Proceedings of 2016 *4th International Connce on Control Engineering & Information Technology* (CEIT -2016)
16-18 December 2016 - Hammamet, Tunisia

Kinematics Analysis and Simulation of a Robotic Arm using MATLAB

Alla N. Barakat[1], Khaled A.Gouda[2], Kenz A. Bozed[3]

Department of Computer System Design , Faculty of Information Technology, Benghazi University
Benghazi, Libya
[1]a2la1101@gmail.com
[2]khaled.gouda@uob.edu.ly
[3] kenz.bozed@uob.edu.ly

***Abstract*— In order to enhance the study of the kinematics of any robot arm, parameter design is directed according to certain necessities for the robot, and its forward and inverse kinematics are discussed. The DH convention Method is used to form the kinematical equation of the resultant structure. In addition, the Robotics equations are modeled in MATLAB to create a 3D visual simulation of the robot arm to show the result of the trajectory planning algorithms. The simulation has detected the movement of each joint of the robot arm, and tested the parameters, thus accomplishing the predetermined goal which is drawing a sine wave on a writing board.**

***Keywords*— Modeling, Simulation, MATLAB, 6 DOF Robot arm, Forward Kinematics, Inverse Kinematics, Trajectory Planning, Jacobian.**

I. INTRODUCTION

In Kinematic analysis process, we determine the DOF of the robot, then we choose the reference frames and calculate its corresponding homogenous matrix, all using D-H convention which is frequently used for choosing reference frames in robotics applications. Using the D-H table, calculating the forward kinematic equations would be easier, faster and more accurate using Symbolic Math Toolbox, which can later be used in simulating the actual robot arm's movement and its interaction with the surrounding environment. In the simulation part, three different algorithms were used to move the robotic arm, which are Cartesian motion algorithm, differential motion algorithm and joint motion algorithm. The results of these algorithms, along with the 3D visual simulation are all conducted in MATLAB.

A. Kinematic equations of the robotic arm

The Denavit-Hartenberg convention is the most frequently used convention for choosing reference frames in robotics applications. Each homogeneous transformation in this convention is the product of four basic transformations :

$$A_{i-1}^{i} = R_{z,\theta_i}\ Trans_{z,di}\ Trans_{x,ai}\ R_{x,\alpha i} =$$

$$\begin{bmatrix} C\theta_i & -S\theta_i . C\alpha_i & S\theta_i . S\alpha_i & a_i . C\theta_i \\ S\theta_i & C\theta_i . C\alpha_i & -C\theta_i . S\alpha_i & a_i . S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the four quantities ai, di, αi and θi are parameters related to link i and joint i. The four parameters are generally given the names link length, link offset, link twist and joint angle respectively.

the reference frames are positioned as follows:

1. the z-axis is in the direction of the joint axis
2. the x-axis is parallel to the common normal:

$$x_n = z_{n-1} \times z_n$$

If the unique common normal(parallel z axes) doesn't exist, then d (below) is a free parameter .The direction of x_n is from z_{n-1} to z_n .

3. The y-axis follows from the x-axis and z-axis by choosing it to be a right-handed coordinate system[1].

ISBN: 978-1-5090-1055-4

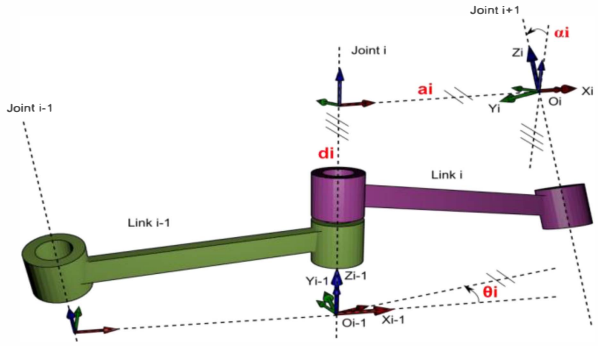

Fig. 1 D-H convention

II. KINEMATIC EQUATIONS OF THE ROBOTIC ARM

The best example to test the simulation is the Stanford robot arm, and figure 2 shows its reference frames chosen using DH convention.

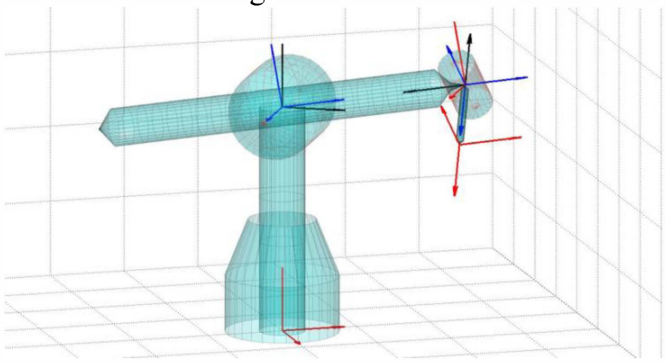

Fig. 2 Robot arm joint axis

The corresponding D-H table (link parameters) is listed in Table 1.

TABLE I
D-H TABLE

joint angle	joint offset	link length	twist angle	cos α	sin α
Θ_i	d_i	a_i	α_i	$C\alpha_i$	$S\alpha_i$
Θ_1	d_1	0	90˚	0	1
Θ_2	0	0	90˚	0	1
$\Theta_{3=0}$	d_3	0	0˚	1	0
Θ_4	0	0	-90˚	0	-1
Θ_5	0	0	90˚	0	1
Θ_6	d_6	0	0˚	1	0

Where d1= 0.6 & d6=0.2 .

A. Forward Kinematics

The homogenous transformation matrix that relates the global coordinate system with the end-effector orientation and position is given by

$$T_0^6 = A_0^1 A_1^2 A_2^3 A_3^4 A_4^5 A_5^6$$

Where A_{i-1}^i can be calculated with D-H convention parameters given in Table 1[2].

(T_0^6) is given by :

$$T_0^6 = \begin{pmatrix} r_{11} & r_{22} & r_{13} & x_6 \\ r_{21} & r_{22} & r_{23} & y_6 \\ r_{31} & r_{32} & r_{33} & z_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where:

$r_{11} = c_6.(c_5.(s_1.s_4 + c_1.c_2.c_4) - c_1.s_2.s_5) + s_6.(c_4.s_1 - c_1.c_2.s_4)$

$r_{12} = c_6.(c_4.s_1 - c_1.c_2.s_4) - s_6.(c_5.(s_1.s_4 + c_1.c_2.c_4) - c_1.s_2.s_5)$

$r_{13} = s_5.(s_1.s_4 + c_1.c_2.c_4) + c_1.c_5.s_2$

$r_{21} = - c_6.(c_5.(c_1.s_4 - c_2.c_4.s_1) + s_1.s_2.s_5) - s_6.(c_1.c_4 + c_2.s_1.s_4)$

$r_{22} = s_6.(c_5.(c_1.s_4 - c_2.c_4.s_1) + s_1.s_2.s_5) - c_6.(c_1.c_4 + c_2.s_1.s_4)$

$r_{23} = c_5.s_1.s_2 - s_5.(c_1.s_4 - c_2.c_4.s_1)$

$r_{31} = c_6.(c_2.s_5 + c_4.c_5.s_2) - s_2.s_4.s_6$

$r_{32} = - s_6.(c_2.s_5 + c_4.c_5.s_2) - c_6.s_2.s_4$

$r_{33} = c_4.s_2.s_5 - c_2.c_5$

$X_6 = d_6.(s_5.(s_1.s_4 + c_1.c_2.c_4) + c_1.c_5.s_2) + d_3.c_1.s_2$

$Y_6 = d_3.s_1.s_2 - d_6.(s_5.(c_1.s_4 - c_2.c_4.s_1) - c_5.s_1.s_2)$

$Z_6 = d_1 - d_6.(c_2.c_5 - c_4.s_2.s_5) - d_3.c_2$

B. Inverse Kinematics (numerical solution)

fsolve(@fun, q0, option) is a built-in function in MATLAB's optimization toolbox that can solve numerically a matrix or vector equation F(q)=0 described in a function subroutine (fun.m) for the initial value q0 of the vector input q and some options. Using fsolve, the inverse kinematics (I-K) can be solved numerically for any robot arm without the need to symbolically solve the I-K. Of course, sometimes the I-K algorithm in its symbolical form is more reliable than the numerical solution, so long as the solution q is not far away from the initial value q(0). Otherwise, some

ISBN: 978-1-5090-1055-4 © 2016 IEEE

unwanted configuration might be the result of using this tool.

The best option for the purposes of this paper is :

```
option = optimset('Algorithm',
  'levenberg-marquardt','Display','off');
```

In mathematics and programming, the non-linear least squares problems can be solved using the damped least-squares (DLS) method, also known as Levenberg–Marquardt algorithm (LMA), which appear specially in least squares curve fitting[3].

III. JACOBIAN MARIX AND SINGULARITIES

Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function. When the matrix is a square matrix. The direct approach of taking the partial derivative of a 6-dimensional orientation/position vector $\xi = \begin{pmatrix} p \\ \rho \end{pmatrix}$ cannot determine the Jacobian matrix J.

for i = 1, … , n if all the n joints are revolute joints, where r_n^{i-1} is the third (last) column of the rotation matrix R_n^{i-1} and represents the z-axis of frame i – 1 with frame n as its reference frame. This shows that:

$$V(n) = \begin{pmatrix} v(n) \\ \omega(n) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} (p_n^{i-1} \times r_n^{i-1}) \dot{\theta}_i \\ \sum_{i=1}^{n} r_n^{i-1} \dot{\theta}_i \end{pmatrix} = J(n)\, \dot{q}$$

If joint j is prismatic, then

$\omega_{j(n)} = 0$, $v_{j(n)} = r_n^{j-1} \times \dot{d}_j$

$$J(n) = \begin{pmatrix} p_n^0 \times r_n^0 & \cdots & p_n^{n-1} \times r_n^{n-1} \\ r_n^{0-1} & \cdots & r_n^{n-1} \end{pmatrix}$$

$$J_{(0)} = \begin{pmatrix} R_0^n & 0_{3\times3} \\ 0_{3\times3} & R_0^n \end{pmatrix} J_{(n)}$$

The resulting Jacobian matrix is a 6×6 matrix. All the possible singular configurations of the 6-DOF robot arm are solutions to the following equation[4] :

$$Det(J) = 0$$

Which, in this case, is :

$$-d_3^{\,2} . \sin\theta_2 . \cos\theta_5 = 0$$

IV. MOTION PLANNING ALGORITHMS

A. Cartesian Motion Planning Algorithm

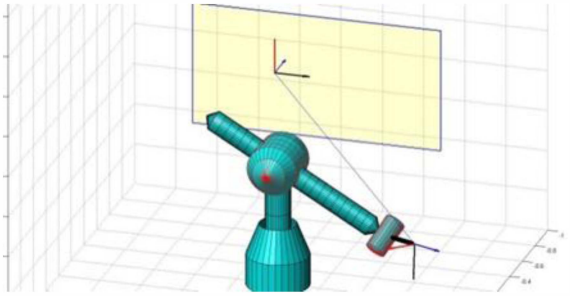

Fig. 3 Cartesian motion path for the Robotic arm end-effector

It's a trajectory sampling process, where you have the chosen end-effector position and orientation (labeled $T_0^6{}_{(N)}$) and the present end-effector orientation and position($T_0^6{}_{(0)}$).

In case that the path of the end-effector was a straight line, the robotic hand's position vector with reference to the base, at the i^{th} sampling step ($0 \le i \le N$), is determined by

$$p_0^{(i)} = p_0^1 + i \cdot \Delta p_0 = p_0^1 + \frac{i}{N} \cdot \left(p_0^2 - p_0^1\right)$$

for the orientation part, we must find R_1^2

$$R_1^2 = R_1^0 \cdot R_0^2$$

Then, using the k–Ø procedure, to convert R_1^2 into the equivalent vector k and angle Ø , and further to sample the angle Ø into N intervals.

$$\Delta_\emptyset = \frac{\emptyset}{N}$$

ISBN: 978-1-5090-1055-4

and then using its forward conversion to determine $R_1^{(i)}$ by using vector k with angle

$$(i \cdot \Delta_{\emptyset}) = \frac{i}{N} \cdot \emptyset$$

into the forward k- Ø procedure. Consequently, at the i^{th} sampling step on the trajectory, the orientation of the end-effector frame with respect to the reference frame is given by[5]

$$R_0^{(i)} = R_0^1 \cdot R_1^{(i)}$$

B. Joint Motion Planning Algorithm

We use the I-K (inverse kinematics) algorithm, in the joint motion path-planning, for a given robot arm at the initial point with H_0^1 and the destination point with H_0^2 of the wanted trajectory. If the I-K solution is q^I for all joint positions based on H_0^1 and that based on H_0^2 is q^{II} , we should be able to calculate how much, in each sampling interval, each joint needs to move by

$$\Delta q = \frac{q^{II} - q^I}{N}$$

Finally, at the i^{th} sampling point ($0 \leq i \leq N$), all the joint positions are given as[5]

$$q^{(i)} = q^I + i \cdot \Delta q$$

C. Differential Motion Planning Algorithm

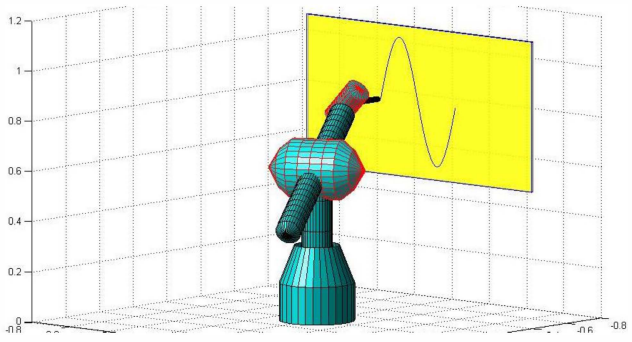

Fig. 4 Differential motion path for the Robotic arm end-effector

By using the following Taylor series:

$$q(t + \Delta_t) = q(t) + \dot{q}_{(t)}\Delta_t + \ddot{q}_{(t)}\frac{\Delta_t^2}{2!} + \cdots$$

We get, after a Δt time interval, the joint position vector, where $q_{(t)} = (\theta_{1(t)} \ \cdots \ \theta_{n(t)})^T$ is the instantaneous value of the joint position vector.

By using the first-order approximation and use the equation $\dot{q}_{(t)} = J^{-1} V$, then

$$q(t + \Delta_t) \cong q(t) + J^{-1} V\Delta_t$$

Where the Jacobian matrix J calculated at q(t) is presumed to be nonsingular and both Cartesian velocity vector V and J must have the same reference frame. The sampling time interval Δ_t must be very small For the tracking accuracy to be acceptable. The second-order approximation can also be used to increase the tracking accuracy[6].

V. SOFTWARE

This project can be logically divided into three phases, All of which are implemented using MATLAB. The Three phases are :

- Analysis of kinematics and calculating the Jacobian.
- implementing the algorithms used to move the arm to get the results.
- drawing a 3D model of the robotic arm and move it to display the results of phase 2 in a comprehensive manner.

Since phase 2 depends on phase 1 in creating crucial part of its code, augmenting the results from phase 1 into phase 2 needs to be done manually by writing the resulting matrixes into its correct line in code. However, this method can be implemented in any other robot arm between 4-DOF and 7-DOF (from other Test cases).

Phase 3 depends on the resulting data from phase 2, which can be sent directly from phase 2, just like calling a function.

ISBN: 978-1-5090-1055-4

Software package was developed to work out the inverse kinematics, forward kinematics, Jacobian and path planning of Stanford Robot arm, however, it can be used for any other serial-link robot arm with a few adjustments in phase1 code, writing the result in phase 2, and creating a new 3D model that fits the new robot arm.

VI. Conclusions

A mathematical model of the robot arm is developed including complete Kinematics analyses. Forward kinematics equations were derived using D-H notation. The Jacobian "Velocity Kinematic", and Trajectory Planning solutions were created and employed by the developed software. Simulation studies were implemented by using MATLAB software's. By creating 3D graphics in MATLAB, structure for the robot arm was built which enable the researchers to investigate robot parameters.

References

[1] Hutchinson, S., Vidyasagar, M. & Spong, M. (2005). *Robot Modeling and Control*
[2] McCarthy, J. (1990). *Introduction to Theoretical Kinematics*, MIT Press, Cambridge, MA
[3] Levenberg, K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares.
[4] Murray, R., Li, Z., Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton
[5] Gu, Edward (2013). *A Journey From Robot To Digital Human*, 1st edition, Springer
[6] Craig, J.(2005). *Introduction to Robotics: Mechanics and Control*, 3rd edition, Prentice Hall, Upper Saddle River, N.J., USA

ISBN: 978-1-5090-1055-4