ADAMS-MATLAB Co-Simulation of A Serial Manipulator

Tejaswin Parthasarathy, Vignesh Srinivasaragavan and Soundarapandian Santhanakrishnan

Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai 600036, India

Abstract. This paper presents the dynamic modelling and simulation of a now redundant robot, Mitsubishi RM-501, and proposes a general algorithm for experimental simulation in kinematics, dynamics and control analysis to any such robot. Through reverse engineering, a model as accurate as the real robot was developed in SolidWorks.The simulations of the same were performed in ADAMS (dynamicmodeling software offered by MSC Software Corp)along with MATLAB for motion studies and control dynamics. Finally, with a user-input path the accuracy and precision of the simulator was verified.

1 Introduction

The institute of robotics in United States of America (USA) gives the definition for a robot as "a reprogrammable, multifunctional manipulator design to move material, parts, tools, or specialized devices through various programmed motions for the performance of variety of tasks." Nowadays, robots in industries have a wide spectrum of applications. The most notable ones are the welding and painting robots in car plants [1], electronic board assembly, repairing nuclear installations [2] etc. Moreover, many academic and research laboratories have been trying and developing new methods (motion planning, manipulationplanning, grasp planning) and algorithms (position and force control algorithms) in order to extend its usage to diverse applications. High amount of cost and time is involved in testing and validation of robots pave the way to use simulated models, as a cost–effective approach and areextensively demanded in research labs.

Simulation and analysis of PUMA 560 [3] and Stabuli TX-4.0 [4] have previously been performed. However, inverse kinematics, dynamic analysis and implementation of a control algorithm have not been done so far. Thus, enough thrust has to be given holistically to satisfy the present needs. This is important in an economic, research and environmental point of view, and the present work is aimed to achieve the said goals.

This paper presents a detailed procedure for virtual testing offive degrees of freedom (5-DOF) type model based on the Mitsubishi RM-501 robot through simulations. All procedures followed are presented in this paper in detail such a way that the reader can model any serial robot manipulators in the given software environments.

The main highlight of the present work is to demonstrate easy ways of co-simulation, accurate controlling of system design as an insight into trajectory planning. After the construction of a three-dimensional (3D) model of the robot manipulator in DDS SolidWorks, the model is then exported to ADAMS dynamic modeling software for simulating kinematics and dynamic behavior of robotic manipulator with an acceptable error (compared to the real models). For obtaining an exact model, inertial and geometric parameters are accurately measured and recorded in the software database, for simulating control algorithms, simulation between ADAMS and MATLAB is performed.

2 Modelling

The robot itself is made up of 5 links namely: base, body, shoulder, elbow and wrist. Each of this entity was designed in SolidWorks. The sketches and dimensions were made using the data extracted from [5] and the model was made accordingly. Then these parts were assembled in the 'Assembly' environment, using appropriate constraints as deemed fit. The process of defining all the constraints was started with fixing the base of the manipulator (to which the universal frame was attached). Then the 'Coincidence' and 'Concentric' commands were used to couple the other parts as deemed fit. The revolute joints were used to couple the links in RM-501, thus simplifying the construction of the model and its assembly.

Once the assembly is done, the "Motion Study" module in SolidWorks was usedto verify the robots predetermined DOF. The "Motion Study" also provides an option to export the geometric model to the ADAMS environment for further processing.

2.1 Importing into ADAMS

ADAMS software was used for the dynamic analysis of the robot model. The basic pre-requisites includes geometric correctness and the validity of constraints were done in SolidWorks. The material of the robot parts (as this determines the mass and other basic dynamic parameters) is equally significant as observed during reverse engineering (RE). Once this is done in SolidWorks the "Motion Study" itself is capable of exporting these details along with the geometric model to ADAMS. Verification of the coherence in inertial parameters can be confirmed in both ADAMS and SolidWorks platforms (Table 1).

Table 1. Inertial parameters of RM-501

Link	m (kg)	I_{xx} x 10^{-3} (kg.m^2)	I_{yy} x 10^{-3} (kg.m^2)	I_{zz} x 10^{-3} (kg.m^2)
Base	NA	NA	NA	NA
Body	2.627	22.218	69.890	61.674
Shoulder	4.875	551.141	123.225	73.646
Elbow	1.839	7.152	7.152	5.399
Wrist	1.202	0.834	0.887	0.952
End Effector	0.185	0.078	0.078	0.033

*m is the mass of each link
**I is the Moment of Inertia about the parts Centre of Mass (CoM)
*** *x,y,z*are defined according to the local (not fixed) coordinate system attached to each and every part

The motion in ADAMS can be given in two different ways. The first one involves providing the direct motion of the joint (in terms of angular displacement/velocity/acceleration) in SolidWorks Motion Manager and importing it to ADAMS. This ensures that the same motion is replicated in ADAMS and later it can be changed according to the needs. The other way is fundamentally cumbersome but more practical, which involves the user giving the necessary joint torques so that the end effector follows the desired path. This involves using the V_TORQUE option in ADAMS in each and every joint, which essentially creates a 3D torque vector. Non-zero values were then filled in for the Z-direction component in each V_TORQUE, while the other two components were made zero as practical actuation was done by using individual servo motors for a joint. The servo motors can provide torque only in one direction. It was noted that chain drives and gear systems should also be incorporated into the model. It was done manually, while defining the parameters of the individual motors. This paper illustrates the second approach, which involves computing the torque (and hence power) required.

3 Kinematic Analysis

Kinematic analysis in a broad sense involves both forward and inverse kinematics. Emphasis should be given equally for both as they directly determines the joint torques and other related parameters in order to make the end effector of the robot to precisely follow a pre-defined path. The coordinate's representation [6] and the modified Denavit-Hartenberg parameters are shown in Figure 1 and Table 2, respectively.

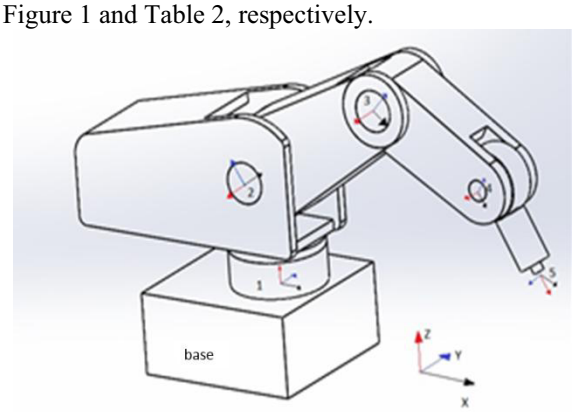

Figure 1. The assembled model of the Mitsubishi RM 501, with the attached frames and coordinate systems

Table 2. Denavit-hartenberg parameters of RM-501

Link (i)	α_{i-1}	a_{i-1}	d_{i-1}	θ_{i+1}
Base	0	0	0	θ_1
Body	90	0	0	θ_2
Shoulder	0	220	0	θ_3
Elbow	0	160	0	90+ θ_4
Wrist	90	0	0	θ_5
End Effector	0	0	137	0

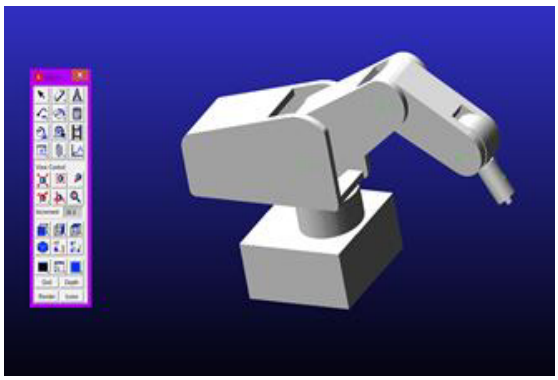

Figure 2. State of the home position of the robotic arm in the ADAMS environment

Care must be taken while importing the CAD model of the robot into ADAMS. The position of joints must eithercorrespond with the default 'Home' position (Figure 2) or a state where all the initial joint parameters, namely the five joint angles are known. Throughout this paper the positive direction of rotation is compatible with the right handed direction of Z axis.

Quintessential to any kinematic calculations are transformation matrices and there are six of them (one for each degree of freedom and one for the end-effector orientation).In this context, one each from the previous coordinate system to the next. Transformation matrices provide complete information about the frames. The information includes the orientation and position of

frames, along with information on any vector of a particular frame, if that vector is viewed from some other frame. Transformation matrices for RM 501 are elaborated in next section.

3.1 Forward kinematics

Given the set of joint angles (θ_i) at any instance, the forward kinematics is simple, as the basic equations of forward kinematics easily determines the end-effector position and orientation. The modified Denavit - Hartenberg (DH) convention and methodology was used to derive the kinematics.

The following convention was followed: (x_0, y_0, z_0) to (x_4, y_4, z_4) represent the local coordinate frames at the five joints respectively, (x_5, y_5, z_5) represented the local coordinate frame at the end-effector, α, γ and θ are the rotation angles about *x*, *y*, and *z* axis respectively. The base local coordinate frame (x_0, y_0, z_0) overlapped with the global coordinate system (X, Y, Z) as it was attached to the base frame.

Based on the DH convention, the transformation

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

matrix from joint n to joint n + 1 is given by:

, where θ and α are DH parameters and cθi, sθi represent cos (θi) and sin (θi), respectively.

Another mathematical entity which is frequently used for describing the frame motion is a 3×3 rotation matrix ($^{i-1}_{i}R$) that is essentially the first 3 rows and 3 columns of the transformation matrix ($^{i-1}_{i}T$). The transformation matrices for all the frames are listed here. Hence, the position and orientation of the end-effector can be calculated if all the joint angles are given. Once the joint angles are found, other kinematic variables can be calculated through forward kinematics using the following equations.

$$ {}^{i+1}_{i+1}\omega = {}^{i+1}_{i}R\,{}^{i}_{i}\omega + \dot{\theta}_{i+1}\,{}^{i+1}_{i+1}\hat{Z} \quad (2) $$

$$ {}^{i+1}_{i+1}\dot{\omega} = {}^{i+1}_{i}R\,{}^{i}_{i}\dot{\omega} + {}^{i+1}_{i}R\,{}^{i}_{i}\omega \times \dot{\theta}_{i+1}\,{}^{i+1}_{i+1}\hat{Z} + \ddot{\theta}_{i+1}\,{}^{i+1}_{i+1}\hat{Z} \quad (3) $$

$$ {}^{i+1}_{i+1}\dot{v} = {}^{i+1}_{i}R\left({}^{i}_{i}\dot{\omega} \times {}_{i+1}{}^{i}P + {}^{i}_{i}\omega \times \left({}^{i}_{i}\omega \times {}_{i+1}{}^{i}P\right) + {}^{i}_{i}\dot{v}\right) \quad (4) $$

$$ {}^{i+1}_{i+1}\dot{v}_c = {}^{i+1}_{i+1}\dot{\omega} \times {}_{i+1}{}^{i}P_c + {}^{i+1}_{i+1}\omega \times \left({}^{i+1}_{i+1}\omega \times {}_{i+1}{}^{i}P_c\right) + {}^{i+1}_{i+1}\dot{v} \quad (5) $$

Here, 'i' varies from 0 to 5. Furthermore ω is the angular velocity of a link, $\dot{\omega}$ is the angular acceleration of a link, $\dot{\theta}$ is the angular velocity of a joint motor, $\ddot{\theta}$ is the angular acceleration of a joint motor, v is the velocity of a link, $\dot{v}$ is the acceleration of a link and $P$is the distance from the joint. A subscript c indicates the reference to the center of mass of a link.$^{i+1}_{i+1}\hat{Z}$is the unit vector along Z-axis of the frame i+1 . All the above mentioned quantities are 3×1 column vectors (for *x*, *y* and *z* axis).

3.2 Inverse kinematics

Inverse kinematics, as the name suggests compliments the forward kinematics. It deals with identifying the joint angles when the position and orientation of end effector is given. This is very much useful if the robot is to follow a predetermined path.

The non-linearity involved in forward kinematicsmakes the solution to the inverse kinematics not only difficult but also non-unique. A single strategy to tackle the challenging of inverse problem is by using translation Jacobian matrices. This approach isused in this work. The results obtained using Jacobian matrices are in close agreement with the actual angles, provided that the motion imparted to the robot is small between two successive measuring positions.

Let X represent a 6×1 column vector containing the end positions and orientation of the end effector tip.Then the Jacobian (J) for the base-tool transformation matrix $^{0}T_{6}$ is defined as:

$$ J = \left[\frac{\partial X}{\partial\theta_1}\frac{\partial X}{\partial\theta_2}\frac{\partial X}{\partial\theta_3}\frac{\partial X}{\partial\theta_4}\frac{\partial X}{\partial\theta_5}\right] \quad (6) $$

where θ_i represents the corresponding D-H parameter. The forward kinematic equation is:

$$ dX = J(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)d\theta \quad (7) $$

where θ represents 5×1 the column vector containing the relevant D-H parameters. As both sides are deterministic, it is possible to invert the Jacobian using erstwhile θ_i and obtain the value of $d\Theta$ and subsequently the next set of θ_i, according to the equation,

$$ d\Theta = J^{-1}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)dX \quad (8) $$

$$ \Theta_{new} = \Theta + d\Theta \quad (9) $$

It is clearly seen that the Jacobian is a matrix of partial derivate of the position and orientation with respect to the five joint angles. Inverse solutions for the velocity and acceleration of each link with respect to its frame i.e. $\dot{\theta}$ and $\ddot{\theta}$ can be obtained by directly differentiating the obtained values of θ. If the simulation is done in discrete steps, it suffices to differentiate using a first order scheme to obtain the required derivatives.

3.3 Trajectory planning

Our simulation requires, the path (trajectory) as closely spaced discrete points in ADAMS environment 3D space and whose coordinates are stored in a database (Microsoft Excel Spreadsheet), using a specially-constructed macro routine. These points are taken as guides in planning the path to be taken by the robot using the command of inbuilt database read. The interpolation between any pair of these consecutive points is given as a cubic polynomial:

$$ \theta(t) = A_i + B_i t + C_i t^2 + D_i t^3 \quad (10) $$

Here i is the storing indexed positions (points) in the

database. The curve is constructed with two basic constraints. Firstly, the velocity at the starting point and the ending point are zero. Secondly, the velocity at the intermediate points are unique. The coefficients of the polynomials is solved using a Tri-Diagonal Matrix Algorithm (TDMA).

The only disadvantage of a cubic interpolation is the jerks in $\theta(t)$. As we can clearly see the second derivative of $\theta(t)$ yields a linear function which will change between a pair of consecutive input points. Though this could have been avoided by a quadratic polynomial, it would render the equation solution-less due to the constraints involved.

4 Dynamic analysis

Dynamic analysis is the study of motion with regard to the causal entities (forces and torques). Dynamic modeling is vital for simulation and implementation of control algorithm. The Newton-Euler methodology used here is based on Newton's second law,in order to determine inverse dynamics that is used in this paper [6]. This equation can describe the behavior of a robot manipulator link-by-link and joint-by-joint from base to end-effector, called forward recursion and also transfer the essential information from end-effector to base frame, called backward recursion.

The forward recursion dynamic equations for the force and net moment acting on a joint are:

$$ {}^{i+1}_{i+1}F = m_{i+1}\,{}^{i+1}_{i+1}\dot{v}_c \tag{11} $$

$$ {}^{i+1}_{i+1}N = {}^{C_{i+1}}_{i+1}I\,{}^{i+1}_{i+1}\dot{\omega} + {}^{i+1}_{i+1}\omega \times {}^{C_{i+1}}_{i+1}I\,{}^{i+1}_{i+1}\omega \tag{12} $$

where *F* is the force acting on a link, *N* is the moment acting on a link, m is the mass of a link, $\omega$is the angular velocity of a link, $\dot{\omega}$ is the angular acceleration of a link, $\dot{v}$ is the acceleration of a link and ${}^{c}_{i}I$ is the moment of inertia of a link 'i' about its center of mass. ${}^{c}_{i}I$is a 3×3 matrix and function of mass of the body and geometry. The rest of the quantities are 3×1 column vectors (for *x, y* and *z* axis).

The inverse recursion dynamic equations for calculating the joint forces and joint torques are:

$$ {}^{i}_{i}f = {}^{i}_{i+1}R\,{}^{i+1}_{i+1}f + {}^{i}_{i}F \tag{13} $$

$$ {}^{i}_{i}n = {}^{i}_{i}N + {}^{i}_{i+1}R\,{}^{i+1}_{i+1}n + {}^{i}_{i}P_c \times {}^{i}_{i}F + {}^{i}_{i+1}P \times {}^{i}_{i+1}R\,{}^{i+1}_{i+1}f \tag{14} $$

$$ \tau_i = {}^{i}_{i}n^T\,{}^{i}_{i}\hat{Z} \tag{15} $$

where$f$ is the force acting on a joint, n is the moment acting on a joint and τ is the component of n along the local rotational axis. These quantities are 3×1 column vectors (for *x, y* and *z* axis). A subscript *c* indicates the reference to the center of mass of a link.

Based on the same, required forces and joint torques can be calculated easily. It is noted that a pseudo-force is applied on each and every link that simulates gravity in real life situations by explicitly defining the acceleration of the base to be equal to g in the upward direction. Also the maximum force and torque that the end effector experiences are clearly specified for any simulation.

5 Control systems

Robotic control is considered to be the spine of robotics. It involves making a robot manipulator to perform operations automatically and precisely with the help of controllers. Hence it is a vital part in all modern robots. Typically, the controllers take the form of an equation or an algorithm which is realized via specialized computer programs.

Present industrial systems use a combination of proportional, integral and derivative (PID) control. This particular method has a wide range of applicability, ease of implementation and is very robust, while sacrificing some accuracy in comparison to non-linear control methods.The values of K_p, K_d and $K_i$in the PID control loop can be fixed based on the response pattern. With any control system, the key parameters are rise time, overshoot, settling time, and the steady state error for a step input. Each of the three parameters (K_p, K_d, K_i) have distinct effects on the four output parameters above, and fine tuning is necessary to obtain the best possible response. MATLAB also offers a PID tuning algorithm that can help in deciding the values of K_p, K_d and K_i after an iterative process. In the given example, the value of K_p, K_d and K_i are approximately 10 each. The values are obtained with the help of MATLAB toolbox mentioned above. In this simulation a simple PID control system was implemented for each individual angle outputs.

6 Simulation and results

Co-simulation between ADAMS and MATLAB is of paramount importance due to their efficient simulation platform. An acceptable format for the inputs and outputs of each program is required to execute the co-simulation. The objective of co-simulation is to establish a connection so that any change in one of the programs is reflected and further affects the other one, thus utilizing the benefits of both the program modules [7]. To provide a simulation that enables ADAMS to recognize the exported output from MATLAB, there is a need for activating the 'Control' plug-in in the 'Plug-in Manager' of ADAMS and defining the robot as a plant. After activation, a new window appears for the determination of plant and its inputs and outputs.

To call the generated plant in MATLAB, the plant name should be entered in the 'MATLAB command window' such that the computer recognizes and displays the input and output information. By executing the 'Adams_sys' command, the block containing the information about the dynamic model is created and loaded in the SIMULINK environment. The 'Adams-sub' block created in the SIMULINK should be configured as shown in Figure3. This is a vital component, as this block is included in the master simulation and control model, which is again defined in SIMULINK as a separate entity.

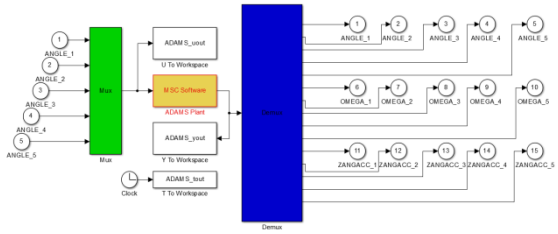

Figure 3. Configured Adams-sub block

The master model, referred to as Master here onwards contains all the modules of the previous sections, which again uses all the concepts (transformation matrices, inverse kinematics) mentioned earlier. The Master contains the trajectory planning module, inverse dynamics module, mainframe model of the robot and its associated control systems. The modules are made of snippets of MATLAB codes hence the blocks are also user-defined and embedded MATLAB functions. Outputs of the dynamics block represent the plant inputs of the robot in ADAMS, namely the joint torques. By starting the simulation in SIMULINK, the signals denoting the joint torques enter the plant and hence runs ADAMS simultaneously which continues to reach the definite final state. The system is also capable of adjusting the size of the discrete steps for each calculation. The generated trajectory functions for each motor yields the end effector to track the desired position and orientation. The flow diagram indicating path generation and inverse dynamics are shown in Figure 4.

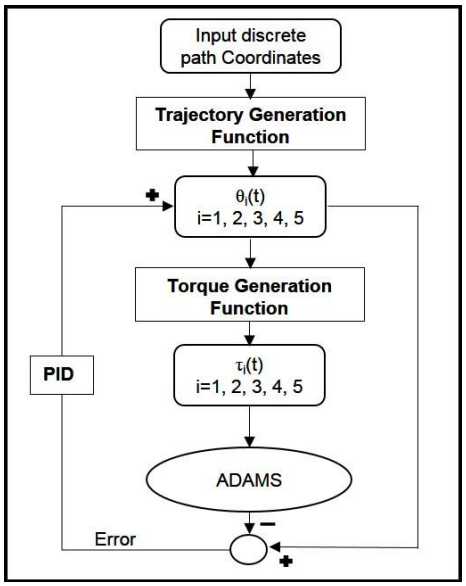

Figure 4. Algorithm for MATLAB/SIMULINK – ADAMS co-simulation

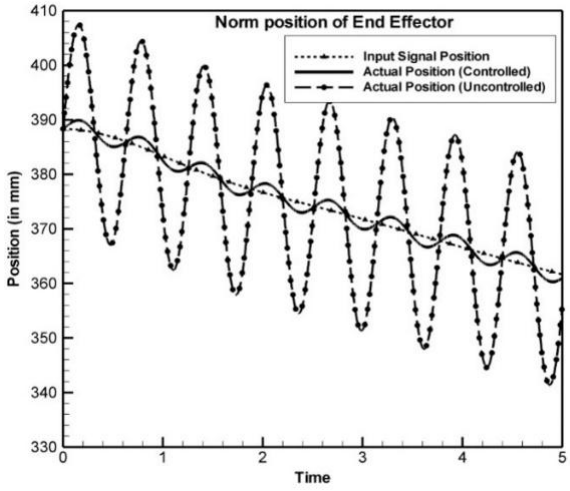

Figure 5. 2nd – Degree Norm Position* of End-Effector (with and without control systems)
*2nd Degree Norm Position is $\sqrt{x^2 + y^2 + z^2}$

By the end of the simulation, the end effector position in the work space is depicted in a scope of MATLAB-SIMULINK. During the simulation, real-time interactive process was shown on the ADAMS interface [8]. By comparing the results with the forward kinematics solution, it can easily be inferred that the simulation runs with the desired accuracy and precision.The graphs were generated in MATLAB-SIMULINK using the 'Scope' module. From simulation results (Figure 5), it can be seen that the links can track the given trajectory curve well with smallerror.

7 Conclusions

In this paper, the virtual development of a 5-DOF robot has been done using SolidWorks and the simulation, testing and validation were carried out using ADAMS and MATLAB-SIMULINK. The validity of the control loop was put to test by imparting a sinusoidal disturbance to the joint torques.Please do note that this observed error was because of the purposeful sinusoidal disturbance. Furthermore, loading the .res file into ADAMS post-processor, it was possible to cross-check all the inertia forces, gravity, centrifugal forces, Coriolis forces and instantaneous power of the driver motorwith the results that were obtained from calculation in MATLAB.

References

1. K.T. Park, Y.J. Shin, C.H. Park, Y.S. Mo, and D.C. Jeong,ICCAS, *"Robot application for assembly process of engine part,"*(2008)
2. D. D. Ray and M. Singh, CARPI,*"Development of a force reflecting Tele-robot for remote handling in nuclear installations,"* (2010)
3. B. Armstrong, O. Khatib, and J. Burdick, ICRA, *"The explicit dynamic model and inertial parameters of the PUMA 560 arm,"* (1986).
4. F. Cheraghpour, M. Vaezi, H. E. ShooriJazeh, and S. A. A. Moosavian, IEEE-ICM "*Dynamic modeling and kinematic simulation of Stäubli© TX40 robot using MATLAB/ADAMS co-simulation,"* (2011).
5. Mitsubishi Electric Corporation Tokyo,*"Movemaster II RM-501 model instruction manual",* (1984).
6. J. J. Craig, *Introduction to Robotics: Mechanics and Control, 3rd ed. United Kingdom: Prentice Hall,* (2003).

7. Z. Yi, X. Min-min, Q. Jin-yi, and Z. Hu, ICCASM,*"Research on co-simulation using ADAMS and MATLAB for automobile active suspension system,"* (2010)
8. R. M. Inigo and J. S. Morton, IEEE Trans. Educ., *"Simulation of the dynamics of an industrial robot,"***vol. 34, no. 1**, pp. 89–99, (1991)