Control and simulation of a robot of two degrees of freedom (Implementation of a new control algorithm)

P. Sánchez-Sánchez$^{\diamond}$, F. Reyes-Cortés$^{\diamond}$, A. Michua-Camarillo$^{\diamond}$, J. Cid-Monjaraz$^{\diamond}$ *BUAP*
and M. Arias-Estrada *INAOE*

***Abstract*— The objectives of this paper are to present a simulator program of a robot of two degrees of freedom and introduce a new controller scheme on joint space. The simulator is designed based on the dynamic model of the prototype and using a controller with stability proof we can locate the end-effector of the robot in a specific point. The kind of control that we use to programming the simulator is the position control type.**

***Index Terms*— Forward and inverse kinematics, Dynamic model, Simulator, Parametric identification, Performance index, Energy shaping, visual C++ compiler, OpenGL libraries, events driver.**

I. INTRODUCTION

The robot manipulators offer interesting theoretical and practical challenges to control researchers due to nonlinear and multivariable nature of their dynamic model [1].

The *dynamic model* describes the behavior from a mechanical system to an applied force. The energy applied at the robot manipulator, as well-known as *controller structure*, takes charge of providing to the mechanical structure the necessary force to fulfill the assigned task [1].

From a practical point of view, the real time implementation of robot controllers can be an expensive project and a time consuming activity if an adequate test system is not available [1], [2].

For this reason, using a simulator programm that describes the relationship between a controller and a specific prototype we can evaluate the system. This evaluation is used to validate the controller's behavior and to guarantee the viability of the design [3], [4].

From the point of view of the robotics, a *simulator* is a program based on the dynamic model of a robot manipulator and the control scheme. The dynamic model is a nonlinear structure with certain very defined characteristics. A computer simulation is an attempt to model a real-life situation on a computer so that it can be studied to see how the system works. By changing variables, predictions may be made about the behavior of the system [1]–[3].

This work focuses on the *position control* for robots manipulators, the goal of this kind of control is to move the manipulator's end-effector from initial position q_0 to a fixed desired target q_d (constant in time).

When we control the position of general manipulators, we are confronted with their nonlinear dynamics in many degrees of freedom. In much of the literature concerned with the dynamic model of manipulators, the complexity of nonlinear dynamic is emphasized and various methods that compensate all nonlinear terms in dynamics in real time are developed in order to reduce the complexity of control systems [5].

However, these methods require a large quantity of complicated calculations so that it is difficult to implement these methods with low level controllers such as micro-computers. In addition, the reliability of these methods may be misplaced when a small error in computation or a little change in the parameters of the system occurs, since these are not considered in the control [1], [3], [5].

Nevertheless, convergence to a target position has not been sufficient investigated for general nonlinear mechanical systems. In this work we have described a simulator program for research and development of control algorithms which allows the easy simulation test of control strategies. We used a particular case of nonlinear position controller, this controller preserves the asymptotic stability in a global way, it's supported by a rigorous stability analysis including the full Lagrangian analysis [3].

This paper is organized as follows: section 2 describes the mathematical representation of the prototype. In section 3 we describe a proposed controller, the control problem formulation and the main stability proof. Section 4 summarizes the main components of the simulator and finally we present some conclusion remarks in section 5.

II. MATHEMATICAL DESCRIPTION OF THE PROTOTYPE

To understand the complete movement of a prototype it is necessary to analyze their kinematics and their dynamics, with the objective of obtaining a mathematical representation of the forces that act in the robot. The following diagram describes the prototype used to make the mathematical analysis.

Observe you that the prototype is located in *home position*.

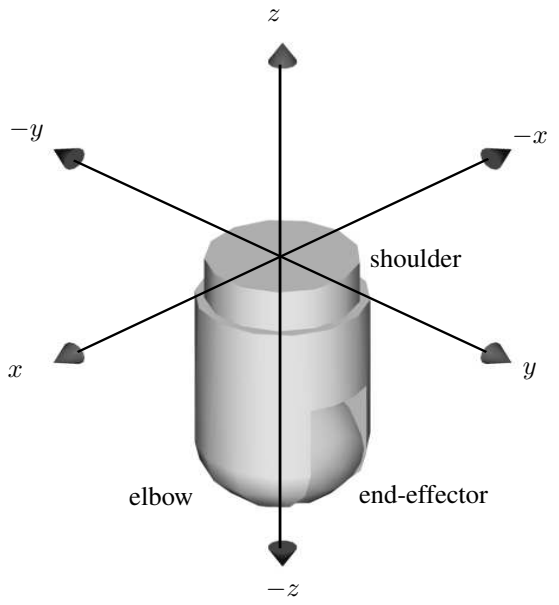

Fig. 1. Experimental prototype (home position)

II-A. Robot kinematics

The robot manipulator's kinematics is used to analyze the forces that act with the system without considering the causes that produce them. In the robotics, this analysis is divided in two: *forward kinematics* and *inverse kinematics*. The forward kinematics is good us to locate the position of the end-effector in function of their joints.

The forward kinematics of the first link (shoulder-elbow) are:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} l_1 \cos(\alpha) \cos(q_1) \\ l_1 \cos(\alpha) \sin(q_1) \\ -l_1 \sin(\alpha) \end{bmatrix} \tag{1}$$

where α is a structural characteristic. While the forward kinematics of the system elbow-shoulder $\begin{bmatrix} x_2 & y_2 & z_2 \end{bmatrix}^T$ are defined as:

$$\begin{bmatrix} -l_2 \cos(\beta) \cos(q_1) - l_2 \sin(\beta) \sin(q_1) \sin(q_2) \\ -l_2 \cos(\beta) \sin(q_1) + l_2 \sin(\beta) \cos(q_1) \sin(q_2) \\ l_2 \sin(\beta) \cos(q_2) \end{bmatrix} \tag{2}$$

where β is a structure characteristic. To obtain the total forward kinematics of the system it is necessary to add the equations (1) and (2).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{3}$$

The forward kinematics is used to be able to obtain the kinetic energy of the system $\mathcal{K}(q, \dot{q})$:

$$\mathcal{K}(q, \dot{q}) = \frac{1}{2} m \underbrace{\left(\frac{d}{dt} \begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \frac{d}{dt} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right)}_{v^2} + \frac{1}{2} I \dot{q}^2 \tag{4}$$

The importance of the forward kinematics resides in application to obtain the kinetic energy of the system which is good us to obtain the dynamics from the robot when applying the equation of Lagrange.

II-B. Robot dynamics

We use Lagrangian's dynamic to obtain the mathematical equations. We begin our development with the general Lagrange's equation of motion [1], [2], [6], [7].

Consider then Lagrange's equations for a conservative system as given by:

$$\frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}} \right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q} = \tau - f(\tau, \dot{q}) \tag{5}$$

where q is a n-vector of generalized coordinates, τ is an n-vector of generalized force, $f(\tau, \dot{q})$ is the vector of friction and the Lagrangian $\mathcal{L}(q, \dot{q})$ is the difference between the kinetic and potential energies [1], [2], [6], [7],

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q). \tag{6}$$

It is well-known that in the absence of friction and other disturbances, the dynamics of a serial n-link rigid robot can be written as [1], [2], [6], [7]:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{7}$$

where $q, \dot{q}, \ddot{q} \in \mathbb{R}^{n \times 1}$ are vectors of joint displacements, velocities and accelerations respectively, $M(q) \in \mathbb{R}^{n \times n}$ is the symmetric positive definite manipulator inertial matrix, $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is the matrix of centripetal and Coriolis torques and $g(q) \in \mathbb{R}^{n \times 1}$ is the vector of gravitational torques obtained as the gradient of the robot potential energy [6], [7].

Although the equation of motion (7) is complex, it has several fundamental properties which can be exploited to facilitate control system design.

We use the following important properties:

Property 1: Considering all revolute joints, the inertial matrix $M(q)$ is lower and upper bounded by [8], [9]:

$$\mu_1(q)I \leq M(q) \leq \mu_2(q)I \tag{8}$$

where I stands for the $m \times n$ Identity matrix. We should consider that $M(q)$ it is symmetric positive definite inertial matrix.

Property 2: The matrix $\dot{M}(q) - 2C(q,\dot{q}) \equiv 0$ is skew-symmetric, that is [8], [9],

$$\dot{M}(q) = C(q,\dot{q}) + C(q,\dot{q})^T. \tag{9}$$

Furthermore, the matrix $C(q,\dot{q})$ is linear on $\dot{q}$ and bounded on q, hence for some $k_c \in \mathrm{R}_+$ [8], [9]:

$$\|C(q,\dot{q})\| \leq k_c(q)\|\dot{q}\|. \tag{10}$$

Property 3: The generalized gravitational forces vector

$$g(q) = \frac{\partial \mathcal{U}(q)}{\partial q} \tag{11}$$

satisfies [8], [9]:

$$\left\| \frac{\partial g(q)}{\partial q} \right\| \leq k_g \tag{12}$$

for some $k_g \in \mathrm{R}_+$, where $\mathcal{U}(q)$ is the potential energy is supposed to be bounded from below [8], [9].

After a rigorous analysis the dynamic model that describes the prototype illustrated in the figure 1 is:

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \tau \tag{13}$$

where:

$$\begin{aligned} m_{11} &= m_1 l_{c_1}^2 \cos^2(\alpha) + m_2 l_{c_2}^2 \cos^2(\beta) \\ &\quad I_1 + m_2 l_{c_2}^2 \sin^2(\beta)\sin^2(q_2) \\ m_{12} &= -m_2 l_{c_2}^2 \sin(\beta)\cos(\beta)\cos(q_2) \\ m_{21} &= -m_2 l_{c_2}^2 \sin(\beta)\cos(\beta)\cos(q_2) \\ m_{22} &= m_2 l_{c_2}^2 \sin^2(\beta) + I_2 \end{aligned} \tag{14}$$

The values of the matrix of centripetal and Coriolis torques are:

$$\begin{aligned} c_{11} &= 2m_2 l_{c_2}^2 \sin^2(\beta)\sin(q_2)\cos(q_2)\dot{q}_2 \\ c_{12} &= m_2 l_{c_2}^2 \sin(\beta)\cos(\beta)\sin(q_2)\dot{q}_2 \\ c_{21} &= m_2 l_{c_2}^2 \sin(\beta)\cos(\beta)\sin(q_2)\dot{q}_2 \\ &\quad + m_2 l_{c_2}^2 \sin^2(\beta)\sin(q_2)\cos(q_2)\dot{q}_1\dot{q}_2 \\ c_{22} &= +m_2 l_{c_2}^2 \sin(\beta)\cos(\beta)\sin(q_2)\dot{q}_1\dot{q}_2 \end{aligned} \tag{15}$$

and the gravitational torque is defined by:

$$\begin{aligned} g_1 &= 0 \\ g_2 &= g m_2 l_2 \sin(\beta)\sin(q_2)\dot{q}_2 \end{aligned} \tag{16}$$

III. Proposed controller

In this section, we present our main results concerning the used controller's stability proof. Typically we propose controllers on joint coordinates using the *energy shaping methodology* [8]–[11].

The energy shaping is a controller method design, this methodology considerate the dynamic model without friction and others disturbances [8]–[11].

We use the following control scheme:

$$\tau = K_p\psi_{\tilde{q}} - K_v\psi_{\dot{q}} + g(q) \tag{17}$$

where $\tilde{q}$ denotes the error position, K_p and K_v are proportional and derivative gains matrices, respectively, and the terms ψ are defined as:

$$\begin{aligned} \psi_{\tilde{q}} &= \frac{\tanh(\tilde{q})\sqrt{2\cosh^2(\tilde{q}) - 1}}{\cosh(\tilde{q})} \\ \psi_{\dot{q}} &= \frac{\tanh(\dot{q})\sqrt{2\cosh^2(\dot{q}) - 1}}{\cosh(\dot{q})}. \end{aligned} \tag{18}$$

The closed-loop system equation obtained by combining the robot model, equation (7), and control scheme, equation (17), can be written as:

$$\frac{d}{dt}\begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\dot{q} \\ M(q)^{-1}\left[K_p\psi_{\tilde{q}} - K_v\psi_{\dot{q}} - C(q,\dot{q})\dot{q}\right] \end{bmatrix} \tag{19}$$

which is an autonomous differential equation and the origin of the state space is its unique equilibrium point. For $\dot{q}$ we have:

$$\begin{aligned} &\dot{q} = 0 \Rightarrow I\dot{q} = 0 \\ &\Rightarrow \dot{q} = 0 \end{aligned} \tag{20}$$

and for $\tilde{q}$ we get:

$$\begin{aligned} &M(q)^{-1} K_p\tilde{q} = 0 \\ &\Rightarrow \tilde{q} = M(q) K_p^{-1}(0) \\ &\Rightarrow \tilde{q} = 0 \end{aligned} \tag{21}$$

To make the stability proof of the equation (17), we proposed the following Lyapunov's candidate function based in the energy shaping's methodology [10], [11]:

$$V(\dot{q},\tilde{q}) = \frac{\dot{q}^T M(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T \cdot K_p \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix} \tag{22}$$

The first term of $V(\dot{q},\tilde{q})$ is a positive define function with respect to $\dot{q}$ because $M(q)$ is a positive definite matrix. The second one of Lyapunov's candidate function (22) is a positive definite function with respect to error position $\tilde{q}$, because K_p is a positive definite matrix.

Therefore $V(\dot{q},\tilde{q})$ is a globally positive definite and radially unbounded function. The time derivative of Lyapunov's candidate function (22) along the trajectories of the closed-loop (19):

$$\dot{V}(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{\ln(\cosh(\tilde{q}_1))} \\ \sqrt{\ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{\ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \frac{\tanh\tilde{q}}{\sqrt{\ln(\cosh(\tilde{q}))}} \end{bmatrix} \dot{\tilde{q}} \tag{23}$$

and after some algebra and using the property 2 it can be written as:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}K_v \begin{bmatrix} \frac{\tanh(\dot{q}_1)\sqrt{2\cosh^2(\dot{q}_1)-1}}{\cosh(\dot{q}_1)} \\ \frac{\tanh(\dot{q}_2)\sqrt{2\cosh^2(\dot{q}_2)-1}}{\cosh(\dot{q}_2)} \\ \vdots \\ \frac{\tanh(\dot{q}_n)\sqrt{2\cosh^2(\dot{q}_n)-1}}{\cosh(\dot{q}_n)} \end{bmatrix} \leq 0. \tag{24}$$

which is a negative semidefinite function, therefore we concluded that the equilibrium point is stable.

In order to prove the asymptotic stability in a global way, we make use of the autonomous nature of closed-loop (19) when we applied the *LaSalle's invariance principle*:

$$\dot{V}(\dot{q},\tilde{q}) < 0. \tag{25}$$

In the region

$$\Omega = \left\{ \begin{bmatrix} \tilde{x} \\ \dot{q} \end{bmatrix} \in \mathrm{R}^n : V(\tilde{q},\dot{q}) = 0 \right\} \tag{26}$$

the unique invariant is $\begin{bmatrix} \tilde{q}^T & \dot{q}^T \end{bmatrix}^T = 0 \in \mathrm{R}^{2n}$.

III-A. Performance Index

Robot manipulators are very complex mechanical system, due to the nonlinear and multivariable nature of the dynamic behavior. For this reason, in the robotics community there are not well established criteria for appropriate evaluation of controllers for robots. However, it is accepted in practice to compare the performance of controllers by using the scalar-valued $\mathcal{L}^2$ norm as an objective numerical measure for an entire error curve. The $\mathcal{L}^2[\tilde{q}]$ norm measures the root-mean-square average of the $\tilde{q}$ position error, which is given by:

$$\mathcal{L}^2[\tilde{q}] = \sqrt{\frac{1}{t-t_0}\int_{t_0}^{t} \|\tilde{q}\|^2 \, dt} \tag{27}$$

where $t_0, t \in \mathbb{R}_+$ are the initial and final times, respectively. A $\mathcal{L}^2[\tilde{q}]$ smaller represents lesser position error and it indicates the best performance of the evaluated controller. The overall results are summarized in figure 2 which includes the performance indexes for the analyzed controllers. To average out stochastic influences, the data presentation in this figure represents the mean of root-mean-square position error vector norm of ten runs.

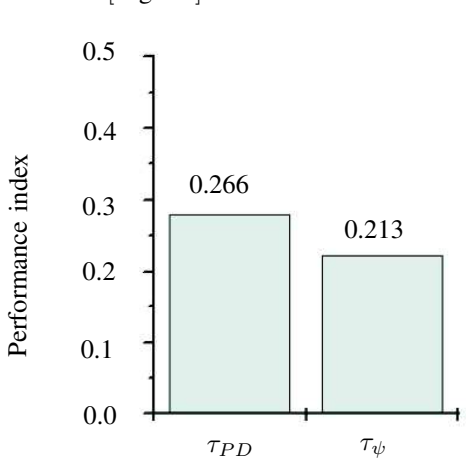

Fig. 2. Performance Index.

We compared the propose controller against the simple PD controller.

In general, the performance of the PD controller is improved roughly $21{,}6\,\%$ by its counterpart, the proposed controller as show in figure 2. τ_{PD} has a $\mathcal{L}^2[\tilde{q}_{PD}] = 0{,}266$ [degrees] over the range of the experimental results, while the performance indexes for τ_ψ are $0{,}213$ [degrees].

IV. Simulator

The steps to design and to build a prototype are the following ones:

1. Obtain the *forward and inverse kinematics* through a geometric analysis of the prototype.
2. Obtain the *dynamic model* by means of the application of the Euler-lagrange's equation.
3. Propose of the *controller with the stability proof.*
4. Draw the prototype in 3D in any design software, in our case AutoCAD was used.
5. Simulator's programming using the dynamic model and the control structure.
6. Propose a *test values* to be able to execute an assigned task using the simulator, in this step we proof the algorithms.
7. Prototype construction
8. Acquisition of the robot's specific parameters by means of the *parametric identification*.
9. To substitute the *real values* of the prototype in the code of the simulator.

Until this moment we have define the dynamic model of the rotational robot and we have prove the stability of the controller; now using this information we will get the simulator.

The platform characteristics of the simulator are:

- The programming of the system was made in Visual C++ 6 compiler of the Microsoft Company.
- To create the 3D graphics we used OpenGL libraries of Silicon Graphics Company Version 1.1.
- The program was implements in a Pentium IV processor a 2 GHz of speed.
- The compiler uses the libraries for made the graphics of the robot using the forward kinematics.

The next diagram of flow shown the process used to display the 3D image used in the simulator of the system.

The figure 3 shows the form in that displays a 3D graphics in Windows, the part where the user can implement his algorithms is in the block *Background*. This place is managed through a events driver, since Windows is a system guided to events.

The *events driver* is the responsible one of managing the entrances and exits of the program, as well as the instructions of conclude the program [12], [13].

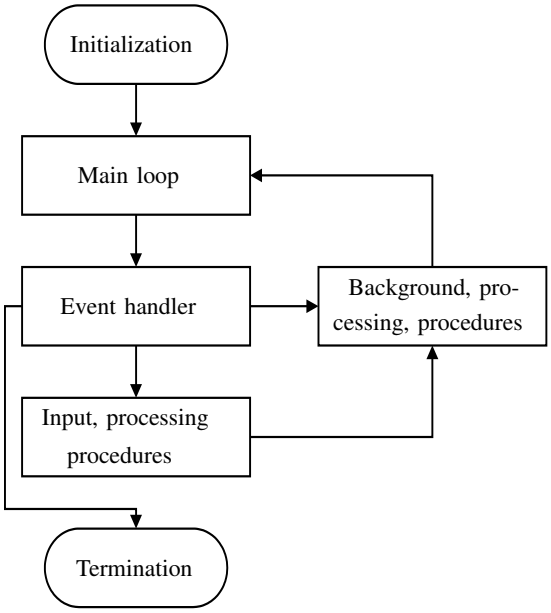

Fig. 3. Diagram to display the 3D image.

The basic steps for graphic in Visual C++ using the OpenGL libraries are the following ones:

- Getting a device context
- Selecting and setting a pixel format
- Creating, making, and setting a rendering context
- Drawing with OpenGL commands
- Deleting the rendering context
- Releasing the device context

In the simulation two routines are believed in charge of evaluating the forward and inverse kinematics, to the forward kinematics we are get the joint values (q_1, q_2) and this give us the (x, y, z) point of the end-effector that is generated by this joints, the next step is generated the graphic of the robot. The second routine receives the point (x, y, z) of the inverse kinematics and it returns the joint values of this point (q_1, q_2). The prototype was designed using AutoCAD, we used this program for the 3D tools. The image of the implemented program is presented next.

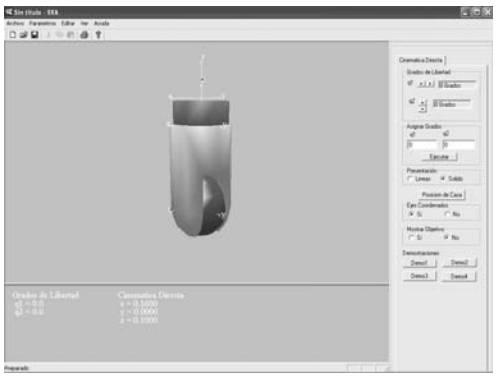

Fig. 4. Main screen of the simulator.

To program the simulator the following *test values* we were used, which allow us to evaluate the controller's behavior and the dynamic model.

$$\begin{array}{lcl} m_1 & = & 1{,}618\,kg \\ m_2 & = & 3{,}236\,kg \\ I_1 & = & 0{,}809 \\ I_2 & = & 0{,}809 \\ l_1 & = & 1{,}00\,m \\ l_2 & = & 1{,}00\,m \\ l_{c_1} & = & 0{,}54\,m \\ l_{c_2} & = & 0{,}63\,m \\ \alpha & = & 60^\circ \\ \beta & = & 45^\circ \end{array} \tag{28}$$

With the simulator we can obtain the curves of the position, error position, torque, what allows to be carried out design considerations.

Substituting the physic parameters, equation (28), on the dynamic model of the simulator, equation (7), we can described many characteristics of the prototype using a graphical interface, figure 4.

The simulator platform was programming using the following structure:

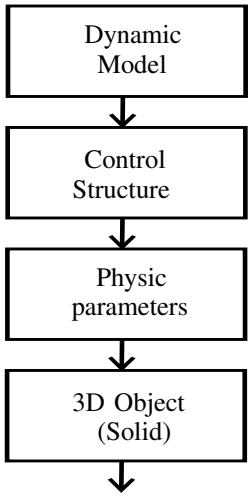

Fig. 5. Programming structure flow

V. Conclusion

In this paper we have described the steps of a robot's design emphasizing our interest in the simulator, it which has the function of evaluating the behavior of the dynamic model and the controller to fulfill an assigned task, the advantage is that by means of this simulation we can readjust calculations and considerations it stops later to build the robot with more security.

Besides the simulator, we were proven used controller's stability and it was applied in the dynamic model to do the position control. The goal of the simulation system is to support the research as well as to develop control algorithms for robot manipulators. We have shown global asymptotic stability for Lyapunov functions.

We can conclude that the realization of a program simulator is important because it allows us to evaluate the characteristics of the prototype before building it.

References

[1] Mark W. Spong and M. Vidrasagov, *Robot Dynamics and Control*, (John Wiley and Sons., 1989)

[2] A. K. Becjzy. *Robot arm dynamics and control*. Technical Memo 33-669, Pasadena, CA: NASA Jet Propulsion Laboratory, 1976.

[3] P. Fritzson, "Principles of object-oriented modeling and simulation with modelica 2.1", (Wiley Interscience, IEEE Press Editorial, 2004)

[4] H. Goldstein, *"Classical Dynamics"* (Reading: MA. Addison-Wesley, 1950).

[5] A. Barrientos, L. Peñin, C. Balaguer and R. Aracil, *"Fundamentos de Robótica"* (Madrid: McGraw Hill, 1997).

[6] L. Sciavicco and B. Siciliano, *"Modeling and Control of Robot Manipulators"* (Napoles: McGraw Hill, 1996).

[7] R. Kelly, R. Haber, R. Haber-Guerra and F. Reyes, "Lyapunov Stable Control of Robot Manipulators: A Fuzzy Self-Tunning Procedure", *Intelligent Automation and Soft Computing*, 5(4), 1999, 313-326.

[8] A. Loria and R. Ortega, "Force/Position Regulation for Robot Manipulators with Unmeasurable Velocities and Uncertain Gravity", *Automatica*, 36(6), 1996, 939-943.

[9] P. Sánchez-Sánchez, F. Reyes-Cortés and R. Reyes-Ruiz, "Cartesian Controllers for Robot Manipulators", *Proceedings of the International Symposium on Robotics and Automation*. Querétaro, México, 2004, 347-351.

[10] R. Kelly, "Regulation of Manipulators in Generic Task Space: An Energy Shaping Plus Damping Injection Approach", *IEEE Transactions on Robotics and Automation*, 15(2), 1999, 381-386.

[11] R. Kelly, V. Santibáñez and F. Reyes, "On Saturated-proportinal derivative feedback with adaptive gravity compensation of robot manipulators", *International Journal of Adaptive Control and Signal Processing*, 10(4-5), 1996, 465-479.

[12] S. Dick, A. Riddle and D. Stein, "Mathematica^R in the Laboratory", (Cambridge University Press, United Kingdom, 1997).

[13] C. Pidgeon, "Tutorials for the biomedical sciences, Animations, Simulations and Calculations using Mathematica^R", (Wiley-VCH, Canáda, 1998)