Chapter 4

Robot's mathematical model

4.1 Introduction

In order to adjust the robot's controllers, a kinematic and dynamic mathematical model of the biped robot (include the robot's physical characteristics) are needed. In this section the deduction of the kinematics and the dynamic model are exposed. The mathematical robot's kinematic model is obtained to implement the simulation of the biped's robot kinematic. The kinematics is obtained by the handle of homogeneous transformation matrix applying the Denavit Hartenverg method. The robot's dynamic is obtained by the use of the inverted pendulum approach, to model the sagittal plane (walking sequence), an artificial neural network used as a system identification to model the ZMP robot's dynamic (balance).

4.2 Kinematics model

The *kinematics* is the study of the robot's movements with regard to a reference system. Is an analytic description of the spacial movement of the robot like a function of time and a relationship, between the position and the orientation (localization) of the robot's final link and the values of their joint coordinates.

The *direct kinematics* consists on place the robot's final link (position and orientation), with regard to a reference system of coordinates, resolving the values of each link and the geometric parameters of the robot's elements. In other words, from the solution of each link coordinates, the final link localization

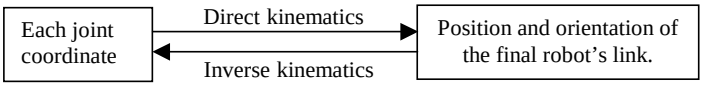

Figure 4.1: Direct and inverse kinematics.

is deduced (without having control where the final link is going to be).

Also is possible to find the position and the orientation of the robot's final link based only on its geometric relations (by a geometric method) but this is not a systematic method and can be used only in a robot with a few degrees of liberty.

The *inverse kinematics* consists on determining the configuration that should adopt the robot at each link to reach a goal position and orientation for the final link. The figure 3.1 shown the relationship between the direct and the inverse kinematics. Table 3.1 shown a summary of the kinematics process.

Kinematic	**Having**	**Finds**
Direct kinematic	Each joint coordinate	Position and orientation of each joint at the final robot's link
Inverse Kinematic	Position and orientation of the final robot link	Each joint coordinate

Table 3.1: Summary of the kinematics process.

4.2.1 Representation by homogeneous transformation matrix

The kinematic model of a robot, can be represented by the *homogeneous transformation matrix* [HTM] (as explained in section 2.3). This representation is necessary for robots with more than two degrees of freedom. From that reason, is convenient to have a systematic method based on homogeneous transformation matrix.

A robot of n degrees of freedom (DOF) is formed by n links assembled by n articulations, in such way that each *articulation-link* constitutes a DOF. To each link a reference system could be associated and using homogeneous

transformations, is possible to represent the rotations and relative translations from the different links which compose the robot [23].

When homogeneous transformation matrix are used to represent the position and the relative orientation between two consequent links, a reference system to each link must be associated. Thus, is possible to represent the translations and relative rotations between the different links. Where the matrix L_{n+1} represents the position and relative orientation between the associated systems of two consequent robot's links. This matrix allows a total or partial representation of the robot's kinematics. For example, ${}^0L_1$ represents the position and orientation of the first link relative to the base reference, and ${}^0L_3 = {}^0L_1{}^1L_2{}^2L_3$ represents the position and orientation of the third link relative to the base reference coordinate system.

When all the DOF are considered, the ${}^nL_{n+1}$ matrix is called T. Thus, for a six degree of freedom robot, the final links position and orientation is represented by the R matrix:

$$R = {}^0L_6 = {}^0L_1{}^1L_2{}^2L_3{}^3L_4{}^4L_5{}^5L_6 \quad [3.7]$$

To locate the coordinate systems from each link and to obtain the robot's kinematics there are systematic methods, like the *Denavit-Hartenberg* (D-H) method.

4.2.1.1 Denavit Hartenberg method (D-H)

D-H, describe robot's kinematics and represent its motions. The method works with the quadruple $\{a_i, \alpha_i, d_i, \theta_i\}$ and label an orthonormal (x,y,z) coordinate system to each robot joint. Relating this way, each joint's reference system to the next and forming a complete robot's geometry representation.

D-H Notation

Anthropomorphic robots generally are made up of links that are connected by each joint to their preceding and subsequent links. The origin herein is the base reference coordinate system. Joints could be rotational or translational. Each joint has one degree of freedom (DOF). Therefore, a concatenation of $n+1$ links by n joints means in total n DOF. "Dany walker" has 5 servos in each leg (a servo represent a rotational joint). Also, in total the robot has 10 DOF.

As *Dieter Kraft* explains in [24], these kinematics parameters are divided in two groups: link parameters and joint parameters. The first group comprises

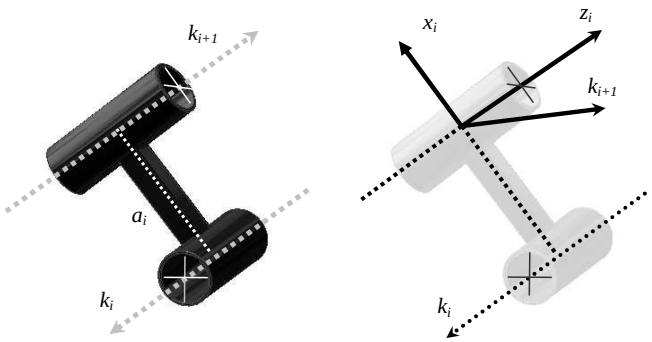

Figure 4.2: Link length (a_i).

link length and link twist, which are determined by the mechanical construction and are thus invariant. The latter group defines the joint distance, which is the drooping of a translational joint and the joint angle, which is the deflection of a rotational joint.

Link Parameters

Link i is the interconnection of the joints k_i and k_{i+1}. The shortest distance of both skew joint axes is the distance between both root points of their common normal. It is called a_i *link length* [25] (Figure 4.2).

The origin of link is the coordinate system $X_i = \{x_i, y_i, z_i\}$ is congruent with the root point of the common normal in the axis of k_{i+1}. Likewise, link *i-1s* coordinate origin $X_{i-1} = \{x_{i-1}, y_{i-1}, z_{i-1}\}$ lies in the foot point of the common normal at the k_i side.

The angle covered by the coordinate axes z_i and z_{i+1}, when shifting X_i into X_{i-1} along a_i, is called *link twist* α_i (Figure 4.3).

describe robot's kinematics and is a standard way to represent and model their motions.

Joint Parameters

These parameters refers to possible movements and are variable. The distance between the origin of the coordinate system X_{i-1} and the root point of the common normal on k_i is referred to by *joint distance* d_i (Figure 4.4). This is the degree of freedom of a translational joint.

More important, in the case of "Dany walker" is the so-called *joint angle* θ_i

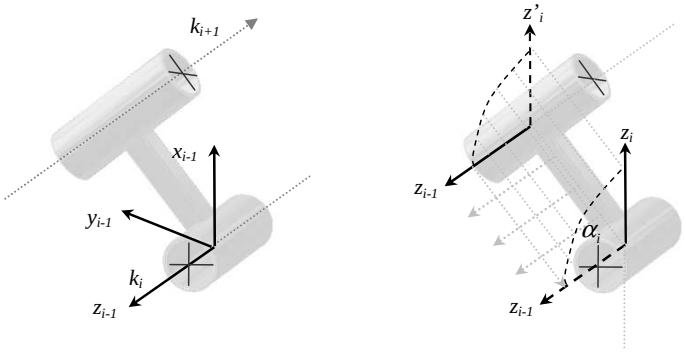

Figure 4.3: Link twist (α_i).

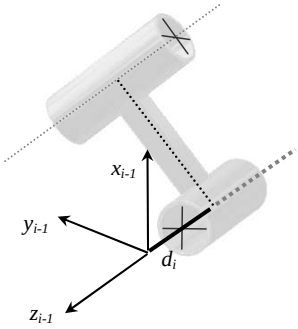

Figure 4.4: Joint distance (d_i).

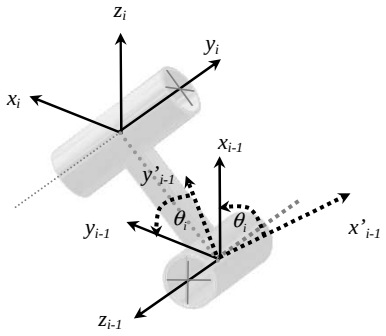

Figure 4.5: Joint angle (θ_i).

(figure 4.5), which represents the variable of rotational joints. The rotation of link i in joint k_i rolls axis x_{i-1} into x_i by moving it by θ_i .

D-H Representation

The D-H method allows the step from a link to the following link by 4 basic transformations that depends only on the robot's constructive characteristics. These are basic transformations that relate the reference system of the element *n+1* with the reference system of the element n (figure 4.6) [26].

1. A rotation θ_{n+i} about the Z_n axis (to bring X_n parallel with X_{n+1})
2. A translation d_{n+i} along the Z_n axis (to make the *x-axes* collinear)
3. A translation a_{n+i} along the X axis (to make the *z-axes* coincide)
4. A rotation α_{n+i} about the X_n axis (to bring Z_n parallel with Z_{n+1})

Together, these four transformations in the above order lead to an unique homogeneous transformation matrix with four variables representing the relationship between these two links. Since the matrix product is not commutative, the operation should be made in that order. In resume [27]:

$$^{n}L_{n+1} = rot(z, \theta_{n+1})trn(0, 0, d_{n+1})trn(a_{n+1}, 0, 0)rot(x, \alpha_{n+1}) \text{ [3.8]}$$

The correspondent matrices are:

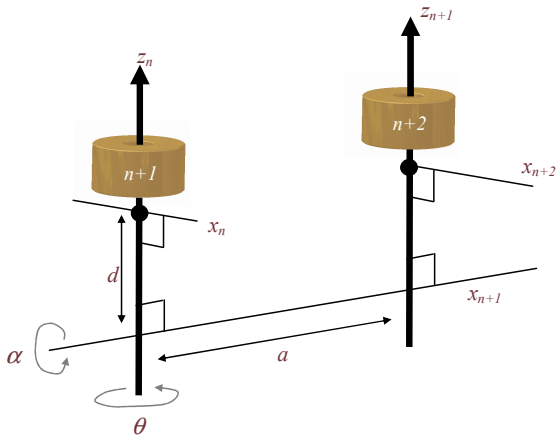

Figure 4.6: Basic transformations that relate the reference system of the element *n+1* with the reference system of the element *n*.

$$^{n}L_{n+1} =$$

$$= \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{n+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

after made the product between the matrix es is obtained:

$$= \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [3.9]$$

Where $\theta_{n+1}, d_{n+1}, a_{n+1}, \alpha_{n+1}$ are the D-H parameters for the i link. Thus, is enough to identify the $\theta_{n+1}, d_{n+1}, a_{n+1}, \alpha_{n+1}$ parameters to obtain the $^{n}L_{n+1}$ matrices and relate each robot's link.

In order to relate the $\{S_1\}$ and $\{S_2\}$ systems by a $^{n}L_{n+1}$ matrix, the systems must be previously conformed according to some norms described by the following algorithm.

D-H Algorithm

Denavit-Hartenberg proposed an algorithm to represent the kinematics of a robot, next a detailed algorithm step by step explanation is presented [28]:

Algorithm 1 Denavit-Hartenberg Algorithm

1: Numerate links beginning with *1* (first mobile link of link's chain) and ending with n (last mobile link). The fixed base reference coordinate system will be numbered as link *0*.
2: Numerate each articulation beginning with *1* (that is the first DOF for a joint) and ending with n.
3: Locate axis of each articulation. If this is *revolving*, the axis will be its own turn axis. If it is *prismatic*, it will be the axis along which the displacement takes place.
4: For $n+1$ of link *0* to n locate Z_{n+1} axis on the axis of articulation n.
5: Place the origin of the base reference coordinate system in any point of z_0 axis. Axes z_0 and y_0 will be located so that they form a right-handed system with z_0.
6: For $n+1$ of link *1* to n, place the $\{S_j\}$ system with regard to the link $n+1$) in the intersection of Z_{n+1} axis with the normal line common to Z_n and Z_j. If both axes cuts, $\{S_j\}$ would be located in the cut point. If they were parallel then $\{S_j\}$ would be located in the articulation n.

According to *D-H* algorithm, all links have to be labeled from *zero* to n, beginning in the base reference coordinate system link. The *concatenated* link chain include links, connected by n joints and establishes a set of $4n$ parameters, n parameters are variable, there are no *translational* and *rotational* joints at the same time. Joint i connects link *i-1* with link i.

Also, Dieter Kraft proposes an algorithm [24]:

Algorithm 2 Dieter Kraft algorithm

1: *Libeling* the joints from 1 to n.
2: Define *base* coordinate system $X_0 = \{x_0, y_0, z_0\}$ in the base body, so that moving axis *1* and coordinate axis z_0 are collinear.
3: *Joint coordinate systems* $\forall(1 \leqslant i \leqslant n-1)$.
Align z_1 axis in direction of joint *i+1*.
Choose origin of coordinate system X_i in:
- intersection of z_i and z_{i-1} axis or
- intersection of common normal $(z_{i-1} \rightarrow z_i)$and the axis.
Determine x_i axis either:
- *ortho normal* to both z-axes

$$x_i = \pm \frac{(z_{i-1} \times z_i)}{|z_{i-1} \times z_i|}$$

or
- along *common normal*, if both z axes are parallel.
Complete *right-handed* coordinate system with y_i axis

$$y_i = \pm \frac{(z_{i-1} \times x_i)}{|z_i \times x_i|}$$

4: *Link Parameter* $\forall(1 \leqslant i \leqslant n)$.
Link length a_i is the distance between the intersection of z_{i-1} axis with x_i axis and the origin of the coordinate system X_i, along x_i axis.
Link Twist α_i is the angle, around x_i axis, that turns z_{i-1} axis into z_i axis.
5: *Joint Paramer* $\forall(1 \leqslant i \leqslant n)$.
Joint distance d_i is the distance between the origin of X_{i-1} and the intersection of its z_{i-1} and x_i axis, along z_{i-1} axis.
Joint angle θ_i is the angle around z_{i-1} axis that x_{i-1} axis into x_i axis.

4.2.1.2 "Dany walker" kinematics model

The *D-H* Algorithm is applied to obtain the "Dany walker" kinematics model. Thus, the first step is to label each joint, with a coordinate system of reference. Figure 4.7 and figure 4.8 shown the reference coordinate system assignation for left and right leg respectively. The tables 3.3 and 3.4 shown the right and left leg's parameters respectively.

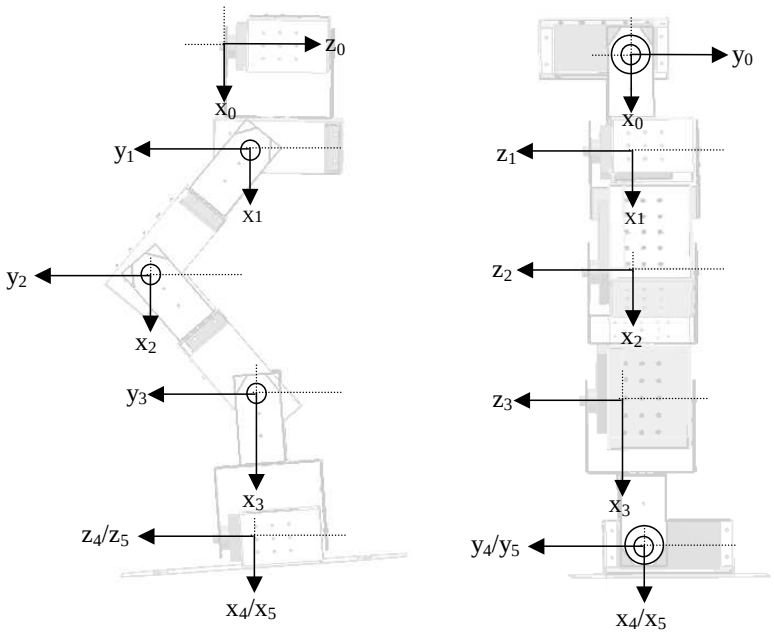

Figure 4.7: Coordinate systems assignation for left leg (left: side view, right: front view)

i	link	joint	a_i/ [mm]	α_i/ [°]	d_i/ [mm]	θ_i/ [°]
0	Waist	None	0	0	0	0
1	90°Joint Top	Waist-Lateral	7	90	0	wl
2	Thigh	Waist-Thigh	11	0	0	wt
3	Shank	Knee	11	0	0	kn
4	90°Joint Down	Ankle-lateral	7	90	0	al
5	Foot	Ankle	0	0	0	an

Table 3.3: D-H Parameters Left leg.

All z-axes are pointed in the direction, so that they perform a right-screw in mathematical term.

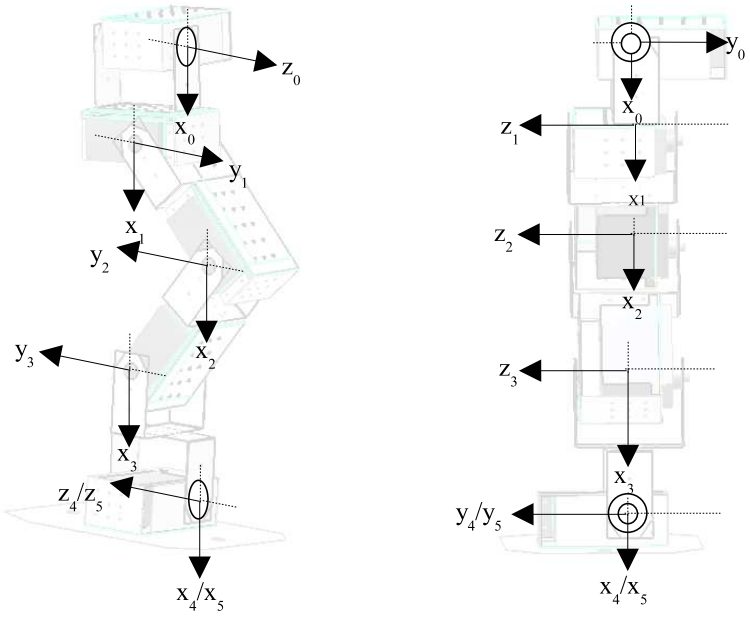

Figure 4.8: Coordinate systems assignation for right leg (left: isometric view, right: back view)

i	link	joint	a_i/ [mm]	α_i/ [°]	d_i/ [mm]	θ_i/ [°]
0	Waist	None	0	0	0	0
1	90°Joint Top	Waist-Lateral	7	90	0	wl
2	Thigh	Waist-Thigh	11	0	0	wt
3	Shank	Knee	11	0	0	kn
4	90°Joint Down	Ankle-lateral	7	90	0	al
5	Foot	Ankle	0	0	0	an

Table 3.4: D-H Parameters Right leg.

Since, the a_i and d_i D-H parameters for both legs are fixed, the only parameters who changes are θ_i (leg's angles of each joint). As convention in this thesis, the positions of each joint, show in figure 4.7 and figure 4.8 mean a 0° angle.

Apparently, from tables 3.3. and 3.4, both legs have the same values, however at walking, the angles are not the same, because the two legs are not in the same position (except when the robot is standing).

To express coordinate system *Xi* in *Xi-l* , the previously determined parameters can be used to calculate the transformation with each two translations and rotations. According to [24], the matrix in 3.10 transforms from *Xi* to *Xi-l*;

Since the two legs are identical, the link-to-link transformations are also identical for both legs:

$$ {}^{i-1}_{i}L = rot(\theta_i) \cdot trn(d_i) \cdot rot(\alpha_i) \cdot trn(a_i) $$

$$ {}^{i-1}_{i}L = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \cdot \sin\theta_i & \sin\alpha_i \cdot \sin\theta_i & a \cdot \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cdot \cos\theta_i & -\sin\alpha_i \cdot \cos\theta_i & a \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad [3.10] $$

$$ {}^{0}_{1}L = \begin{pmatrix} \cos\theta_{wl} & 0 & \sin\theta_{wl} & 7 \cdot \cos\theta_{wl} \\ \sin\theta_{wl} & 0 & -\cos\theta_{wl} & 7 \cdot \sin\theta_{wl} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} [3.11] $$

$$ {}^{1}_{2}L = \begin{pmatrix} \cos\theta_{ht} & -\sin\theta_{ht} & 0 & 11 \cdot \cos\theta_{ht} \\ \sin\theta_{ht} & \cos\theta_{ht} & 0 & 11 \cdot \sin\theta_{ht} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} [3.12] $$

$$ {}^{2}_{3}L = \begin{pmatrix} \cos\theta_{kn} & -\sin\theta_{kn} & 0 & 11 \cdot \cos\theta_{kn} \\ \sin\theta_{kn} & \cos\theta_{kn} & 0 & 11 \cdot \sin\theta_{kn} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} [3.13] $$

$$ {}^{3}_{4}L = \begin{pmatrix} \cos\theta_{al} & 0 & \sin\theta_{al} & 10 \cdot \cos\theta_{al} \\ \sin\theta_{al} & 0 & -\cos\theta_{al} & 10 \cdot \sin\theta_{al} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} [3.14] $$

$$ {}^{4}_{5}L = \begin{pmatrix} \cos\theta_{an} & -\sin\theta_{an} & 0 & 0 \\ \sin\theta_{an} & \cos\theta_{an} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} [3.15] $$

The transformation matrix from foot to hip is the concatenated transformation from *link-to-link* ${}^{0}_{5}L = \prod_{i=1}^{5} {}^{i-1}_{i}L$. Because both legs have the same mathematical representation, only one matrix had to be calculated for both legs. This matrix, considers as a base coordinate system, the origin of the hip servos axis of each leg. But now, considering an unique base reference coordinate system, for both legs, which be locate between the two former ones (in the middle of the hip), a constant must be add to the base reference coordinate system on the y_i-axis. In the matrix calculation, element (4,1) must be add by a constant value, which specifies the displacement along the y_i-axis. The matrix is represented as follows:

$$ {}^{0}_{5}T = \begin{pmatrix} \cos\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\sin\theta_{an}\sin\theta_{wl} & -\sin\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\cos\theta_{an}\sin\theta_{wl} & \cos\theta_{wl} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \cos\theta_{wl}\begin{pmatrix} 7\cdot(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) \\ +11\cos(\theta_{kn}+\theta_{wt}) \\ +11\cos\theta_{wt} \end{pmatrix} \\ \cos\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\sin\theta_{an}\sin\theta_{wl} & -\sin\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\cos\theta_{an}\cos\theta_{wl} & -\sin\theta_{wl} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \sin\theta_{wl}\begin{pmatrix} 7\cdot(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) \\ +11\cos(\theta_{kn}+\theta_{wt}) \\ +11\cos\theta_{wt} \end{pmatrix} \\ \cos\theta_{an}\sin(\theta_{al}\theta_{kn}+\theta_{wt}) & -\sin\theta_{an}\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & -\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) & 7\cdot\sin(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) \\ & & & +11\sin(\theta_{kn}+\theta_{wt}) \\ & & & +11\sin\theta_{wt} \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

[3.16]

Not only transformations from one foot to the hip are needed, but also vice verse. Therefore, the matrix shown above must be inverted. In addition to this, matrices translating from shank to hip, and from thigh to hip are necessary. These matrices are also needed in both directions, bottom-up and top down [29].

Thus, twelve matrices are calculated: *Foot-to-Hip, Shank-to-Hip,* and *Thigh-to-Hip*, all of them for the left and right leg, and all inverted. Below, all matrices are shown, to point out the similarities:

Foot-to-Waist with common-base extension:

$$
{}^{Waist}_{Foot}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\sin\theta_{an}\sin\theta_{wl} & -\sin\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\cos\theta_{an}\sin\theta_{wl} & \cos\theta_{wl} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \cos\theta_{wl}\begin{pmatrix}7\cdot(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}))\\ +11\cos(\theta_{kn}+\theta_{wt})\\ +11\cos\theta_{wt}\end{pmatrix} \\
\cos\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\sin\theta_{an}\cos\theta_{wl} & -\sin\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\cos\theta_{an}\cos\theta_{wl} & \sin\theta_{wl} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \sin\theta_{wl}\begin{pmatrix}7\cdot(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}))\\ +11\cos(\theta_{kn}+\theta_{wt})\\ +11\cos\theta_{wt}\end{pmatrix}\pm 6 \\
\cos\theta_{an}\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & -\sin\theta_{an}\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & -\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) & \begin{matrix}7\cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt})\\ +11\sin(\theta_{kn}+\theta_{wt})\\ +11\sin\theta_{wt}\end{matrix} \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.17]

Waist-to-Foot:

$$
{}^{Foot}_{Waist}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\sin\theta_{an}\sin\theta_{wl} & \cos\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\sin\theta_{an}\cos\theta_{wl} & \cos\theta_{an} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \begin{matrix}\cos\theta_{an}\begin{pmatrix}7\cdot(1+\cos(\theta_{al}+\theta_{kn}+\theta_{wt}))\\ +11\cos(\theta_{al}+\theta_{kn})\\ +11\cos\theta_{al}\end{pmatrix}\\ \pm 6\begin{pmatrix}\cos\theta_{an}\sin\theta_{wl}(\theta_{al}+\theta_{kn}+\theta_{wt})\\ -\sin\theta_{an}\cos\theta_{wl}\end{pmatrix}\end{matrix} \\
-\sin\theta_{an}\cos\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) +\cos\theta_{an}\sin\theta_{wl} & -\sin\theta_{an}\sin\theta_{wl} \cdot\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) -\cos\theta_{an}\cos\theta_{wl} & -\sin\theta_{an} \cdot\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \begin{matrix}\sin\theta_{an}\begin{pmatrix}7\cdot(1+\cos(\theta_{an}+\theta_{wl}+\theta_{wt}))\\ +11\cos(\theta_{al}+\theta_{kn})\\ +11\cos\theta_{al}\end{pmatrix}\\ \pm 6\begin{pmatrix}-\sin\theta_{an}\sin\theta_{wl}c\cos(\theta_{al}+\theta_{kn}+\theta_{wt})\\ -\cos\theta_{an}\cos\theta_{wl}\end{pmatrix}\end{matrix} \\
\cos\theta_{wl}\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & \sin\theta_{wl}\sin(\theta_{al}+\theta_{kn}+\theta_{wt}) & -\cos(\theta_{al}+\theta_{kn}+\theta_{wt}) & \begin{matrix}-7\sin(\theta_{al}+\theta_{kn}+\theta_{wt})\\ -11\sin(\theta_{al}+\theta_{kn})-11\sin\theta_{al}\\ \pm 6(\sin(\theta_{al}+\theta_{kn}+\theta_{wt})\sin\theta_{wl}\end{matrix} \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.18]

Shank-to-Waist:

$$
{}^{Waist}_{Shank}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{wl}\cos(\theta_{wt}+\theta_{kn}) & -\cos\theta_{wl}\sin(\theta_{wt}+\theta_{kn}) & \sin\theta_{wl} & \cos\theta_{wl}\begin{pmatrix}11\cos(\theta_{kn}+\theta_{wt})\\+11\cos\theta_{wt}+7\end{pmatrix}\\
\sin\theta_{wl}\cos(\theta_{wt}+\theta_{kn}) & -\sin\theta_{wl}\sin(\theta_{wt}+\theta_{kn}) & -\cos\theta_{wl} & \sin\theta_{wl}\begin{pmatrix}11\cos(\theta_{kn}+\theta_{wt})\\+11\cos\theta_{wt}+7)\pm 6\end{pmatrix}\\
\sin(\theta_{kn}+\theta_{wt}) & \cos(\theta_{kn}+\theta_{wt}) & 0 & 11\sin(\theta_{kn}+\theta_{wt})+11\sin\theta_{wt}\\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.19]

Waist-to-Shank:

$$
{}^{Shank}_{Waist}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{wl}\cdot\cos(\theta_{wt}+\theta_{kn}) & \sin\theta_{wl}\cdot\cos(\theta_{wt}+\theta_{kn}) & \sin(\theta_{wt}+\theta_{kn}) & -(\cos(\theta_{wt}+\theta_{kn})\\
 & & & \cdot(\pm 6\sin\theta_{wl}+7)+11\cos(\theta_{kn})+11)\\
-\cos\theta_{wl}\cdot\sin(\theta_{wt}+\theta_{kn}) & -\sin\theta_{wl}\cdot\sin(\theta_{wt}+\theta_{kn}) & \cos(\theta_{wt}+\theta_{kn}) & \sin(\theta_{wt}+\theta_{kn})\\
 & & & \cdot(\pm 6\sin\theta_{wl}+7)+11\sin(\theta_{kn}))\\
\sin(\theta_{wt}) & -\cos(\theta_{wt}) & 0 & \pm 6\cos\theta_{wl}\\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.20]

Thigh-to-Waist:

$$
{}^{Waist}_{Thigh}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{wl}\cdot\cos(\theta_{wt}) & -\cos\theta_{wl}\cdot\sin(\theta_{wt}) & \sin(\theta_{wl}) & \cos(\theta_{wl})\cdot(11\cos(\theta_{wt})+7)\\
\sin\theta_{wl}\cdot\cos(\theta_{wt}) & -\sin\theta_{wl}\cdot\sin(\theta_{wt}) & -\cos(\theta_{wl}) & \sin(\theta_{wl})\cdot(11\cos(\theta_{wt})+7)\pm 6\\
\sin(\theta_{wt}) & \cos(\theta_{wt}) & 0 & 11\cdot\sin\theta_{wt}\\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.21]

Waist-to-thigh:

$$
{}^{Thigh}_{Waist}T_{\substack{left\\right}} = \begin{pmatrix}
\cos\theta_{wl}\cdot\cos(\theta_{wt}) & \sin\theta_{wl}\cdot\cos(\theta_{wt}) & \sin(\theta_{wl}) & -\cos(\theta_{wt})\cdot(\pm 6\sin\theta_{wl}+7)-11\\
-\cos\theta_{wl}\cdot\sin(\theta_{wt}) & -\sin\theta_{wl}\cdot\sin(\theta_{wt}) & \cos(\theta_{wl}) & \sin(\theta_{wt})\cdot(\pm 6\sin\theta_{wl}+7)\\
\sin(\theta_{wl}) & -\cos(\theta_{wl}) & 0 & \pm 6\cos\theta_{wl}\\
0 & 0 & 0 & 1
\end{pmatrix}
$$

[3.22]

All matrices show a small number of different trigonometrical functions, which are used in altered combinations. Substitutions can reduce the number of trigonometrical calculations by using pre-calculation and a look up table.

The matrices obtained in this section, allow to calculate the relative link's position for the biped robot structure and build a kinematics simulator. Dynamic is analyzed in section 4.3.

4.2.1.3 Kinematics simulator

After implemented the kinematics's representation of the "Dany walker" biped robot by *Denavit-Hartenberg* method and having analyzed the *walking sequence control algorithm* (described in section 5.6). A program to visualize and simulate the robot's movements during the walking process, was developed (Figure 4.9), in this program the different output angles (produced by the *walking sequence control algorithm*) were took to feed the robot's *D-H* model. Also some useful functions were added to the simulator allowing to modify different parameters like: size of the step, simulation time, maximum height of the step, etc. This tool was very helpful to obtain an efficient real-time walking routine which was finally implemented on the real robot. This program also was able to output the signals to the real robot, thus allows to prove in real time the walking routine. The simulator was developed to allow 6 degrees of freedom (degrees that intervene during the walking), the other 4 degrees of freedom concern only to the robot's balance. This kinematics simulator was able to simulate the walking pattern by using the robot's kinematics.

4.3 Dynamic model

In this section, two approaches to model the dynamic of the robot's structure are propose. The first, is the inverted pendulum to model the sagittal robot's movements. The second, is a neural network as a system identification to model the balance process (lateral robot's movements). Both models, together represent the robot's dynamics. With those models more information about the robot's dynamics can be used to simulate the robot's walking. Figure 4.10 shown the robot's dynamic process divided on two parts and the part that each approach model attends.

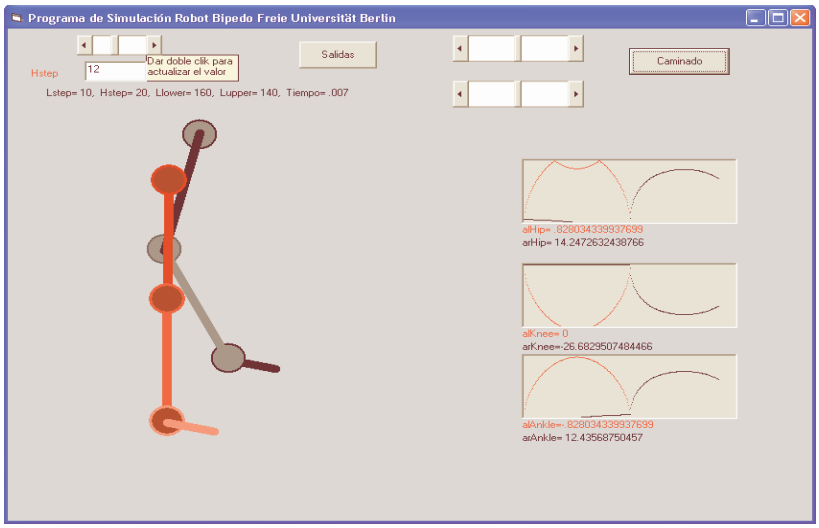

Figure 4.9: Simulator for the robot's kinematic.

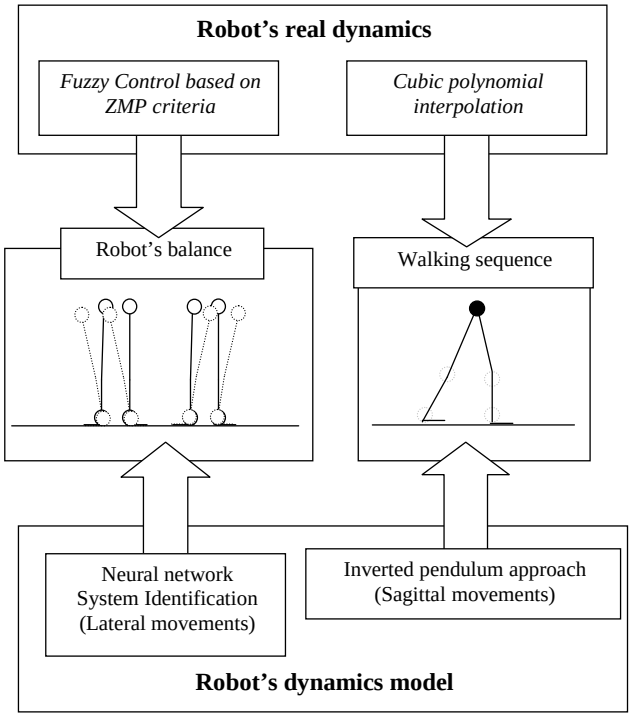

Figure 4.10: Biped robot dynamic approach models.

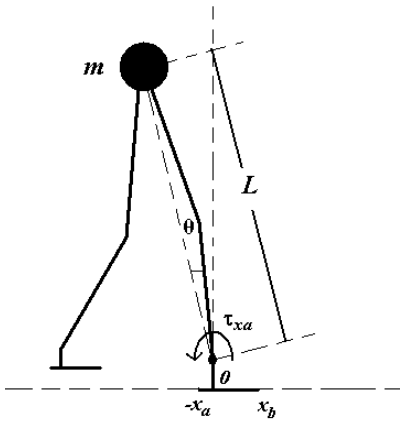

Figure 4.11: Biped robot's model obtained from the inverted pendulum model.

4.3.1 Inverted pendulum approach

This approach is used to model the sagittal dynamic of the biped's movements using the inverted pendulum model. Suppose that the biped robot has a mass point in the hip and the support knee is in a constant position, for these conditions the inverted pendulum model can be used to model the dynamics movements of the robot's structure [15][30]. The figure 4.11 shows the robot's model based on the pendulum model used to determine the ankle torque.

Where, L is the leg longitude, $\tau_{xa}(\theta)$ and $\tau_{xb}(\theta)$) are the maximum and minimum possible torques for the ankle in the sagittal plane. Thus, this relationship can be described as:

$$\tau_{xa}(\theta) = m(g + (\frac{\tau_{xa}(\theta)}{Lm} - g\sin(\theta))\sin(\theta) - \frac{v^2}{L}\cos(\theta))x_a \quad [3 \cdot 23]$$

where v is the hip velocity and $\frac{v^2}{L}$ is the centripetal acceleration.

The centripetal acceleration is smaller than the others components so it can be eliminated from equation 3.23, then the ankle torque can be redefined as:

$$\tau_{xa} = \frac{mgx_a(1 - \sin^2(\theta))}{1 - \frac{\sin(\theta)}{L}x_a} \quad [3.24]$$

$$\tau_{xb} = \frac{mgx_b(1 - \sin^2(\theta))}{1 - \frac{\sin(\theta)}{L}x_b} \quad [3.25]$$

This model approach is useful also to know the maximum torques in the ankle and make a decision on these motors value but, don't take on count the lateral forces add it by the balance.

4.3.2 Artificial neural network approach

In this section, a neural network used to model the nonlinear biped robot's lateral movements dynamic was implemented. The strategy was to use a neural network as a system identificator, in this case the system to be identified is the biped robot's lateral movements dynamics. A part of the lateral movements are generated by the fuzzy controller to correcting the ZMP. The ZMP dynamic, will be the parameter to be learned by the neural network. Some different training methods were used to compare the performance of the neural network to approximate the real robot's ZMP dynamic at walking. In all the different training methods, a back-propagation neural network architecture was choose. The following section describes the system identification process for the lateral robot's movements.

4.3.2.1 System Identification

System identification is the task of inferring a mathematical description, *a model* of the dynamics system from a series of measurements on the system. A typical system identification application is the simulation of a dynamics system.

Neural networks have been applied in the control of dynamics systems and its identification. The approximation capabilities of the multilayer perceptron make it an interesting option for modeling nonlinear systems [31].

In this thesis, to implement the system identification was necessary to train a neural network to represent the ZMP dynamic for the biped robot. The structure of the neural network plant model is given in the figure 4.12. The neural network plant model uses previous inputs and previous plant outputs to predict future values of the plant output.

The figure 4.13 shows the architecture used to train a back-propagation neural network to identify the biped robot's ZMP dynamic model. First, from the real biped robot (real robot's dynamics) the ZMP is obtained (ZMP(k)) and feed to the incremental fuzzy PD controller. The controller produces an output (lateral motors output) to correct the ZMP inside of the support polygon.

Thus, the inputs to the neural network are *M(k), M(k-1), M(k-2),* and *ZMP(k-1).*They are respectively, the output produced by the incremental fuzzy

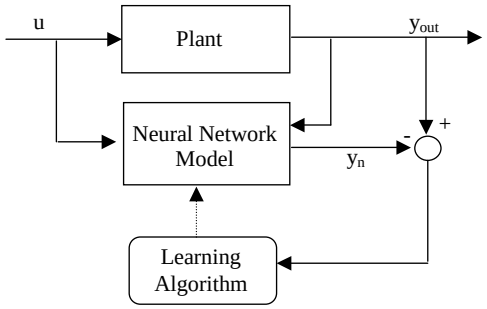

Figure 4.12: Structure of the neural network plant model.

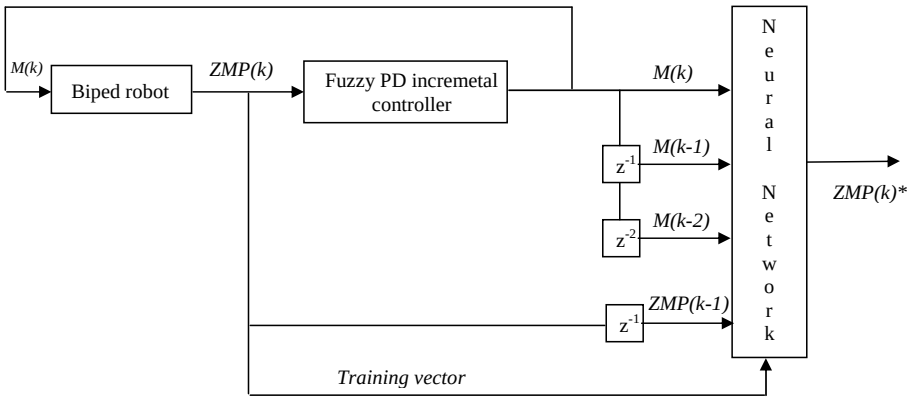

Figure 4.13: Architecture for the system identification neural network training.

PD controller (lateral motors output), this output delayed one time unit *(k-1)*, and the same output delayed two time units *(k-2)*. Finally *ZMP(k-1)* is the *ZMP(k)* delayed one time unit. The neural network output is *ZMP(k)**. This *ZMP(z)** is the ZMP learned by the neural network and should be very similar to the real ZMP (*ZMP(k)*).

In resume, to model the biped robot's balance dynamics, a back propagation neural network with four input neurons and an output neuron and with linear output activation function, was choose .

The network was trained offline in batch mode, using data collected from the real walking operation of the biped robot. Some different training algorithms were tested for the network training, each, obtain a different biped robot's ZMP dynamic model behavior.

4.3.2.2 Neural Network model's performance

It is difficult to establish a criteria to know which training algorithm will better describes the ZMP robot's dynamic at walking. However, the criteria used in this thesis will be a compromise between the velocity and economy of the algorithm. The algorithm's performance could depend on many factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and the application it self (discriminant analysis, regression, etc). The last, is the case of this thesis, since the goal is to find, means a neural network, a function approximation which model the biped robot's ZMP dynamic. Next, a graphical comparison of the biped robot's models performance obtained using some training algorithm is exposed.

Training algorithm comparison

This section present the results of a back-propagation neural network architecture used to identify the biped robot's ZMP dynamics dynamics. The neural network was training using different training methods. To test the performance of each of them, the controller's output at walking was feed to the neural network. Expecting that the neural network, now trained with the biped's ZMP dynamics, be able to predict the ZMP that the real biped robot will produce. In the following figures, a data set of ZMP real values obtained at walking, is compared with the ZMP produced by the neural network using different training algorithms.

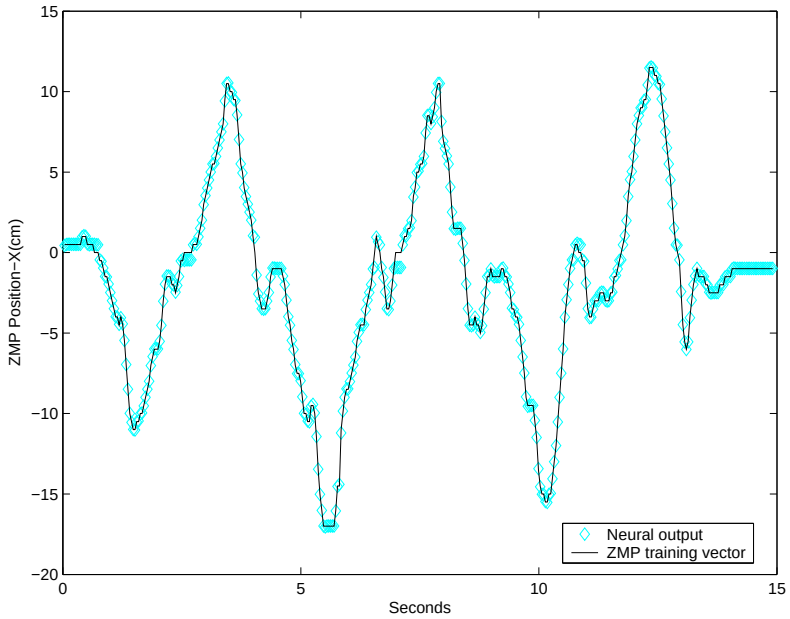

Figure 4.14: Levenberg-Marquardt algorithm's performance to model the biped robot's ZMP dynamics.

Levenberg-Marquardt training algorithm

In general, on function approximation problems, for networks that contain up to a few hundred weights, the Levenberg-Marquardt algorithm will have the fastest convergence. This advantage is especially noticeable if very accurate training is required. In many cases, Levenberg-Marquardt training algorithm is able to obtain lower mean square errors than any of the other algorithms tested. However, as the number of weights in the network increases, the advantage of the Levenberg-Marquardt training algorithm decreases.

Figure 4.14 shows the performance of the Levenberg-Marquardt training algorithm to model the biped robot's ZMP dynamics.

However, the storage requirements of Levenberg-Marquardt training algorithm are larger than the other algorithms tested.

Resilient Back-propagation training algorithm

The Resilient Back-propagation training algorithm is the fastest algorithm on discriminant analysis problems. However, in general it does not perform well on

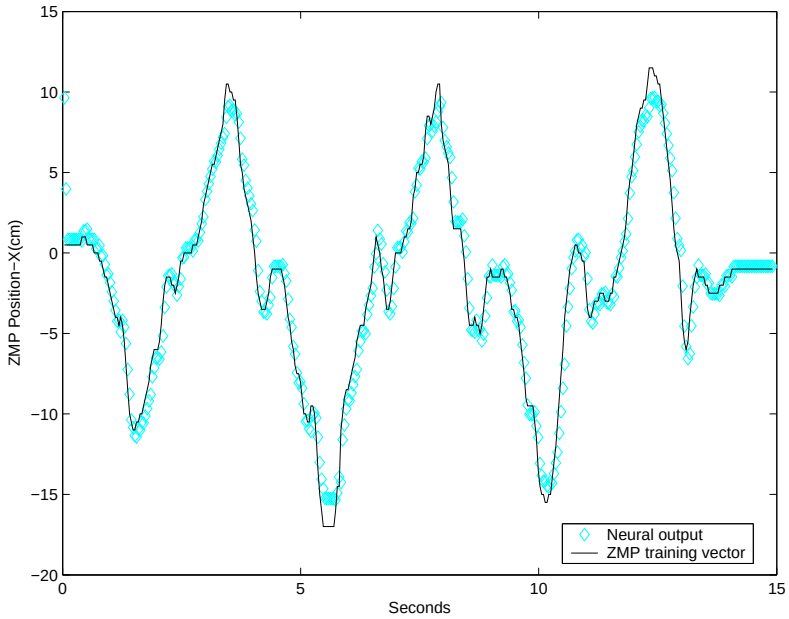

Figure 4.15: Resilient back-propagation algorithm's performance to model the biped robot's ZMP dynamics.

function approximation problems. Its performance also degrades as the error goal is reduced. The memory requirements for this algorithm are relatively small in comparison to the other algorithms considered. Figure 4.15 shows the performance of the resilient back propagation training algorithm to model the biped robot's ZMP dynamics.

Scaled Conjugate Gradient (SCG) training algorithm

The conjugate gradient algorithms, in particular Scaled Conjugate Gradient (SCG) training algorithm, seem to perform well over a wide variety of problems, particularly for networks with a large number of weights. The SCG algorithm is almost as fast as the Levenberg-Marquardt training algorithm on function approximation problems (faster for large networks) and is almost as fast as Resilient Back-propagation training algorithm on discriminant analysis problems. Its performance does not degrade as quickly as Resilient Back-propagation training algorithm. Figure 4.16 shows the performance of the scaled conjugate gradient training algorithm to model the biped robot's ZMP dynamics.

The conjugate gradient algorithms have relatively modest memory require-

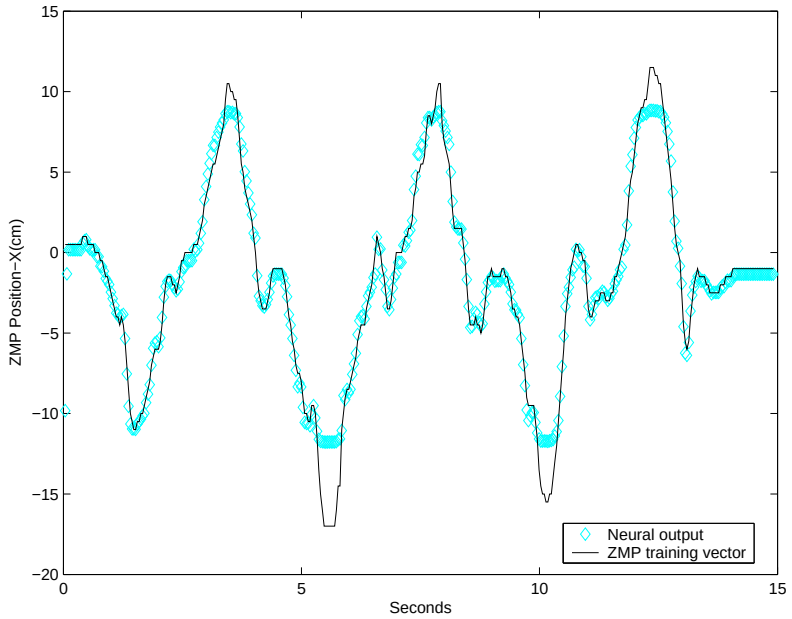

Figure 4.16: Scaled Conjugate Gradient algorithm's performance to model the biped robot's ZMP dynamics.

ments.

BFGS Quasi-Newton training algorithm

The quasi-Newton method that has been most successful in published studies is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS). The Quasi-Newton training algorithm performance is similar to that of Levenberg-Marquardt training algorithm. It does not require as much storage as Levenberg-Marquardt training algorithm, but the computation required does increase geometrically with the size of the network, since the equivalent of a matrix inverse must be computed at each iteration. Figure 4.17 shows the performance of the Quasi-Newton training algorithm to model the biped robot's ZMP dynamics.

One-Step Secant training algorithm.

Since the BFGS algorithm requires more storage and computation in each iteration than the conjugate gradient algorithms, there is need for a secant approximation with smaller storage and computation requirements. The one step secant

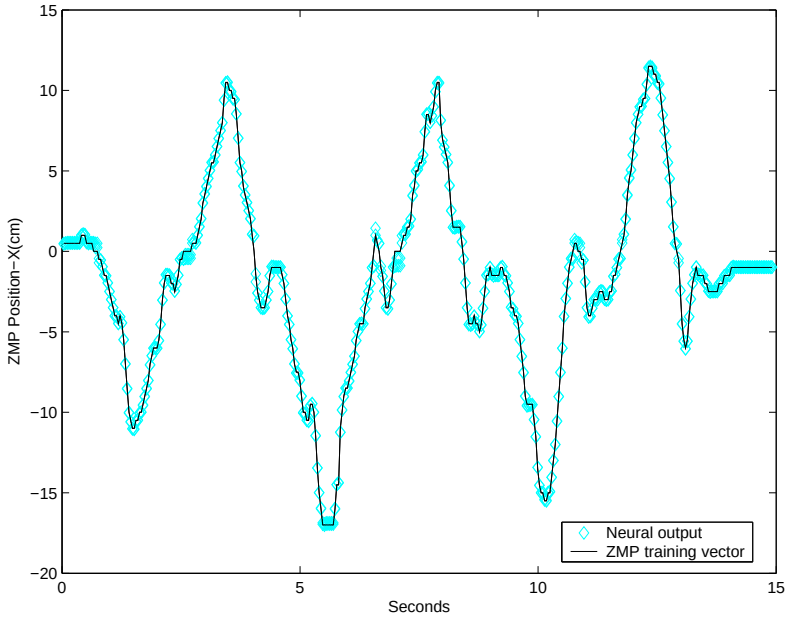

Figure 4.17: BFGS Quasi-Newton algorithm's performance to model the biped robot's ZMP dynamics.

(OSS) method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration, the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse.

This algorithm requires less storage and computation per epoch than the BFGS algorithm. It requires slightly more storage and computation per epoch than the conjugate gradient algorithms. It can be considered a compromise between full quasi-Newton algorithms and conjugate gradient algorithms.

Figure 4.18 shows the performance of the one-step secant training algorithm to model the biped robot's ZMP dynamics.

In summary, due the graphical performance comparison, the best training algorithm to model the robot's ZMP dynamics is the BFGS Quasi-Newton training algorithm. It is selected for its convenient relationship between economy and fast convergence.

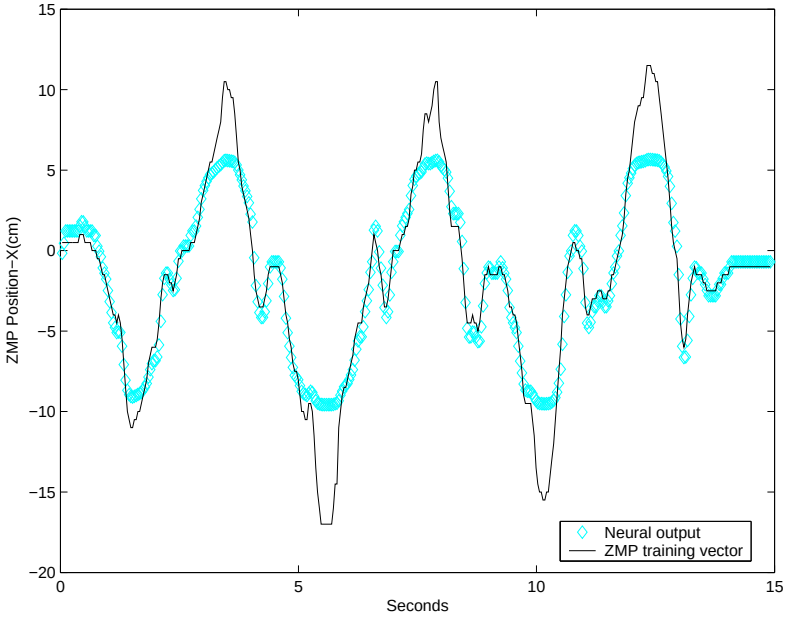

Figure 4.18: One-step secant algorithm's performance to model the biped robot's ZMP dynamics.