View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Biblioteca Digital do IPB

# Humanoid Robot Simulation with a Joint Trajectory Optimized Controller

José L. Lima, José C. Gonçalves, Paulo G. Costa, A. Paulo Moreira
Department of Electrical and Computer Engineering
Faculty of Engineering of University of Porto
*jllima@ipb.pt, goncalves@ipb.pt, paco@fe.up.pt, amoreira@fe.up.pt*

## Abstract

*This paper describes a joint trajectory optimized controller for a humanoid robot simulator following the real robot characteristics. As simulation is a powerful tool for speeding up the control software development, the proposed accurate simulator allows to fulfil this goal. The simulator, based on the Open Dynamics Engine and GLScene graphics library, provides instant visual feedback.*

*The proposed simulator, with realistic dynamics, allows to design and test behaviours and control strategies without access to the real hardware in order to carry out research on robot control without damaging the real robot in the early stages of the development. The joints controller techniques, such as acceleration, speed and energy consumption minimization are discussed and experimental results are presented in order to validate the proposed simulator.*

## 1. Introduction

In last years, studies of research in biped robots have been developed and resulted in a variety of prototypes that resemble the biological systems. Legged robots have the ability to choose optional landing points, an advantage to move in rugged terrains, and two legged robots are also able to move in human environment. So, studies about biped robots are very important and stimulating [1]. Locomotion under influence of external disturbances is a challenging task for a humanoid robot, once if disturbances are large enough, a fall might become unavoidable. Closed loop controllers should minimize the number of falls [2] and if a fall happens, the robot should detect it and get back into an upright posture [3]. On the one hand, simulation is a powerful tool for speeding up the control software development. On the other hand, developing new control software for robots can be a difficult and challenge task. The ability to rapidly prototype software, within a simulation environment, can be of great benefit to develop robot control if the resulting software can be easily transfered from simulation to real world systems. Therefore, the simulator must capture the most important environment characteristics; however, developing simulators with high-fidelity dynamic models that can be simulated in real-time is a non trivial problem [4]. The simulator must also be able to measure the consumed energy providing a good efficiency planning. The planning for humanoid movements should result in minimum energy consumption, like it happens in the human body. Joints angles and torques limits must also be handled.

There are several robot simulators, such as Simspark, Webots and MURoSimF, that provide humanoid simulation capability. Meanwhile, the developed simulator allows to build and to test the low and high level controllers, with a configurable control period, in a way that can be mapped with the reality, although with some overhead. Furthermore, this simulator can be controlled by network and script language avoiding installations of development applications. As an important feature, robots can be built with a configurable structure based on a *xml* description file. It is also possible to create several humanoid robots in the environment.

Code migration from general realistic simulators to real world systems is the key for reducing development time of robot control, localization and navigation software. Due to the complexity of robot, world, sensors, and actuators modelling it is not an easy task to develop such simulator. The motivation of developing a realistic humanoid robot simulator is to produce a personalized and versatile tool that will allow in the future the production and validation of robot software reducing considerably the development time. This simulator deals with robot dynamics and how it reacts for several controller strategies and styles.

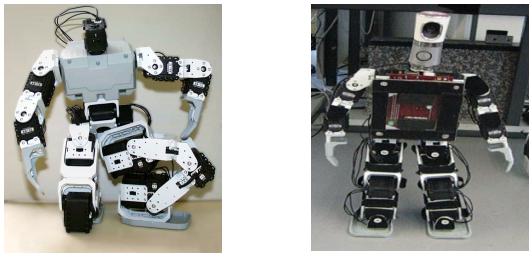
This paper proposes a simulator, based on the Open Dynamics Engine [5], for a humanoid robot and presents its low level controller. The proposed simulator allows to design and test behaviours without access to real hardware in order to carry out research on robot control once it is developed having in mind the real robot: dimensions, masses, inertias, joints angles and velocities limits are accurately resembled.

The paper is organized as follows: Initially, the real robot, where mechanical design, communication and control application are described, is presented. Then,

section 3 presents the developed simulator basis and how simulator robot is built. Section 4 presents the joint trajectory planning where minimum acceleration, minimum speed and minimum energy consumption methods are described. Experimental results are presented further in section 5. Finally, section 6 rounds up with conclusions and future work.

## 2. Real Humanoid Robot

The commercially available Bioloid [6] robot kit, from Robotis, is the basis of the used humanoid robot. The overview of the proposed biped robot is shown in Figure 1. The suggested robot was modified and differs from the original kit, to follow the dimensional rules of RoboCup [7] Humanoid League [8]. Next subsections present the physical robot in which was based the developed humanoid simulator.

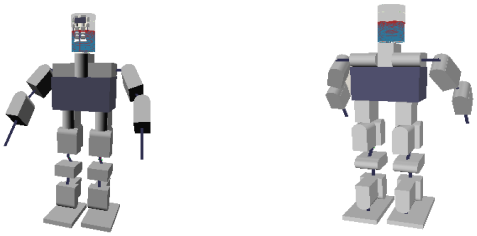


a) b)

**Figure 1. Real humanoid robot poses.**

### 2.1. Mechanical Design

The presented humanoid robot is driven by 19 servo motors: 6 per leg, 3 in each arm and one in the head. Three orthogonal servos set up the 3DOF (degree of freedom) hip joint. Two orthogonal servos form the 2DOF ankle joint. One servo drives the head (a vision camera holder). The shoulder is based on two orthogonal servos allowing a 2DOF joint and elbow has one servo allowing 1DOF. The total weight of the robot is about 2 *kg* and its height is 38 *cm*. The modelled humanoid robot is presented in Figure 2 that allows to visualize real humanoid posture in the interface software.



a) b)

**Figure 2. Modelled robot. a) frontal view, b) back view.**

### 2.2. Communication Architecture

Multiple layers that run on different time scales contain behaviours of different complexity. The layer map is presented in Figure 3.

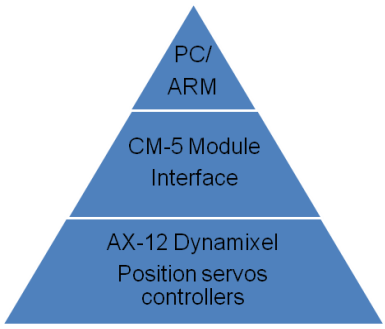



**Figure 3. Layers diagram.**

The lowest level of this hierarchy, the control loop within the Dynamixel actuators (AX-12), has been implemented by Robotis [9]. Each servo is able to be programmed with not only the goal position, the moving speed, the maximum torque, the temperature and voltage limits but also with the control parameters. This communication layer is based on a 1Mbps half-duplex serial bus where each individual servo can be addressed or a broadcast can be sent. These limitations are placed in the simulator for a faithful representation.

At the next layer, an interface unity CM-5 module, based on an Atmel ATMega128 microcontroller, allows a communication interchange. It receives messages from the upper layer and translates them to the servos bus. Answers from servos are also translated and sent back to the upper layer as presented in Figure 4.

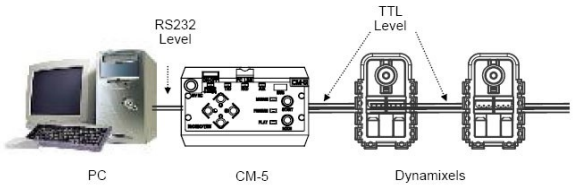



**Figure 4. Interface layer.**

The original firmware was replaced in order to get higher performances and low level controller achieve. At the higher layer, target angle and moving speed for the individual joints are generated from a personal computer or from an embedded system.

### 2.3. Behaviour and Control

Perception assumes a major role in an autonomous robotics, and must be therefore reliable or abundant [10]. For this robot, the joint position, speed and torque perception was planned. For enlarge the closed loop control, as a future feature, the accelerometers information and feet force sensing perceptions were also planned [11]. At the present, the real humanoid robot is

not equipped with accelerometers and for similarity the simulator sensors were not included.

As a first approach, an open-loop system can be used (accelerometers and feet force information are disabled). This can be done sending pre-programmed joint angles and angular speeds for each joint. Walk and stand up movements can be achieved. The developed software that communicates with CM-5 module and controls the robot is presented in Figure 5. The main task is to send preprogrammed joints angles and velocities that compose a movement and show, in a real-time 3D window, the real robot posture. Preprogrammed positions can be planned off-line (disconnected from the robot) once user can observe the robot postures. It also allows to obtain the real attitude of the robot.

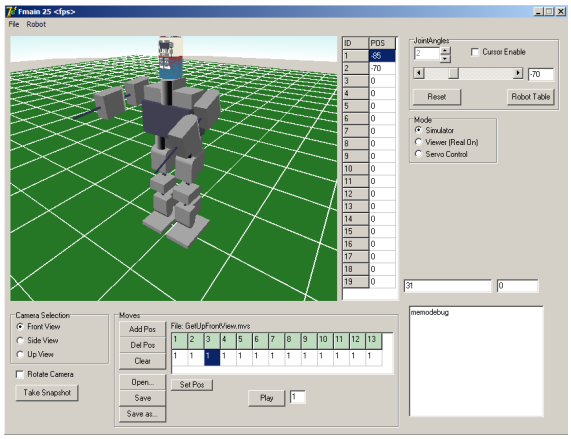



**Figure 5. Developed humanoid software controller.**

## 3. Open Dynamics Engine Simulation

Design behaviour without real hardware is possible due to a physics-based simulator implementation. The physics engine is the key to make simulation useful in terms of high performance robot control. Although there are a number of open source simulation engines available, most focus on producing fast pseudo realistic simulations for use in computer games. These engines are therefore fast, but produce motions that look good as opposed to being accurate. In contrast, there exist a number of simulation engines for rigid body motion that are unusable for simulating the mechanical interactions of rigid parts [4]. For real-time simulation, an accurate but fast simulation engine must be used. ODE, Open Dynamics Engine [5] checks these requisites. As an open source rigid body simulation engine, developed by Russell Smith, has reached a maturity level ensuring that produced code is stable. It is essentially a simulation library that provides support for rigid body motion, rotational inertia and collisions treatment where the world to be simulated is built. It also allows to use Open GL (graphics library) routines to render the 3D simulated environment. The graphic routines are based on Open GLScene library. It provides visual components and objects allowing description and rendering of 3D scenes in an easy, no-hassle, yet powerful manner. It has grown to become a set of founding classes for a generic 3D engine with RAD (Rapid Application Development) in mind [12].

### 3.1. Humanoid Construction

A complex humanoid model can be avoided due to the ODE usage. Humanoid body simulator construction is based in body masses and joint connections. Each body mass imitates the servo motors and connection pieces weights from the real robot. ODE joints, imitate the servo motors axis movements and must be defined its types, angles and torques limits. Joints types can be classified as hinges or universal joints: a hinge that allows both bodies to be connected and roll such as arms and forearms, femur and leg; a universal joint must be introduced when there are two or more degrees of freedom between two bodies. It happens when two servo motors are physically combined. A universal joint allows two bodies to roll on both axes. As example, presented in the simulator, these joints connect trunk and arms, trunk and legs, legs and feet.

### 3.2. XML model description

The *Extensible Markup Language* (XML) is a general specification language that allows its users to define their own elements. It defines a generic syntax used to mark up data with simple human-readable tags [13]. A description of the humanoid robot model in done resorting to XML description language. Positions, sizes, masses perform the description of bones and positions, axis, limits and types perform the description of joints as presented in the next XML excerpt.

```
<robot>
 <kind value='humanoid'/>
 <solids>
  <cuboid>
   <ID value='6'/>
   <pos x='-0.040' y='0' z='0.010'/>
   <size x='0.082' y='0.102' z='0.137'/>
   <mass value='0.635'/>
   <desc Pt='Tronco'/>
   <desc Eng='Trunk'/>
  </cuboid>
 </solids>
 <articulations>
  <joint>
   <ID value='0'/>
   <pos x='-0.039' y='0.030' z='-0.170'/>
   <axis x='0' y='1' z='0'/>
   <connect B1='7' B2='17'/>
   <limits Min='-98' Max='32'/>
   <type value='Hinge'/>
   <desc Pt='Joelho Esq'/>
```

```
    <desc Eng='Left Knee'/>
  </joint>
 </articulations>
</robot>
```

The humanoid robot simulator is built based on the XML description. By this way, it is an easy task to modify the robot structure once there is no necessity to compile a new application, making the simulator useful to others beyond the programmer. The same language is used to store the movements of each joint.

GLScene is used to render the 3D graphics appearance enhancing visualization including shadows, textures, projections, illuminations and it also provides depth perception. Zooming and camera positioning become also an easy task. A screenshot of the developed simulator is shown in Figure 6, where a 3D scene shows some humanoid robots and several obstacles, a table shows the desired joint variables such as angle and angular speed.

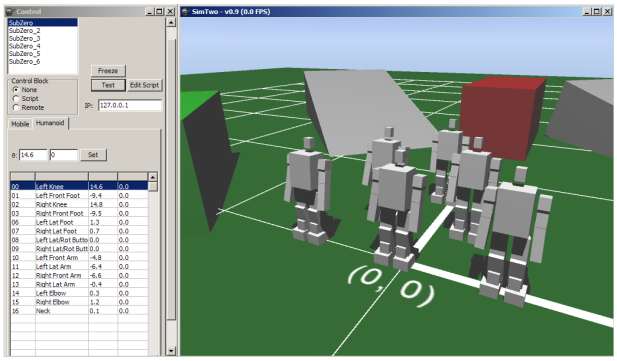


**Figure 6. Developed simulator screenshot.**

## 4. Humanoid Joint Trajectory Controller

This controller level accepts, for each servo, angles, angular speeds and timings requirements from the higher level. The main objective of this controller is to build and to follow the trajectories established by angles and angular speeds requirements having in mind the acceleration, speed and energy consumption minimization.

### 4.1. Servo-Motor model

The servo motor response, such as dynamics, maximum acceleration and speed, must be known in order to draw simulator trajectories compatible to the real robot. The joints closed loops controllers must also have the same response as servos have.

An input step, from 0 to 50 degrees with a sample frequency of 30 Hz and an inertial mass, is presented in Figure 7 (orange lozenges) and allows to obtain the desired parameters. The maximum speed can be found by the servo motor angle maximum derivative ($\omega_{max}$=281 *deg/s*) and the acceleration can be found by maximum second derivative ($a_{max}$=1400 *deg/s²*). This test was made assuming that friction and wind-up saturation non linearities are despised.

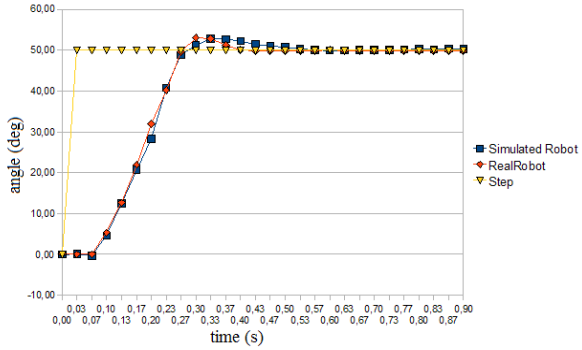



**Figure 7. Real and simulator servo motors response to a step input.**

The implemented servo motor model in simulator, based in the real servo motor step response, is tested for the same input step and with the same inertial mass in order to validate its similarity with the real one and presented in the same figure (blue squares). The overlapped curves allow to validate the servo motor model implemented in the simulator.

### 4.2.Trajectory planning

The joints controller finds the intermediate trajectories that take joints to the desired states and follow them. It is also able to minimize the accelerations in order to save consumed energy.

Suppose that for $t=t_1$ (actual time) it is measured angle $\theta_1$ and angular speed $\omega_1$, and for $t=t_2$ (next period) it is desired position $\theta_2$ and angular speed $\omega_2$, as illustrated in Figure 8 a) and b), that shows the used symbology and some examples of possible trajectories, assuming a constant acceleration in $[t_1,t_m]$ and $[t_m,t_2]$.

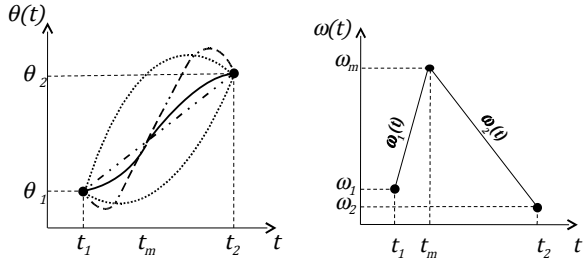



**Figure 8. Joint state: a) angle ref. and b) speed ref.**

It is necessary to calculate the angle and angular velocities equations that result in the desired conditions: the $\omega_m$ and $t_m$, assuming a constant angular acceleration that allows angular speed to follow a piece-wise linear equation: the angular speed is linear by parts, and

angular acceleration is constant by parts. The $\omega_m$ (for $t_m$ instant) must be determined and depends on the adopted method: acceleration, speed or energy can be minimized as presented in next subsections. By this way, angular reference and angular speed equations can be found as a smooth movement, following the desired conditions. There are several solutions for $t_m$, for the same initial and final conditions as presented in next equations. The covered angle ($\theta_2$-$\theta_1$) can be expressed by the triangle areas ($A_I$, $A_{II}$ and $A_{III}$) presented in Figure 9 and equation 1. Equation 2 allows to find the linear function $\omega_m=f(t_m)$, represented by *L2* line in Figure 9.

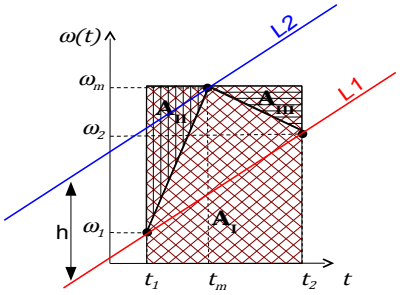



**Figure 9. $\omega_m$ position freedom.**

$$\theta_2-\theta_1=A_I-A_{II}-A_{III}= \omega_m(t_2-t_1)-(\omega_m-\omega_1)(\frac{t_m-t_1}{2})- (\omega_m-\omega_2)(\frac{t_2-t_m}{2}) \quad (1)$$

$$\omega_m=\frac{2(\theta_2-\theta_1-\frac{t_m-t_1}{2}\omega_1-\frac{t_2-t_m}{2}\omega_2)}{t_2-t_1} \quad (2)$$

In fact, equation 2 is a linear function of $t_m$ and its slope (derivative function) can be found by $(\omega_2-\omega_1)/(t_2-t_1)$, the same slope as *L1* line. These lines, *L1* and *L2* are deviated by *h* that depends on the covered angle and it can be found by equation 3.

$$h=\omega_m(t_1)-\omega_1 \quad (3)$$

So, the *h* value can be determined by equation 4.

$$h=2\frac{\theta_2-\theta_1}{t_2-t_1}-\omega_2-\omega_1 \quad (4)$$

The angle equation, $\theta_{ref}(t)$, can be found resorting to the formula of uniform linear accelerated movement for each time interval $[t_1,t_m]$ and $[t_m,t_2]$ as presented in equations 5 and 6. By this way, $t_m$ can be placed in $[t_1,t_2]$ interval, according to equation 2, that allows to cover the desired angle. Next subsections discusses $t_m$ positioning from the point of view of acceleration, speed and energy consumption.

$$\theta^{Ref}(t)=\theta_1+\omega_1(t-t_1)+\frac{1}{2}\frac{\omega_m-\omega_1}{t_m-t_1}(t-t_1)^2 \quad (5)$$

$$\theta^{Ref}(t)=\theta^{Ref}(t=t_m)+\omega_m(t-t_m) +\frac{1}{2}\frac{\omega_2-\omega_m}{t_2-t_m}(t-t_m)t^2 \quad (6)$$

### 4.3. Acceleration minimization controller method

During the $[t_1,t_m]$ interval, the joint angular acceleration $a_1$ can be expressed in equation 7, whereas during the $[t_m,t_2]$ interval, $a_2$ can be expressed in equation 8 as presented in Figure 10.

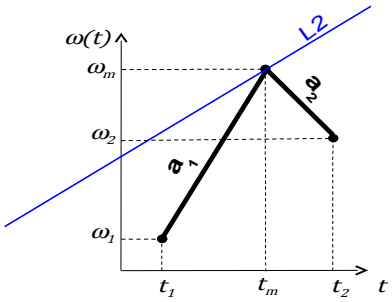



**Figure 10. Acceleration minimization.**

$$a_1=\frac{(\omega_m-\omega_1)}{(t_m-t_1)} \quad (7)$$

$$a_2=\frac{(\omega_2-\omega_m)}{(t_2-t_m)} \quad (8)$$

It can be shown that if $t_m$ moves from $t_1$ to $t_2$, $a_1$ becomes lower and, otherwise, $a_2$ becomes higher. The optimal $t_m$ can be found when both accelerations modules match, as presented in equation 9.

$$|a_1|=|a_2| \quad (9)$$

Equation 9 allows to find the $t_m=f(\omega_m(t))$ function for a desired $\omega_m$ as presented in equation 10, having in mind the particular case of accelerations minimization $a_1$ and $a_2$.

$$t_m^{Amin}=\frac{\omega_m(-t_2-t_1)+\omega_1 t_2+\omega_2 t_1}{\omega_1+\omega_2-2\omega_m} \quad (10)$$

The $(t_m,\omega_m)$ point can be found by equation 2 for the $t_m^{Amin}$ timing, calculated in equation 10, that results the desired $\omega_m$ presented in equation 11.

$$\omega_m=\omega(t_m)=\frac{1}{2}(\omega_2+\omega_1+h) \pm\frac{1}{2}\sqrt{\omega_2^2-2\omega_2\omega_1+\omega_1^2+h^2} \quad (11)$$

The valid $\omega(t)$ solution is the one that takes $t_m$ into the $[t_1,t_2]$ time window. The trajectory references $\theta(t)$ and $\omega(t)$, which minimize both accelerations $a_1$ and $a_2$, can now be drawn.

As result, presented in Figure 11, a *Matlab* function draws the expressed equations for this example:

- $\theta_1$=0 *deg*
- $\theta_2$=27 *deg*
- $\omega_1$=20 *deg/s*
- $\omega_2$=30 *deg/s*
- $t_1$=3 *s*
- $t_2$=4 *s*.

The maximum reached acceleration is about 14.77 *deg/s*$^2$.

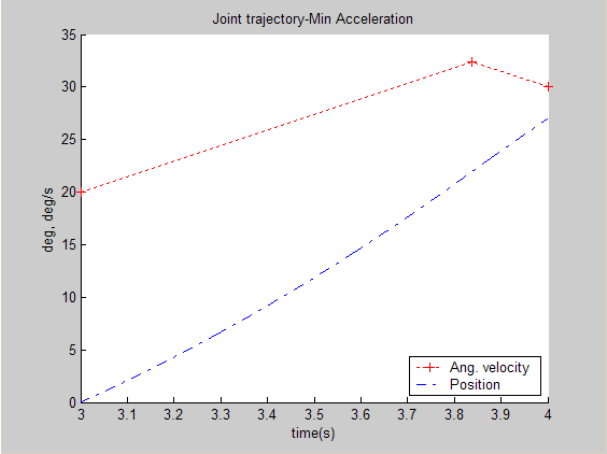



**Figure 11. Joint trajectory - minimum acceleration.**

### 4.4. Speed minimization controller method

The way to minimize the speed is to reach the *L2* line as fast as possible while keeping the maximum acceleration limits.

From all the feasible solutions, the one that minimizes the joint speed is presented in Figure 12 where $a_{max}$ is the servo motor maximum acceleration.

The $t_m$ instant can now be determined by *L2* and *L3* intersection as expressed in equation 12. Equation 2 allows to find the $\omega_m$ value for the desired instant $t_m$.

$$t_m=\frac{h}{a_{max}-\frac{\omega_2-\omega_1}{t_2-t_1}}+t_1 \quad (12)$$

As result, presented in Figure 13, a *Matlab* function draws the expressed equations for the same desired conditions as presented in previous subsection and an acceleration of 200 *deg/s*$^2$.

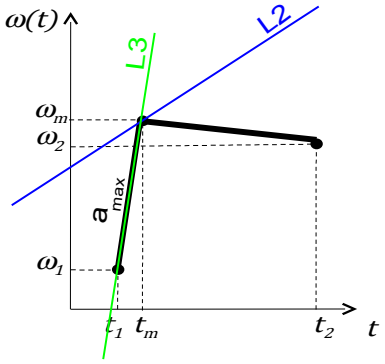



**Figure 12. Joint trajectory ($\omega_{max}$ minimized).**

The maximum reached speed is 30 *deg/s* (the desired final speed $\omega_2$).

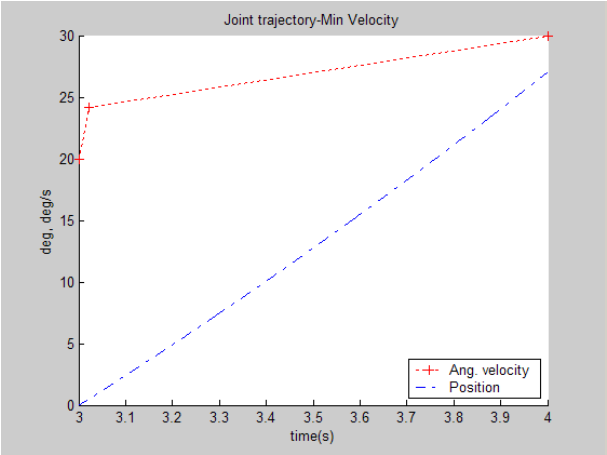



**Figure 13. Joint trajectory - minimum speed.**

### 4.5. Energy consumption minimization controller method

As humanoid robot is powered by on-board batteries, energy consumption must be reduced as much as possible. Trajectories design task should care energy consumption and minimize it.

Assuming that the instant power consumption by servo motor can be determined as the torque and the angular speed $\omega$ product ($P=k.I.a.\omega$), where $I$ is the moment of inertia, $k$ a scalar gain and $a$ the angular acceleration, it is possible to place $t_m$ in the energy minimization instant as described in this subsection. The moment of inertia depends on each joint and robot posture but can be considered constant in energy minimization timing achieve for simplicity As example, an extended arm that measures approximately 0.2 *m* and

weights 0.174 *kg* has an moment of inertia of 0.00232 $kg.m^2$ ($I=mL^2/3$, where $m$ is the mass, and $L$ the body length). The $k$ value allows the conversion from *deg/s* to S.I. units expressed as $4\pi^2/360^2$.

For the first time part (where $t \in [t_1,t_m]$) there is an angular speed $\omega_1(t)$ with an acceleration $a_1$ and for the second time part (where $t \in [t_m,t_2]$) there is an angular speed $\omega_2(t)$ with an acceleration $a_2$. The instant power consumption can be described by equation 13 where $a_1$ and $a_2$ depend on $t_m$ instant and the total energy can be described by the power integral as presented in equation 14.

$$P_1(t)=k\,I\,a_1\omega_1(t)$$
$$P_2(t)=k\,I\,a_2\omega_2(t) \qquad (13)$$

$$E_{Tot}=\int_{t_1}^{t_m} P_1(t)+\int_{t_m}^{t_2} P_2(t)\,dt \qquad (14)$$

The minimum energy consumption $t_m$ instant can be found when its first derivative equals zero as presented in equation 15 that allows to find the $t_m$ instant presented in equation 16. Equation 2 allows to find the $\omega_m$ value for the desired instant $t_m$.

$$\frac{d\,E_{Tot}}{d\,t}=0 \qquad (15)$$

$$t_m=\frac{w_1 t_1-2\,w_2 t_2-2\,\theta_1+2\,\theta_2+w_2 t_1}{-w_2+w_1} \qquad (16)$$

As result, presented in Figure 14, a *Matlab* function draws the expressed equations for the same desired conditions as presented in previous subsections. The consumed energy is 0.1767 *mJ*, following the previous presented conditions.

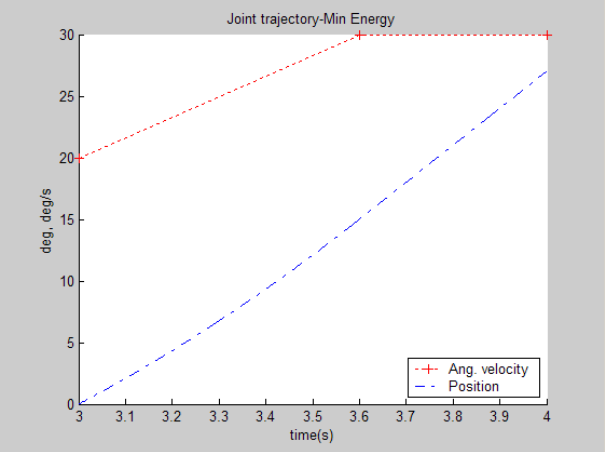



**Figure 14. Joint trajectory - minimum energy consumption.**

Minimum energy consumption function from *t1* to *t2* timing is also illustrated in Figure 15 that shows the optimal $t_m$ value at 3.6 seconds. If the optimal $t_m$ timing is outside $[t_1,t_2]$ interval, maximum acceleration should be done at $t_1$ if $t_m < t_1$ or to $t_2$ if $t_m > t_2$.

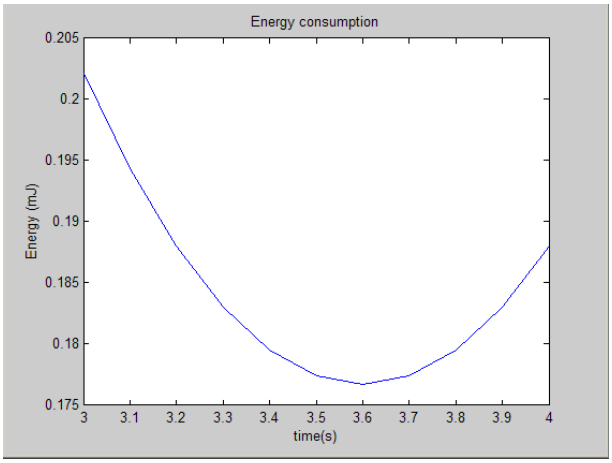



**Figure 15. Energy consumption in $[t_1,t_2]$ interval.**

### 4.6. Trajectory planning methods comparison

In order to compare the effectiveness of the presented trajectory planning methods, table 1 presents the $t_m$ instant, the $\omega_m$ value, the maximum acceleration $a_{max}$ and the energy consumption $E_{cons}$ for each method ($A_{min}$-minimum acceleration, $\omega_{min}$-minimum speed and $E_{min}$-minimum energy consumption). Notice that $\omega_m$ for the $\omega_{min}$ method is lower than the requested angle (30 *deg/s*).

**Table 1. Trajectories planning comparison.**

| Method | $t_m$ (s) | $\omega_m$ (deg/s) | $\lvert a_{max}\rvert$ (deg/s²) | $E_{cons}$ (mJ) | $E_{cons}$ (%) |
|---|---|---|---|---|---|
| $A_{min}$ | 3.84 | 32.39 | 14.77 | 0.1807 | 2.26 |
| $\omega_{min}$ | 3.02 | 24.21 (30) | 200.0 | 0.2004 | 13.41 |
| $E_{min}$ | 3.6 | 30 | 16.67 | 0.1767 | - |

The shown results allow to validate the presented methods. Energy consumption saving in the presented examples is about 13 percent and can be done withou any hardware change. With this, $E_{min}$ method is the most suitable having in mind the energy efficiency as battery energy is a limited resource. On the other hand, accelerations minimization method also allows to decrease the energy spent and can be sometimes adopted to minimize joints efforts on the robot.

## 5. Experimental Results

This chapter presents, in a short way, the results that simulator can achieve when the conditions previously presented are applied. The energy consumption and acceleration minimizations are tested with successful results. As example, the same conditions were applied to the neck joint. The angle and reference angle are presented in Figure 16 where, at the left side, user can add which robot, joint and variables are requested to appear in the graphic at the right side.

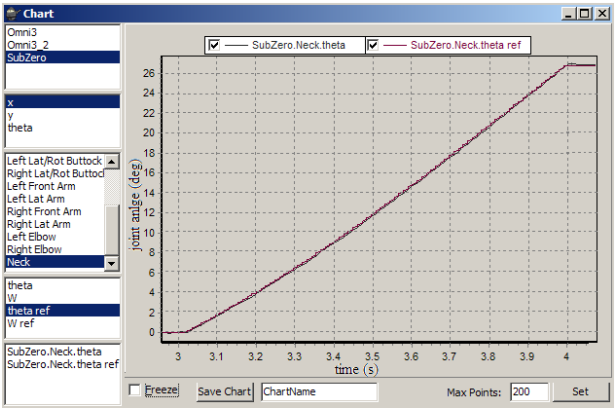



**Figure 16. Neck joint minimum consumption energy test.**

In the presented example, neck joint angle follows the reference from 0 to 27 degrees in 1 second using the minimum energy consumption trajectory. Other example can be shown in Figure 17 where neck joint follows the reference array [0, 45, 90, -45, -90] degrees with minimum acceleration control method.

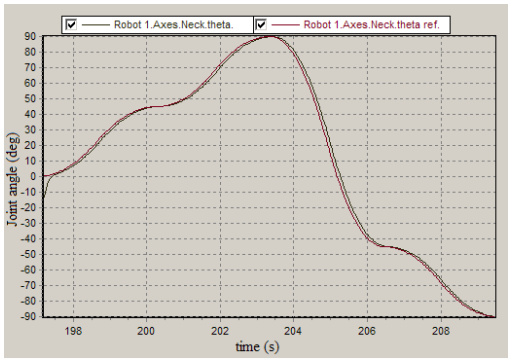



**Figure 17. Neck joint minimum acceleration test.**

## 6. Conclusion and Future Work

In this paper a humanoid simulator, based on a dynamics engine and a 3D visualization engine, is presented. The real robot limitations, such as servo motors angular speed and accelerations are taken into account. A low level trajectories controller that allows to minimize energy consumption is presented. The presented results allow to validate the simulator and show the realistic simulation. The whole body controller was implemented and humanoid simulator behaves like real robot.

As future work, more control strategies will be implemented and tested using the high level programming based on a Pascal script dialect that allows users to create their own control programs while results are real-time presented.

Enhancing the simulator with realistic sensors, such as accelerometers and gyroscopes, is also a future step.

### References


[1] T. Suzuki, K. Ohnishi, "Trajectory Planning of Biped Robot with Two Kinds of Inverted Pendulums", *Proceedings of 12th International Power Electronics and Motion Control Conference*, 2006.

[2] R. Renner, S. Behnke, "Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes", *Proceedings of International Conference on Inteligent Robots and Systems*, 2006.

[3] J. Stückler, J. Schwenk, S. Behnke, "Getting Back on Two Feet: Reliable Standing-up Routines for a Humanoid Robot", *Proceedings of 9th International Conference on Inteligent Autonomous Systems*, 2006.

[4] B. Browning, E. Tryzelaar, "UberSim: A Realistic Simulation Engine for RobotSoccer", *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.

[5] Russell Smith, Open Dynamics Engine, http://www.ode.org, 2000.

[6] Tribotix, http://www.tribotix.com/index.html, 2004.

[7] Robocup, http://www.robocup.org/, 2007.

[8] Humanoid League, http://www.humanoidsoccer.org/, 2007.

[9] S. Behnke, M. Schreiber, J. Stückler, R. Renner, H. Strasdat, "See, Walk, and kick: Humanoid robots start to play soccer", *Proceedings of International Conference on Humanoid Robots*, 2006.

[10] V. Santos, F. Silva, "Design and Low-Level Control of a Humanoid Robot Using a Distributed Architecture Approach", *Journal of Vibration and Control, Vol.*, pp. 1431-1456, 2006.

[11] S. Kagami, Y. Takahashi, K. Nishiwaki, M. Mochimaru, H. Mizoguchi,"High-speed matrix pressure sensor for humanoid robot by using thin force sensing resistance rubber sheet", *Proceedings of IEEE Sensors Conference*, 2004

[12] GLScene, http://glscene.sourceforge.net, 2000.

[13] E. Harold and W. Means, *XML in a Nutshell*, O'Reilly, 2004.