World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(Research Article)

Codes application in trajectory generation of simulated robot arm dynamics

Okechukwu Stanley Ikwunze [1], kelechi Kingsley Igbokwe [2] and Victor Ikedichi Okparaku [3, *]

[1] *Department of Electrical/Electronics, Abia State Polytechnic, Aba, Abia State, Nigeria.*
[2] *Department of physics/Electronics, Abia State Polytechnic, Aba, Abia State. Nigeria.*
[3] *Uma Ukpai Polytechnic, Asaga, Ohafia, Abia State, Nigeria.*

World Journal of Advanced Engineering Technology and Sciences, 2022, 06(01), 047–052

Publication history: Received on 21 April 2022; revised on 31 May 2022; accepted on 02 June 2022

Article DOI: https://doi.org/10.30574/wjaets.2022.6.1.0061

Abstract

The robot arm used in this research had already been modelled and simulated using Simulink software (i.e. software environment) to yield a robot arm that will mimic the functionality and control of real human arm. The main objective is not to rebuild the robot arm to further enhance tasks performance by armless people in the society, rather to verify the significance of the code (program) application in trajectory generation of simulated robot arm or when simulating a robot arm. The common problem encountered with trajectory generation is to connect the initial configuration to a final configuration while satisfying other specified constraints at the endpoints (e.g., velocity constraints). The trajectory parameters (such as angle and angular velocity) of the robot arm joint had already been simulated with reference to the real human arm in order to generate a robot dynamics (trajectory) of same and equal functionality to real human arm. The code system which implies writing of appropriate languages was confirmed to contain the performance and controlling information of the robot arm. This performance and control of the robot arm appears to be the trajectory generation of the simulated robot arm. Hence, the input languages into the software environment (Simulink) were presented as codes which resulted to (have output of the desired functions) to trajectory generation of the simulated robot arm.

Keywords: Trajectory Planning; Code; Computer Language; Robot Dynamics; Software Environment; Simulation

1. Introduction

Robot dynamics entails the relationship between the forces acting on a robot mechanism and the corresponding accelerations produced by them. Characteristically, the robot mechanism is modelled as a rigid-body system in which case robot dynamics is applying rigid-body dynamics to robots [2], [5]. The robotic kinematics is essentially describing the end-effector's position, orientation as well as motion of all the joints, while dynamics modeling is crucial for analyzing and synthesizing the dynamic behavior of robot [1].

Trajectory planning is act of moving from point A to point B while avoiding collisions over time and it can be computed in both discrete and continuous methods. Also, trajectory planning is a major area in robotics as it gives way to autonomous vehicles [3]. The aim of the trajectory generation is to generate inputs to the motion control system which ensures that the planned trajectory is executed. The user or the upper-level planner describes the desired trajectory by some parameters, usually the initial and final point (point-to-point control) [8].

The trajectory planning could be achieved by coding the language and input it into the software environment. Coding is essentially written instructions that a robot or computer program can read and then execute [8]. One must determine the task one wants to complete through a robot, design the code to make it happen, and then send it to the robot to view

[*] Corresponding author: Okparaku, Victor Ikedichi
Uma Ukpai Polytechnic, Asaga, Ohafia, Abia State, Nigeria.

the outcome. Coding is a list of step-by-step instructions that get computers to do what you want them to do. It makes it possible for us to create computer software, games, apps and websites. The coders or programmers are people who write the programmes behind everything we see and do on a computer [7].

The purpose of this research is actually to determine the pragmatic implication of applying code as computer language or instruction in trajectory generation in robotic and simulation.

2. Limitation of Trajectory

A problem with trajectory is to connect the initial to a final configuration while satisfying other specified constraints at the endpoints (e.g., velocity constraints). Without loss of generality, it is the benefit of the researcher to consider planning the trajectory for a single joint, since the trajectories for the remaining joints will be created independently and in exactly the same way [6].

3. Illustration of a Robot Trajectory Planning

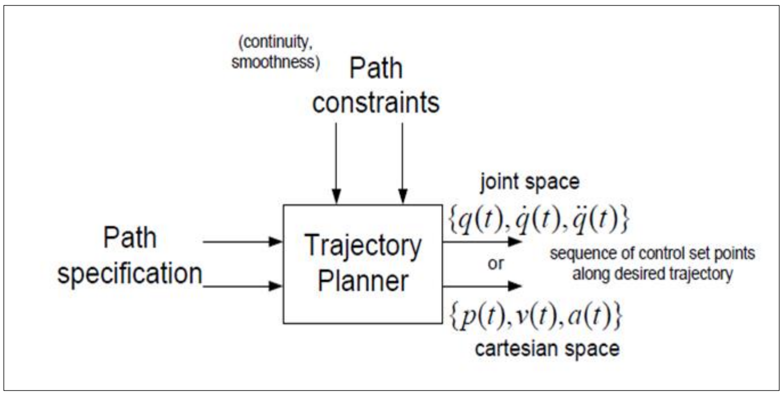


Figure 1 Block diagram of common/existing trajectory planning

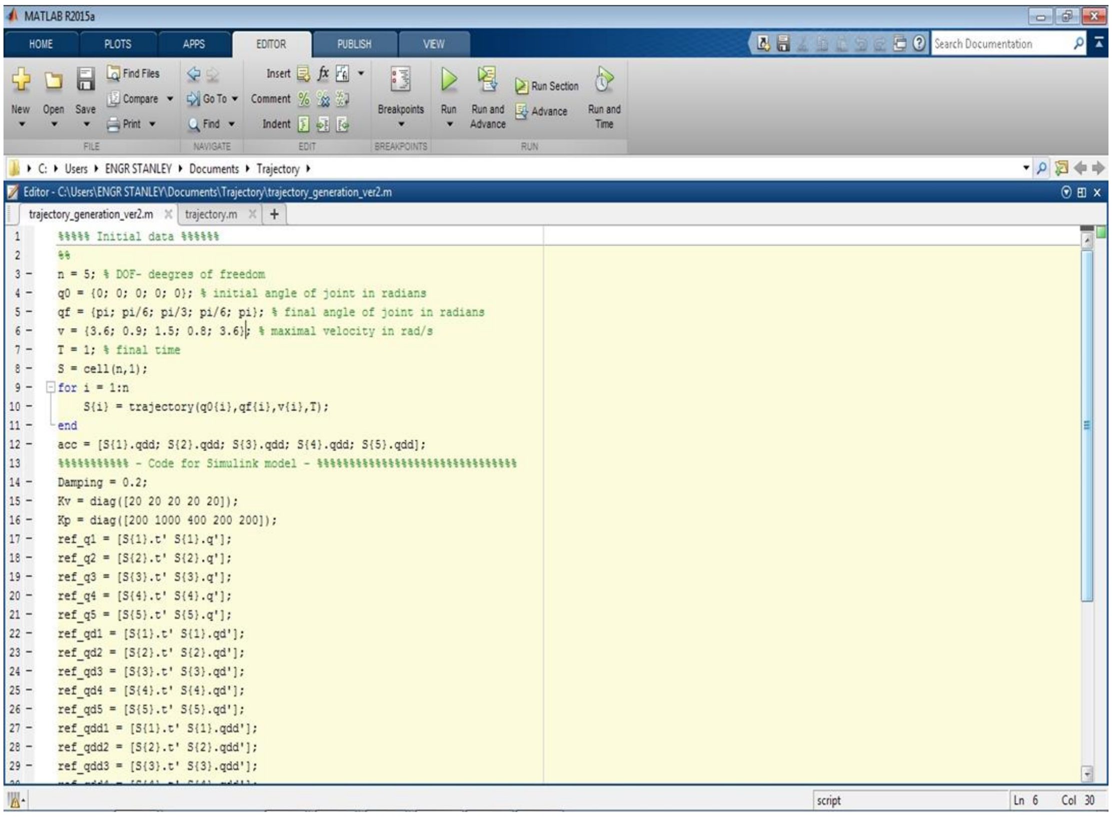
4. Methodology

In achieving the control of a robot arm, personal computer (PC) is used and there must be connection between the robot and PC. This connection is called interface connection and it is done by using a microcontroller. The microcontrollers are devices commonly used in embedded computing applications to impart computing and smart decision-making capabilities to machines, products, and processes. They are designed to interface and interact with electrical/electronic devices, sensors and actuators, and high-tech gadgets to automate systems [4]. Microcontrollers are not meant to interface with human beings, however, they do not have graphical user interface (GUI) capabilities that are common in many personal computer (PC) applications. The complete control process can be divided into two categories: hardware and software. The suitable environment (approach) for the Robot arm control is software environment.

Software environment can be divided into two parts: the CUBLOC microcontroller program and MATLAB Program. In CUBLOC program, a code is written to make the interfacing between PC and the robot arm. The MATLAB program consists of the Serial Communication codes and the graphical user interface (GUI). The codes input into the software environment in simulating the robot arm are languages (instructions) to building parameters of the robot joints, angular velocity and the acceleration.

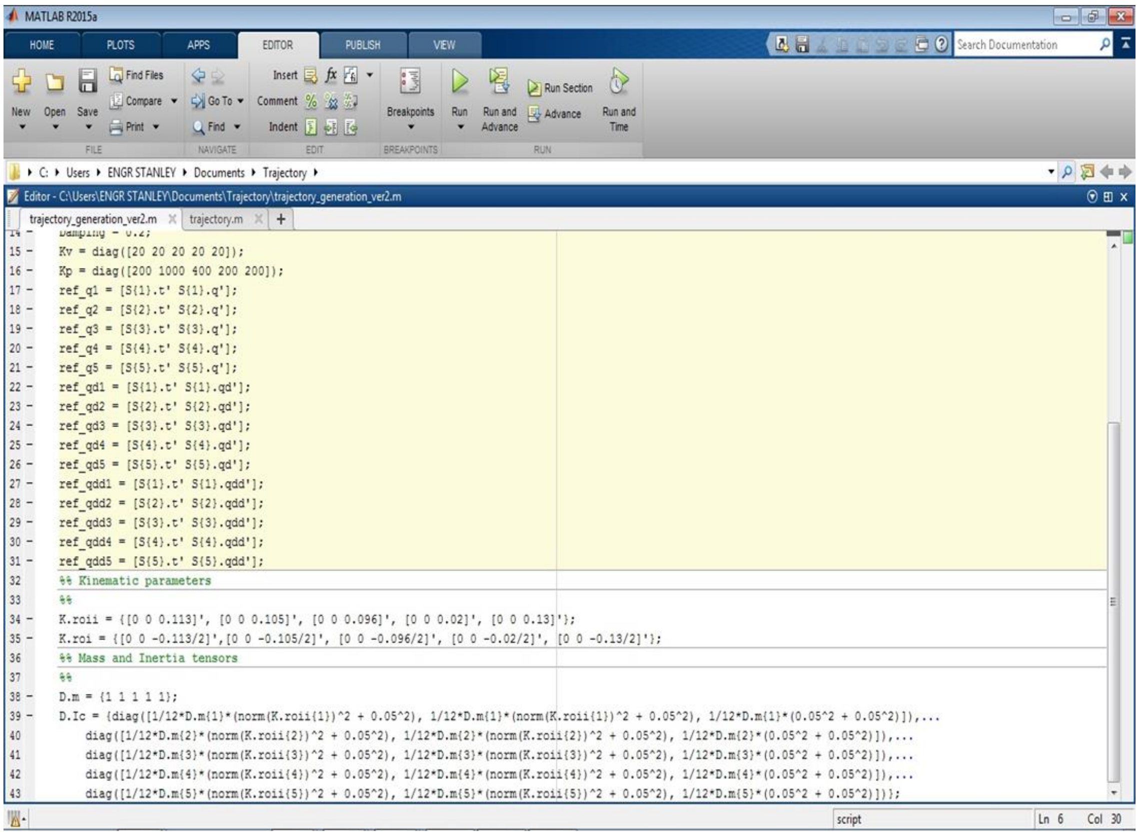
5. Results and Discussion of Trajectory Generation (Via Coding Application)

The figures below are the input and output of trajectory generation for simulated robot arm dynamics. The input figure is categorized into part A and part B because of large data which could not be visibly clear to the reader. However, part B is the continuation of part A. Codes were used to input the language functions into Simulink software to generate the resulting trajectory.


MATLAB R2015a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
C: ▸ Users ▸ ENGR STANLEY ▸ Documents ▸ Trajectory ▸
Editor - C:\Users\ENGR STANLEY\Documents\Trajectory\trajectory_generation_ver2.m
trajectory_generation_ver2.m
trajectory.m
script
Ln 6 Col 30

```
%%%%% Initial data %%%%%%
%%
n = 5; % DOF- deegres of freedom
q0 = {0; 0; 0; 0; 0}; % initial angle of joint in radians
qf = {pi; pi/6; pi/3; pi/6; pi}; % final angle of joint in radians
v = {3.6; 0.9; 1.5; 0.8; 3.6}; % maximal velocity in rad/s
T = 1; % final time
S = cell(n,1);
for i = 1:n
    S{i} = trajectory(q0{i},qf{i},v{i},T);
end
acc = [S{1}.qdd; S{2}.qdd; S{3}.qdd; S{4}.qdd; S{5}.qdd];
%%%%%%%%%%%% - Code for Simulink model - %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Damping = 0.2;
Kv = diag([20 20 20 20 20]);
Kp = diag([200 1000 400 200 200]);
ref_q1 = [S{1}.t' S{1}.q'];
ref_q2 = [S{2}.t' S{2}.q'];
ref_q3 = [S{3}.t' S{3}.q'];
ref_q4 = [S{4}.t' S{4}.q'];
ref_q5 = [S{5}.t' S{5}.q'];
ref_qd1 = [S{1}.t' S{1}.qd'];
ref_qd2 = [S{2}.t' S{2}.qd'];
ref_qd3 = [S{3}.t' S{3}.qd'];
ref_qd4 = [S{4}.t' S{4}.qd'];
ref_qd5 = [S{5}.t' S{5}.qd'];
ref_qdd1 = [S{1}.t' S{1}.qdd'];
ref_qdd2 = [S{2}.t' S{2}.qdd'];
ref_qdd3 = [S{3}.t' S{3}.qdd'];
```

Figure 2 Initial code (data) into Simulink for modelling (part A)


MATLAB R2015a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
C: ▸ Users ▸ ENGR STANLEY ▸ Documents ▸ Trajectory ▸
Editor - C:\Users\ENGR STANLEY\Documents\Trajectory\trajectory_generation_ver2.m
trajectory_generation_ver2.m
trajectory.m
script
Ln 6 Col 30

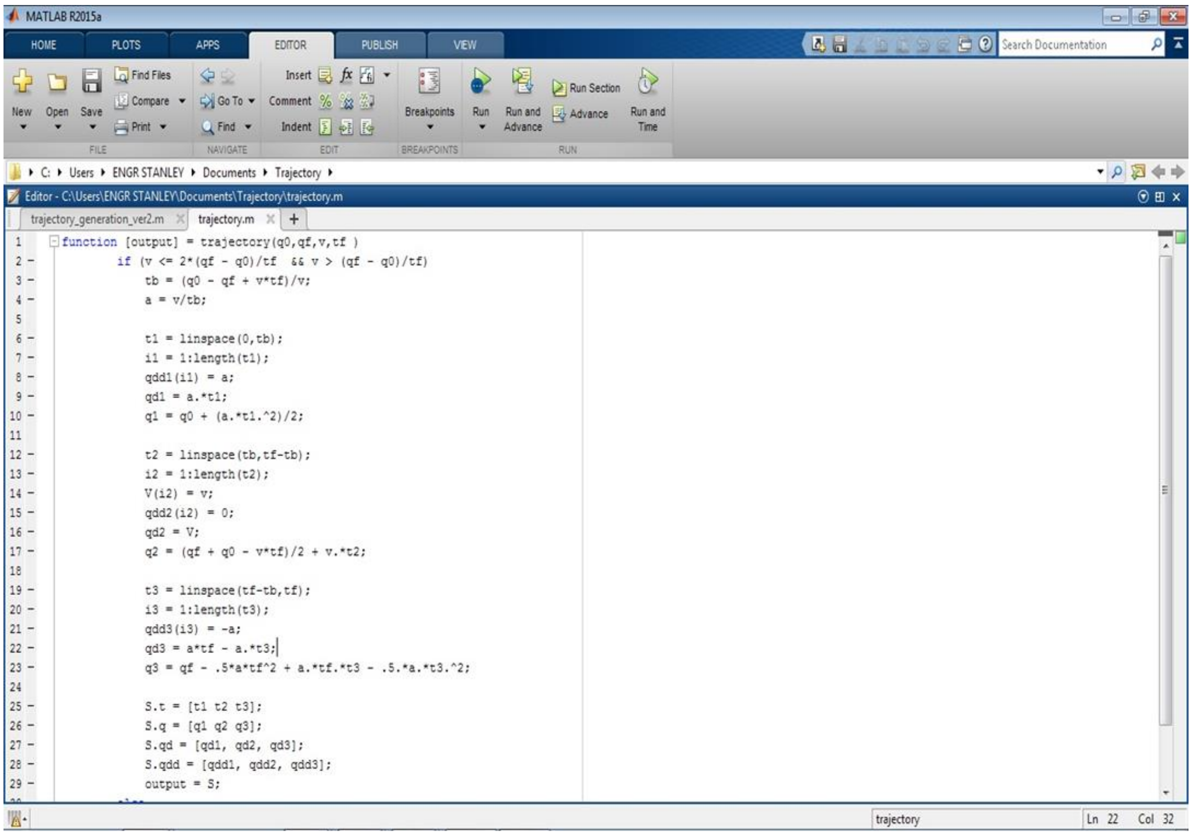
```
Kv = diag([20 20 20 20 20]);
Kp = diag([200 1000 400 200 200]);
ref_q1 = [S{1}.t' S{1}.q'];
ref_q2 = [S{2}.t' S{2}.q'];
ref_q3 = [S{3}.t' S{3}.q'];
ref_q4 = [S{4}.t' S{4}.q'];
ref_q5 = [S{5}.t' S{5}.q'];
ref_qd1 = [S{1}.t' S{1}.qd'];
ref_qd2 = [S{2}.t' S{2}.qd'];
ref_qd3 = [S{3}.t' S{3}.qd'];
ref_qd4 = [S{4}.t' S{4}.qd'];
ref_qd5 = [S{5}.t' S{5}.qd'];
ref_qdd1 = [S{1}.t' S{1}.qdd'];
ref_qdd2 = [S{2}.t' S{2}.qdd'];
ref_qdd3 = [S{3}.t' S{3}.qdd'];
ref_qdd4 = [S{4}.t' S{4}.qdd'];
ref_qdd5 = [S{5}.t' S{5}.qdd'];
%% Kinematic parameters
%%
K.roii = {[0 0 0.113]', [0 0 0.105]', [0 0 0.096]', [0 0 0.02]', [0 0 0.13]'};
K.roi = {[0 0 -0.113/2]',[0 0 -0.105/2]', [0 0 -0.096/2]', [0 0 -0.02/2]', [0 0 -0.13/2]'};
%% Mass and Inertia tensors
%%
D.m = {1 1 1 1 1};
D.Ic = {diag([1/12*D.m{1}*(norm(K.roii{1})^2 + 0.05^2), 1/12*D.m{1}*(norm(K.roii{1})^2 + 0.05^2), 1/12*D.m{1}*(0.05^2 + 0.05^2)]),...
    diag([1/12*D.m{2}*(norm(K.roii{2})^2 + 0.05^2), 1/12*D.m{2}*(norm(K.roii{2})^2 + 0.05^2), 1/12*D.m{2}*(0.05^2 + 0.05^2)]),...
    diag([1/12*D.m{3}*(norm(K.roii{3})^2 + 0.05^2), 1/12*D.m{3}*(norm(K.roii{3})^2 + 0.05^2), 1/12*D.m{3}*(0.05^2 + 0.05^2)]),...
    diag([1/12*D.m{4}*(norm(K.roii{4})^2 + 0.05^2), 1/12*D.m{4}*(norm(K.roii{4})^2 + 0.05^2), 1/12*D.m{4}*(0.05^2 + 0.05^2)]),...
    diag([1/12*D.m{5}*(norm(K.roii{5})^2 + 0.05^2), 1/12*D.m{5}*(norm(K.roii{5})^2 + 0.05^2), 1/12*D.m{5}*(0.05^2 + 0.05^2)])};
```

Figure 3 Initial code (data) into Simulink for modelling (part B)

Fig. 2 and Fig. 3 show a clear initial data (codes) input into the software environment (Simulink) which underwent simulation (modelling) with the following coding descriptions. The coding instructions are for degree of freedom. The other parameters modelled in the Simulink can be seen on the screenshotted figures.

n = s; degree of freedom
q0 = initial angle of joint in radians
qf = final of joint in radians
v = maximum velocity in rad/s
T = 1; final time
S = cell (n,1)
S (i) = trajectory [q0 (i), qf (i)), v (i), T]
acc. = s (1). qdd; s (2).qdd; s (3). qdd; s (4). qdd; s (5). qdd.
ref-qn = run from ref-q1: [s (1). t' s (1). q'] to ref-q5: [s (5) t'. s (5). q']
ref-qdn = from ref-qd1: [s (1). t' s (1). qd'] to ref-qd5: [s (5). t' s (5). s (5). qd']
ref-qddn = from ref-qdd1: [s (1). t' s (1). qdd'] to ref-qdd5: [s (5). t' s (5). qdd']

These codes were simulated based on the language connotation allocated to them. The process yielded the output in fig. 4 below.



```
function [output] = trajectory(q0,qf,v,tf )
        if (v <= 2*(qf - q0)/tf  && v > (qf - q0)/tf)
            tb = (q0 - qf + v*tf)/v;
            a = v/tb;

            t1 = linspace(0,tb);
            i1 = 1:length(t1);
            qdd1(i1) = a;
            qd1 = a.*t1;
            q1 = q0 + (a.*t1.^2)/2;

            t2 = linspace(tb,tf-tb);
            i2 = 1:length(t2);
            V(i2) = v;
            qdd2(i2) = 0;
            qd2 = V;
            q2 = (qf + q0 - v*tf)/2 + v.*t2;

            t3 = linspace(tf-tb,tf);
            i3 = 1:length(t3);
            qdd3(i3) = -a;
            qd3 = a*tf - a.*t3;
            q3 = qf - .5*a*tf^2 + a.*tf.*t3 - .5.*a.*t3.^2;

            S.t = [t1 t2 t3];
            S.q = [q1 q2 q3];
            S.qd = [qd1, qd2, qd3];
            S.qdd = [qdd1, qdd2, qdd3];
            output = S;
```

Figure 4 Output/Result of the code (data) into Simulink after modelling

Fig. 4 gives the function (output) of the trajectory generation. The output is detailed as follows:

Function (output) = trajectory (q0, qf, v, tf)

If V ≤ 2* (qf – q0)/ tf &
V > (qf – q0)/ tf
tb = (q0 – qf + v* tf)/v;
a = v/tb;
t1 = linespace (0, tb)

```
i1 = l; length (t1);
qdd1 = (i1) = a;
qd1 = a* t1
q1 = q0 + (a*. t1^2)/2;
t1 = linespace (tb, tf - tb);
i2 = l; length (t2);
v (i2) = v;
qdd2 (i2) = 0,
qd2 = v; q2 = (qf + q0 – v* lf)/2 + v*. t^2
t3 = linespace (tf – tb, tf);
i3 = l: length (t3);
qdd3 (i3) = -a;
qd3 = a* tf – a* t3;
q3 = qf - .5* a* tf^2 + a* tf. *t3 - .5* a*t3^2;
s.t = (t1, t2, t3)
s.q = (q1, q2, q3)
s.qd = (qd1, qd2, qd3)
s.qdd = (qdd1, qdd2, qdd3)
Output = s
Hence, n = s; degree of freedom.
```

This result/output is a generated trajectory like angle of the robot joints, angular velocity and the acceleration. Due to the ambiguity in the output, the researchers extracted the output of the angle and angular velocities of the joints as dynamic parameters. The angle of the joints and angular velocity are described as q and qd respectively.

6. Conclusion

The codes are inevitably essential in simulating a robot arm in a software environment – Simulink. Codes are languages/instructions that the personal computer understands which are translated into the robot arm as functions/roles/duties/assignments/tasks they are planned/programmed to achieve/execute. The coding output of the angle and angular velocities of joints of the robot (represented as q and qd respectively) as dynamic parameters excerpted from the simulation resembles the real human arm in terms of angle and angular velocities. This implores those codes (computer languages) input into the software environment in simulation successfully generated the required trajectory.

Recommendations

It is with great sense of understanding that the researchers recommend that code application is a primary and essential component of trajectory planning in robotics.

When encoding during trajectory planning, ensure it is done error free because that becomes functionality of the simulating robot and the instruction/command it obeys.

Compliance with ethical standards

Acknowledgments

We acknowledge the efforts of every researcher and the research assistant who enormously contributed towards the completion of this research.

Disclosure of conflict of interest

The interest of the researchers is equal and evenly distributed.

References

[1] Almeida F, Lopes A. A force-impedance controlled industrial robot using an active robotic auxiliary device. Robotics and Computer-Integrated Manufacturing. 2008; 24(3): 299-309.

[2] Arbib MA. Perceptual structures and distributed motor control: In V.B. Brooks, editor. *Handbook of Physiology*, volume II of *Section 2: The Nervous System, Motor Control, Part 1*, American Physiological Society. 1981; 1449 – 1480.

[3] Arya W, Habibollah BH. Industrial Robot Simulation Software Development Using Virtual Reality Modeling Approach (VRML) and MATLAB-Simulink Toolbox. University of Teknologi, Malaysia. 2004; 1-100.

[4] Chun HA, Khin TL, Yin MM. SS Modeling Motion Control System for Motorized Robot Arm using MATLAB, Proceedings of World Academy of Science, Engineering and Technology. 2008; 20-50.

[5] Mark WS, Seth H, Vidyasagar M. Robot Modeling and Control. John Wiley & Sons, 1st Edition. 2005; 1-60.

[6] Martin R. Web Based Robot Simulation Using VRML. Proceedings of the 2000 Winter Simulation Conference. 2000; 10-35.

[7] Mataric M, Cliff D. Challenges in evolving controllers for physical robots. Robot. Autonomous System. 1996; 19: 67-83.

[8] Shala A, Bruqi M. Trajectory Tracking of Mobile Robot using Designed Optimal Controller. *International Journal of Mechanical Engineering and Technology.* 2017; 8(8): 649 – 658.