# 3D Graphical Simulation of Biped Dynamic Locomotion

Gonçalo PIRES
IST, Av. Rovisco Pais 1, 1096 Lisboa Codex, PORTUGAL, L34100@alfa.ist.utl.pt
Pedro LIMA
IST/ISR, Torre Norte, Av. Rovisco Pais 1, 1096 Lisboa Codex, PORTUGAL, pal@isr.ist.utl.pt

## Abstract

*In this paper we describe a graphical simulator of biped locomotion. The simulation is accomplished in two stages: given the step period and the step length, joint torques are first determined off-line to ensure postural and gait control, in the absence of external disturbances. Afterwards, the torques are passed to the dynamics simulation module, which generates on-line the joint angular positions and velocities. Those are used by a 3D graphical display module to produce a realistic animation of the biped robot. The simulator is intended for future development and test of biped locomotion controllers. The paper presents results for a 8 DOF biped robot, showing the effect of different step periods and lengths on the robot walking behaviour, as well as on the net power which would be required in a real implementation.*

## 1. Introduction

Walking robots may be used, with advantage over wheeled robots, to simulate human locomotion, to move along odd surfaces, or to implement active suspension through careful control of locomotion.

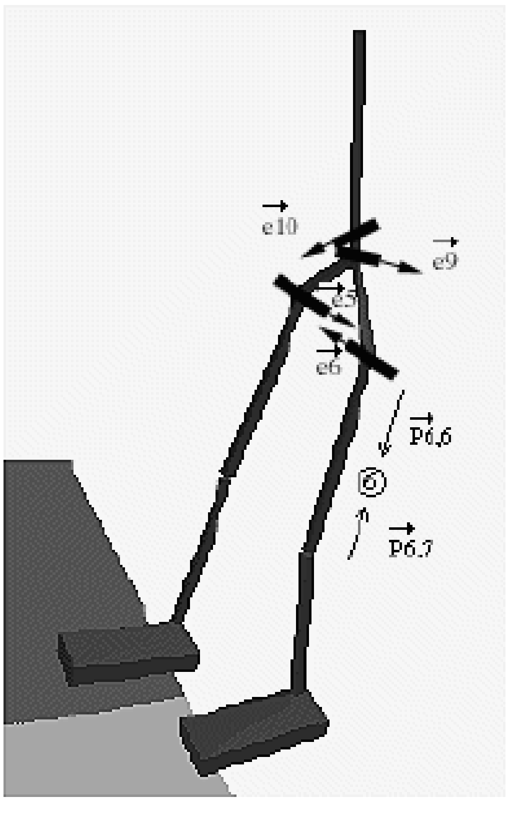

Figure 1 - Biped simulator image.

The first steps in this area were given by *static walking* robots. In those pioneer walkers, the projection of the robot centre of mass on the ground was always within the area inscribed by the feet, resulting in some restrictions to the structure and efficiency. Those robots had big feet and walked with small steps. A summary of those works can be found in Todd [2] and Raibert [3].

Since then, many walking robots have been built. Kajita, Yamaura and Kobayashi developed a simple model of linear differential equations, assuming massless legs. Afterwards, they built a prototype with very light legs that could walk over flat surfaces [4]. Another example of pioneer work in dynamic gait is the Kenkyaku-2, a robot that works in 2 dimensions, counting on wide feet to keep its sagittal balance [5]. Miller developed a robot controlled by a neural network that after some learning could walk with small steps, without any prior knowledge of its structure [6]. Prototypes with dynamic, three-dimensional (3D) gait over flat surfaces, such as BLR-G1 [7], and its latest version BLR-G2, were also developed. The robots WABOT WL - 10RD and its latest version WABOT WL - 12 were the first capable of climbing stairs and small slopes [8].

Nevertheless, none of the published prototypes is based upon an algorithm truly flexible, that allows the robot to walk over odd surfaces, turn, walk at any speed allowed by the actuators and start or stop at any moment.

This paper describes a 3D graphical simulator of biped locomotion, based on the dynamic model introduced by Vukobratovic *et al* [1]. The model allows any prescribed movement of a subset of the robot joints (e.g., the leg joints) and computes the motion of the remaining joints (e.g., the trunk joints) such that the whole system keeps its balance. The motion of prescribed joints can be determined from measures of human gait as in this work, even though it could also result from minimisation of energy consumption [9], or any other suitable method. The model assumes that the links are rigid and the friction forces between the ground and the feet are big enough to prevent the robot from slipping. The only restriction is the need for the robot to keep at least one foot on the ground. In Section 2 we summarize the fundamentals of Vukobratovic´s model.

Section 3 describes the implementation of the simulator. Results of the simulation of an eight degree-of-freedom (DOF) biped robot, with trunk but no arms, are presented in Section 4. The results show the effect of changing the step period and length on the robot walking behaviour and also on the net power which would be required by a real implementation.

## 2. Basic Concepts

In this section we summarize the work of Vukobratovic *et al* [1], used in our simulator.

The dynamic behaviour of a biped robot of $n$ joints can be described by the following equation:

$$\mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} = \mathbf{P} \qquad (1)$$

where **P** is the $n \times 1$ vector of the torques applied to the joints, **q** is the relative joint angles $n \times 1$ vector, **H** is a $n \times n$ massic matrix that depends on the systems state ($\dot{\mathbf{q}}$ and **q**) and on the geometric parameters, and finally **h** is a $n \times 1$ vector that contains the centripetal, Coriollis and gravitic forces which depends on **q** and on the geometric and dynamic parameters of the robot.

The torques **P** can be directly obtained from equation (1) and from the knowledge of the joint accelerations $\ddot{\mathbf{q}}$ and the system state **q** and $\dot{\mathbf{q}}$. Alternatively, knowing the torques **P** and the system state **q** and $\dot{\mathbf{q}}$, the accelerations $\ddot{\mathbf{q}}$ can be determined solving a second order non-linear differential equation system.

A third possibility is to start with the system state, part of the joint accelerations and part of the torques, and determine the remaining accelerations and torques. This intermediate solution is used to solve the dynamics of biped robots where some torques $\mathbf{P_0}$ and the corresponding initial states $\mathbf{q_x}(0)$ and $\dot{\mathbf{q}}_\mathbf{x}(0)$ are prescribed. In this work, the prescription of two torques, concerning the trunk two degrees of freedom, allows the robot to keep the frontal and sagittal balance. Two initial states (the two joints of the trunk) are determined to ensure that the gait is laterally and sagittally symmetric. The kinematics of the remaining joints (denoted as $\mathbf{q_0}$, $\dot{\mathbf{q}}_\mathbf{0}$ and $\ddot{\mathbf{q}}_\mathbf{0}$) is obtained from biometric measures of a normal human gait.

Therefore, with the prescription of the two torques $\mathbf{P_0}$, two accelerations $\ddot{\mathbf{q}}_\mathbf{x}$ will be determined at each instant. From the knowledge of all the accelerations, the $n$ - 2 unknown torques $\mathbf{P}_\mathrm{x}$ can be obtained. This approach to the bipeds dynamic problem solves the problem without simplifications or linearizations.

The algorithm to compute the dynamics of a forward step can be summarized as follows:

1. Computation of **H** and **h** matrices;
2. Prescription of the two torques $\mathbf{P_0}$ so that the robot keeps its frontal and sagittal balance;
3. **Do** Determination of the unknown joint accelerations **until** initial step state $\mathbf{q_x}(0)$ and $\dot{\mathbf{q}}_\mathbf{x}(0)$ is equal (or symmetric) to the final state $\mathbf{q_x}(T)$ and $\dot{\mathbf{q}}_\mathbf{x}(T)$, where $T$ is the step period;
4. Computation of nominal torques **P** to ensure forward steps at stabilized speed.

The details of the algorithm are described in the following two subsections.

### 2.1. Dynamics

In this subsection we cover step 1 of the algorithm, presenting a method to obtain the **H** and **h** matrices. The method will be described first for simple chains, and subsequently modified to include complex chains. Complex chains correspond to robot that besides legs, may have trunk and eventually arms.

#### 2.1.1. Simple chains

The robot is modelled as a set of rigid bodies, interconnected by one degree of freedom revolution joints.

A *link* of the robot is defined by the set of parameters $\mathbf{L}_\mathrm{i} = (\mathbf{C}_\mathrm{i}, \mathbf{D}_\mathrm{i})$, where $\mathbf{C}_\mathrm{i}$ represents the subset of kinematic parameters and $\mathbf{D}_\mathrm{i}$ the subset of dynamic parameters of link i. The mechanism kinematics is defined by the set $\mathbf{C}_\mathrm{i} = ({}^0_\mathrm{i}\mathbf{R}, \mathbf{P}_\mathrm{i}, \mathbf{E}_\mathrm{i})$, where ${}^0_\mathrm{i}\mathbf{R}$ defines the local frame orientation, with its origin in the mass centre of link i. $\mathbf{P}_\mathrm{i} = \{\vec{\mathrm{p}}_\mathrm{ik}\}$ is the set of position vectors of link i mass centre referred to the centre of joint k. $\mathbf{E}_\mathrm{i} = \{\vec{\mathrm{e}}_\mathrm{ki}\}$ is the set of rotation axes unit vectors of the joints that connect link i with link k, where i>k. When k=i–1, $\vec{e}_{ki}$ will be abbreviated to $\vec{e}_i$. Examples of position and $\vec{e}_i$ vectors are shown in Figure 1. The mechanism dynamics is defined by the set $\mathbf{D}_i = (m_i, \mathbf{J}_i)$ where $m_i$ is the mass of link i and $\mathbf{J}_i = \begin{bmatrix} J_{ix} & J_{iy} & J_{iz} \end{bmatrix}^T$ are the inertial moments of link i local frame axis.

A *kinematic pair* $\mathbf{P}_{ik}$ represents a pair of links $\{\mathbf{L}_i, \mathbf{L}_k\}$ interconnected by a joint at point $\mathbf{Z}_{ik}$, where i<k.

A *chain* $\Psi_n$ is the set of *n* kinematic pairs, $\Psi_n = \{\mathbf{P}_{ik}\}$ where $i \in N$, $k \in N$ and N={1, 2, ..., n}.

A *simple chain* is a chain where none of the links $\mathbf{L}_i$ $\forall i \in N$, is part of more than 2 kinematic pairs.

An *open chain* is a chain with at least a link $\mathbf{L}_i$ $i \in N$, that belongs to only one kinematic pair.

The method used to compute matrices **H** and **h** is divided in four stages. A comprehensive explanation of those stages can be found in Vukobratovic *et al.* [1].

In the first stage, the position and orientation of the local frames are computed with respect to the reference frame, assuming that all the joint relative angles are zero.

In the second stage, the position and orientation of the local frames are computed again with respect to the reference frame, but this time as a function of the joint relative angles.

In the third stage, the linear and angular accelerations of the links mass centres are computed in terms of the joint positions, velocities and accelerations.

Finally, in stage 4 the matrices **H** and **h** are obtained. First, the joint forces $\vec{F}_i^L$ and torques $\vec{M}_i^L$ that act on each link are computed from link *i* = 1 to *n*. Next, the joint forces $\vec{F}_i^J$ and torques $\vec{M}_i^J$ are computed from joint *n* to 1. Finally, the net joint torques **P** are determined and matrices **H** and **h** built.

The forces and torques that act on each link are composed of inertial and gravitic terms.

$$\vec{F}_i^L = \vec{F}_i^I + \vec{F}_i^G, \qquad \vec{M}_i^L = \vec{M}_i^I + \vec{M}_i^G \qquad (2)$$

Inertial forces and torques can be determined from the following equations

$$\begin{aligned} \vec{F}_i^I &= \begin{bmatrix} \vec{a}_{i1} & \cdots & \vec{a}_{ii} & 0 & \cdots & 0 \end{bmatrix} \ddot{\mathbf{q}} + \vec{a}_i^0 \\ \vec{M}_i^I &= \begin{bmatrix} \vec{b}_{i1} & \cdots & \vec{b}_{ii} & 0 & \cdots & 0 \end{bmatrix} \ddot{\mathbf{q}} + \vec{b}_i^0 \end{aligned} \qquad (3)$$

where the vectors *a* and *b* are obtained from Newton´s law and Euler´s dynamic equations applied to the accelerations determined in the 3rd stage.

Gravitic forces are applied to the mass centre of links only:

$$\vec{F}_i^G = m_i \vec{g}, \qquad \vec{M}_i^G = \vec{0} \qquad (4)$$

The joint forces and torques can now be computed from equations (2)-(4) by

$$\vec{F}_i^J = -\sum_{j=i}^{n} \vec{F}_j^L, \qquad \vec{M}_i^J = -\sum_{j=i}^{n} \left( \vec{M}_j^L + \vec{p}_{ji} \times \vec{F}_j^L \right) \qquad (5)$$

The forces and the torque component orthogonal to the rotation axis have no direct effect on the joint torques, but they may be useful modelling friction. However, this was not considered in our work. Hence, the *i<sup>th</sup>* component of the torques vector **P** is given by

$$P_i = \vec{e}_i \bullet \vec{M}_i^J \qquad (6)$$

where $\bullet$ represents the dot product. Rewriting (6), and using equations (2)-(5), we finally obtain:

$$\mathbf{P} = \mathbf{H}\ddot{\mathbf{q}} + \mathbf{h}, \quad \text{with}$$

$$H_{ik} = -\vec{e}_i \bullet \sum_{j=\max(i,k)}^{n} \left( \vec{b}_{jk} + \vec{p}_{ji} \times \vec{a}_{jk} \right) \qquad (7)$$

$$h_i = -\vec{e}_i \bullet \sum_{j=i}^{n} \left( \vec{b}_j^0 + \vec{p}_{ji} \times \left( \vec{a}_j^0 + \vec{F}_i^G \right) \right)$$

### 2.1.2. Complex chains

A *complex chain* is a chain where at least one link $\mathbf{L}_i$ $\forall i \in N$ is part of more than two kinematic pairs.

A complex chain can be subdivided in simple chains. For each chain, the first link is the common support to all the chains and the last link belongs to only one kinematic pair and is different of the terminal links of the other chains.

The algorithm described in the previous subsection can be applied separately to each simple chain, with the resulting complex chain torque $P_i$ equal to the sum of the torques of the simple chains that share the common joint.

## 2.2. Prescribed torques

The prescription of the two torques $\mathbf{P_0}$ will allow the robot to keep its balance. The initial state ($\mathbf{q_x}(0)$ and $\dot{\mathbf{q}}_\mathbf{x}(0)$) is computed so that the beginning of each robot step is equal or symmetric to its end. In this section we explain how to compute the two accelerations $\ddot{\mathbf{q}}_\mathbf{x}$ given those constraints. Then, all the system accelerations will be known, allowing the computation of all the torques using equation (1) - step 4 of the dynamics simulation algorithm.

### 2.2.1. Torques

The gait of bipeds has two phases: the single-support phase, where only one foot is touching the ground; and the double-support phase, where both feet are touching the ground. Nevertheless, should we consider that only one foot is supporting the robot weight at each instant, the model of the mechanism will be an open complex chain. This is physically possible because, when both feet are touching ground, the robot weight may be moved from one foot to the other, by changing the joint torques. This can be done almost instantly. Therefore, only open complex chains will be used to model the robot.

Since all the reaction forces under the foot have the same direction, they can be replaced by a resulting force $\vec{N}$ and a resulting torque $\vec{M}$ at the centre of the foot.

There is however a point, denoted as the *zero horizontal moment point* (ZMP) - shown in Figure 2 - where the reaction forces can be replaced by an equivalent force $\vec{R}$ and a vertical friction torque $\vec{M}_Z \neq 0$. The components of the torque parallel to the foot support plane are zero:

$$\vec{M}_X = \vec{0}, \quad \vec{M}_Y = \vec{0} \tag{8}$$

It is sufficient to keep the ZMP within the area inscribed by the feet, to ensure that the robot will show balanced motion.

When the ZMP position is prescribed, the biped looses, at each instant, two degrees of freedom, leading to the determination of $\ddot{\mathbf{q}}_\mathbf{x}$. Notice that, by prescribing the ZMP position, we are indirectly prescribing the torques $\mathbf{P_0}$.

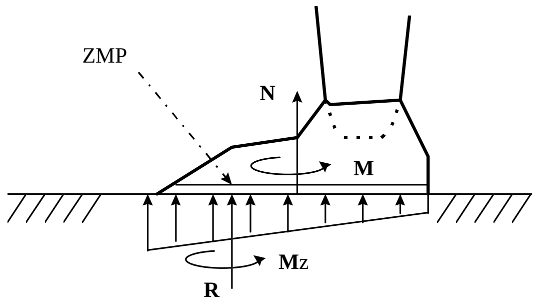

Figure 2 - Longitudinal distribution of the foot pressure and *zero horizontal moment point* (ZMP).

Considering the forces $\vec{F}^L$ and the torques $\vec{M}^L$ that operate in the mass centre of each link and the link i mass centre position with respect to the ZMP vector ($\vec{p}_i^{ZMP}$), equations (8) may be rewritten as

$$\vec{u}_k \bullet \sum_{i=1}^{n} \left(\vec{p}_i^{ZMP} \times \vec{F}_i^L + \vec{M}_i^L\right) = 0, \quad k = x, y \tag{9}$$

where $\vec{u}_k,\ k = x, y$ are unit vectors which span a plane parallel to the foot support surface. Separating $\mathbf{q_0}$ from $\mathbf{q_x}$, rewriting (9) in matrix form and using equation (2),

$$\begin{aligned}
&\mathbf{\Phi}\,\ddot{\mathbf{q}}_\mathbf{x} + \boldsymbol{\phi} = \mathbf{0}, \quad \text{where} \\
&\Phi_{jk} = \vec{u}_k \bullet \sum_{i=1}^{n_x} \left(\vec{p}_{X_i}^{ZMP} \times \vec{a}_{X_i j} + \vec{b}_{X_i j}\right), \; j = 1, \dots, n_x \\
&\phi_k = \vec{u}_k \bullet \sum_{i=1}^{n} \left( \begin{aligned} &\sum_{j=1}^{n_0} \left(\vec{p}_i^{ZMP} \times \vec{a}_{iZ_j} + \vec{b}_{iZ_j}\right) \cdot \ddot{q}_{0j} + \\ &+ \vec{p}_i^{ZMP} \times \left(\vec{a}_i^0 + \vec{F}_i^G\right) + \vec{b}_i^0 \end{aligned} \right) \\
&\vec{p}_i^{ZMP} = \vec{p}_{ii} - \vec{p}_{0i} - \vec{p}^{ZMP}, \quad k = x, y
\end{aligned} \tag{10}$$

where $\vec{p}^{ZMP}$ is the position of the ZMP with respect to the reference frame 0, $n_0$ is the number of joints of $\mathbf{q_0}$, $n_x$ is the number of joints of $\mathbf{q_x}$, **Z** is a vector whose elements are the indices of the joints belonging to $\mathbf{q_0}$, and **X** is a vector whose elements are the indices of the joints belonging to $\mathbf{q_x}$.

The solution of this system of differential equations is the acceleration vector $\ddot{\mathbf{q}}_\mathbf{x}$ of the dynamic compensation joints (between the legs and the trunk, in this work).

### 2.2.2. Initial state

The symmetry conditions of the initial and final step states must be checked to complete the implementation of step 3 of the dynamics simulation algorithm. The initial state will be linearly approximated until the condition

$$\mathbf{W}\begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0 = \begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_T \qquad (11)$$

is met. The matrix **W** is diagonal, with diagonal elements equal to +1 or −1 depending on if the joint position and velocity in the end of the step is equal or symmetric to the initial position and velocity.

Assuming that an approximate solution of the problem is known, a solution with a pre-defined accuracy can be obtained based on linear approximations around the final solution.

Assuming that small changes in the initial state produce small changes in the final state, we obtain

$$\mathbf{W}\left(\begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0 + \begin{bmatrix}\Delta\mathbf{q_x}\\ \cdots\\ \Delta\dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0\right) = \begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_T + \begin{bmatrix}\Delta\mathbf{q_x}\\ \cdots\\ \Delta\dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_T . \qquad (12)$$

Furthermore, for small changes,

$$\mathbf{UW}\begin{bmatrix}\Delta\mathbf{q_x}\\ \cdots\\ \Delta\dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0 = \begin{bmatrix}\Delta\mathbf{q_x}\\ \cdots\\ \Delta\dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_T , \quad \text{with}$$

$$\mathbf{U} = \frac{1}{10\cdot 2\cdot n_x}\left[\begin{array}{c:c}\dfrac{\Delta q_{x_i}(T)}{\Delta q_{x_j}(0)} & \dfrac{\Delta q_{x_i}(T)}{\Delta \dot{q}_{x_j}(0)}\\ \hdashline \dfrac{\Delta \dot{q}_{x_i}(T)}{\Delta q_{x_j}(0)} & \dfrac{\Delta \dot{q}_{x_i}(T)}{\Delta \dot{q}_{x_j}(0)}\end{array}\right] \qquad (13)$$

$$i = 1, \dots, n_x \qquad j = 1, \dots, n_x$$

Hence, from (13) and (12), and solving with respect to the correction on the initial state,

$$\begin{bmatrix}\Delta\mathbf{q_x}\\ \cdots\\ \Delta\dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0 = (\mathbf{U} - \mathrm{I})^{-1}\left(\begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_0 - \begin{bmatrix}\mathbf{q_x}\\ \cdots\\ \dot{\mathbf{q}}_\mathbf{x}\end{bmatrix}_T\right) \qquad (14)$$

In summary, the dynamics simulation algorithm works as follows: first, matrices **H** and **h** are computed using equation (7). Then, two torques $\mathbf{P_0}$ are prescribed to ensure that the robot keeps its sagittal and longitudinal balance. From equation (10) it is possible to determine the two unknown accelerations of the trunk joints. This operation is repeated, correcting the initial conditions through equation (14), until the initial state is equal or symmetric to the final state, as stated by matrix **W** in equation (11), ensuring the repeatability of the gait. After convergence, all joint positions, velocities and accelerations are determined, as well as the corresponding torques, using equation (1).

## 3. The Graphical Simulator

The graphical simulator was implemented in C++, running on a DEC AlphaStation under OSF1 Operating System. The SPHIGS graphical library was also used for image rendering. The program structure emulates a team producing a cartoon motion picture. Figure 3 shows the organisation chart of the team.

The `Director` is the responsible for all the operations. He receives the texts from the `Writer`, translates them into simple commands and distributes them by the `Drawer`, `Motion Coordinator` and `BopBop World` modules. The `Drawer` draws the robot and any world objects or details. The `Motion Coordinator` computes the motion of the guest star: the biped BopBop (see Figure 1).

`BopBop World` simulates the virtual world where BopBop lives, including external objects, rough surfaces, stairs, etc.

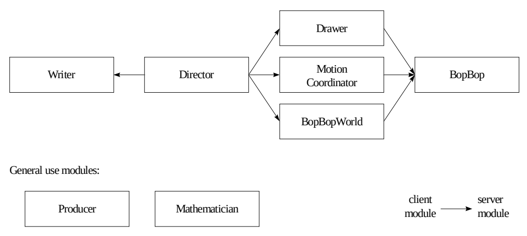

Figure 3 - Organisation chart

This module is not currently implemented. BopBop still lives in an ideal world, moving along flat surfaces with no objects around him.

Additional modules of general use are the `Producer`, which keeps statistics of CPU usage, and the `Mathematician`, which is needed for relatively complex arithmetic operations. All modules are implemented as objects which communicate with each other through mailboxes.

### 3.1. Writer

The `Writer` has the mission of collecting data from the user, classifying it as normal or urgent, packing it and mailing it to the `Director`.

### 3.2. Director

The main loop of the Director checks if there is any message in its mailbox, executes a command from the list of ready-to-execute commands and loops through messages sent sequentially to the `Drawer`, `Motion Coordinator` and `BopBop World` modules. Each of these three objects has an internal watch which is incremented after each cycle. The `Director` starts by triggering the `Drawer´`s cycle. Then, it triggers the `Motion Coordinator` until the watch of this module is greater than or equal to the `Drawer´`s watch. Finally, the `BopBop World´`s cycle is executed until its watch is greater than or equal to the `Drawer´`s watch, so that the three watches are synchronised. This allows the three modules to have different sampling rates.

### 3.3. Drawer

The `Drawer´`s job consists of building 3D images of the world and BopBop. To accomplish this task, the module emulates a virtual camera that can be translated, rotated, zoomed in and out. The camera may also follow the robot or keep a static viewpoint. The graphical information is passed to SPHIGS through poliethrons and 4x4 rotation/translation matrices.

### 3.4. Motion Coordinator

This module includes the computation of BopBop motion, such that postural and gait control are ensured, in the absence of external disturbances.

To compute BopBop moves, Vukobratovic´s algorithm was used (see Section 2). The dynamic compensation differential equation system was integrated by a Runge-Kutta method that can be found in reference [11]. Each move step is computed by Runge-Kutta formulas of 4th and 5th order with Cash-Karp parameters. In each integration step, two acceleration approximations are computed, one of the 4th order the other of the 5th order. In general, this algorithm is faster than constant step Runge-Kutta algorithms. However, there were small changes to the original algorithm in order to include 2nd order differential equations and save the acceleration values in constant intervals, despite the variable steps.

## 4. Results

The behaviour of an eight DOF robot walking step forward at constant speed was studied for different step periods and lengths. The joints are located at the ankle, knee and hip of the supporting and suspended legs, with another two joints at the waist, between the legs and the trunk, allowing trunk motion which is a combination of lateral and sagittal plane motions. There are two additional joints between the foot of the supporting leg and the floor to allow foot lateral and longitudinal motion. The kinematics were prescribed for the leg joints, while the waist joints move the trunk to control locomotion.

Results are shown for situations close to the values assumed for the prescribed kinematics: step length of 0.204m and step period of 0.75s. The *zero horizontal moment point* with respect to the foot centre (see Section 2) was established as in the following table,

| Time | Position |
|---|---|
| Until 20% of step | -0.02 m |
| From 20% to 75% | 0 m |
| After 75% | +0.035 m |

corresponding  to a “natural” sequence of the support foot motion, where body load is sequentially moved from the back (heel) to the front (toes) of the foot.

The plots of Figures 4 and 6 were obtained as if there were a pen at the top of the trunk recording a trajectory on a plane perpendicular to the sagittal and lateral planes and located at the waist height.

The results for a step length of 0.204m and different step periods are shown in Figure 4. In general, the longer the step period, the further the trunk has to move sideways, in order to keep its lateral balance.

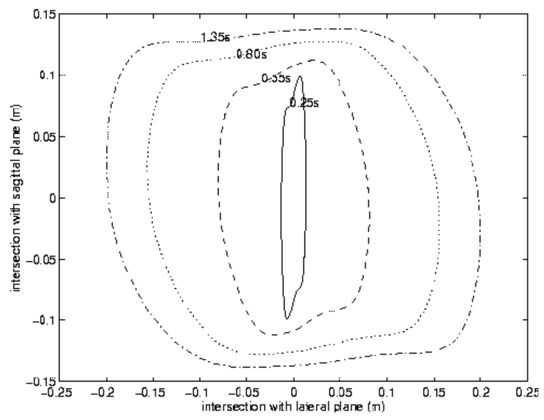

Figure 4 - Position of the top of the trunk for different step periods, given a step length of 0.204m.

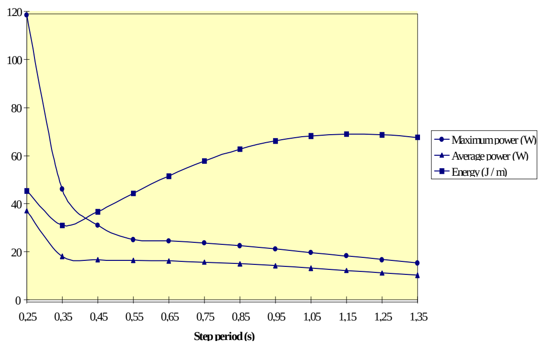

Figure 5 - Power/energy vs step period, for a step length of 0.204m.

Next, we studied the effect of the step period on the net power required by the joint actuators, given a constant step length (Figure 5). Generally, the slower the step, the less power it takes, but the energy spent per meter increases, since the robot must spend more energy to keep its balance.

The next investigation concerned the effect of the step length on the robot motion. The position of the top of the trunk is plotted in Figure 6 for a step period of 0.75s. In general, the greater the step length, the more the trunk has to move back and forth to keep its sagittal balance.

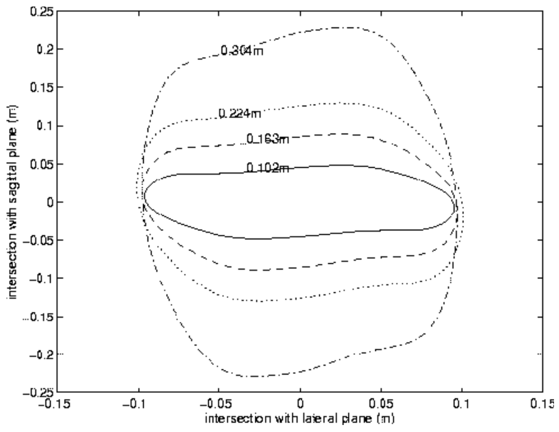

Figure 6 - Position of the top of the trunk for different step lengths, given a step period of 0.75s.

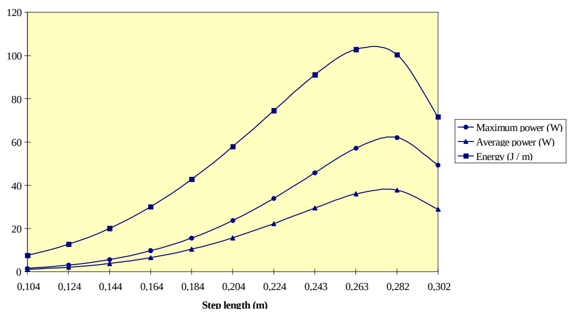

Figure 7 - Power/energy vs step length, for a step period of 0.75s.

Finally, we studied the effect of step length in the net power required by the joint actuators (Figure 7). In general, the larger the step, the greater is the required power. However, for step lengths greater than 0.270m the trend is reversed. This step length corresponds to an average trunk inclination of 45º. For larger inclinations (corresponding to larger steps) the benefits of the centripetal forces generated by the quasi-elliptic motion of the top of the trunk are more important, thus reducing the actuators effort.

## 5. Conclusions and Future Work

The 3D graphical simulator described in this paper is the first step towards the development and test of a biped robot. The simulation of the locomotion dynamics was based on an algorithm proposed by Vukobratovic and his associates. A postural and gait control planner module generates torques for the simulator.

The current implementation can only plan forward steps of fixed period and length for the robot motion. Future work will replace this by a more general planner, already developed [10], ensuring general trajectory planning, with way points described by frames. Trajectories will be based either on human data (as in the implementation here described), or as a result of the minimisation of some performance criterion, such as the power required. Results presented in this paper allow the comparison among the power requirements of different step lengths and periods, thus helping the future design of a real prototype. Should prescribed data come from antropomorphic measures, biped construction and control may lead to structures helpful to handicapped people.

Such a prototype will require the development, test and implementation of a locomotion controller. The controller will integrate information from feet sole pressure sensors and joint positions to compute the dynamic compensation of the trunk and keep the balance, despite external disturbances and uncertainties in sensor data and model parameters.

## 6. References

[1] Vukobratovic M., Borovac B., Surla D. and Stokic D., Biped Locomotion: Dynamics, Stability, Control and Application. Berlin-Heidelberg, Alemanha: Springer-Verlag, 1990.

[2] Todd D., *Walking Machines: An Introduction to Legged Robots.* New York, NY: Chapman and Hall, 1985.

[3] Raibert M., *Legged Robots That Balance.* Cambridge, MA: M.I.T. Press, 1986.

[4] Kajita S., Yamaura T., and Kobayashi A., *Dynamic walking control of a biped robot along a potential energy conserving orbit.* IEEE Trans. Robot. Auto., vol. 8, pp. 431-438, August 1992.

[5] Yamada M., Furusho J. and Sano A., *Dynamic Control of Walking Robot with Kick-Action*. Tokyo, Proc. of 85' International Conference on Advanced Robotics, pp. 405-412, 1985.

[6] Miller W., *Real-Time Neural Network Control of a Biped Walking Robot.* IEEE Control Systems, pp. 41-48, February de 1994.

[7] Sano A., Furusho J., *3D Steady Walking Using Active Control of Body Sway Motion and Foot Pressure*. Proc. of USA - Japan Symp. on Flexible Automation, pp. 665-672, Minneapolis, 1988.

[8] WASEDA ROBOT, Publ. Kato Laboratory, Waseda University, Tokyo, 1986.

[9] Ferreira P., Costa J., *Planeamento de Trajectórias de Robots na Presença de Obstáculos.* Actas do 1º Encontro Português de Controlo Automático, Volume II, pp. 185-190, 1994.

[10] Pinto P., Pires G., and Sequeira J. *Kinematic analysis of a Biped Robot.* Actas do 1º Encontro Português de Controlo Automático, Volume II, pags 203-206, 1994.

[11] William H. Press, William T. Vetterling, Saul A. Tenkolsky and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.