Development of the Programming and Simulation System of 4-axis Robot with Hybrid Kinematic

Nikola Slavkovic
Associate Professor
University of Belgrade
Faculty of Mechanical Engineering

Sasa Zivanovic
Full Professor
University of Belgrade
Faculty of Mechanical Engineering

Nikola Vorkapic
Teaching Assistant
University of Belgrade
Faculty of Mechanical Engineering

Zoran Dimic
Research Associate
LOLA Institute Belgrade

This paper presents an approach for developing the programming and off-line simulation systems for low-cost industrial robots in the MatLab/Simulink environment. The approach is presented in the example of a virtual model of a 4-axis robot with hybrid kinematics intended for manipulation tasks. The industrial robot with hybrid kinematics consists of the well-known 5R planar parallel mechanism to which two serial axes have been added. The programming system developed in a MatLab environment involves generating G-code programs based on given pick and place points. The virtual model included in the simulation system is configured in the Simulink environment based on the CAD model of the robot and its kinematic structure. The kinematic model and the inverse kinematic problem have to be included in the virtual model to realize the motion of the virtual robot. The system of programming and simulation has been verified through several examples that include object manipulation to perform various tasks.

Keywords: *kinematics, programming, virtual model, MatLab/Simulink, CAD system.*

1. INTRODUCTION

The paper discusses the programming and simulation of a 4-axis robot with hybrid kinematics at the level of its virtual model. The considered virtual model is based on a robot with hybrid kinematics consisting of the well-known planar parallel mechanism (five-bar planar mechanism, BiSCARA, or pantograph) [1], to which two serial axes (translatory and rotary) have been added. The development of a five-bar mechanism started from a US patent in 1934 [1,2]. Only after that, in 1978, Prof. Makino patented the well-known SCARA robot [3]. After that, Donald C. Fyler came up with the idea to use this five-bar mechanism as a robot [1,4].

Such robots are popular in academic institutions [5] because they are suitable for education and experimental work. The hybrid kinematics allows this robot to use three degrees of freedom to reach a position and one degree of freedom to reach orientation around the vertical axis, making it suitable for manipulation tasks. The significant advantage of the used parallel robot is that two of its rotary motors are fixed at the base, which results in lighter moving parts [1].

Most industrial robots are programmed either by teaching (on-line) or by a programming language (off-line). Industrial robots are successfully programmed for some of today's tasks using their programming languages and adequate software for simulation. The off-line robot programming complexity lay in the fact that each robot manufacturer uses its robot programming language. This fact can be solved by using specialized CAM software for programming of robots that generates directly native robot language using appropriate postprocessors for robots. There are also solutions for industrial robots that use G-code as a programming language [6,7].

The subject of this paper will be the development of the programming and simulation system for the virtual robot model when the G-code is used as a program. The programming and simulation system is developed in the MatLab/Simulink environment. In this way, before the pre-planned realization of the physical prototype, an adequate environment for programming and simulation is realized to verify the control programs. The G-code was selected as a programming language because of the pre-planned development of an open-architecture control system based on the LinuxCNC software [8,9]. The open-architecture control system enables the implementation of kinematic models of the different robots and is suitable for education in the field of robot control system development.

Working in a virtual environment allows the verification of a program prepared for the real robot, i.e. it is possible to prevent errors that occur during the programming process, which would lead to a collision between the robot segments themselves or robot elements and the environment [10]. The virtual robot programmed in G-code can be implemented within the CAD/CAM system or open-architecture control system of the robot itself.

The off-line robot simulation and programming software include almost all robotic arms on the market to easily implement them in a simulation environment. To resolve the issues for low-cost robots, i.e. simulation and programming of a new laboratory prototype of robots many researchers developed a system that

Received: January 2008, Accepted: March 2008
Correspondence to: Prof. Nikola Slavkovic
Faculty of Mechanical Engineering,
Kraljice Marije 16, 11120 Belgrade 35, Serbia
E-mail: nslavkovic@mas.bg.ac.rs

integrated kinematics and motion control simulation using MatLab/Simulink environment [11-15].

In [11] is developed a system that integrated kinematics and motion control simulation using MatLab/Simulink which can be connected with a considered robot to verify the results of the simulation. The paper [12] derived the theoretical model of the kinematics analysis of the Gough Stewart mechanism that has been built into the Simulink/MatLab package to obtain the lengths, position, and orientation for the manipulator at any time of motion. In paper [13] is designed a DELTA parallel robot to realize the seed sorting, which simplifies the FANUC parallel robot. The CAD model of the robots is converted into a SimMechanics model. The simple motion simulation experiments are carried out to verify the correctness of the model. The paper [14] presents the model-based motion planning of a delta parallel robot in Simulink/Simscape environment. A model was developed and simulated for motion study and has been simulated to solve the direct kinematics of the parallel manipulator and to check its efficacy. In [15] is proposed software to solve the problem of insufficient robots in schools. The authors combined MatLab/Simulink and AutoCAD to establish forward/inverse kinematics and trajectory planning.

Configuring of the virtual robots, that include complete kinematic model, in the Simulink/Simscape environment [16-18] for the purpose of programming and simulation of low-cost robots, i.e. a new prototype of robots is part of presented research.

2. OUTLINE OF THE CONCEPT

A virtual model of the considered robot is a software implementation of its structure and kinematic model, which can execute generated G-code programs in the developed Simulink environment. The approach considered in this paper for developing industrial robots programming and simulation systems in MatLab/Simulink environment is presented in Fig. 1.

Considered approach for the development of simulation system for low-cost robot consists of three modules:

- creating a file that define pick and place points significant for the manipulation tasks,
- the development of the program for the joint space trajectory generation in MatLab environment, and
- configuring of the virtual robot in the Simulink environment.

The first module is simple and involves the definition of a file that includes vectors of the points at which the robot performs the tasks, i.e. the pick and place points.

The module developed in the MatLab environment consists of two parts. One refers to the development of functions for generating a G-code program based on a defined file of significant points. The other one refers to the development of MatLab functions to generate the joint space trajectories, based on generated or loaded G-code. Joint space trajectory is generated based on the inverse kinematic problem, which is a function included in this module.

The main idea of this paper is to present the whole complete approach that includes both of these parts. The reason for this lies in the fact that there is almost no software that can be used for programing robot manipulation tasks by G-code. To simulate any other robot applications, such as robot laser engraving or 3-axis milling, the first part does not have to be involved and the second part could interpret the external G-code generated in any other available market CAD system. On the other hand, these two parts are independent which enables the developed system to be used in different robot applications. Here is highlighted laser engraving, 3-axis milling or 3D printing but uses of the developed system refers to any tasks that can be performed to considered robot according to its configuration and workspace.

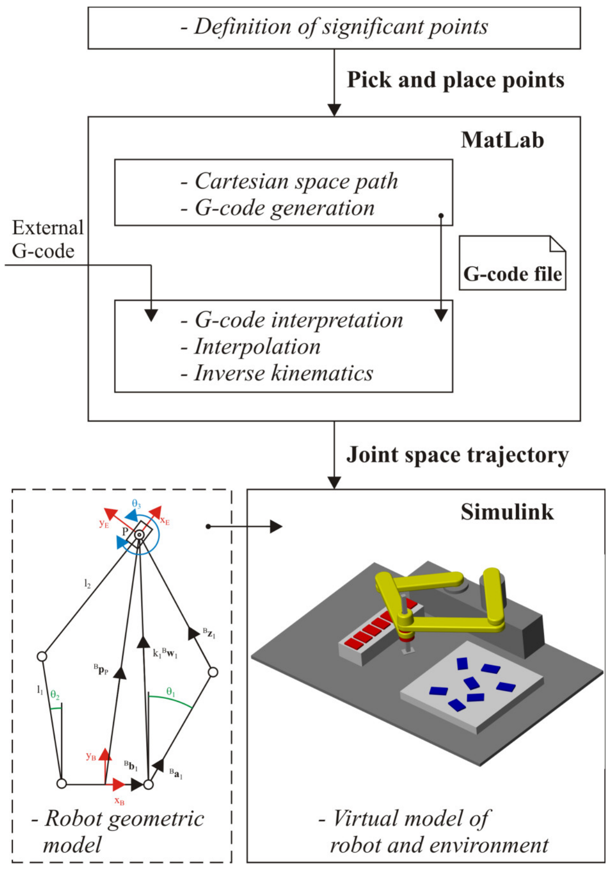


Figure 1. Developed system for the robot programming and simulation

The third module, developed in the Simulink/Simscape environment, covers the implementation of robot elements in STEP format, generated in the CAD system, in the Simscape model or *.slx file. In this research, the commercial PTC Creo software is used [19]. The configuration of a virtual robot is done according to its kinematic structure that also includes kinematic connections (joints). To transfer the physical robot structure in the Simscape model it is necessary to create a geometric model of a robot, Fig. 1., because of the kinematic chains, the direction of robot joint axes, robot joint limits, reference robot frame, etc. The geometric model of a robot will be explained in detail in Section 3.

3. KINEMATIC MODELLING

The 4-axis robot with hybrid kinematics was chosen as an example to show the application of the approach to the development of programming and simulation systems. Figure 2 shows the virtual model of the considered robot. The parallel robot structure consists of a base, a platform, and two kinematic chains with two struts with lengths l_1 and l_2. All elements of the parallel mechanism are connected by rotary joints with one degree of freedom. The parallel mechanism allows the moving of end-effectors in the XY plane, while two added serial axes (translatory and rotary) enable the moving in the Z direction and orientation of the end-effector about a vertical axis.

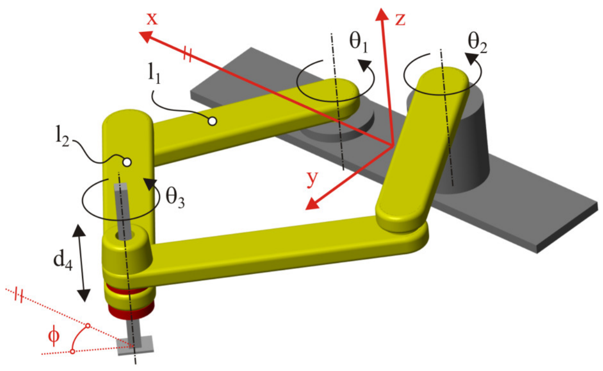


Figure 2. The 4-axis BiSCARA robot

Since the robot is based on a mechanism with hybrid kinematics, the inverse kinematics problem (IKP), for this robot, could be divided into two parts [20]:

- IKP for parallel mechanism, and
- IKP for serial parts of the mechanism.

3.1. Inverse kinematic problem

To solve IKP the geometric model of the robot is established, Fig. 3. The world coordinate vector, Fig3a, is defined as

$$\boldsymbol{x} = {}^B\boldsymbol{p}_E = [x_E \quad y_E \quad z_E \quad \phi]^T \tag{1}$$

In the same manner the joint coordinate vector is derived as

$$\boldsymbol{q} = [\theta_1 \quad \theta_2 \quad \theta_3 \quad d_4]^T \tag{2}$$

The IKP for the parallel mechanism, Fig. 3b, is presented in [8,21] with the exception that the robot is in the initial position in a different configuration, which leads to different equations for the first two joint coordinates.

The vectors necessary for derivation of the solution of angle θ_1 and θ_2 are:

- position vectors of the join centre at the base ${}^B\boldsymbol{b}_i = [b_{ix} \quad 0]^T$,
- unit vectors ${}^B\boldsymbol{a}_i$ and ${}^B\boldsymbol{z}_i$ along struts with lengths l_1 and l_2, and
- unit vectors ${}^B\boldsymbol{w}_i$

where i=1,2 represents the number of the kinematic chains. The unit vector ${}^B\boldsymbol{a}_i$ is defined as

$${}^B\boldsymbol{a}_i = \begin{bmatrix} -\sin(\theta_i) \\ \cos(\theta_i) \end{bmatrix} \tag{3}$$

Based on the defined vectors, according to Fig. 3b, observing one kinematic chain, the following vector equations can be derived

$$\begin{aligned} {}^B\boldsymbol{p}_P &= {}^B\boldsymbol{b}_i + k_i\,{}^B\boldsymbol{w}_i \\ k_i\,{}^B\boldsymbol{w}_i &= l_1\,{}^B\boldsymbol{a}_i + l_2\,{}^B\boldsymbol{z}_i \end{aligned} \tag{4}$$

where vector ${}^B\boldsymbol{p}_P$ represent the position of moving platform (point P) in the XY plane and is defined as

$${}^B\boldsymbol{p}_P = [x_P \quad y_P]^T \tag{5}$$

From the system of equations (4), squaring the second equation and determining the vector $k_i\,{}^B\boldsymbol{w}_i$ from the first equation, the inverse kinematic problem of parallel mechanisms is solved in a closed-form solution.

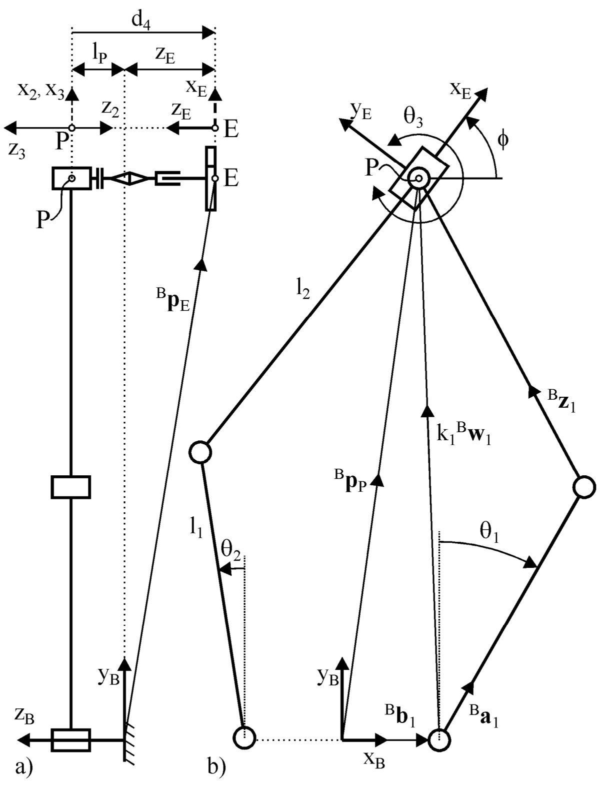


Figure 3. Geometric model of the robot

In this way, the well-known trigonometric equation is derived

$$A_i\cos(\theta_i) + B_i\sin(\theta_i) = C_i \tag{6}$$

where

$$\begin{aligned} A_i &= -y_P \\ B_i &= x_P - b_{ix} \\ C_i &= \frac{l_1^2 - l_2^2 - x_P^2 - y_P^2 - b_{ix}^2 + 2x_P b_{ix}}{2l_1} \end{aligned} \tag{7}$$

By introducing the expression $t = tg\left(\frac{\theta_i}{2}\right)$ in equation (6) a well-known quadratic equation is obtained whose solutions are

$$t_{1/2} = \frac{B_i \pm \sqrt{A_i^2 + B_i^2 - C_i^2}}{A_i + C_i} \tag{8}$$

From equations (8) the joint coordinates θ_1 and θ_2 is determined as

$$\theta_i = 2Atan(t_{1/2}) \tag{9}$$

where i=1,2 represents the number of the kinematic chains.

There are two solutions of the inverse kinematic problem for parallel mechanism and the appropriate solution has to be adapted to the part of the workspace necessary to robot perform the task.

The IKP of added two axes, i.e. serial part of mechanism covers the solution for the remaining two joint coordinates θ_3 and d_4. Figure 4 describes the procedure for solving joint coordinate θ_3.

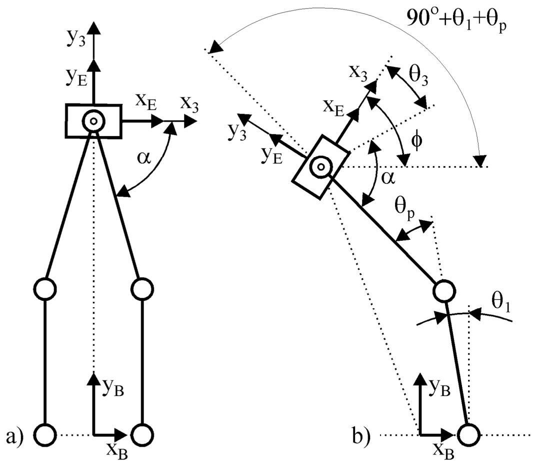


Figure 4. Determination of the orientation angle

To solve joint coordinate θ_3 it is necessary to determine angle α, Fig. 4a. The angle α is calculated using equation

$$\alpha = Atan2(\sqrt{l_2^2 - b_{1x}^2}, b_{1x}) \tag{10}$$

Another important angle θ_P, Fig. 4b, has to be calculated by equation

$$\theta_P = Atan2(s\theta_P, c\theta_P) \tag{11}$$

where

$$s\theta_P = \sqrt{1 - c\theta_P^2} \tag{12}$$

and

$$c\theta_P = \frac{(b_{1x} - x_P)^2 + y_P^2 - l_1^2 - l_2^2}{2l_1 l_2} \tag{13}$$

Now using equations (10) – (13) the joint coordinates θ_3, Fig. 4b, could be calculated as

$$\theta_3 = \phi - \theta_1 - \theta_p - \alpha + 90° \tag{14}$$

It is obvious from Fig. 3a that joint coordinates d_4 can be calculated from equation

$$d_4 = -z_E + l_P \tag{15}$$

The equations (9), (14), and (15) represent the solution of inverse kinematic problem of considered 4-axis robot with hybrid kinematics. These equations are required for the realization of the virtual model.

3.2. Workspace analysis

Another characteristic of the robot structure, that is not crucially important for the development of the virtual model, is the robot workspace. It is necessary to determine the dimensions and shape of the robot workspace to program and simulate robot tasks in the right way.

To determine the workspace, the method presented in [21] can be used, which determines if the point defined in Cartesian space is reachable or not, according to the limits in the joints, based on solutions of the inverse kinematic problem. Determining the dimensions and shape of the workspace, in the case of designing a new prototype, is an iterative procedure in which the parameters of the mechanism are changed and at the end adopted based on the satisfactory dimensions of the workspace.

The adopted parameters of robot are $l_1 = 210mm$ and $l_2 = 270mm$. Figure 5 presents the shape and dimension of workspace of considered 4-axis robot in XY plane. The third dimension of the workspace depends on joint coordinate d_4, which is adapted to have a motion range of 100mm. According to robot structure, it is obvious that the shape and dimensions of workspace in the XY plane are the same along the Z direction, i.e. adopted 100mm.

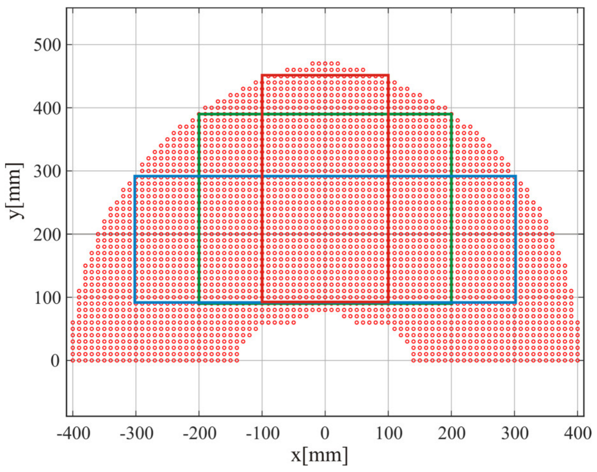


Figure 5. Workspace of considered robot

For the application of robot tasks, the operators could use all portions of the workspace with irregular shapes or reduced workspace to appropriate parallelepiped according to tasks, Fig. 5.

4. PROGRAMMING AND SIMULATION SYSTEM

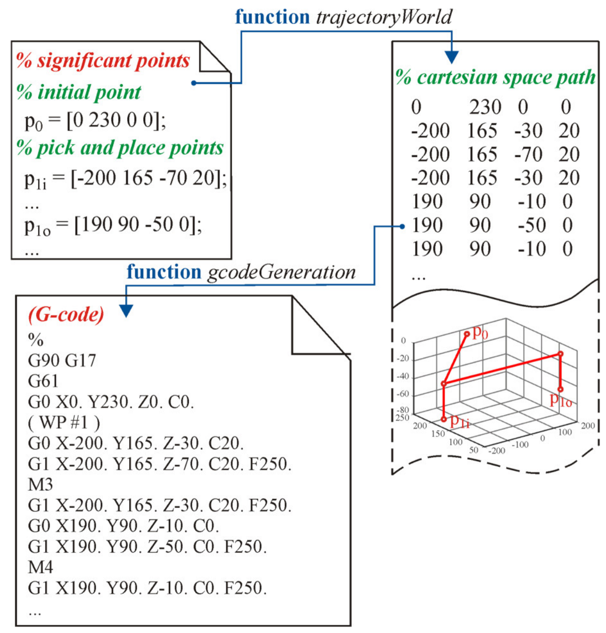
Briefly, the developed system for programming and off-line simulation for new low-cost laboratory prototype robots starts from the generation of programs in a MatLab environment. The G-code program is interpreted and a joint space trajectory is determined. Appropriate functions have been developed for each of these steps. The joint space trajectory is then loaded to a virtual robot configured in the Simulink/Simscape environment. Using joints trajectory the virtual robot simulate the motion of programmed tasks.

4.1. MatLab programming system

The MatLab programming system is developed only for manipulation tasks in G-code. For any other tasks, the virtual environment can use G-code generated in some available CAD systems.

The input parameter for the programming system is the file that consists of the initial position of robot configuration and significant points in which the robot performs the task. This file is formatted as presented in Fig. 6. The initial point is defined with vector $\boldsymbol{p}_0$. The points of object picking are represented with vectors $\boldsymbol{p}_{1i}$, $\boldsymbol{p}_{2i}$, etc., while the points of an object placing are represented with vectors $\boldsymbol{p}_{1o}$, $\boldsymbol{p}_{2o}$, etc. All these vectors are defined in the robot reference coordinate system.

Based on created file the MatLab function *trajectoryWorld* generates the robot trajectory in Cartesian space, Fig. 6. The generated trajectory is something similar to the well-known CLF (Cutter Location File). Since the programming system was developed for manipulation tasks, the trajectory was generated based on the programmed rule that the gripper goes first to the point above the manipulation object, then to the point of picking the object, and at the end again to the point above the object. The same rule applies to the generation of the trajectory part when the gripper places the object.



```
% significant points
% initial point
 p0 = [0 230 0 0];
% pick and place points
 p1i = [-200 165 -70 20];
 ...
 p1o = [190 90 -50 0];
 ...
```

```
% cartesian space path
 0     230   0    0
 -200  165  -30  20
 -200  165  -70  20
 -200  165  -30  20
 190   90   -10   0
 190   90   -50   0
 190   90   -10   0
 ...
```

```
(G-code)
%
G90 G17
G61
G0 X0. Y230. Z0. C0.
( WP #1 )
G0 X-200. Y165. Z-30. C20.
G1 X-200. Y165. Z-70. C20. F250.
M3
G1 X-200. Y165. Z-30. C20. F250.
G0 X190. Y90. Z-10. C0.
G1 X190. Y90. Z-50. C0. F250.
M4
G1 X190. Y90. Z-10. C0. F250.
...
```

Figure 6. Developed programming system

In addition to the coordinates of the points, the file in which the trajectory in world space is written also contains information on the feed rate, whether the gripper is moving at a rapid or not. It also indicates the points of picking and placing objects. Now the file of trajectory in Cartesian space is post-processed using the function *gcodeGeneration* to obtain the G-code.

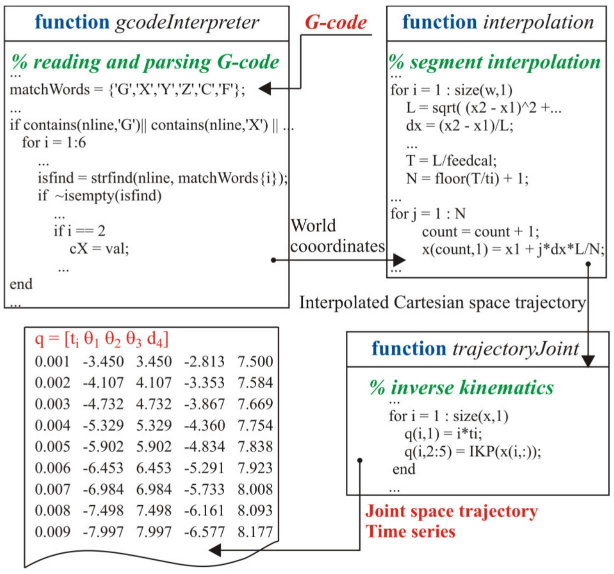
Figure 6 shows part of the G-code from the example shown in Section 5. Besides the standard beginning of the program that defines programming in absolute coordinates and interpolation in the XY plane, parts of the code related to object manipulation can also be seen. In these parts, crucial commands are M3 and M4 that activate the gripper to pick or place an object. These auxiliary functions can be remade in LinuxCNC software [7].

4.2. Joint space trajectory generation

This module, although it uses some of the information defined in the previous module when generating G-code for manipulation tasks, is programmed to be completely independent due to the possibility of simulating some other tasks on a virtual robot. The developed functions for joint space trajectory generation stars from G-code generated in a developed module or market CAD systems.

First, the function *gcodeInterpreter* reads and parses one by one line of G-code. The result of it is an inter-file that consists of the points defined in reference frame, speed, etc. In other words, the file defines the segments of the trajectory and the end-effector feed rate on those segments, Fig. 7.

Then, based on the inter-file the function *interpolation* divided all segments by using the rule of linear interpolation in Cartesian space resulting in the interpolated trajectory. This rule is programmed according to segments length, speed, and time increment.



```
function gcodeInterpreter
% reading and parsing G-code
...
matchWords = {'G','X','Y','Z','C','F'};
...
if contains(nline,'G')|| contains(nline,'X') || ...
  for i = 1:6
     ...
     isfind = strfind(nline, matchWords{i});
     if ~isempty(isfind)
        ...
        if i == 2
           cX = val;
        ...
end
...
```

```
function interpolation
% segment interpolation
...
for i = 1 : size(w,1)
   L = sqrt( (x2 - x1)^2 +...
   dx = (x2 - x1)/L;
   ...
   T = L/feedcal;
   N = floor(T/ti) + 1;
...
for j = 1 : N
      count = count + 1;
      x(count,1) = x1 + j*dx*L/N;
...
```

```
q = [ti θ1 θ2 θ3 d4]
0.001  -3.450  3.450  -2.813  7.500
0.002  -4.107  4.107  -3.353  7.584
0.003  -4.732  4.732  -3.867  7.669
0.004  -5.329  5.329  -4.360  7.754
0.005  -5.902  5.902  -4.834  7.838
0.006  -6.453  6.453  -5.291  7.923
0.007  -6.984  6.984  -5.733  8.008
0.008  -7.498  7.498  -6.161  8.093
0.009  -7.997  7.997  -6.577  8.177
```

```
function trajectoryJoint
% inverse kinematics
...
for i = 1 : size(x,1)
   q(i,1) = i*ti;
   q(i,2:5) = IKP(x(i,:));
end
...
```

Figure 7. Generating the joint space trajectory

After the linear interpolation of trajectory in Cartesian space the function *trajectoryJoint* involving the inverse kinematic solution, equations (9), (14), and (15), generates the joint space trajectory with an added vector of interpolated time as the first column. The result presents the time-series of joint coordinates necessary for the simulation of generated Simscape virtual robot.

4.3. Simulink model of 4-axis robot

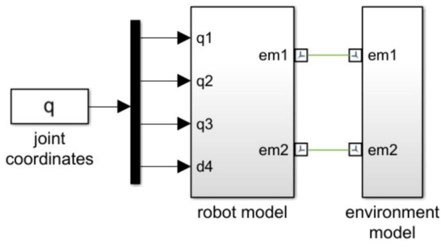
The CAD model of considered robot, necessary for the realization of virtual robot in Simulink environment, is realized in PTC Creo software. From CAD software the STEP files of robot segments are exported and they present the input in the Simulink environment to configure the virtual robot. Configuring a virtual robot involves implementing a kinematic structure, which of course involves kinematic connections between segments, in a software environment. Figure 8 present the structure of the considered robot realized in the Simulink environment taking into account the direction of robot joint axes, robot joint limits, reference robot frame, etc.

To simulate robot tasks it is grateful to have a completely virtual environment not only a virtual robot. Figure 8a presents the complete structure (robot and environment) in Simulink for the example described in Section 5. It is divided into three subsystems in the Simulink to easily change the robot environment for simulating other tasks. These subsystems are:

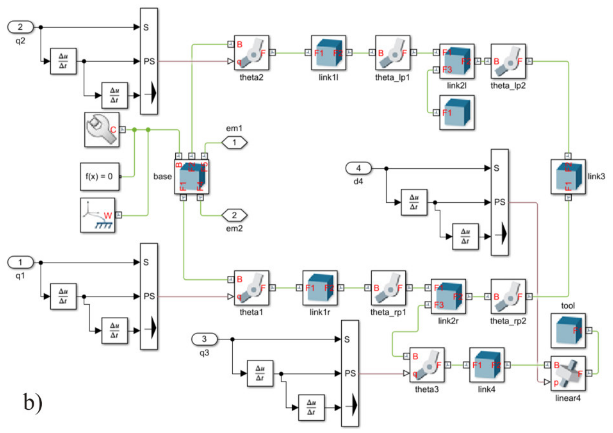
- joint time-series in MatLab environment,
- model of robot, and
- model of the environment.

The generation of joint time-series is presented in previous sections and uses only *From Workspace* element in Simulink. The model of environment can be easily modelled in a CAD environment and implemented by appropriate connection in the Simulink model. These connections are the created coordinate frames in CAD or *Brick Solid* element in the Simulink.

a)



b)



c)

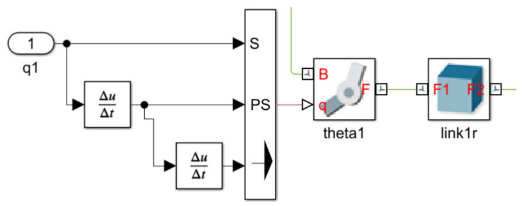


Figure 8. Simulink model of the robot

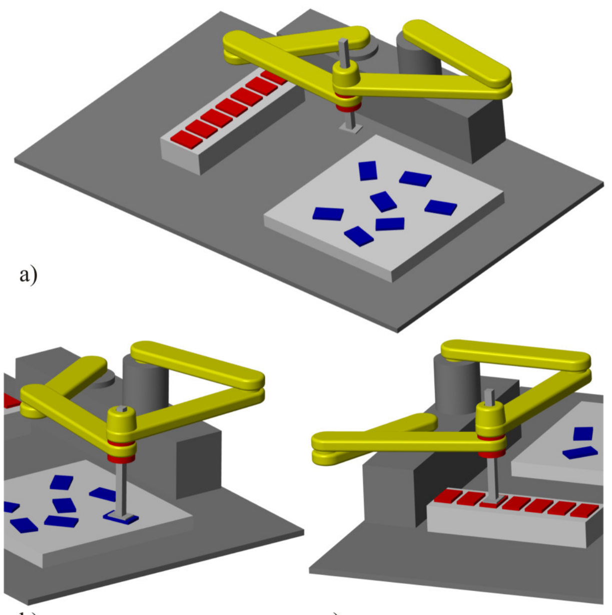
The structure of configured virtual robot in Simulink is presented in Fig. 8b. It consists of a base and three kinematic chains. The base is represented with the *Brick Solid* element connected with *World Frame*, *Solver Configuration*, and *Mechanism Configuration* elements. Two of the three kinematic chains present the parallel part of the considered mechanism, while the third present the serial part. The first two kinematic chains are interconnected with joint connections. Each of them is connected to the base. The third (serial) chain is connected to one of the first two, depending on kinematic calculation. Any kinematic chain is consists of *Brick Solid* elements and appropriate *Revolute Joint* and/or *Prismatic Joint* elements.

The active joints in these kinematics chains have to be actuated by joint time series generated in MatLab while passive joints numerical compute their position. The actuated revolute joint is presented in Fig. 8c. It is actuated with joint angle, velocity, and acceleration, Fig. 8c, which are provided by appropriate connection to receive the signal from the joint coordinate subsystem, Fig. 8a.

5. SYSTEM VERIFICATION

The developed system for programming and off-line simulation has been verified through several examples that include object manipulation to perform various tasks.

One example of an objects manipulation experiment is shown in Fig. 9a. The planned experiment is implied of taking rectangular cross-section parts (blue parallelepipeds) of arbitrary position and orientation and placing them (red parallelepipeds) in a precisely defined position on a pallet site. The placing site can be also the conveyor belt, which can be modelled in a CAD environment and placed by appropriate connection in a virtual environment.



a) b) c)

Figure 9. An example of the system verification

The start position and orientation of the objects are defined in the file at the beginning of experiments, Fig. 1. When a real robot performs a task, these positions could be obtained from a camera mounted on the robot itself. Before the simulation is started the G-code program is generated based on created file. The virtual robot executes a program in G-code, which allows a simulation of robot motion according to the generated trajectory in joint space. It is of importance to simulate the robot's approach to each position in the appropriate orientation for taking objects, Fig. 9a, as well as display the gripper position and orientation at the time of placing the object, Fig 9b. The simulation covers all the movements of the robot, but not the display of the gripper picking, manipulating, and placing the objects. This is not important for verifying the accuracy of the program, because it is planned to attach on robot plate the vacuum or electromagnetic gripper.

After reviewing the complete simulation, it can be concluded that the virtual robot works correctly according to the generated G-code program and that approaches all positions with appropriate orientations for both picking and placing of objects.

6. CONCLUSION

The main advantages of using virtual robots are the ability to check the motion of the end-effector along the programmed path taking into accounts the limitations in the joints motions, and visual detection of collisions between segments and end-effector with the environment.

This paper presents an approach for developing the programming and simulation system of a 4-axis virtual robot with hybrid kinematics in the MatLab/Simulink environment. A virtual model of the considered robot is a software implementation of its CAD and kinematic models in the Simulink environment, which can execute generated G-code. The inverse kinematic problem is implemented in the virtual model to realize the motion of the virtual robot, as the real robot does. The developed program in a Matlab environment based on realized functions generates a G-code for the robot manipulation task. The developed virtual robot can simulate any other robot task such as laser engraving, 3-axis milling, 3D printing, i.e. the virtual robot can use the G-code generated in some other CAD system. The developed system for programming and simulation of the configured virtual robot has been verified through several examples that include object manipulation to perform various tasks.

The presented approach of developing the programming and off-line simulation systems is crucial for the new laboratory prototypes of low-cost industrial robots that are not implemented in market software for simulation. The further research direction will cover the optimization of robot parameters and development of a prototype of a considered 4-axis robot with hybrid kinematics.

ACKNOWLEDGMENT

This work has been financially supported by the Ministry of Education, Science and Technological Development of the Serbian Government, through the project “Integrated research in macro, micro, and nano mechanical engineering (contract No. 451-03-9/2021-14/200105), and by the Science Fund of the Republic of Serbia, grant No. 6523109, AI - MISSION4.0, 2020-2022.

REFERENCES

[1] DexTAR, User’s Manual, Version 1.0, by Mecademic Inc., 2014–2015.

[2] Pollard Jr., W.L.G.: Spray Painting Machine, US Patent 2,213,108, filed October 29, 1934, issued August 27, 1940.

[3] Makino, H., Kato, A., and Yamazaki, Y.: Research and commercialization of SCARA robot, International Journal of Automation Technology, Vol. 1, No. 1, pp. 61–62, 2007.

[4] Fyler, D.C.: Control Arm Assembly, US Patent 4,712,971, filed February 13, 1985, issued December 15, 1987.

[5] Cano-Ferrer, X.: *Educational Five-Bar Parallel Robot*, from https://hackaday.io/project/173325-educational-five-bar-parallel-robot, accessed on 2022-01-30.

[6] Milutinovic, D., Glavonjic, M, Slavkovic, N., Dimic, Z., Zivanovic, S., Kokotovic, B., Tanovic, Lj.: Reconfigurable robotic machining system controlled and programmed in a machine tool manner, International Journal of Advanced Manufacturing Technology, Vol. 53, No. 9-12, pp. 1217-1229, 2011.

[7] Milutinovic, D., Slavkovic, N., Kokotovic, B., Milutinovic, M., Zivanovic, S., Dimic, Z.: Kinematic modeling of reconfigurable parallel robots based on DELTA concept, Journal of Production Engineering, Vol.15, No.2, pp. 71-74, 2012.

[8] Slavkovic, N., Vorkapic, N., Zivanovic, S., Dimic, Z., Kokotović, B.: Virtual Biscara robot integrated with open-architecture control system, in: *Proceedings of the 14th International Scientific Conference MMA 2021 – Flexible Technologies,* 23-25.09.2021, Novi Sad, pp. 63-66.

[9] LinuxCNC, from http://linuxcnc.org/, accessed on 2021-05-07.

[10] Vokapic, N., Zivanovic, S., Dimic, Z., Kokotovic, B., Slavkovic, N.: Virtual Horizontal Machining Center LOLA HBG 80 for Program Verification and Monitoring, FME Transactions, Vol. 49, No. 3, pp. 696-703, 2021.

[11] Lee, W.-c., Kuo, S.-a.: Simulation and Control of a Robotic Arm Using MATLAB, Simulink and TwinCAT, in: *Proceedings of 2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS) IEEE*, 2020, pp. 1-5.

[12] Alwan, H.M., Sarhan, R.A.: Kinematics Simulation of Gough-Stewart Parallel Manipulator by Using Simulink Package in Matlab Software, Journal of University of Babylon for Engineering Sciences, Vol. 27, No, 2, pp. 10-20, 2019.

[13] Li, M., Bi, D., Xiao, Z.: Mechanism simulation and experiment of 3-DOF parallel robot based on MATLAB, in: *Proceedings of the 2015 International Power, Electronics and Materials Engineering Conference*, May 2015, pp. 489-494.

[14] Makwana, M.A., Patolia, H.P.: Model-based motion simulation of delta parallel robot, Journal of Physics: Conference Series, Vol. 2115, No. 1, p. 012002, 2021.

[15] Qassem, M.A., Abuhadrous, I., Elaydi, H.: Modeling and Simulation of 5 DOF educational robot arm, in: *Proceedings of 2010 2nd International Conference on Advanced Computer Control IEEE*, March 2010, pp. 569-574.

[16] Import a Robotic Arm CAD Model, from https://www.mathworks.com/help/physmod/sm/ug/import-robot-arm-model.html, accessed on 2022-02-01.

[17] Model and Control a Manipulator Arm with Robotics and Simscape, from https://www.mathworks.com/help/robotics/ug/model-and-control-a-manipulator-arm-with-simscape.html, accessed on 2022-02-01.

[18] Multi-Loop PI Control of a Robotic Arm, from https://www.mathworks.com/help/control/ug/multi-loop-pid-control-of-a-robot-arm.html, accessed on 2022-02-01.

[19] PTC Creo, webpage, from https://www.ptc.com/, accessed on 2022-02-01.

[20] Milutinovic M., Slavkovic N., Milutinovic D.: Kinematic Modeling of Hybrid Parallel-Serial Five-Axis Machine Tool, FME Transactions, Vol.41, No.1, pp. 1-10, 2013.

[21] Slavkovic, N., Zivanovic, S., Vorkapic, N.: Configuring a virtual prototype of a BiSCARA robot (in Serbian), TEHNIKA, Vol. 70, No.3, pp. 311-317, 2021.

РАЗВОЈ СИСТЕМА ЗА ПРОГРАМИРАЊЕ И СИМУЛАЦИЈУ 4-ОСНОГ РОБОТА СА ХИБРИДНОМ КИНЕМАТИКОМ

Н. Славковић, С. Живановић, Н. Воркапић, З. Димић

У раду је приказан приступ развоју система за програмирање и *off-line* симулацију развијених прототипова робота, који нису имплементирани у постојеће софтвере за роботе, у *MatLab/Simulink* окружењу. Приступ је приказан на примеру развоја виртуелног 4-осног робота са хибридном кинематиком, намењеног за задатке манипулације. Разматрани индустријски робот са хибридном кинематиком се састоји од добро познатог *5R* раванског механизма са паралелном кинематиком, коме су додате две серијске осе. Систем програмирања развијен у *MatLab* окружењу обухвата генерисање *G*-код програма на основу задатих тачака у којима робот извршава задатак. Виртуелни модел је конфигурисан у *Simulink* окружењу на основу *CAD* модела робота и његове кинематичке структуре. Кинематичко моделирање и инверзни кинематички проблем су решени у циљу реализације кретања робота према генерисаном програму. Развијени систем за програмирање и симулацију је верификован кроз неколико примера који укључују манипулацију објектима при обављању различитих задатака.