dot{x}_f(t) = 0 & t = kT \\ \dot{x}_f(t) = 0 & t = kT + T_s \end{cases} \quad [4.58]$$

$$z_f(t) = \begin{cases} z_f(t) = z_{fs} & t = kT \\ z_f(t) = z_{fe} & t = kT + T_s \\ \dot{z}_f(t) = 0 & t = kT \\ \dot{z}_f(t) = 0 & t = kT + T_s \end{cases} \quad [4.59]$$

Knowing the initial and final states in x and z axis, a smooth trajectory can be generated by the cubic polynomial interpolation. Defined as follows:

for $x_f(t)$:

$$x_f(t) = x_{fs} + 3(x_{fe} - x_{fs})\frac{(t-kT)^2}{T_s^2} - 2(x_{fe} - x_{fs})\frac{(t-kT)^3}{T_s^3} \quad kT < t \leqslant kT + T_s$$

[4.60]

for $z_f(t)$:

$$\begin{cases} z_{fs} + 3(z_{fm} - z_{fs})\frac{(t-kT)^2}{T_m^2} - 2(z_{fm} - z_{fs})\frac{(t-kT)^3}{T_m^2} kT < t \leqslant kT + T_m \\ z_{fm} + 3(z_{fe} - z_{fm})\frac{(t-kT-T_m)^2}{(T_s-T_m)^2} - 2(z_{fe} - z_{fm})\frac{(t-kT-T_m)^3}{(T_s-T_m)^3} kT + T_m < t \leqslant kT + T_s \end{cases}$$

[4. 61]

Determination of the *Tp,vxhs,T1,Tm* and *zfm* parameters

The determination of the parameters *Tp,vxhs,T1,Tm* and *zfm* is not trivial because the modification of one of them supposes a change in the trajectory and conditions of smoothness in the velocity and accelerations. One of the main characteristics of this approach is that the velocity at the end of *single*

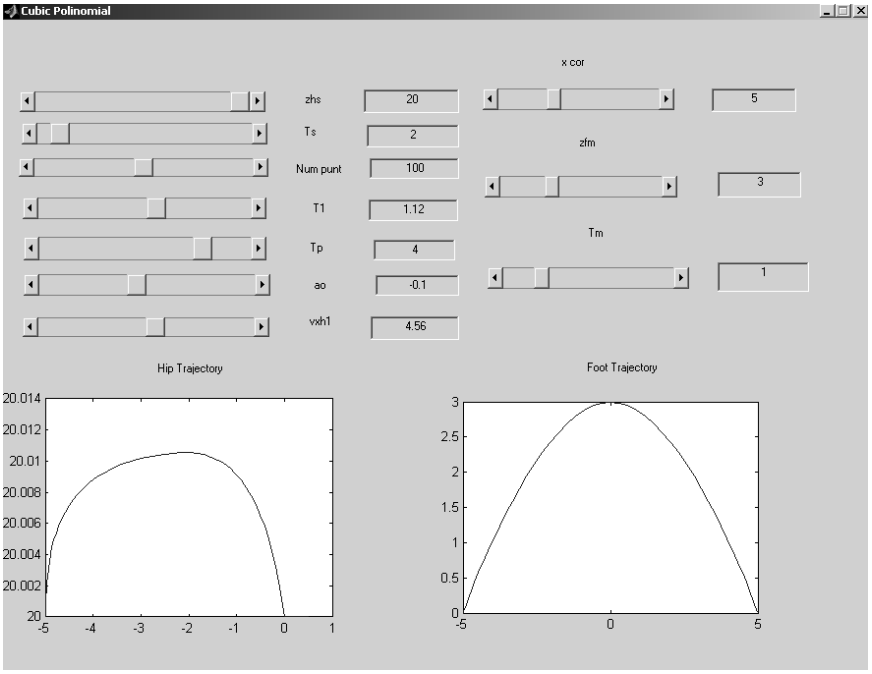

Figure 5.20: Matlab program to calculate the algorithm parameters.

support phase is zero or near to zero, therefore a smooth contact with the floor surface is guaranteed [15].

To determine these parameters a *Matlab* interface was programmed, it allows to modify all the important parameters (represented at the equations [4.55], [4.57], [4.60] and [4.61]) quickly. Thus, we can easily vary the parameters and verify the consistency of the trajectories. The figure 5.20 show the program interface.

The trajectories for the hip and the foot at single support phase are calculated using equations [4.55], [4.57], [4.60] and [4.61]. A real walk sequence on "Dany walker" biped robot obtained using these equations is shown in figure 4.21. This sequence is only for a leg, but the same sequence is used for the other leg, completing the walk sequence.

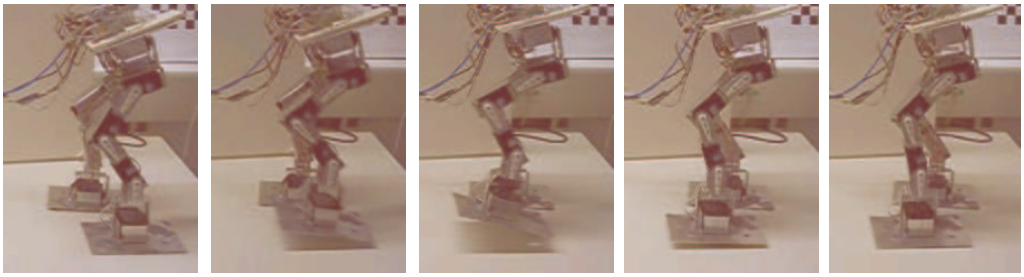

Figure 5.21: Real walk sequence on "Dany walker" biped robot.