A MODELING APPROACH OF MANIPULATOR ROBOT TO USE IN REINFORCEMENT LEARNING CONTROL

RANAIVOSON Narilala Daniel[1], RAKOTOMIRAHO Soloniaina [2]

[1] PhD student, SE-I-MSDE, ED-STII, Madagascar
[2] Thesis director, SE-I-MSDE, ED-STII, Madagascar
[3] Thesis co-director, SE-I-MSDE, ED-STII, Madagascar

ABSTRACT

In order to apply a reinforcement learning concepts to robotic, it is useful to have hands on a robot model. In this paper we relate the work of modeling a robot. The point is to propose a way of designing the model not to model any particular robot. From geometrical measurements, one can write the Denavit –Hartenberg parameters of each joint. Then the homogeneous transformation matrix, associated with each joint, fallows. For the entire robot the homogeneous transformation is given by multiplying consecutively each matrix. But in order to be efficient, those multiplications will be done with numerical values instead of literal expressions. Information's about mass, density and geometric shapes allow computing every element of the inertia tensor. To simplify, we will use cylindrical forms for the joints. And then the equation of motion in the formulations of Lagrage-Euler and Newton-Euler will be written. From these equations we derive, two classes of equations. The inverse dynamic equations correspond to the control problem i.e. the question: what is the force to apply for a given trajectory? And the direct dynamic respond to another question: what is the trajectory induced by a given force? PID controls are formulated from dynamic equations. And the tuning works are done by Ziegler-Nichols methods. The modelling and simulation are carried on ROS environment. We propose a manipulator robot with five joints. Descriptions of the robot are made in URDF format. The response curves are plotted from the information given by each joint. Simulation results and experimental data have shown similarities with real system. This makes us rather confident about the validity of the model.

Keyword : robot modeling, robot manipulator, Denavit –Hartenberg parameters, homogeneous transformations, inertial tensor, mass center, Lagrange-Euler equation, Newton-Euler equation, Kinetic energy, Potential energy, Dynamic direct, dynamic inverse, ROS, URDF, PID, Ziegler-Nichols tuning method, ROS controller.

1. INTRODUCTION

The use of model is crucial in robotics. Often it is difficult to get hands on a real robot for various reasons. This is particularly the case for the study of reinforcement learning algorithm on a robot. Indeed, the development of these algorithms often requires a huge amount of data. Actually, the data is collected from the simulation of the robot and its environment during the execution of various commands. That's why the quality, precision and realism of the model used are very important. The model must reflect all physical characteristics of the robot. In this paper we propose robot's modelling in taking into account kinematic and dynamic characteristics. The point is to propose a way of designing the model not to model any particular robot. From physical specification, we derive the Denavi-Hartenberg parameters, the homogenous transformation matrix, speed and acceleration. Next, we study the dynamic with the inertia tensor and the equation of motion. And for validation, we design a command control for each joint.

2. CINEMATIC

The robot for this case study is a manipulator with five joints and six links. A simple sketch is given by Fig -1.

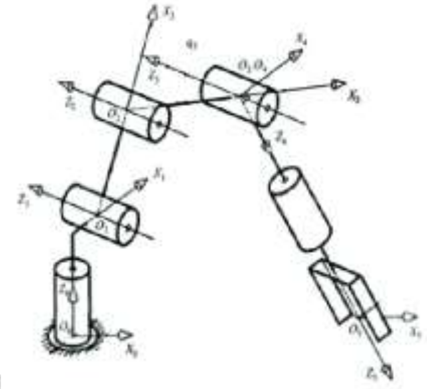


Fig -1: Elementary particle in a link

2.1 Denavit Hartenberg parameters

The DH parameters are taken from the structure of the robot to be modelled, for our robot they are given by Table 1:

Table -1: Denavit Hartenberg parameters for the robot in the Fig -1:

Joint #i	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	α_1	a_1	d_1	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	α_i	0	0	$\frac{\pi}{2}+\theta_4$
5	0	0	d_i	θ_5

Where,

- d_i - the distance between the previous x-axis and the current x-axis, along the previous z-axis.
- θ_i - the angle around the z-axis between the previous x-axis and the current x-axis.
- a_i - the length of the common normal, which is the distance between the previous z-axis and the current z-axis
- α_i - the angle around the common normal to between the previous z-axis and current z-axis.

For a more detailed explanation, we recommended [1].

2.2 Homogeneous transformation

The transformation matrix derives from the DH parameter table. Once the DH coordinate system has been established for each link, each homogeneous transformation matrix can easily be written considering the frame number {i-1} and {i}. The five transformations of each articulation are:

$$H_0^1 = \begin{bmatrix} c\theta_1 & -s\theta_1 c\alpha_1 & s\theta_1 s\alpha_1 & a_1 c\theta_1 \\ s\theta_1 & c\theta_1 c\alpha_1 & -c\theta_1 s\alpha_1 & a_1 s\theta_1 \\ 0 & s\alpha_1 & c\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$H_1^2 = \begin{bmatrix} c\theta_2 & -s\theta_2 c\alpha_2 & s\theta_2 s\alpha_2 & a_2 c\theta_2 \\ s\theta_2 & c\theta_2 c\alpha_2 & -c\theta_2 s\alpha_2 & a_2 s\theta_2 \\ 0 & s\alpha_2 & c\alpha_2 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$H_2^3 = \begin{bmatrix} c\theta_3 & -s\theta_3 c\alpha_3 & s\theta_3 s\alpha_3 & a_3 c\theta_3 \\ s\theta_3 & c\theta_3 c\alpha_3 & -c\theta_3 s\alpha_3 & a_3 s\theta_3 \\ 0 & s\alpha_3 & c\alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$H_3^4 = \begin{bmatrix} c\theta_4 & -s\theta_4 c\alpha_4 & s\theta_4 s\alpha_4 & a_4 c\theta_4 \\ s\theta_4 & c\theta_4 c\alpha_4 & -c\theta_4 s\alpha_4 & a_4 s\theta_4 \\ 0 & s\alpha_4 & c\alpha_4 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$H_4^5 = \begin{bmatrix} c\theta_5 & -s\theta_5 c\alpha_5 & s\theta_5 s\alpha_5 & a_5 c\theta_5 \\ s\theta_5 & c\theta_5 c\alpha_5 & -c\theta_5 s\alpha_5 & a_5 s\theta_5 \\ 0 & s\alpha_5 & c\alpha_5 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The matrix H_0^5 giving the transformation of coordinates between the base of the robot and the fifth joint is obtained by the matrix product:

$$H_0^5 = H_0^1 \times H_1^2 \times H_2^3 \times H_3^4 \times H_4^5 \quad (6)$$

3. DYNAMICS

Kinematic equations are used to establish the dynamic equation of motion. And dynamics allows writing the model in its details.

3.1 Mass

Each link is considered separately. The first step is to calculate its mass, knowing its geometry and density function. This link is considered as a system of particles. Each particle has an elementary mass dm , a position characterized by the vector$x$ and an elementary volume dv.

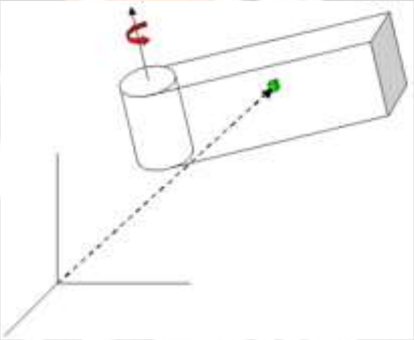


Fig -2: Elementary particle in a link

Given the density ρ(x), function of the position x, the mass of the solid is given by the sum:

$$m = \int_{V_B} \rho(x)\, dv \quad (7)$$

3.2 Centre of mass

The point of center of mass has for vector position, the vector CM whose expression is:

$$CM = \frac{1}{m} \int_{V_B} \rho(x)\, x\, dv \quad (8)$$

3.3 Inertia tensor

For a complete description of the robot, it is necessary to know the inertia tensor of each link. The inertia tensor has the general form:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yx} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{9}$$

The elements of this matrix are calculated by:

$$I_{xx} = \iiint (y^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \tag{10}$$

$$I_{yy} = \iiint (x^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \tag{11}$$

$$I_{zz} = \iiint (x^2 + y^2)\rho(x, y, z)\, dx\, dy\, dz \tag{12}$$

$$I_{xy} = I_{yx} = -\iiint xy\, \rho(x, y, z)\, dx\, dy\, dz \tag{13}$$

$$I_{xz} = I_{zx} = -\iiint xz\, \rho(x, y, z)\, dx\, dy\, dz \tag{14}$$

$$I_{yz} = I_{zy} = -\iiint yz\, \rho(x, y, z)\, dx\, dy\, dz \tag{15}$$

Where ρ(x,y,z) is the density of the object, giving its position.

3.4 Inertial tensor of cylindrical shape

To simplify the robot links are considered to be cylindrical solid with uniform density. So the body is a cylinder of height h, radius r of mass M and with constant density.

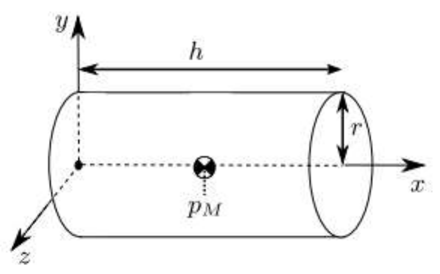


Fig -3: Shape of simplified link

The inertia tensor relative to the Oxyz frame as shown in Fig -3 is:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{16}$$

where

$$I_{xx} = \frac{1}{2}Mr^2 \tag{17}$$

$$I_{yy} = I_{zz} = \frac{1}{12}M(3r^2 + h^2) \tag{18}$$

3.5 Equation of motion

Generally, there are two formulation of the equation of motion: the formulations of Lagrange – Euler and Newton – Euler.

Lagrange – Euler formulation:

Lagrange – Euler write the equation in terms of kinetic energy and potential energy. The Lagrangian $\mathcal{L}$ is defined as the difference between kinetic energy and potential energy. For a manipulator robot, the Lagrangian is :

$$\mathcal{L}(q, \dot{q}) = K(q, \dot{q}) - V(q) \tag{19}$$

In which $K(q, \dot{q})$ is the kinetic energy and $V(q)$ is the potential energy. Then the equation of motion is:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = Y \qquad (20)$$

Where, Y is the generalized external force.

Kinetic energy consists of two terms, linear and angular kinetic energy. It is written as follows:

$$K = \frac{1}{2}mv^T v + \frac{1}{2}\omega^T I\omega \qquad (21)$$

m is the total mass of the object, v and ω are the linear and angular velocities respectively, and I is the inertia tensor. In which the inertia tensor is written as:

$$I = RIR^T \qquad (22)$$

A more developed writing of K are:

$$K = \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n}\{m_i J_{v_{mi}}(q)^T J_{v_{mi}}(q) + J_{\omega_{mi}}(q)^T R_i(q) I_{mi} R_i(q)^T J_{\omega_{mi}}(q)\}\right]\dot{q} \qquad (23)$$

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q} \qquad (24)$$

For our case (case of non-deformable solids) the only potential source of energy is gravity. The potential energy of link number i can be calculated by considering that the mass of the whole object is concentrated at its mass centre and given by:

$$P_i = m_i g_0^T r_{mi} \qquad (25)$$

where m_i is the mass, g_0 is a vector giving the direction of gravity and the vector r_{mi} gives the coordinates of the mass center. The total potential energy of the n-links robot is therefore:

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g_0^T r_{mi} \qquad (26)$$

Finally, the equation of motion of Lagrange – Euler is:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \qquad (27)$$

where D(q) is the inertia matrix defined by the expression of K.
The elements of the $C(q,\dot{q})$ matrix are defined by:

$$C_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \qquad (28)$$

c_{ijk} are Christoffel’s symbols.

Newton – Euler formulation:
This formulation is based on expressions of force and Momentum. It results from two equations: the Newton dynamic equation which governs the rectilinear motion of a solid, and the Euler equation which concerns the rotational motion.

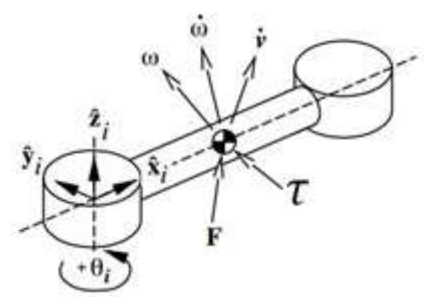


Fig -4: Link subjected to a force F and a couple τ

Newton’s equation expresses the equality of the external force and the variation of the momentum.

$$F = \frac{d}{dt}(mv) = m\dot{v} \tag{29}$$

Euler’s equation expresses the equality of the external torque and the angular momentum.

$$\tau = \frac{d}{dt}(I\omega) = I\dot{\omega} + \omega \times I\omega \tag{30}$$

Written in matrix form, the two equations can be summarized as one.

$$\begin{bmatrix} mI_3 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times mv \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{31}$$

I_3 is the identity matrix, I the inertial tensor with respect to the center of mass of the link.

3.6 Inverse dynamic

The problem of the inverse dynamic is: how to find the force F and/or torque **τ** to apply to obtain a given trajectory q(t) ? If you look at the equation of motion:

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{32}$$

this is a type of function : $(q,\dot{q},\ddot{q}) \longmapsto \tau$. It can be directly used for commanding the robot.

3.7 Direct dynamic

The problem of direct dynamics is: how to find the trajectory followed by the robot knowing the force and or the torque applied? In this case, he wants to know the robot’s response if a given force or torque is applied.
By rearranging the equation of motion:

$$\ddot{q} = D^{-1}(q)\,\{\tau - C(q,\dot{q})\dot{q} - g(q)\} \tag{33}$$

$$\dot{q} = \int \ddot{q}\ dt \tag{34}$$

$$q = \int q\ \dot{}dt \tag{35}$$

This is a type function: $\longmapsto (q,\dot{q},\ddot{q})$. It’s used to simulate the robot.

4. PID CONTROL

The low-level control i.e. the control at the level of each joint can be considered as an internal routine of the robot. The PID type controller is very common in engineering and robotics. We will use a PID controller for our modelling and simulations. Directly, at the input of joints the motor applies force or torque. With PID controller, we have:

$$\tau = K_p\theta_e + K_i \int \theta_e(t)\,dt + K_d\dot{\theta}_e \tag{36}$$

K_p, K_i, K_d are proportional, integral and derivative gains respectively. $\theta_e = \theta_d - \theta$ is the error with respect to the reference θ_d . From this equation, we have the bloc diagram of the Fig 5.

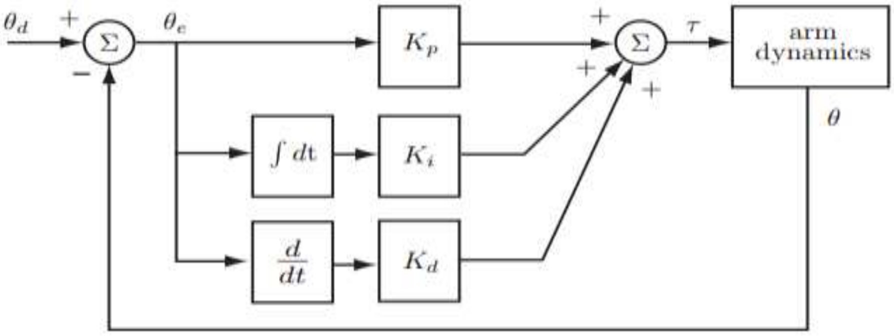


Fig -5: PID control of the joint angle

4.1 PID tuning

To have stable and robust controls, the values of the PID controller parameters have to be ajusted in order to find the optimal values. Fig -6 shows the bloc diagram of PID control.

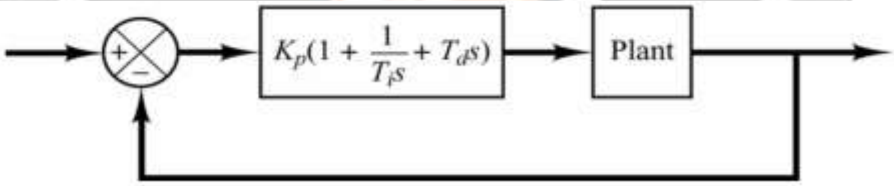


Fig -6: Bloc diagram of PID control

The transfer function of the controller is:

$$\frac{U(s)}{E(s)} = G_{PID}(s) = K_p + K_i \frac{1}{s} + K_d s = K_p(1 + \frac{1}{T_i s} + T_d s) \qquad (37)$$

To study the dynamic of the system, the step response has to be examined. The step response characteristics are:

- Rise time (T_m) is the time taken by a signal to change from 10% to 90% of the finale value.
- Overshoot (D) is the occurrence of a signal or function exceeding its target.
- Setting time (T_r) is the time required for the response curve to reach and stay within a range of 5% of the final value.
- Static error (e_{rp}) is the difference between final value and the input.

These characteristics of the index response are illustrate in Fig 7.

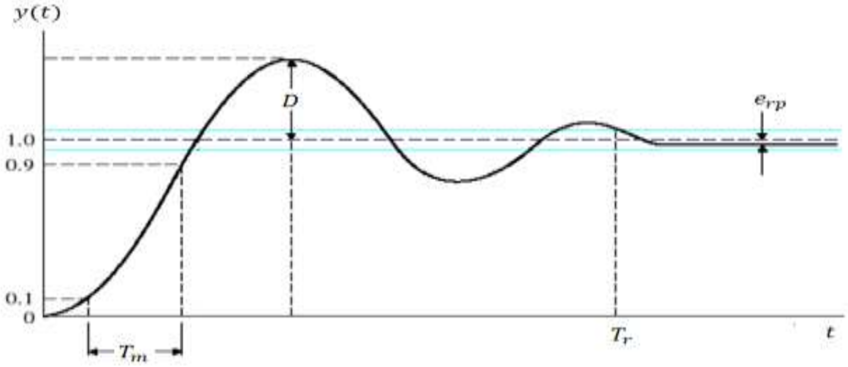


Fig -7: Step response characteristics

System impacts for increasing values of the parameters K_p, K_i , $K_d$are summarized in the Table -2.

Table -2: Summary of the influence of PID gains on system dynamics

	Rise time	**Overshoot**	**Setting time**	**Static error**
$\boldsymbol{K_p}$	Decreases	Increase	Sans Effet	Decreases
$\boldsymbol{K_i}$	Decreases	Increase	Increase	Removes
$\boldsymbol{K_d}$	No effect	Decreases	Decreases	No effect

Ampirical procedure for setting the PID parameters is given by the following recipies.

- First, looks which parameter need to be improved.
- Secondly, change K_p in order to decrease T_m.
- Thirdly, change K_d in order to decrease D.
- Fourth, use K_i to eliminate e_{rp}.
- return to the first stage if not yet satisfied.

This procedure gives good results quite quickly, most of the time provided that the initial choices of $K_p, K_d$and K_i are not very far from the optimal values. But there is no guarantee that the search for initial parameters will be so quick or so easy.

4.2 Ziegler-Nichols methods

Two cases are to be considered.
First, there is the particular case where the step response is an S-shaped curve with no significant overshoot. The system has neither an integrator nor dominant conjugate poles. Fig -8 shows the characteristics of this S-shaped curve.

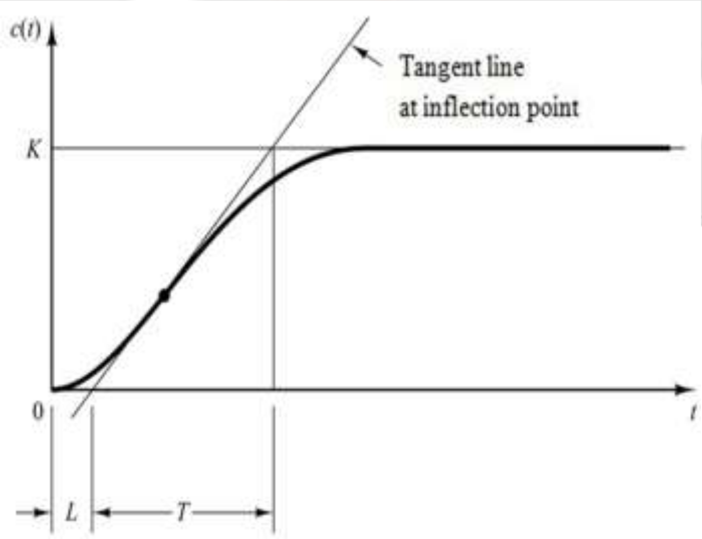


Fig -8 : S-shaped curve step response

The tangent line at the inflection point gives delay L and time constant T. The controller gains proposed by Ziegler and Nichols are those of Table 3.

Table -3: Ziegler and Nichols gains for the Fig -8 case

Controller types	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\ \frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\ \frac{T}{L}$	$2\ L$	$0{,}5\ L$

Secondly, the most general case where the step response looks more like the curve in Fig -7.

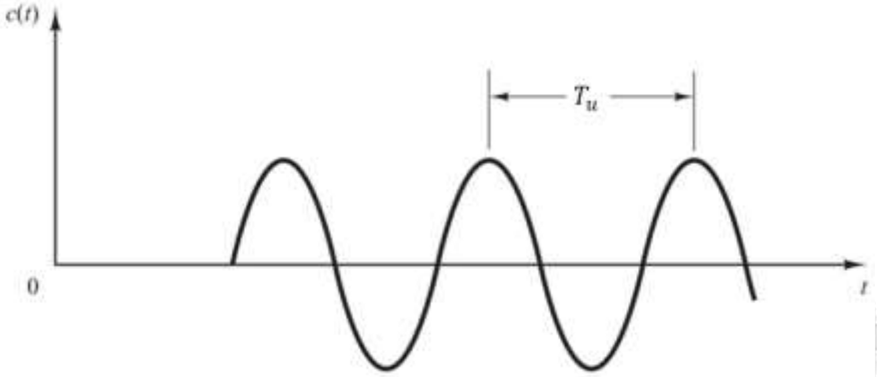


Fig -8: Oscillation curve at the output when $(K_i, K_d) = (0, 0)$ and $K_p = K_u$

The method is as follows. First the integral and derivative gains are cancelled. Then the proportional gain is increased until the closed loop output signal oscillates in a sustained manner (Fig -8). This critical proportional gain K_u and the corresponding oscillation period T_u are then recorded.

In this case Table -4 is to be used.

Table -4: Ziegler-Nichols gains for the second case

Controller types	K_p	T_i	T_d
P	$0{,}5\ K_u$	∞	0
PI	$0{,}45\ K_u$	$\frac{T_u}{1{,}2}$	0
PID	$0{,}6\ K_u$	$0{,}5\ T_u$	$0{,}125\ T_u$

Each gain is calculated based on the numerical values of Ku and Tu. Table -4 also shows the parameters for type P and PI controllers which are simpler PID variations.

5. MODELLING

The modeling is done with ROS. Its description is written in XML file in URDF format. An overview of the robot in Rviz is given by Fig -9.

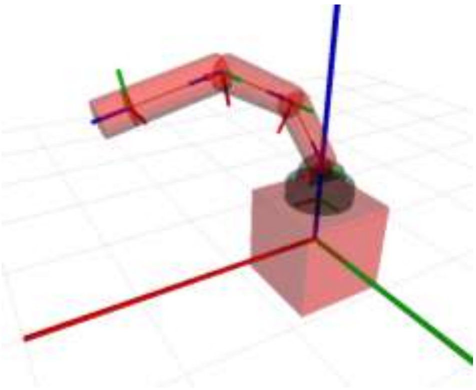


Fig -8: Visualization of the robot model

4.1 Implementation

There are quantities variables needed in order to implement the model. Variables and values of the model implement cinematic, dynamic and geometric properties. Table -5 repertories link's variables and their respective values.

Table -5: Link's variables and values

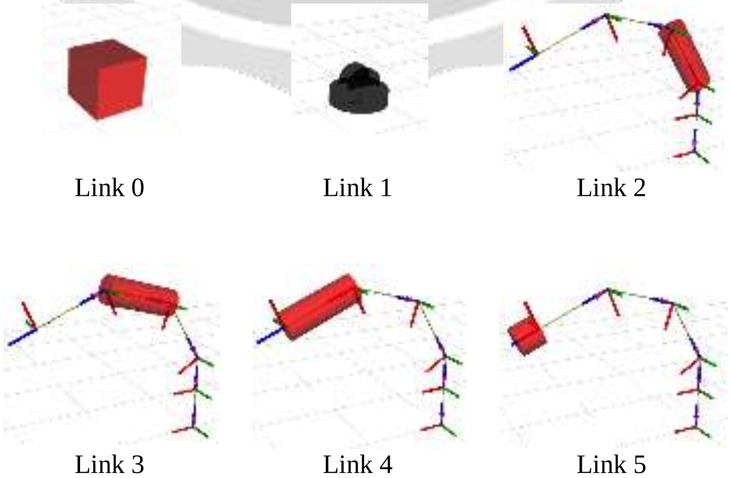
Link #i	Mass	Inertial tensor	shape	Dimensions
0	1024	$I = \begin{bmatrix} 170.667 & 0 & 0 \\ 0 & 170.667 & 0 \\ 0 & 0 & 170.667 \end{bmatrix}$	cubic	edge=1
1	9.655	$I = \begin{bmatrix} 13.235 & 0 & 0 \\ 0 & 13.235 & 0 \\ 0 & 0 & 9.655 \end{bmatrix}$		

2	57.906	$I = \begin{bmatrix} 12.679 & 0 & 0 \\ 0 & 12.679 & 0 \\ 0 & 0 & 0.651 \end{bmatrix}$	cylindrical	radius=0.15
				height=0.8

3	57.906	$I = \begin{bmatrix} 12.679 & 0 & 0 \\ 0 & 12.679 & 0 \\ 0 & 0 & 0.651 \end{bmatrix}$	cylindrical	radius=0.15
				height=0.8

4	57.906	$I = \begin{bmatrix} 12.679 & 0 & 0 \\ 0 & 12.679 & 0 \\ 0 & 0 & 0.651 \end{bmatrix}$	cylindrical	radius=0.15
				height=0.8

5	18.056	$I = \begin{bmatrix} 0.479 & 0 & 0 \\ 0 & 0.479 & 0 \\ 0 & 0 & 0.204 \end{bmatrix}$	cylindrical	radius=0.15
				height=0.25



Link 0 Link 1 Link 2

Link 3 Link 4 Link 5

Fig -9: Visualization of individual link shape

The images of each articulation are shown in Fig -9. Local frames and relative position vectors are represented to illustrate relative joint positions. The articulations number 2, 3 and 4 are geometrically identical. Table -5 shows that their dynamic properties are also identical. i.e. they have the same inertia tensor, the same mass, the same dimensions and shape.

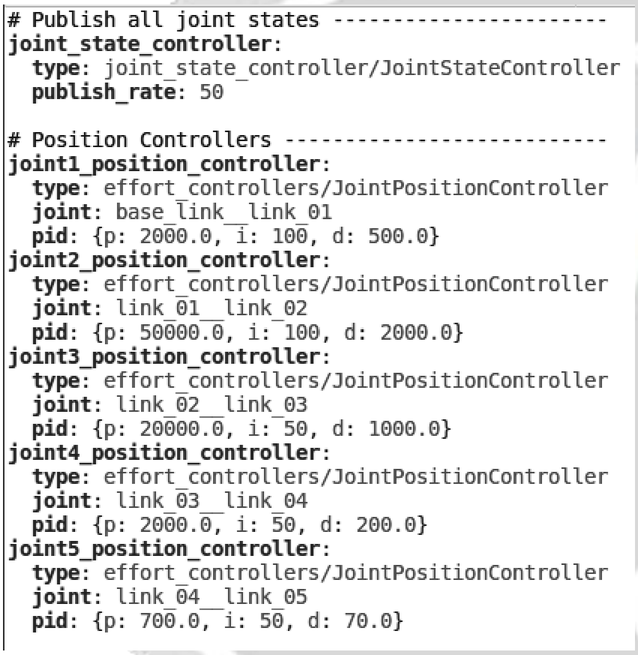
4.2 PID controller

Joint level controls are implemented with the ros_control, a set of packages that include controller interfaces, controller managers, transmissions and hardware_interfaces. Fig -10 present an overview of how it works. For more information we recomand [09], an ROS control documentation.

PID gains and the types of controllers and the joints on which they work must be specified in a YAML file. Table -6 is summarizes these information:

Table -6: PID implementation on the model

YAML code	**PID parameters**


# Publish all joint states ---------------------- 	
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50
 	
# Position Controllers -------------------------- 	
joint1_position_controller:
  type: effort_controllers/JointPositionController
  joint: base_link__link_01
  pid: {p: 2000.0, i: 100, d: 500.0}
joint2_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_01__link_02
  pid: {p: 50000.0, i: 100, d: 2000.0}
joint3_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_02__link_03
  pid: {p: 20000.0, i: 50, d: 1000.0}
joint4_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_03__link_04
  pid: {p: 2000.0, i: 50, d: 200.0}
joint5_position_controller:
  type: effort_controllers/JointPositionController
  joint: link_04__link_05
  pid: {p: 700.0, i: 50, d: 70.0}	

	Joint #**1** $K_p, K_i, K_d = [2000 \quad 100 \quad 500]$
	Joint #**2** $K_p, K_i, K_d = [50000 \quad 100 \quad 2000]$
	Joint #**3** $K_p, K_i, K_d = [20000 \quad 50 \quad 1000]$
	Joint #**4** $K_p, K_i, K_d = [2000 \quad 50 \quad 200]$
	Joint #**5** $K_p, K_i, K_d = [700 \quad 50 \quad 70]$

ROS works with various module of program. The module called "JointStateController" returns information from joint positions. And the "JointPositionController" module receives the input controls for the robot's model.

4.3 Simulations

Les simulations sont faites (are carried out) dans ROS. Les rendus graphique du robot sont produites par les logiciels Rviz. Les courbes sont tracées à partir des données fournies par ROS.

Simulations are carried out in ROS. Graphics are produced by Rviz software. The curves are plotted from the data provided by ROS.

For each joint, the following signal curves are displayed:

- input signal, i.e. the command curve,
- output signal, i.e., the actual position of the joint,
- error curve at the input of the PID controller,
- control law curve at the output of the PID controller.

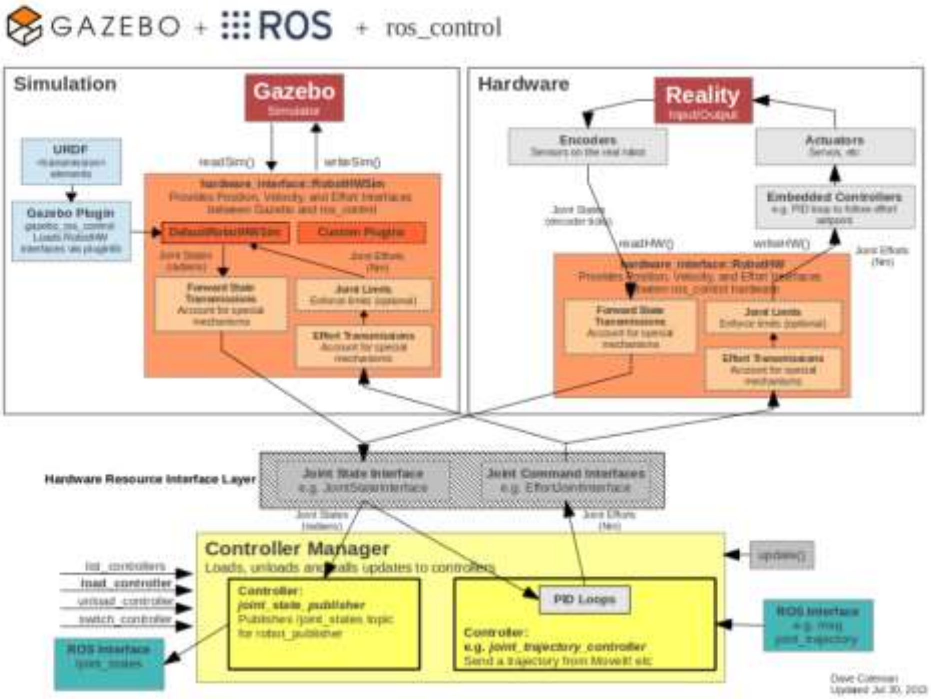

GAZEBO + ROS + ros_control
Simulation
Gazebo
Simulator
URDF
Gazebo Plugin
readSim()
writeSim()
hardware_interface::RobotHWSim
Provides Position, Velocity, and Effort Interfaces between Gazebo and ros_control
DefaultRobotHWSim
Custom Plugins
Joint States
Joint Efforts
Forward State Transmissions
Account for special mechanisms
Joint Limits
Enforce limits (optional)
Effort Transmissions
Account for special mechanisms
Hardware
Reality
Input/Output
Encoders
Sensors on the real robot
Actuators
Servos, etc
Embedded Controllers
e.g. PID loop to follow effort setpoint
readHW()
writeHW()
hardware_interface::RobotHW
Hardware Resource Interface Layer
Joint State Interface
e.g. JointStateInterface
Joint Command Interfaces
e.g. EffortJointInterface
Controller Manager
Loads, unloads and calls updates to controllers
list_controllers
load_controller
unload_controller
switch_controller
ROS Interface
joint_states
Controller:
joint_state_publisher
Publishes /joint_states topic for robot_publisher
PID Loops
Controller:
e.g. joint_trajectory_controller
Send a trajectory from MoveIt! etc
update()
ROS Interface
e.g. msg
joint_trajectory
Dave Coleman
Updated Jul 30, 2013

Fig -10: Overview of the use of the package ros_control in ROS and GAZEBO

Chart -1 to 5 are the curves for the robot's five joints.

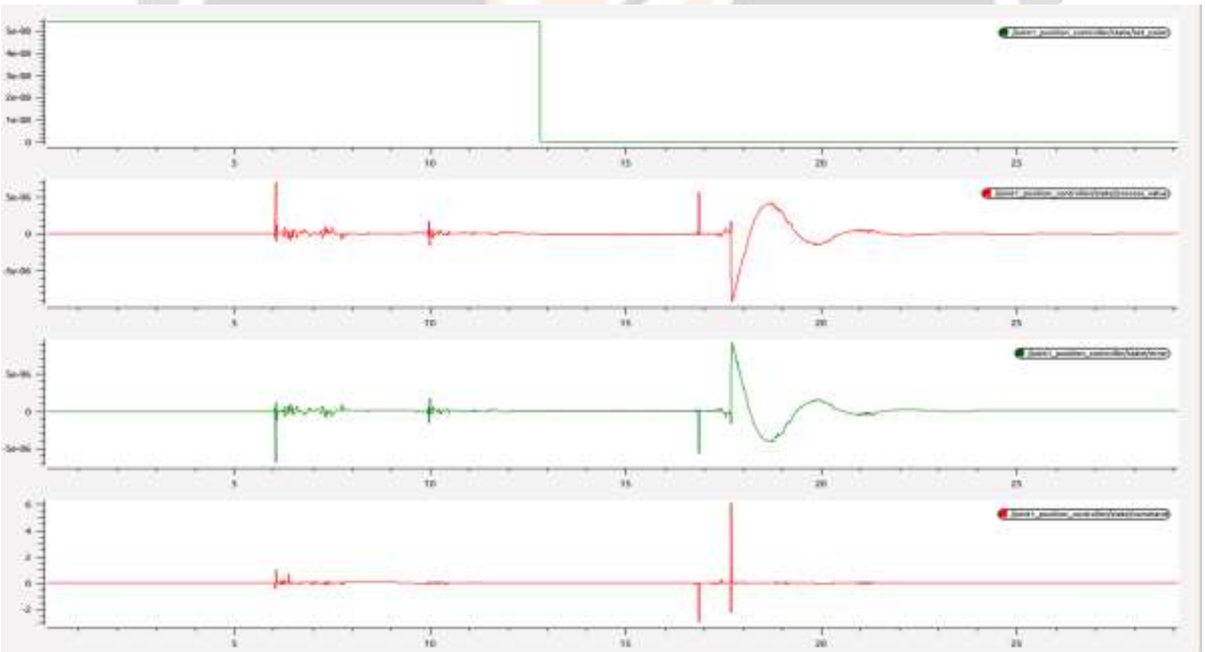


Chart -1: Curve for Joint 1

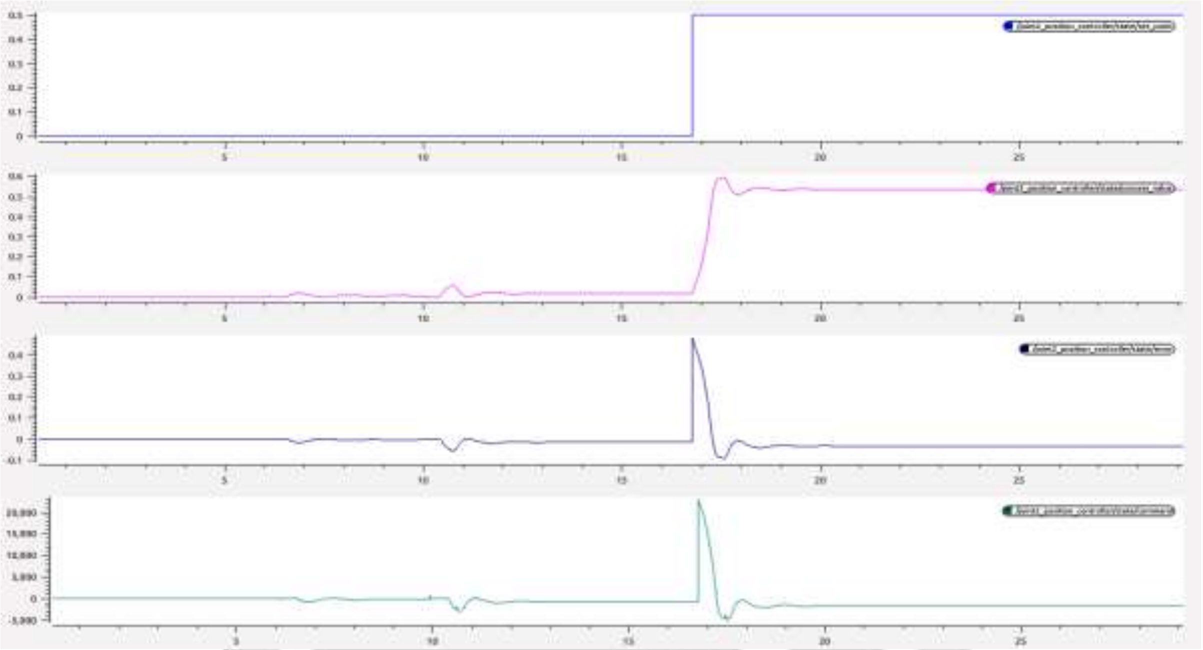


Chart -2: Curve for Joint 2

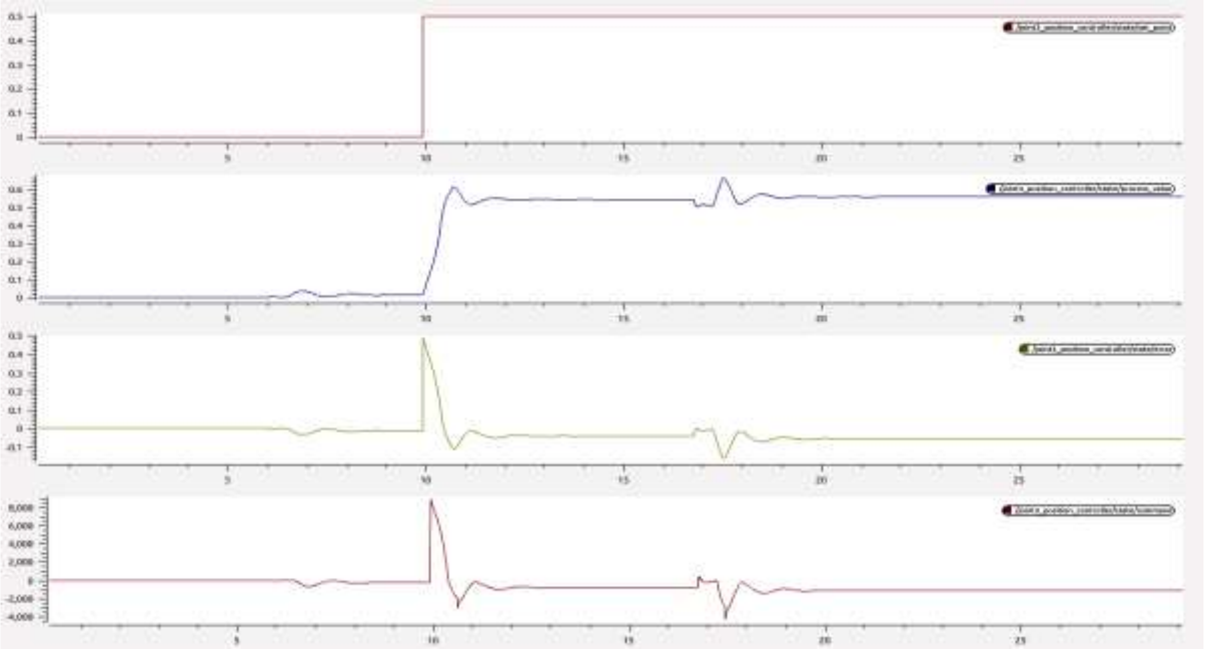


Chart -3: Curve for Joint 3

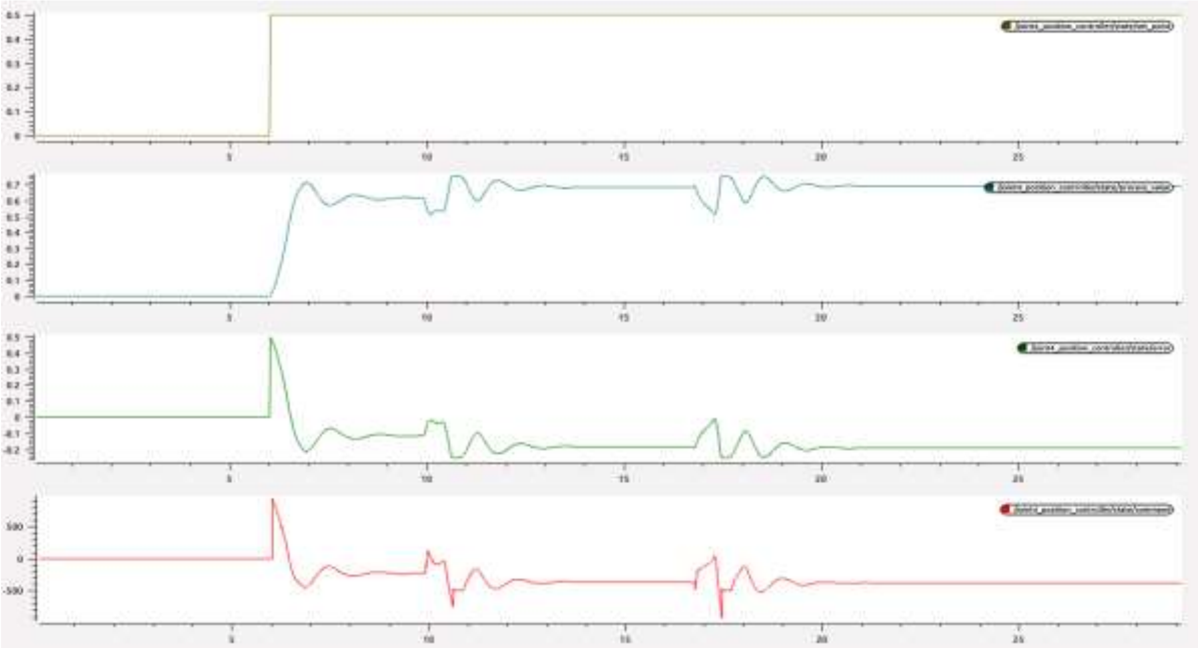


Chart -4: Curve for Joint 4

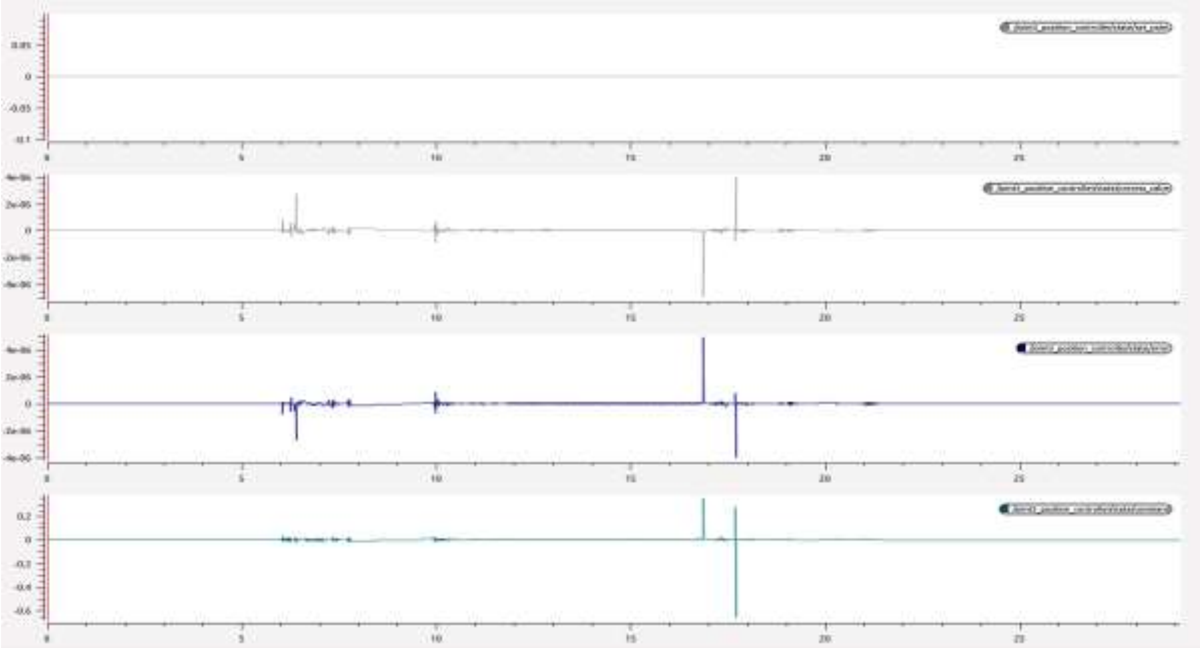


Chart -5: Curve for Joint 5

4.4 Observations

The first joint to be solicited by an order is the number 4 on Chart-4. It presents a very realistic behavior. We also notice that the movements on one joint cause small jolts on the others. This is normal because they are mechanically chained in the same robot. The reactions are quite similar for the other joints except for number 5 (Chart -5). The explanation is that the axis of the axis of rotation of this latter joint is orthogonal to those of the others.

5. CONCLUSIONS

The robot model is functional and has fairly realistic kinematic and dynamic behaviors. The controllers work correctly. We are rather confident about its use in the rest of our work. This model will then be used to study commands using reinforcement learning algorithms. We think that this paper presents a way to model a robot, which will not necessarily be a manipulator. Hopefully our work can be useful to others.

6. REFERENCES

[1]. S.K.Saha, Tata McGraw-Hill, “Introduction to Robotics”, New Delhi, July 28, 2010

[2]. I. Zamora, N. G. Lopez, V. M. Vilches, A. H. Cordero, “Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo”, Article, Erle Robotics, 2017.

[3]. C. Yang, “Advanced Technologies in Modern Robotic Applications”, Science Press and Springer Science+Business Media Singapore, 2016.

[4]. K. Kufieta, “Force Estimation in Robotic Manipulators:Modeling, Simulation and Experiments”, Thesis, Departement of Engieneering Cybernetics, NTNU Norwegian University of Science and Technology, 2014.

[5]. N. Munro, F. L. Lewis, “Robot Manipulator Control Theory and Practice”, Second Edition, University of Manchester, UK, Automation & Robotics Research Institure – University of Texas Arlington, 2004.

[6]. Prof. A. De Luca, “Dynamic model of robots: Newton-Euler approach”, Dipartimento Di Ingegneria Informatica Automatica, Università Di Roma.

[7]. M. W. Spong, “Motion Control of Robot Manipulators”, Article, The Coordinated Science Laboratory, University of Illinois at Urbana-Champaign,

[8]. S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. R. Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Ludtke, E. F. Perdomo, “ros_control: A generic and simple control framework for ROS”, Article, The Journal of Open Source Software, 2017.

[9]. R. M. Murray, California Institute of Technology, Z. Li, Hong Kong University of Science and Technology, S. S. Sastry, University of Califorina, Berkeley, “A Mathematical Introduction to Robotic Manipulation”, CRC Press, 1994.

[10]. B. Ponleitner, “Fundamental Mathematical Concepts for Problems Arising in Robotics”, Thesis, University of Wien, 2013.

[11]. A. R. Tsouroukdissian, “ROS control, an overview”, Conference, ROSCon 2014, 2014.

[12]. A. Mahtani, L. Sanchez, E. Fernandez, A. Martinez, “Effective Robotics Programming with ROS”, Third Edition, Packt Publishing, Livery Place 35 Livery Street Birmingham B3 2PB, UK, 2016.

[13]. K. M. Lynch, F. C. Park, “Modern Robotics – Mechanics, Planning, and Control”, Cambridge University Press, 2017.