Stabilization of a Biped Robot with its arms

A Practical Approach

Florian Barras

May 2010

Semester Project

Biorobotics Laboratory

Ecole Polytechnique Fédérale de Lausanne

Supervisors:

Prof. A. J. Ijspeert (EPFL, Switzerland)

Sarah Degallier (EPFL, Switzerland)

Jesse van den Kieboom (EPFL, Switzerland)

Abstract

Nowadays, there is still no robust solution to stabilize a humanoid robot while it is walking; all proposed solutions work only approximately for certain assumptions. This can be easily understood knowing that there is still no sufficient criterion that implies the stability of a walking robot.

This project consists of finding a solution to keep a robot stable while it is walking implying the analysis of some stability criteria applied to bipedal locomotion. My research is mainly about how to use the arms efficiently to stabilize the robot. The reason I have chosen these parts of the body comes from natural observations (intuitively, the human babies tend to stabilize themselves notably with their arms when they first try to walk) and also from the fact that they are not directly involved in the bipedal locomotion (in comparison with the legs for instance).

My practical work can be separated in two main parts. Firstly, the construction of a stabilization algorithm handling the arms for quasi-static situations (when the robot does not move) and secondly, the same but for dynamic cases. For the latter, I also implemented (and optimized) an open-loop robot controller managing the walk of the robot.

Technically, for the first part, I implemented a controller to stabilize the HOAP-2 robot (a humanoid robot realized by Fujitsu) that was standing on an oscillating platform. For this, my strategy is to predict the next position of the center of mass (according to its velocity and acceleration) in order to evaluate the stability of the robot and to move the arms (if necessary) to adjust its position (according to some margins). For the second part, I implemented a CPG-based controller integrating a direct feedback control that managed the bipedal locomotion of the humanoid robot, namely the walk, on horizontal surfaces.

Contents

1 Introduction

Humanoid robots are far from the bipedal locomotion capability of humans. Indeed, currently there is still no robust solution to stabilize a humanoid robot while it is walking (for instance on complex surfaces or if the robot is sliding). This can be easily understood knowing that there is still no sufficient condition that implies the stability of a walking robot. Nevertheless, there is a necessary and sufficient criterion for static cases (when the robot does not move at all) and many different stability criteria for dynamic cases (that are valid only for specific assumptions).

1.1 Description and Goal

Globally, this project tends to find a solution in order to stabilize a robot while it is walking what implies among others the analysis of some stability criteria applied to bipedal locomotion.

Precisely, the work consists of implementing a controller (with feedback control) for the HOAP-2 robot[1] in order to stabilize it when it is walking.

In this project, I analysed the effect of the robot's arms on its stability and described how they can be used efficiently for stabilization. Firstly, I examined quasi-static situations (when the robot did not walk but was standing on a moving platform) and secondly dynamic cases.

Technically, for the first part, I implemented a controller to stabilize the HOAP-2 robot that was standing on an oscillating platform. For this, I used the *extrapolated projection of the center of mass*[2], namely *EPCM*, to evaluate the stability of the robot and the arms to adjust its position according to some margins. Then, I analysed quantitatively the effects of these parts of the body on the stability and optimized the use of them. For the second part, I implemented a *CPG-based controller*[3] integrating a direct feedback control that managed the bipedal locomotion of the humanoid robot, namely the walk, on horizontal surfaces. Finally, I used the results from the previous part and optimized them to stabilize the walking robot.

[1]A humanoid robot, see subsection 1.2 on the following page for details

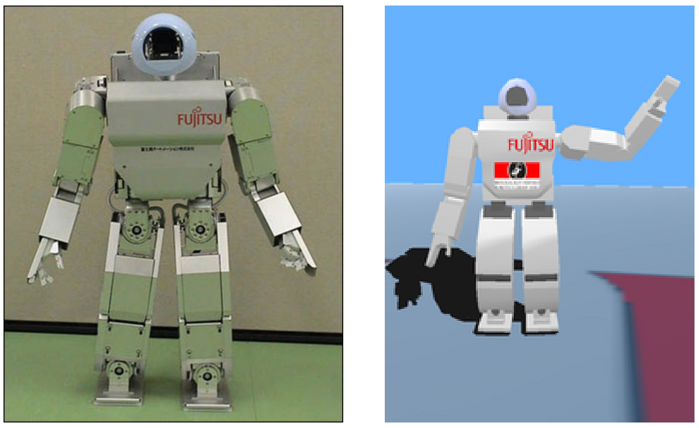
[2]A support point introduced in this report and defined in subsection 2.2 on page 6

[3]A controller using a high level dynamical system detailed in subsection 2.1 on page 5

In section 2 on the next page, I explain these terms and other strategies and in section 3 on page 11, I justify the reasons of these choices. In section 4 on page 15, I technically expose the construction of the robot controller and in section 5 on page 24, I optimize the controller for stabilization for the static and dynamic cases. Finally in section 6 on page 29, I conclude notably on the efficiency of the arms for stabilization.

1.2 Environment

The robot used for the experiments is the *HOAP-2* (an advanced humanoid robot platform manufactured by Fujitsu Automation). An illustration of this humanoid is given in the figure 1(a)[4].



(a) (b)

Figure 1 – *The real HOAP-2 robot on the left and the Webots model on the right*

The HOAP-2 robot was modelled in a previous project in ‘Webots (n.d.)’ (see figure 1(b)) which is a development environment used to model, program and simulate mobile robots. All simulations of this project are done in this commercial mobile robot simulation software developed by ‘Cyberbotics Ltd’. It includes notably a physics module that allows to simulate the gravity and any forces that are exerted on the robot. This is particularly useful to test the gait stability for instance.

A ‘Webots physics plugin’ created by J. van den Kieboom is also used in order to get the positions of several support points (notably the center of mass of the robot) and his software to construct networks of oscillators namely ‘CPG Studio’.

[4] Illustration taken from reneeteunissen.nl

2 Literature Review

In this section, I explain the different terms I use into details. First, I summarize the definition of a *CPG* and its specific meaning in this context. Secondly I explain how this mechanism can be integrated in a robot controller. And finally, I expose different stability criteria that exist in the literature and the one I introduce notably for this project.

2.1 Central Pattern Genetrators (CPGs)

This term comes from the neuroscience and precisely from the study of rhythmic motor patterns. Central pattern generators (CPGs) are neural networks that can produce rhythmic patterned outputs without the need of rhythmic inputs. A large range of behaviour (walking, scratching, breathing, ...) are typical rhythmic motor patterns produced by the nervous system (Ijspeert 2008). In the early 1900s, physiologists such as T. Graham Brown or C.S. Sherrington belonging to the Royal Society of London tried to explain the source of biological rhythmic productions allowing for instance the locomotion. In summary, there were two main hypothesis; C.S. Sherrington explained that the rhythms came from a chain of reflexes while Brown (1911) stated that they can be produced independently from sensory feedback, i.e. that they were centrally generated. The last hypothesis became more significant after the experiments of Wilson (1961); indeed, he showed that the locust nervous system can still produce rythmic patterns even if it was isolated from the insect. This was the first evidence that rhythmic motor patterns are indeed centrally generated for many organisms. Moreover, further work has shown that these CPGs (neural networks) were only small and autonomous in comparison with the whole nervous system (Hooper 1999).

In my work, I model the high-level behaviour of the CPG with a dynamical system based on oscillators. This mechanism (that can produce periodical and stable outputs) is used to create robust rhythmic outputs for the joints angles when the robot is walking. In this project, each oscillator creates independently a rhythmic signal (without patterned input) that depends on its parameters (such as amplitude or frequency). The network I use in the robot controller consists then of several oscillators that can be phase coupled together to produce rhythmic coordinated output signals.

2.2 Stability Criteria

Different criteria exist in the literature. For static cases (when the robot does not move), there is a sufficient and necessary criterion: A posture is denoted static stable if and only if the projection of the center of mass on the ground (PCM) is within the convex hull of the support polygon.

But for dynamic cases (when the robot is in movement), the problem becomes much more complex. At current date there is still no sufficient and necessary criterion, but there are many necessary criteria for some assumptions. Here I briefly present the useful support points and a well-known criterion, namely the ZMP.

2.2.1 Center of Mass (CoM)

The center of mass OG (in a coordinate system O) of a system of n particles (or segments) is defined as the average of their positions p_i, weighted by their masses m_i:

$$OG = \frac{\sum_{i=1}^{n} m_i * p_i}{\sum_{i=1}^{n} m_i} \tag{1}$$

The figure 2[5] illustrates the position of the CoM and its projection on the ground for two body postures.

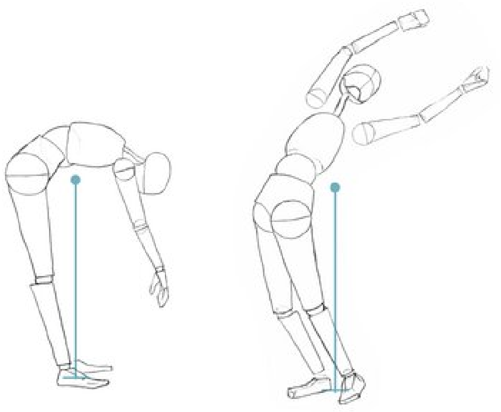


Figure 2 – *The projection of the CoM for two body postures*

[5] Illustration taken from hippie.nu

2.2.2 Extrapolated Center of Mass ($XcoM$)

Hof et al. (2004) introduced a condition for dynamic stability based on the extrapolation of the center of mass. They proposed an extension of the necessary and sufficient criterion of static cases for dynamic situations: 'The vertical projection of the CoM plus its velocity times a factor $\sqrt[2]{\frac{l}{g}}$ should be within the base of support, l being leg length and g the acceleration of gravity'. This vector quantity has been named $XcoM$. Concretely at time t:

$$XcoM(t) = PCM(t) + \sqrt[2]{\frac{l}{g}} * v_{PCM}(t) \approx PCM(t+1), \tag{2}$$

where l is the leg length and g is the acceleration of gravity. The CoM trajectory is then extrapolated in the direction of its velocity.

This result comes from the idea to consider the body reacting like one (or several) inverted pendulum(s) and therefore to compute the next position of the PCM according to this body model. The figure 3(a)[6] shows the inverted pendulum on a cart and the figure 3(b)[7] shows a schematic diagram of the inverted pendulum model representing a leg on the ground (with a single mass m, the center of mass CoM and the length of the leg l).

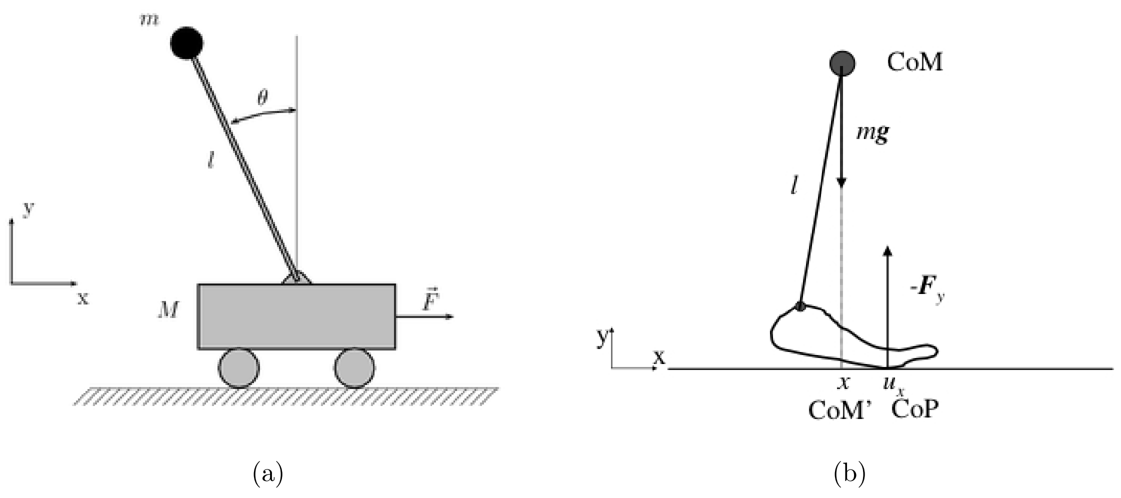


Figure 3 – *On the left, the inverted pendulum on a cart and on the right, a schematic diagram of the inverted pendulum model*

[6] Illustration taken from Wikipedia

[7] Illustration taken from Hof (2008)

2.2.3 Extrapolated Projection of the Center of Mass ($EPCM$)

This support point is introduced and used in this project. The $EPCM$ is a generic way to predict the next position of the CoM. Its definition is the following: The $EPCM$ of an object o is the extrapolation of its PCM according to the position, velocity and acceleration of the PCM. In simple words, the $EPCM$ is used to predict the next position of the center of mass in taking into account some information about its position, speed and acceleration. Concretely at time t:

$$EPCM(t) = PCM(t) + k_v * v_{PCM}(t) * \Delta t + k_a * \frac{1}{2} * a_{PCM}(t) * \Delta t^2 \approx PCM(t + \Delta t), \quad (3)$$

where PCM is the CoM's projection on the ground, k_v is the coefficient factor of the velocity, v_{PCM} is the velocity of the PCM, k_a is the coefficient factor of the acceleration, a_{PCM} is the acceleration of the PCM and Δt is the timestep.

The robot is then considered stable if its $EPCM$ remains in the convex hull formed by the contact points. Note that $XcoM$ is equal to $EPCM$ with $\Delta t = 1$, $k_v = \sqrt[2]{\frac{l}{g}}$ and $k_a = 0$; one assumption of the $XCoM$ is to consider the robot with a constant velocity (Hof et al. 2004). In section 5, experiments show that acceleration can increase the precision of the CoM's prediction and thus, the stability of the robot.

2.2.4 Center of Pressure (CoP)

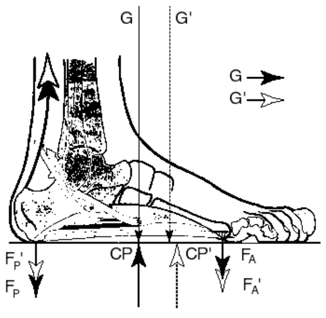
The center of pressure is the point where all the forces exerting on a body can be summed to form a single resultant force and where there is no moment about it. The illustrations of the CoP for two feet postures is shown in figure 4 on the next page[8].

For horizontal contact surfaces, it corresponds to:

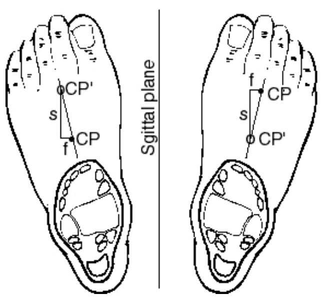
$$F_R^{CoP}|_{horizontal} = 0, \quad (4)$$

where this force is the horizontal resultant of the ground reaction forces exerted on the body at the CoP.

[8] Illustration taken from Pierre-Marie Gagey, Institut de Posturologie, Paris



(a)



(b)

Figure 4 – *The CoP for two postures of the feet*

2.2.5 Zero Moment Point (ZMP)

Vukobratovic and Juricic define in 1969 a specific support point (notably for stability) that is relatively near of the CoP (Popovic et al. 2005). They defined the Zero Moment Point (or ZMP) as the 'point of resulting reaction forces at the contact surface between the extremity and the ground'. Arakawa and Fukuda found a concrete definition in 1997:

$$F_R^{ZMP}|_{horizontal} = 0, \tag{5}$$

where this force is the horizontal resultant of the ground reaction forces exerted on the body at the ZMP. There is then only a non-zero vertical component at that point. This coincides with the CoP for horizontal surfaces.

Hirai and others introduced an alternative definition of this point in 1998: ZMP is also the 'point on the ground at which the net moment due to inertial and gravitational forces has no component along the horizontal axes' (ICC 1998). Concretely:

$$F_{I+G}^{ZMP}|_{horizontal} = 0, \tag{6}$$

where this force is the horizontal resultant of the inertial and gravitational forces exerted on the body at the ZMP. Again, there is then only a non-zero vertical component at that point.

The figure 5 on the following page[9] illustrates this support point with the forces mentioned above.

[9] Illustration taken from Popovic et al. (2005)

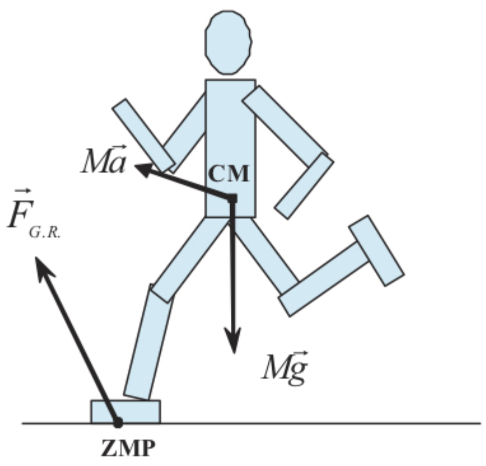


Figure 5 – *The position of the ZMP and different forces exerted on a human body*

For stability, this support point must also remain in the convex hull of the support polygon. This criterion is widely use as a stability criterion on horizontal surfaces (even if it is not sufficient).

3 Strategy

The goal is to construct a controller able to manage the walk of the HOAP-2 robot in a robust way, i.e. notably against exterior perturbations that can destabilize it. To do it, I used a CPG network with a feedback control. The latter checks if the robot is stable or not and moves some parts of the body, the arms, to stabilize it, according to the extrapolated projection of the center of mass (*EPCM*).

3.1 Arms

Concerning the use of parts of the body for robot stabilization, several people propose different techniques depending on the situation. The most common parts used in the bipedal stabilization process are first the legs that ensure the direct support of the body on the ground (Hauser et al. 2007). For the HOAP-2 robot, each leg is composed of five joints (one for each degree of freedom). As you can imagine, the problem becomes rapidly complex to control simultaneously these parts, since modifying one joint changes the position of another. Hauser et al. (2007) propose a new paradigm for balance control of humanoid robots based on biologically inspired kinematic synergies (that briefly simplifies the complexity of the model). Secondly the less common parts used for stabilization are the upper body joints (at the hip level, torso or arms).

My work consists essentially in exploring possible solutions to stabilize the HOAP-2 robot in using efficiently the arms like stabilisers. The reason I have chosen these parts of the body comes first from natural observations; intuitively, the human babies tend to stabilize themselves notably with their arms when they first try to walk or the tightrope walkers who generally also use the arms to stabilize themselves for instance. Secondly they are uncoupled from the joints directly involved in the walk of the robot (e.g. from the feet or legs) what facilitates the manipulations of them (any changes in the arms positions will not directly affect the locomotion). However they are not the only parts that humans use of course, but for a question of time I focused on them. For a future work, it could be interesting to explore the efficiency of other components like the hips or legs for stabilization.

3.2 CPG-Based Controller

As seen in the previous section, a CPG-based controller is a controller using a CPG network (modelled with oscillators in this project). The next subsection sheds light on the choice for this kind of mechanism that is particularly well-suited for cyclic patterns like walk. Finally, I explain my choice concerning the stability criterion for walking.

3.2.1 CPG Network

Rythmic Patterns First, we can notice that human *walking* is a cyclic motion bringing into play many muscles. The first ones are of course those of the legs. But there are also many other muscles playing a minor role in the locomotion that are indispensable (like the muscles involved in the stabilization; it could be the abdominals or the muscles of the arms as well). This kind of locomotion is a typical example of *rhythmic motor patterns* that is a part of many organisms (Hooper 1999).

Similarly for humanoid robots, walking involves almost the intervention of all joints (legs, hips, arms, etc.). Of course all these elements are not independent (because they are mechanically linked together) and then form a real complex mechanism to manipulate. For example, the figure 6[10] illustrates the skeleton of the HOAP-2 robot.

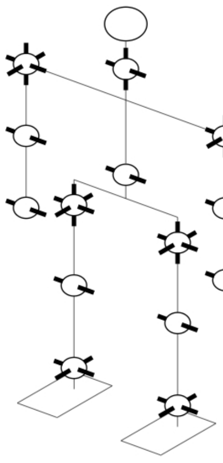


Figure 6 – *The skeleton of the HOAP-2 robot*

As we can see, each joint is connected to another forming what we call a skeleton. A CPG network in this case can be used to output the value of each articulation's angle

[10] Illustration taken from Biorobotics lab, EPFL

(without the need of specific rhythmic inputs) and can allow the coordination of the joints (with phase coupling for instance). An example of a CPG implementation with coupled oscillators for bipedal locomotion can be seen in Righetti (2006) (it has also been done for the HOAP-2 robot).

Easy integration of active dynamic elements CPGs facilitate the integration of dynamic elements. For instance, the feedback can easily modulate the phase, frequency and amplitude of every oscillator. The figure 7[11] illustrates this.

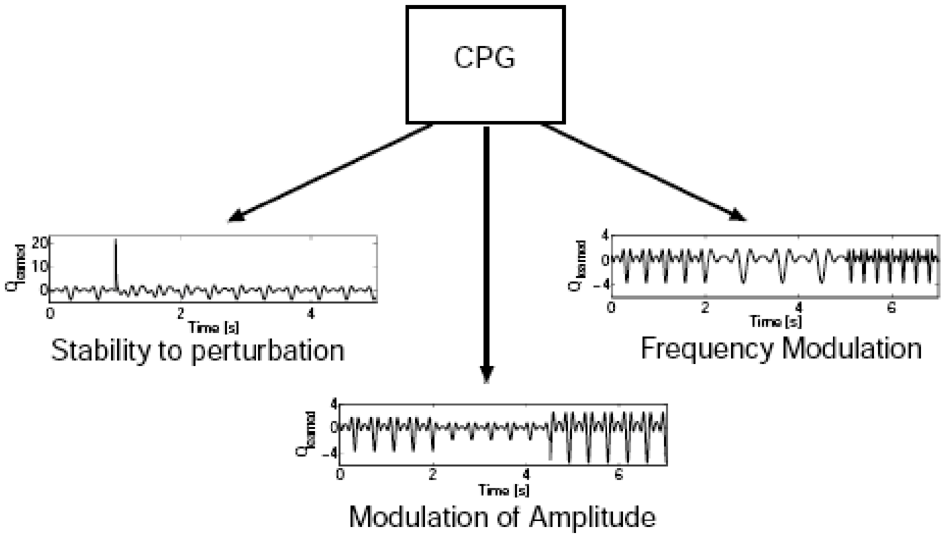


Figure 7 – *Some CPG capabilities*

CPGs modelled with oscillators are also robust to perturbations; these have a stable limit cycle implying self sustained oscillations. Thus, any perturbation in the system influencing the output signal would cause it to come back to the limit-cycle. Therefore, active elements introduced in the network will not change the signal forever thanks to the nice property of each oscillator that behaves like an 'attractor'. These elements can then be set to modify the original pattern during a certain period (that can be short or long) what is particularly useful if the signal needs some particular adjustments for a certain interval of time.

Smooth corrections Since each active dynamic element can be integrated in the CPG network, any adjustments of the feedback control produce smooth corrections to the

[11] Illustration taken from Biorobotics Laboratory, EPFL

output signal (thanks to the integration). For instance, this is particularly useful when one wants to adjust rapidly a part of the body without brutal changes (e.g. discontinuous input signals given directly to joints). In this case, all he needs is to integrate its feedback and the mechanism will act smoothly.

3.3 Stability Criterion

In the project, I mainly use the $EPCM$ calculated notably thanks to the angular velocity provided by the gyroscope of the robot. The first reason is that the projection of the CoM is involved in the necessary and sufficient condition for static stability. Moreover, it can be easily modified so that it becomes possible to apply it for dynamic situations. And the second reason is that the other support points are more complex to use practically as I explain below.

I use then the following criterion: if the $EPCoM$ is within the convex hull of the support polygon (with some specific margins), then the robot is considered stable or else it needs some adjustments. The difficulty remains in finding first the PCM (according to the mass distribution of the body) and secondly the extrapolation of the this point according to its inertia.

The reason I do not use the ZMP is because this point does not inform on the instability but only on the stability. Indeed, by definition the ZMP cannot be outside of the convex hull formed by the contact points. Therefore, if the robot is unstable, the point will be only on a side of the hull, giving thus no information on the instability; for instance, it could be at the extremity of the convex hull but the robot can be still stable. Moreover, this kind of situations often happen with the HOAP-2 robot, on account of its feet's rigidity. Since the latter are not flexible, each foot is almost always on one (or eventually two) of its four sides when the robot is walking. This also implies that the ZMP varies in a discontinuously way, making it really hard to use in this case.

4 Implementation

First, I describe the stabilization algorithm for static situations (when the robot does not walk) and secondly, I detail the construction of the CPG network that models the walking gait of the robot with the feedback control based on the stabilization algorithm. The project has been done in C++, an object-oriented programming language, with the appropriated 'Webots' libraries.

4.1 Static Situations

In this subsection, I present the stabilization algorithm that is based on the *EPCM* criterion and uses the arms of the robot to stabilize it (to adjust the position of the *EPCM*).

4.1.1 Stabilization algorithm

This algorithm contains two main parts. First the computation of the *EPCM* according to the velocity and acceleration of the *PCM* at each step. And secondly, the reaction function that moves the arms according to the *EPCM* and some thresholds (margins). Concretely in the 'Webots' main loop:

```
while (wb_robot_step(CONTROL_STEP) != -1) {
    ...
    // - Update the support points
    Points::update();
    // - Update the arms positions for stabilization
    Arms::updatePositions();
    ...
}
```

The class *Points* contains the information about the useful support points (*CoM*, *EPCM*, ...) and the class *Arms* manages the arms positions for stabilization.

Extrapolated Projection of the Center of Mass The prediction of the *PCM* follows the equation 3 and it is computed as follows:

```
// CoM prediction
Vector velocity = {0.0, 0.0, 0.0}, acceleration = {0.0, 0.0, 0.0};

// - Get the velocity
velocity.x = kv * Points::velocity.x;
velocity.y = kv * Points::velocity.y;
velocity.z = kv * Points::velocity.z;

// - Get the acceleration
acceleration.x = ka * Points::acceleration.x;
acceleration.y = ka * Points::acceleration.y;
acceleration.z = ka * Points::acceleration.z;

// - Compute the next supposed position of the CoM
EPCM.x = COM.x + velocity.x * TIME_STEP + acceleration.x * pow(TIME_STEP, 2);
EPCM.y = COM.y + velocity.y * TIME_STEP + acceleration.y * pow(TIME_STEP, 2);
EPCM.z = COM.z + velocity.z * TIME_STEP + acceleration.z * pow(TIME_STEP, 2);
```

The resulting point $EPCM$ is the approximation of the CoM's next position. If it is inside the support polygon formed by the contact points according to some margins, it indicates that the robot is stable, else the robot needs some adjustments to avoid the instability. Note that there are still two parameters that remains unknown, namely k_v and k_a. These coefficients depends mainly on the robot and they are found in section 5. The goal of these parameters is to calibrate the influence of the velocity and acceleration on the computation of the $EPCM$ in order to reduce the prediction error; the less is the difference between the $EPCM$ and the future PCM's position, the better are the parameters. Moreover, further experiments show that the optimal values of k_v and k_a do not change if the robot is moving differently from one to another. The next paragraph deals with the stabilization problem and explain how to use the arms to stabilize the robot, i.e. how to re-center the $EPCM$ inside the convex hull.

Arms Reaction Functions Once the $EPCM$ has been computed, we can see if it is far from ideal positions (where the robot does not need to move any more to stabilize itself). From now, I assume for simplicity that the contact points between the robot and the ground are only due to the feet touching it (i.e. the support polygon is formed with the contact points between the feet and the ground only). Let's consider the three-dimensional referential r of the robot, where the center is the ideal $EPCM$'s position when the robot is standing erect, the y axis corresponds to the normal to the surface of contact and it is oriented from the feet to the head, the z axis from the center to the front and the x axis from the center to the left of the robot. The referential is shown in figure 8.

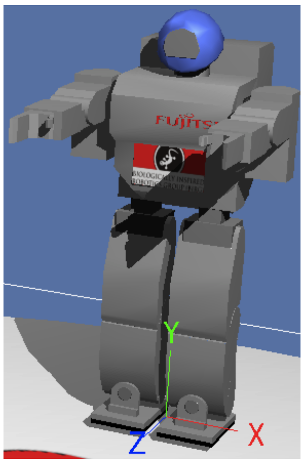


Figure 8 – *The referential r (relative to the robot)*

Thus, if the $EPCM$'s position does not match with the center, the arms will perhaps move to adjust its position according to some thresholds; we can imagine that the larger is the difference between the ideal and current positions of this point, the faster and/or more significant should be the corrections. Concretely the $EPCM$ should move in the following direction:

$$D = DPCM - EPCM = (0,0,0) - EPCM = -EPCM, \tag{7}$$

where $DPCM$ is the desired PCM's position that is the center of r.

To reduce the complexity, the arms will correct the position of the $EPCM$ following only and independently the z and x axis; the arms joints 1 and 2 that move respectively the arms in the z and x direction will only correct the $EPCM$'s position relatively to the z and x axis (and so independently from one to another). Thus, there are two reaction functions (one for each axis) that proportionally increase or decrease the angles of their joints depending on the $EPCM$ position. Concretely the angles of the arms joints 1 and 2 are:

$$\alpha_i = BASE + C_i * MAX, \tag{8}$$

where α_i is the angle value of the arms joints i, $BASE$ is the default value of the angle (when the robot is stable), MAX is the maximum value that can take the angle and C_i is a coefficient between 0 and 1 (defining how much the angle of the arms joints i must change to adjust the $EPCM$'s position). The figure 9 on the following page show optimal values for C_i depending on the difference D (see equation 7) for both the z and x axis (respectively C_1 with D_z and C_2 with D_x). These results come from several simulations explained in section 5.

4.2 Dynamic Situations

4.2.1 CPG Network

The role of the mechanism I use is to provide for each joint of the robot the appropriate value of its angle so that the HOAP-2 robot can robustly walk. Note that the joints angles for this bipedal locomotion were already given by a pre-existing data set (given in the form of a set of points). The latter contains the angles of each joint during a normal walking gait cycle (when the robot is not subjected to perturbations). The figure 10 on page 20[12] illustrates this type of cycle for a human.

The network of oscillators was constructed in the software 'CPG Studio' (mentionned in section 1.2 on page 4). Each oscillator is associated to a joint of the robot and outputs the value of its angle. The next paragraphs explain how they does it.

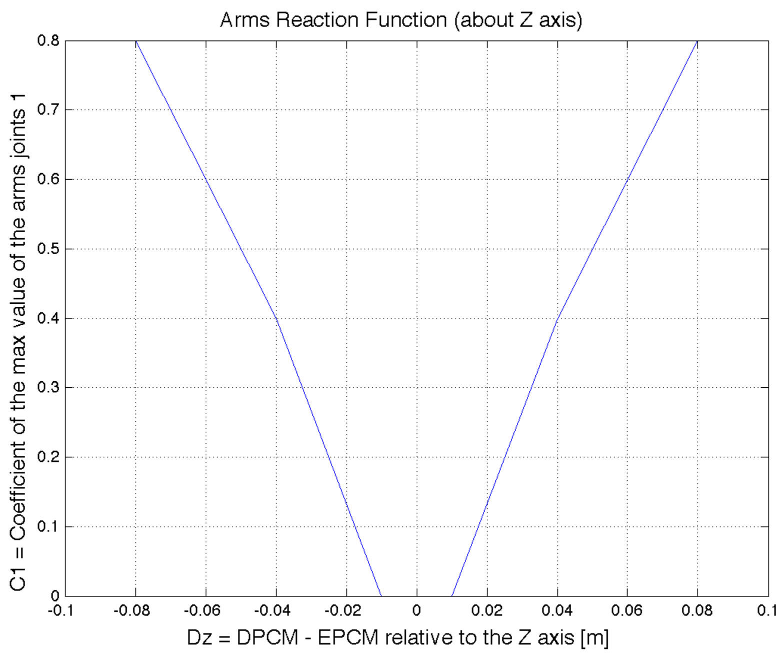
Update Functions Each oscillator updates the angle's value of its corresponding joint i in the following way:

$$\dot{y_i} = \alpha \cdot (f_j(\theta_i) - y_i) + \frac{df_i}{d\theta_i} \cdot \dot{\theta}_i + C_i \tag{9}$$

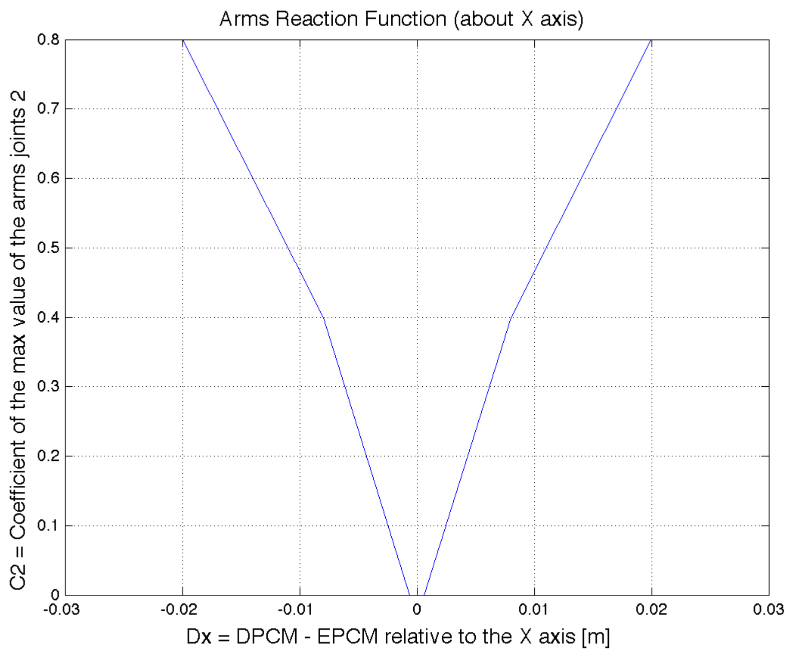
where y_i is the angle's value of the joint (the output), α is the convergence factor, f_i is the polynomials function of the joint (outputs the desired value for its angle), C_i is an arbitrary perturbation in the system and θ_i is the phase of the oscillator. The latter is defined as follows:

$$\dot{\theta}_i = \omega + \sum_j sin(\theta_j - \theta_i - \phi_{ij}), \tag{10}$$

[12] Illustration taken from Kevin Deluzio, Queen's University



(a)



(b)

Figure 9 – *The optimal values for the arms joints coefficients according to the position of the EPCM* $(= -D)$

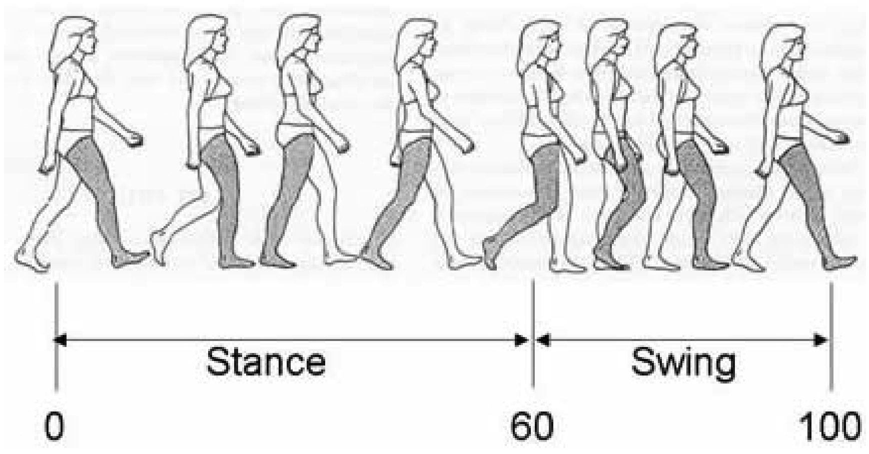


Figure 10 – *An example of the walking gait cycle*

where ω is the angular frequency and ϕ_{ij} the phase bias of the coupling between oscillators i and j.

In equtation 9, the first term is the difference between the desired and the current values (times a factor α). This kind of 'attractor' is used to converge to the desired value (namely $f_j(\theta_i)$) and α indicates how fast it does it (the larger is this value, the more important is this term and then the convergence to the desired value). The second term is the derivative of the polynomials function $f_i(\theta_i)$ with respect to θ_i. This part is the driver, or feed-forward term, used to update the previous value of the joints angle.

Polynomials Functions For each timestep t, the data-set provides an angle's value v for each legs joint. Let be σ_i the map between t and v for the joint i. For each joint i of the robot, I associate a function f_i that uses some polynomials $p_{i1}(t), ..., p_{in}(t)$ to return the angle's value of i at time t. The polynomials $p_{i1}(t), ..., p_{in}(t)$ interpolate the angle's values of the joint i given by $\sigma_i(t)$. The polynomial interpolation is made with the least squares method. Given m, the latter finds the optimal polynomial p_m of degree m that minimizes the sum S of squared residuals $r_1, ..., r_n$. Concretly:

$$\min S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (v_i - p_m(t_i))^2, \tag{11}$$

where r_i is the difference between the desired value (t_i, v_i) and the value obtained with the polynomial $p_m(t_i)$. The figure 11 on the next page shows the result of this method applied to a set of point; the resulted polynomial functions minimize the sum s of the squared residuals. Note that there is a different polynomial function for each degree and the higher is it, the less is s.

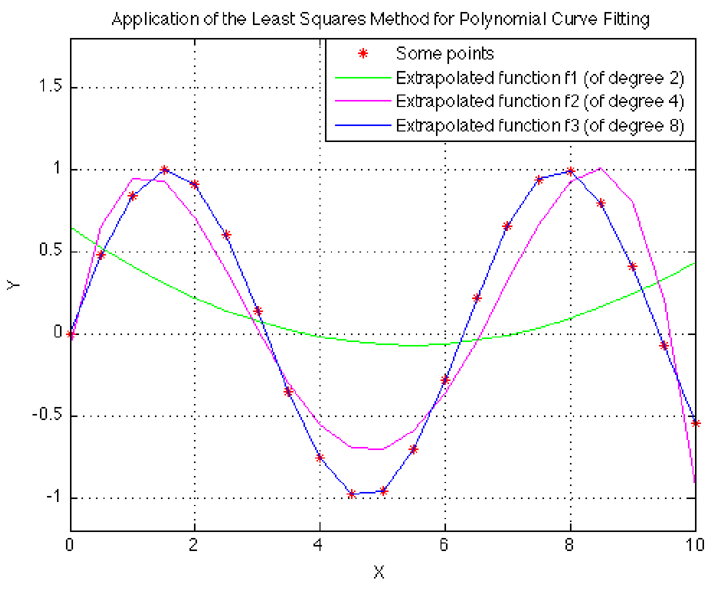
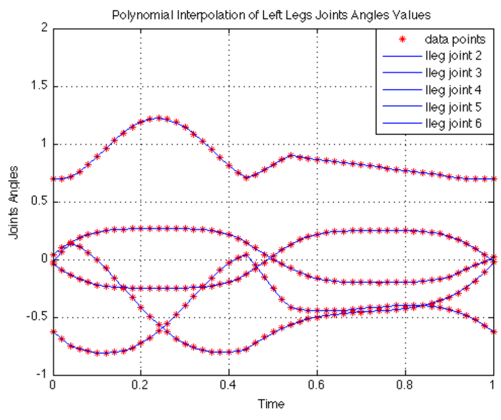


Figure 11 – *Polynomial curve fitting using the least squares method*

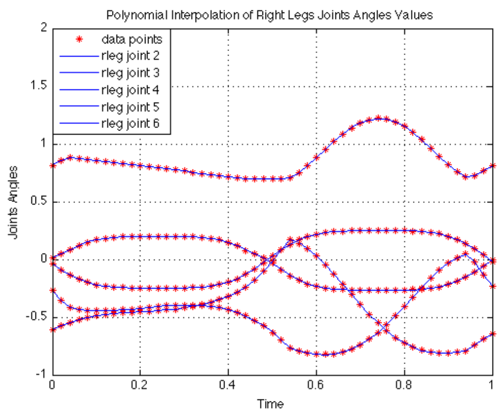
Therefore, in order to find a function f_i that interpolates the map σ_i with several polynomials, the method is applied several times and each time to a different piece of σ. Thus, the function f_i will be made up of several polynomials, each extrapolating a piece of σ_i. The reason for which I use several polynomials is to decrease the degrees of them in still keeping a good approximation. The next figures shows the interpolation of the angles values of the robot's joints following this method.

Initialization Since the initial posture does not belong to the walking cycle, I use the convergence factor and some initial values to adapt the movements of the robots so that it can meet up with the walking cycle in staying stable. Moreover, I adjusted the positions of the arms to increase the stability. The figure 14 on the following page shows the angles values of the arms obtained from the CPG when the robot is walking.

Results The output signals for the legs generated by the CPG network can be seen in the figure 13 on the next page. These correspond to the angles values of the legs joints during the walk of the HOAP-2 robot. We can see that these signals meet up with

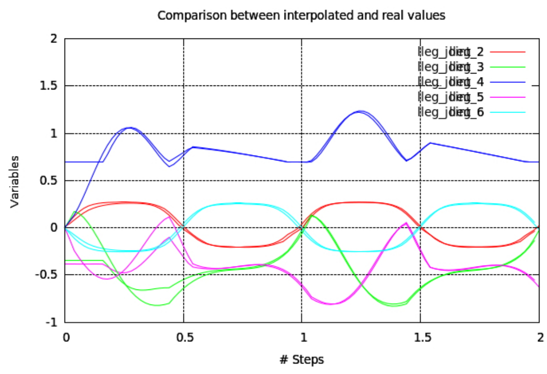


(a)

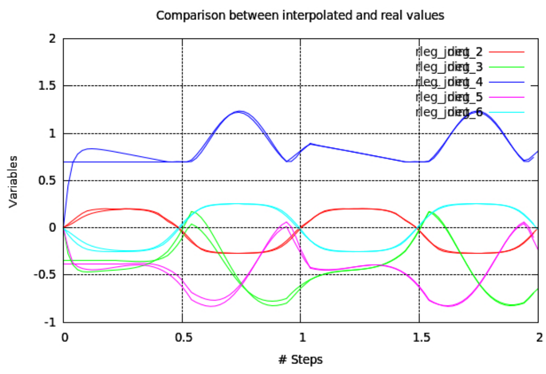


(b)

Figure 12 – *Interpolation of the joints angles values of the left and right legs*

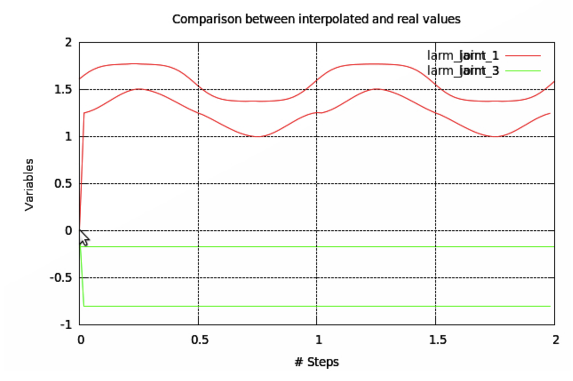


(a)

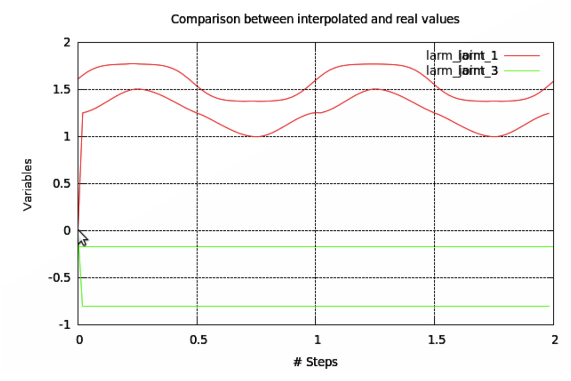


(b)

Figure 13 – *Values of the left and right legs joints angles during the walk of the robot*



(a)



(b)

Figure 14 – *Values of the left and right arms joints angles during the walk of the robot*

the original values and varies periodically. The same for the arms can be seen in the figure 14 on the preceding page (except that there exist an offset to stabilize the robot as explained before). The graphs also show the comparison between the values obtained from the original pattern (real values) and those obtained from the CPG network (interpolated values). As expected by construction, the angle's values of every joint varies in a rythmic pattern.

5 Research and Experimentation

My research and experiments tend to find optimal values for the parameters involved in the calculation of the *extrapolated projection of the center of mass (EPCM)* and in the *arms reaction functions*.

5.1 Extrapolated Projection of the Center of Mass

The $EPCM$ is useful to predict the stability of the robot and also the direction of the corrections. This prediction is configurable with two parameters namely k_v and k_a (that are respectively the coefficients of the velocity and acceleration of the PCM). In this subsection, I explain the method and discuss the results.

5.1.1 Goal

The goal is to find optimal values for k_v and k_a, i.e. that minimize the difference between the $EPCM$ and the next PCM. Concretely:

$$\min \delta = \min |EPCM(t) - PCM(t + \Delta t)| \qquad (12)$$

5.1.2 Method

Since this must be optimized for all scenarios (when the robot is walking, falling, ...), I created a moving platform in 'Webots' (that can oscillate in every direction) on which the HOAP-2 robot is standing and trying to stay stable. For this, it has only the arms to stabilize itself. Then, I programmed a set of n simulations (tests) with different k_v and k_a values and compute for each test the average of δ, namely the prediction quality; the less is δ, the better is the PCM's prediction.

5.1.3 Results

The figure 15 on the next page shows the results of different simulations, notably when the robot is falling.

As we can see, the velocity is very important in order to predict the PCM's position. Moreover, we can observe that the acceleration brings additional information that is not

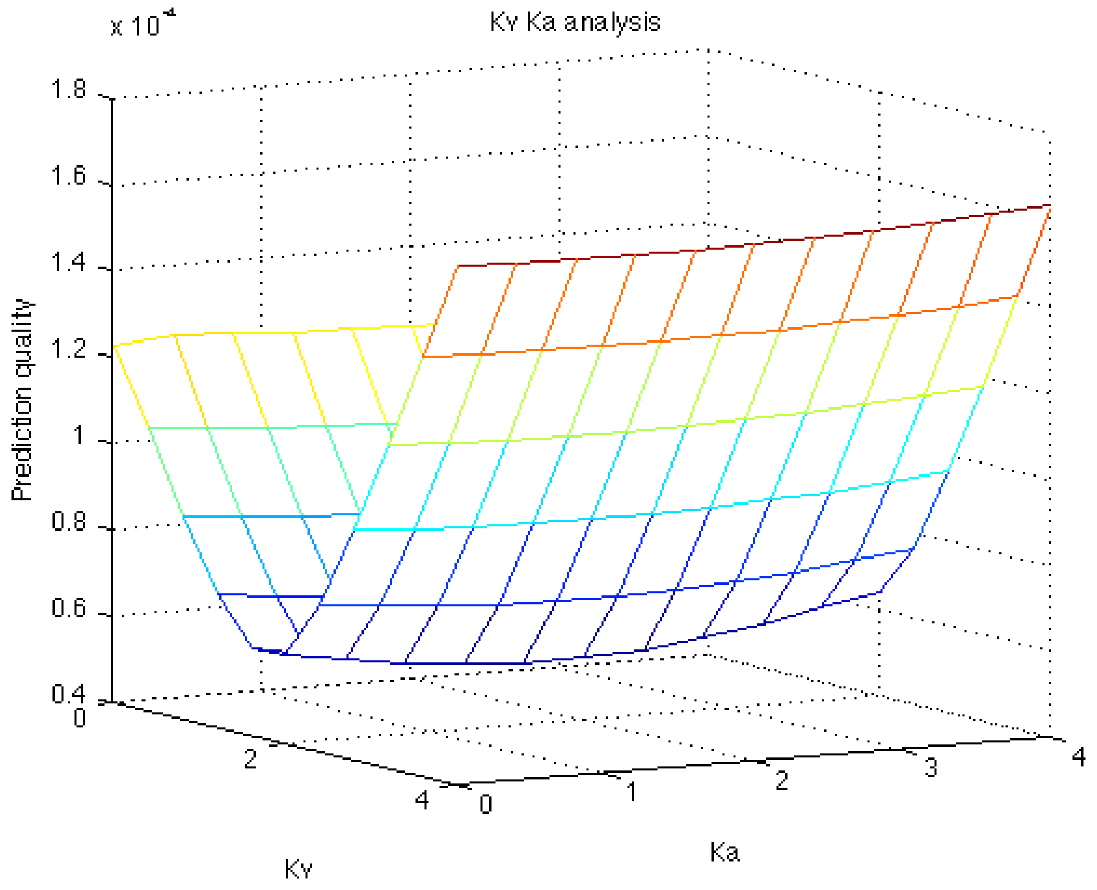


Figure 15 – *PCM's prediction quality according to the parameters k_v and k_a*

negligible. And finally the the shape of the graph clearly shows that there exists a unique optimal solution that is, for this case, when $k_v \approx 1.65$ and $k_a \approx 1.40$.

5.1.4 Conclusion

Once the calibration has been done, the *EPCM* can work as a good stability indicator for dynamic situations and can of course be used as a sufficient criterion for static cases. Moreover, the acceleration does not bring any noise but a real information about the next *PCM*'s position. Comparing to the *XCoM* that does not take into account the acceleration, it seems to be more precise. And finally, the parameters are definitively fixed for all scenarios in the moving platform for the HOAP-2 robot.

5.2 Arms Reaction Function

First note that the mass of the robot's hands has been increased by a factor of 5 in order to allow a better stabilization with the arms. Now let's consider the difference D_i defined in equation 7 on page 17 and the thresholds T_1, T_2 and T_3 that are the parameters of the arms reaction function f_i. This function takes as input D_i and according to the thresholds returns the coefficient of the arms joints i, namely C_i (see equation 8 on page 17 for details). This coefficient (defined between 0 and 1) determines how much the angle must be changed in order to increase the stability. Each threshold is associated to a fixed value (a coefficient value) and represents a value that can take D_i (that is not fixed). The function f_i interpolates linearly the thresholds to be continuous.

5.2.1 Goal

The goal is to find the optimal thresholds of the arms reaction functions for both the z and x axis, i.e. to find at which value D_i the function must output the corresponding C_i.

5.2.2 Method

Similarly to the previous experiment, the robot was standing on the oscillating platform and a set of simulations (tests) has been made separately for both the z and x axis (once the platform was oscillating around the x axis and once around the z axis). Each threshold has an associated value, respectively 0.2, 0.4 and 0.8 and each can take any D_i values. And for instance, if D_i is exactly in the middle of T_1 and T_2, $f_i(D_i)$ will output $\frac{(f_i(T_1)+f_i(T_2))}{2}$ (since $f_i(T_1)$, $f_i(T_2)$ and $f_i(T_3)$ are known and f_i linearly interpolates the thresholds). The thresholds quality (or fitness function) takes into account the amount of time the robot has been stable and the average variation of the PCM (to know if the robot was constantly trying to stabilize itself for instance). The amplitude of the platform's oscillations could also be changed to reach critical cases (when the robot must absolutely use some parts of its body to stabilize itself else it falls).

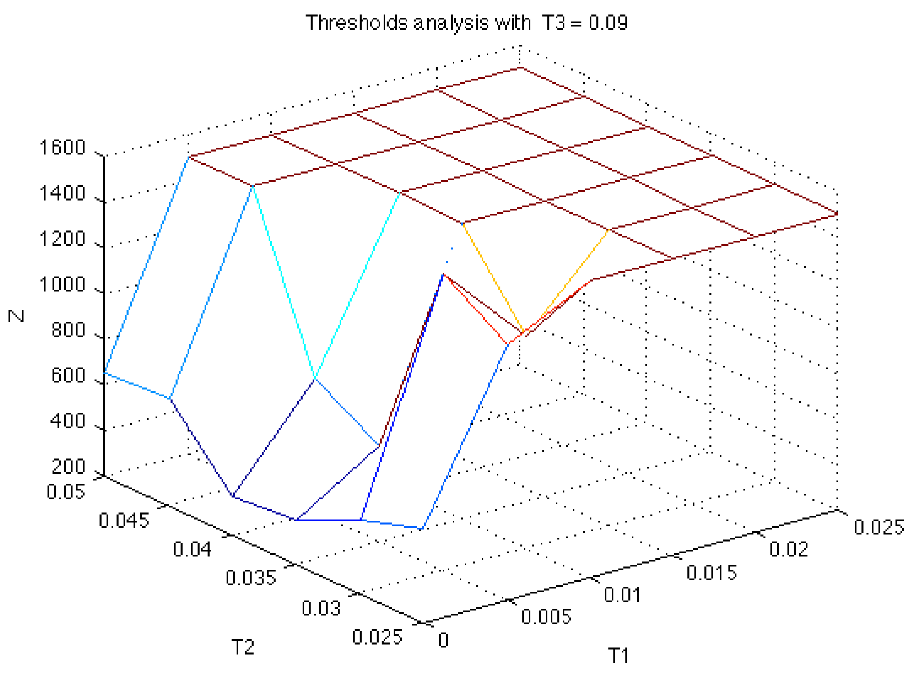
5.2.3 Results

The results for an extreme case can be shown in the figure 16 on the following page. It indicates the score of the arms reaction functions correcting respectively the position of the $EPCM$ along the z and x axis. The maximal value (1000) along the vertical axis corresponds to the maximal number of timesteps per simulation (test); if the robot falls, the test is immediately stopped and the time is recorded. So if the robot successfully stabilizes itself during the simulation, it gets a score of 1000. We can see that good thresholds values are crucial for the stability of the robot. If the robot is very sensible to PCM changes, it can risk to constantly try to stabilize itself implying a constant instability and at the opposite, in extreme cases, if it is not sensible at all, it can simply fall before remarking the instability.

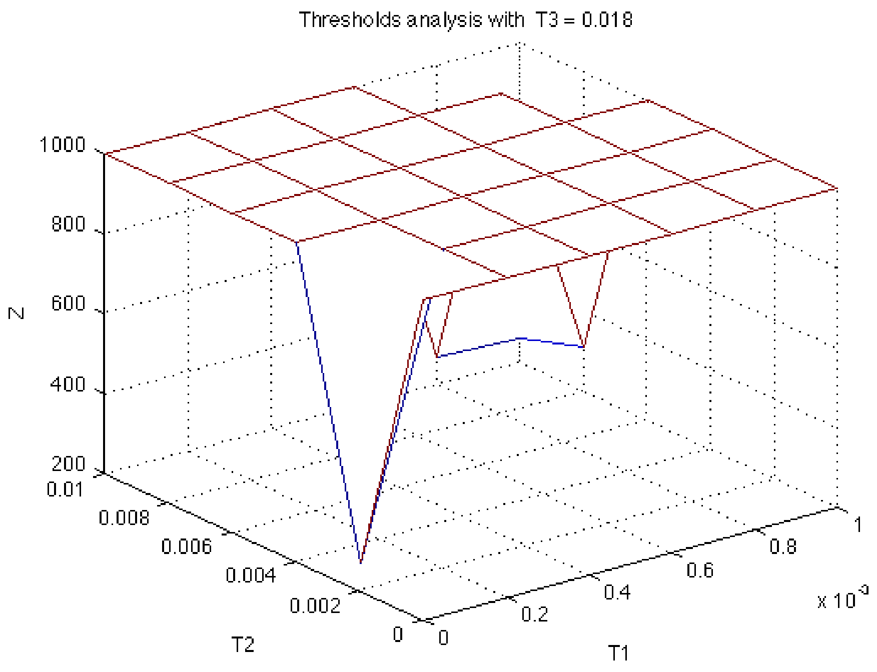
5.2.4 Conclusion

Combined with a good prediction of the PCM such as the $EPCM$, the robot shows good results in simulations. Indeed, with optimal values found in the tests, the robot rapidly stabilizes itself with the arms without constantly self-balancing. As shown in the following figure, the robot is standing on the moving platform and stabilizing itself against the perturbations.

And since the results can be significantly remarked visually, some videos showing the stabilization of the robot on the moving platform can be seen on the Biorob website.



(a)



(b)

Figure 16 – *The resulting robot scores for different thresholds in the arms reaction functions (respectively relative to z and x axis)*

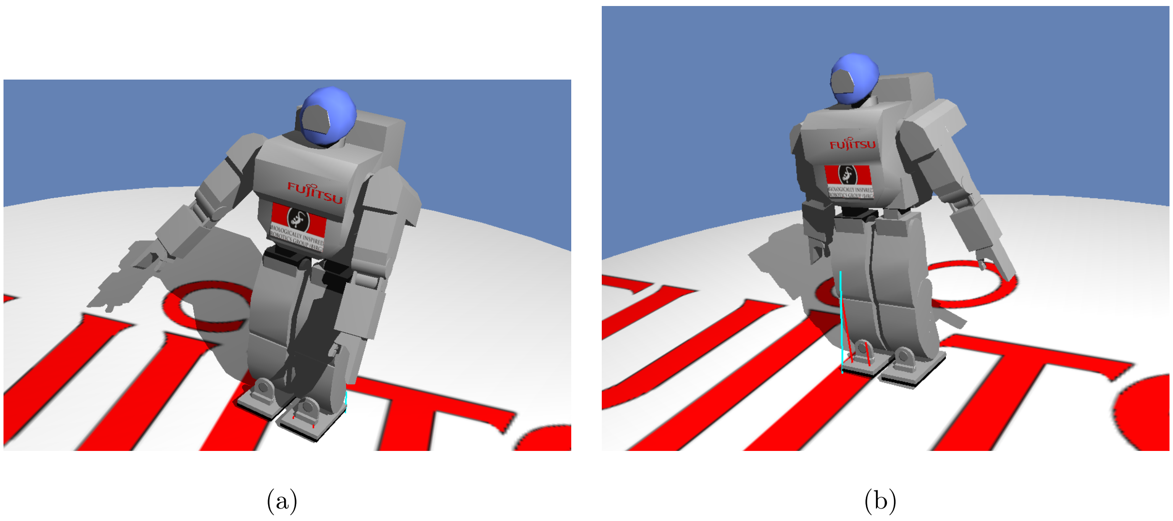


Figure 17 – *The HOAP-2 robot stabilizing on the oscillating platform*

6 Conclusion

6.1 Discussion

The stabilization problem is known to be difficult in dynamic situations. However, I introduced some simple methods that work according to some assumptions to help the stabilization process notably in dynamic cases (when the robot is trying to stay stable on the moving platform for instance). Finding a good stability criterion was an interesting problem for the HOAP-2 robot and many solutions has been first exploited (such as the ZMP) before introducing the $EPCM$. The concept and implementation of the reaction functions were also a challenge; many questions arose such as how to interpret the $EPCM$ to stabilize correctly the robot and how to do it. Here came the idea of the stabilization according to the axis (to simplify the model) that works in simulations even when both joints arms are acting together along different axes.

6.2 Future Work

For a future work, it could be interesting to deal with other parts of the body (such as the torso, the hips or even the legs) for the stabilization process. Another interesting topic could be also the enhancement of the stability criteria, for instance for more complex contact surfaces (until now I considered only horizontal contact surfaces). Finally, it could also be nice to have some more complex reaction functions (such as sigmoid functions) that can still improve the stability of the robot according to some stability criteria.

6.3 Acknowledgements

I was happy to work in a fascinating project such as this one and thank my supervisors for their patience and assistance.

References

Brown, T. G. (1911), ‘The intrinsic factors in the act of progression in the mammal’, *Proceedings of the Royal Society of London* .

Hauser, H., Neumann, G., Ijspeert, A. J. & Maass, W. (2007), ‘Biologically inspired kinematic synergies provide a new paradigm for balance control of humanoid robots’.

Hof, A. L. (2008), ‘The ‘extrapolated center of mass’ concept suggests a simple control of balance in walking’.

Hof, A. L., Gazendam, M. G. J. & Sinke, W. E. (2004), ‘The condition for dynamic stability’.

Hooper, S. L. (1999), ‘Central pattern generators’, *Embryonic ELS* .

ICC (1998), *The development of Honda humanoid robot. Proceedings of the IEEE International Conference on Robotics and Automation*, Leuven (Belgium).

Ijspeert, A. J. (2008), ‘Central pattern generators for locomotion control in animals and robots: A review’.

Popovic, M. B., Goswami, A. & Herr, H. (2005), ‘Ground reference points in legged locomotion: Definitions, biological trajectories and control implications’.

Righetti (2006), ‘Programmable central pattern generators: An application to biped locomotion control’.

Webots (n.d.), ‘http://www.cyberbotics.com’. Commercial Mobile Robot Simulation Software.
URL: *http://www.cyberbotics.com*

Wilson, D. M. (1961), ‘The central nervous control of flight in a locust’, *Journal of Experimental Biology* .