International Journal on Interactive Design and Manufacturing (IJIDeM) (2019) 13:567–595
https://doi.org/10.1007/s12008-018-0519-z

ORIGINAL PAPER

Quantitative and qualitative study of methods for solving the kinematic problem of a planar parallel manipulator based on precision error optimization

Sid-Ahmed Dahmane[1] · Abdelwahab Azzedine[1] · Abdelkader Megueni[1] · Abdelkader Slimane[2,3]

Received: 13 August 2018 / Accepted: 29 November 2018 / Published online: 13 December 2018

Abstract
The main objective of our work is to optimize the positional and orientation error of a platform using the direct geometric model of a parallel plane manipulator robot. It is well known that the main disadvantage of parallel manipulators is the existence of singularities within its workspace, the adaptive neuro-fuzzy solution is proposed in this study. Intermediate methods have been used to determine the optimal solution. The first method is a graphical method which determines all possible positions of the platform based on the intersection of circles. The second method is the polynomial method used to calculate the coordinates of the center of gravity and the orientation of the platform. Matlab programming simulation of these methods makes it possible to find all the solutions deduced from these methods. The analysis shows that the polynomial method is the one that provides the optimal solution.

Keywords Parallel plane manipulator · Error · Direct geometric model · Singularities · ANFIS · Polynomial method · Optimization

1 Introduction

The parallel robot manipulators, owing to their closed-loop structure possess a number of advantages over traditional series manipulators such as high rigidity, high value of power density, high natural frequencies, speeds and high precision [1]. However, they also have a few disadvantages such as a relatively small workspace, relatively complex forward kinematics as well as the existence of singularities inside the workspace. [2]. The geometrical model analysis of parallel manipulators is divided into two types namely direct and inverse models. The inverse geometrical model which maps the operational space in the joint space, is not difficult to solve. Unlike direct geometrical model that performs the reverse operation presents difficulties in problem solving. Moreover, the existence not only of multiple direct geometrical solutions (or work patterns) is another problem in the geometrical analysis [3]. The analysis of the singularity of planar parallel manipulators has been studied by many researchers [4, 5]. Efforts to solve the direct geometry of planar parallel manipulators have focused on the manipulator plane 3RPR due to its inherent simplicity. It is established that the direct geometrical solution of this type of manipulator leads to a polynomial [6, 7]. However, the study of the problem of direct geometry leads to a maximum of six real solutions. A neural network-based approach was developed to curb the problem of the inverse geometric model. Many efforts have been made on the applications of adaptive neuro-fuzzy inference system (ANFIS) in different types of parallel machine tools due to their extreme flexibility and nonlinear approximation ability of the correspondence function output/input [8–12]. Multiple neural networks can solve the problem of multiple branches of the direct or inverse geometrical problem. This approach also overcomes the problem of singularities and uncertainties arising from the path plan-

✉ Sid-Ahmed Dahmane
pro_dah@hotmail.com

1 Laboratory of Structures and Solids Mechanics (LMSS), University Djilali Liabès of Sidi-Bel-Abbes, 22000 Sidi Bel Abbès, Algeria

2 Laboratoire de mécanique appliquée, Département de Génie Mécanique, Université des Sciences et de la technologie d'Oran Mohamed Boudiaf, USTO-MB, BP 1505, El M'naouer, 31000 Oran, Algeria

3 Laboratory of Materials and Reactive Systems (LMSR), Department of Mechanical Engineering, University of Sidi-Bel-Abbes, Bp 89, cité Ben M'hidi, 22000 Sidi Bel Abbès, Algeria

ning. In this approach, a network is formed using the data generated by the inverse geometrical model.

An analysis and comparison of the three used methods are made for determining the best optimal solution concerning the minimum error with minimum execution time. The analysis shows that the optimal solution is obtained by the polynomial method.

2 Main objective of the study

This study represents the comparison of three methods, graphical method, analytical method and ANFIS interactive method which represents the main part of our project. The objective of this study was the general formalization of a direct geometry problem for parallel robots, which proves that this method is an interactive method used to solve problems of the optimal position of industrial robots. The results obtained by this method encourage us to consider the following points:

- Modeling of nonlinear systems by the fuzzy hybrid approach (ANFIS).
- Inject the neuro-fuzzy control into the application of industrial robots.
- Build programs that are able to perform increasingly complex and precise calculations in the robots' work space.

After presenting the degeneracy of the system we examine the situations where the direct geometry degenerates.

After we go on to the main problem of our study, which consists of conducting an investigation in the direct geometrical model that we tried to solve with several methods with a good precision at the beginning with the polynomial solution that sought the existence of the angle φ then find the coordinates of the platform p (x, y).

3 Kinematics of parallel manipulators

Kinematic analysis of parallel manipulators includes solution for both forward and inverse kinematic problems. The forward kinematics of a manipulator deals with the computation of the position and orientation of the manipulator end-effector in terms of the active joints variables. Forward kinematic analysis is one of essential parts in control and simulation of parallel manipulators. Contrary to the forward kinematics, the inverse kinematics problem deal with the determination of the joint variables corresponding to any specified position and orientation of the end-effector. The inverse kinematics problem is essential to execute manipulation tasks. Most parallel manipulators can admit not only multiple inverse kinematic solutions but also multiple forward kinematic solutions. This property produces more complicated kinematic models but allows more flexibility in trajectory planning [13]. In other words a manipulator configuration can be defined either by actuator coordinates $q = [q_1,\ldots, q_n]^T$ or by cartesian end effector coordinates $x = [x_1,\ldots, x_n]^T$ with n the DOF of the manipulator under study. The transformation between actuator coordinates and cartesian coordinates is an important issue from the viewpoint of kinematic control. Computation of the end effector coordinates from given actuator coordinates (forward kinematics) can be written in the general form as:

$$X = f(Q) \tag{1}$$

The inverse task which is to establish the actuator coordinates corresponding to a given set of end effector coordinates (inverse kinematics) can be also expressed in the general form by:

$$Q = f^{-1}(X) \tag{2}$$

The forward kinematics model is given by:

$$\dot{X} = J\dot{Q} \tag{3}$$

where J is the Jacobian matrix.

The inverse kinematics model is expressed by the following relation:

$$\dot{Q} = J^{-1}\dot{X} \tag{4}$$

J is the inverse matrix of J^{-1}

Inverse kinematic singularity occurs when different inverse kinematic solutions coincide that happens usually at the workspace boundary. Hence the manipulator loses one or more degrees of freedom. Mathematically they can detected by det (Jq) = 0. Forward kinematic singularity occurs when different forward kinematic solutions coincide. Hence the manipulator gains one or more degrees of freedom. That happens inside the workspace so it is a great problem. Mathematically they can detected by det (Jx) = 0.

Let $X = [x, y\ \varphi]^T$, the vector of the operational coordinates (3 degrees of freedom DOF of the robot) which corresponds to the position $[x, y]^T$ of the geometric center P_1 of the reference of the platform and to the orientation of the latter With respect to the x-axis of the base marker located as shown in Fig. 1.

Let $\boldsymbol{Q} = [\boldsymbol{q}_1\boldsymbol{q}_2\boldsymbol{q}_3]^{\boldsymbol{T}}$ be the vector of the articular variables corresponding to the elongations.

From the diagram of the robot 3RPR (Fig. 1) one can deduce the solution to the inverse geometric problem.

It can be easily written as follows:

$$q1 = \sqrt{x^2 + y^2} \tag{5}$$

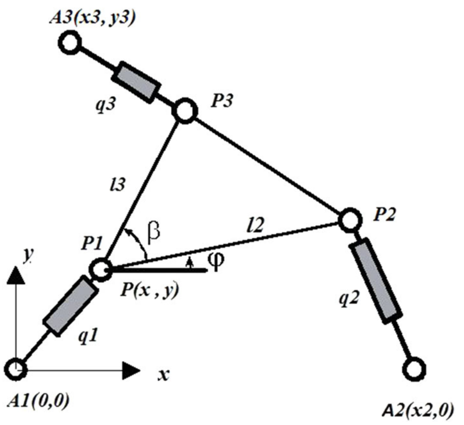


Fig. 1 Schematic representation of the robot 3 RPR

$$q2 = \sqrt{(x + l_2 \cos\varphi - x_2)^2 + (y + l_2 sin\varphi)^2} \quad (6)$$

$$q3 = \sqrt{(x + l_3 \cos(\varphi + \beta) - x_3)^2 + (y + l_3 \sin(\varphi + \beta) - y_3)^2} \quad (7)$$

The direct geometric problem can be solved using the analytical method proposed by [14]. The loop closure equations are expressed:

$$A_i P_i = q_i^2 i = (1, 3) \quad (8)$$

From the system of Eqs. (7) the equations of the inverse geometric problem can be written as follows:

$$q_1^2 = x^2 + y^2 \quad (9)$$

$$q_2^2 = (x + l_2 \cos\varphi - x_2)^2 + (y + l_2 sin\varphi)^2 \quad (10)$$

$$q_3^2 = (x + l_3 \cos(\varphi + \beta) - x_3)^2 + (y + l_3 \sin(\varphi + \beta) - y_3)^2 \quad (11)$$

4 Workspace

The workspace corresponds to the set of poses (positions and orientations) that the characteristic point of the end effector can reach. The workspace of a robot manipulator is considered a key feature for the various applications in robotics. It is used to analyze the performance of robot manipulators and also for optimal design. The working space is determined by the geometrical parameters of the robot manipulator such as the lengths of the elements as well as the articular limits of the motorized and passive links. From the industrial point of view, the working space of a manipulator robot is a primordial and fundamental datum. Indeed, when a trajectory is programmed we must verify that all the poses are located inside the workspace of the robot manipulator. In other more complex cases such as collision and avoidance much of the research involves analyzing the work space of a manipulator robot. This work presents different methods and techniques for determining, representing and characterizing the different types of workspaces. For our robot this is the operative zone of the robot where it is possible to fix the position G (x, y) and orientation $\boldsymbol{\varphi}$ of platform. It is obtained by drawing the three circles centered on the fixed points $\boldsymbol{A_i}$, and having for their respective radii the three articular coordinates $\boldsymbol{q}_1$, $\boldsymbol{q}_2$ and $\boldsymbol{q_3}$. In the graphics window (Fig. 2), this procedure has been automated by selecting the Work Plan menu and then the Work Area submenu. The software then displays an editor which allows the intervals of variation to be entered for each joint. The domain delimited by the three circles thus defines the working space.

5 Geometric method

5.1 Method of graphic construction

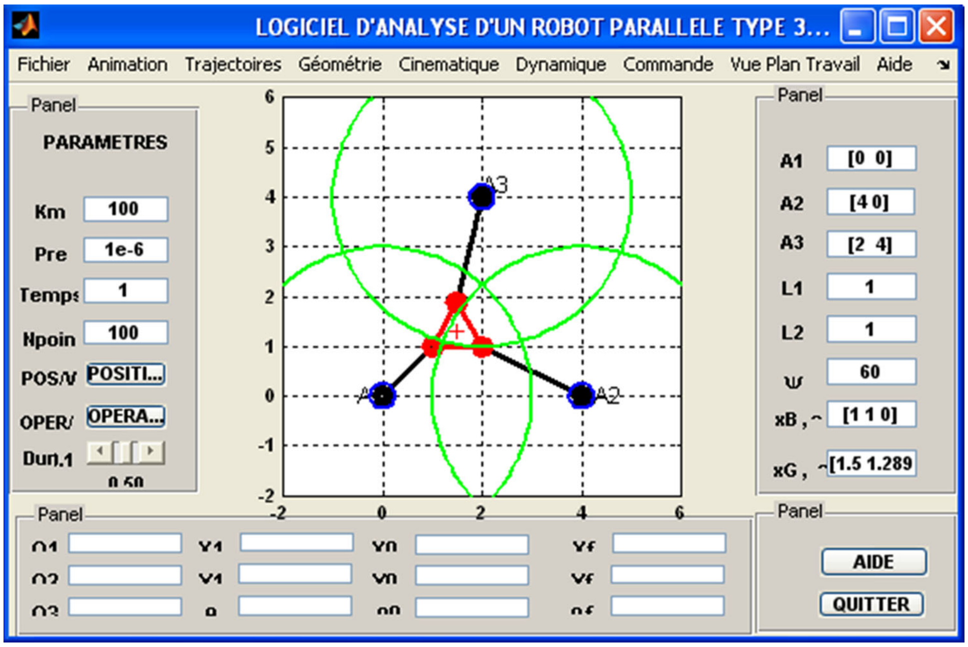
The geometric method makes it possible to determine by graphic construction the direct geometry of the robot by drawing three concentric circles $\boldsymbol{C_i}(\boldsymbol{i} = 1:3)$, the centers of which are placed at the fixed points of articulation $\boldsymbol{A_i}(\boldsymbol{i} = 1:3)$ and whose radius correspond to the articular coordinates $\boldsymbol{q_i}(\boldsymbol{i} = 1:3)$. It is then sufficient to place the platform in such a way that the vertices of the triangle $\boldsymbol{P_i} = (1:3)$ are positioned on the corresponding circles. This procedure can be automated as follows:

1. The vertex 1 will be moved in such a way that it scans the entire circle. Let $\boldsymbol{P}_1$ be its initial position $\boldsymbol{P}_1(\boldsymbol{q}_1, 0)$.
2. For each position $\boldsymbol{P}_1$, we will trace a circle of radius $\boldsymbol{l}_2$ will be drawn which will intersect the circle $\boldsymbol{C}_2$ at a point or two points ($\boldsymbol{P}_2$ and $\boldsymbol{P}'_2$). The intersection condition must be verified beforehand: distance between the two centers less than the sum $\boldsymbol{q}_1 + \boldsymbol{q}_2 + \boldsymbol{l}_2$
3. From $\boldsymbol{P}_2$, one will draw a circle of radius $\boldsymbol{l}_1$ and another circle of center P_1 and having a radius equal to $\boldsymbol{l}_3$. These two circles intersect at a point $\boldsymbol{P}_3$

It will be checked whether the point $\boldsymbol{P}_3$ belongs to the circleC_3, by calculating the distance $\boldsymbol{A}_3\boldsymbol{P}_3$ which must be equal to $\boldsymbol{q}_3$ to a precision fixed in advance.

4. The point will be retained in this eventuality, otherwise we will proceed in the same way with the second point $\boldsymbol{P}'_2$
5. The point $\boldsymbol{P}_1$ will be moved on the circle until it is completely swept. It is obvious that the condition of intersection of the circles makes it possible to sweep only certain points of the circle $\boldsymbol{C}_1$ to the detriment of others.

The sweeping step is necessarily limited (we have taken 1000 points or an increment of 0.36°), once the solution is



Km 100
Pre 1e-6
Temps 1
Npoin 100
POS/V POSITI...
OPER/ OPERA...
Dun.1 0.50

A1 [0 0]
A2 [4 0]
A3 [2 4]
L1 1
L2 1
ψ 60
xB , ~ [1 1 0]
xG , ~ [1.5 1.289]

Fig. 2 Graphical view of the Workspace

found, the procedure is resumed by determining the successive positions of the solutions that will be retained, refining the step to find the solution Optimal corresponding to the minimum error (quadratic criterion)

5.2 Calculation of the intersection coordinates of two circles known by their centers and radius $C_1(x_1, y_1, r_1)$ and $C_2(x_2, y_2, r_2)$

- Case where $x_1 = x_2$

$$y_c = \frac{r_1^2 - r_2^2 + y_2^2 - y_1^2}{2(y_2 - y_1)} \tag{12}$$

$$x_c = x_1 \pm \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + (y_2 - y_1)^2}{2(y_2 - y_1)}\right)^2} \tag{13}$$

- Case where $y_1 = y_2$

$$x_c = \frac{r_1^2 - r_2^2 + x_2^2 - x_1^2}{2(x_2 - x_1)} \tag{14}$$

$$y_c = y_1 \pm \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + (x_2 - x_1)^2}{2(x_2 - x_1)}\right)^2} \tag{15}$$

- General case

$$Ax^2 - 2Bx + C = 0 \tag{16}$$

$$A = 1 + \left(\frac{x_2 - x_1}{y_2 - y_1}\right)^2 \tag{17}$$

$$B = x_1 + \frac{(x_2 - x_1)}{(y_2 - y_1)^2}\left(r_1^2 - r_2^2 + x_2^2 - x_1^2 + (y_2 - y_1)^2\right) \tag{18}$$

$$C = x_1^2 - r_1^2 + \left(\frac{r_1^2 - r_2^2 + x_2^2 - x_1^2 + (y_2 - y_1)^2}{2(y_2 - y_1)}\right)^2 \tag{19}$$

- Case where $B^2 - AC = 0$

$$x_c = \frac{B}{A} \tag{20}$$

$$y_c = y_1 \pm \sqrt{r_1^2 - (x_c - x_1)^2} \tag{21}$$

- Case where $B^2 - AC > 0$

$$x_c = \frac{B \pm \sqrt{B^2 - AC}}{A} \tag{22}$$

$$y_c = \frac{r_1^2 - r_2^2 + x_2^2 - x_1^2 + y_2^2 - y_1^2 - 2x_c(x_2 - x_1)}{2(y_2 - y_1)} \tag{23}$$

- Case where $B^2 - AC < 0$ No Solutions

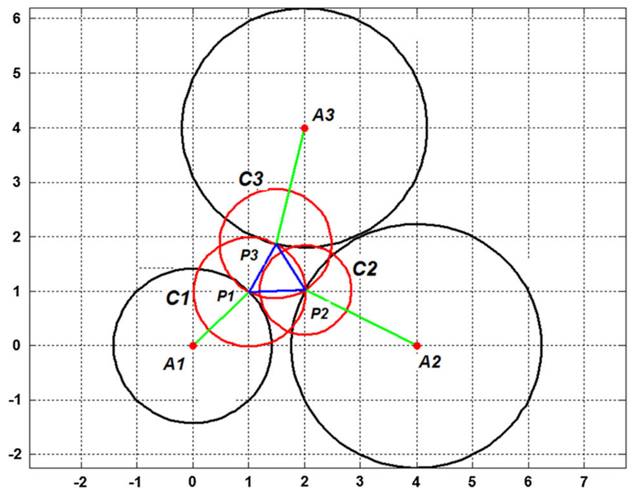


Fig. 3 Graphic construction of the solution

5.3 Graph construction

This graph was made for the following data:

$A_1(0,0)$, $A_2(3,0)$, $A_3(2,3)$, $A_1(0, 0)$, $A_2(3, 0)$, $A_3(2, 3)$, $T(1, 1, 60°)$, and $Q = [1.31322.23612.1918]$

Corresponding to the initial pose P (1, 1, 0)

The polynomial method gives us two solutions:

$$P_1 = [1.000\ -1.0000],\quad P_2 = [2.0000\ -1.0000],\quad P_3 = [1.5000\ -0.1330]$$

and:

$$P_1 = [1.0150\, 0.9838],\quad P_2 = [2.0131\, 1.0277],\quad P_3 = [1.3773\, 1.8715]$$

We have represented the graph corresponding to the first solution (Fig. 3).

The sweeping of the circle requires a not inconsiderable calculation time which depends on the power of the processor. This method is systematic and finds all possible solutions even in the case of degeneration. However, it can generate several solutions according to the fixed accuracy criterion. The higher the accuracy, the more care will be taken to retain only the exact solutions. However, if an acceptable precision is sufficient, we find solutions that the analytical calculus cannot give.

(a) Case of non-degeneration

A numerical example of the geometric method is now presented to show the case of non-degeneration given assembly. The geometric parameters are

$$x_2 = 4,\ x_3 = 2,\quad y_3 = 4,\quad l_2 = 1,\ l_3 = 1\quad \beta = \pi/3,$$

$$q_1 = 1.4142,\quad q_2 = 3.1196 \text{ and } q_3 = 2.6084$$

Table 1 Solutions for a non-degenerate manipulator

P_1	#1	#2
φ (*degrés*)	− 53.9651	60.0
x	0.3434	1
y	1.3719	1

Table 2 Positions Pi for a non-degenerate manipulator

P_i	#1		#2	
P_1	0.3434	1.3719	1	1
P_2	0.9316	0.5632	1.500	1.866
P_3	1.3378	1.4770	0.500	1.866

Table 3 Positions Pi, the articular vector and the overall resultant error for a non-degenerate manipulator

P_1	#1	#2
q_1	1.414213562373095	1.414213562373095
q_2	1.414213562373095	3.119623504137780
q_3	2.608419547897759	2.608419794289794
Error (10^{-6})	0.95087	1.19726

From the direct geometry it is an optimization which allows finding the optimal solution, presenting the minimal quadratic error, and the poses of the corresponding robot. A Matlab program was developed to calculate the minimal squared error and the robot poses (Tables 1, 2, 3).

To which correspond the two positions Pi (i = 1: 3) of the vertices of the platform of the robot.

The two poses of the platform that corresponds to these two solutions of which the data below, the arrow segments point to the numbers of the solutions of the preceding table (Fig. 4).

(b) Case degeneration of the first order

A numerical example of the geometric method is now presented to show the case of first-order degeneracy given assembly. The geometric parameters are

$$x_2 = 2,\ x_3 = 0,\quad y_3 = 1,\quad l_2 = 2,\ l_3 = 1.5,\quad \beta = \pi/3,\quad q_1 = q_2 = q_3 = 1$$

Four solutions are found in (Table 4).

With the corresponding positions Pi in Table 5.

With the corresponding articular vector and the overall resultant error in Table 6.

The graphs of the four poses are shown below (Fig. 5).

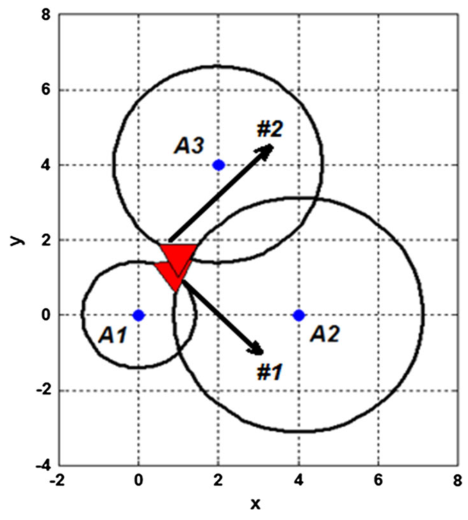


Fig. 4 The assembly mode corresponding to Table 1 for solution geometric non-degenerate

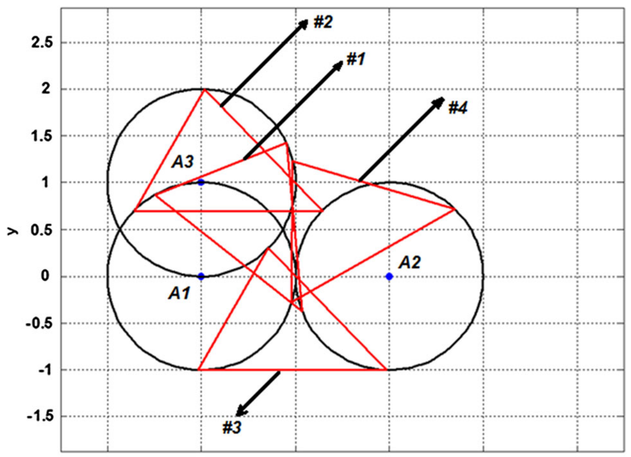


Fig. 5 The four assembly modes corresponding to Table 3 for two non-degenerate solutions, one degenerate

Table 4 The four solutions for a first order degenerate manipulator

P_1	#1	#2	#3	#4
φ(*degrés*)	– 38.6320	0	0	29.5331
x	– 0.4885	– 0.7138	– 0.0036	– 0.9618
y	0.8723	0.7003	– 0.9993	– 0.2736

(c) Case of degeneracy of the first order: four roots non degenerate, one degenerate
A numerical example of the geometric method is now presented to show the case of first order degeneracy (four degenerate roots, one degenerate) given assembly. The geometric parameters are $x_2 = 2$, $x_3 = 0.5$, $y_3 = 1$, $l_2 = 2$, $l_3 = 1.5$, $\beta = \pi/3$, $q_1 = q_2 = 1$, $q_3 = 0.7$

We find the four solutions, including one ($\varphi = 0$) degenerate (Table 7).

The corresponding poses P_i are in the Table 8.

With the corresponding articular vector and the overall resultant error in Table 9.

The four positions are shown in the following Fig. 6.

(d) Case of degeneration of order three
A numerical example of the geometric method is now presented to show the case of degeneracy of order three assembly data. The geometric parameters are

$$x_2 = 2,\ x_3 = 1,\quad y_3 = 1,\quad l_2 = 1,\ l_3 = 1,$$
$$\beta = -\pi/2,\quad q_1 = 0.8,\ q_2 = q_3 = 1.5$$

The possible solutions are shown in the Table 10.

The corresponding Pi positions are in the Table 11.

For this example we have determined the articular vector and the overall resultant error in Table 12.

The different poses are described below (Fig. 7).

The geometric method is a systematic method which allows finding all the solutions generated by the polynomial

Table 5 Positions Pi for a first-order degenerate manipulator

P_i	#1		#2		#3		#4	
P_1	– 0.4885	0.8723	– 0.7138	0.7003	– 0.0036	– 0.9993	– 0.9618	– 0.2736
P_2	1.0734	– 0.3722	1.2861	0.7003	1.9638	– 0.9993	2.7019	0.7122
P_3	0.9080	1.4188	0.0361	1.9993	0.7138	0.2996	0.9740	1.2263

Table 6 The articular vector and the overall resultant error for a first-order degenerate manipulator

P_1	#1	#2	#3	#4
q_1	1.000000000000000	0.9999999999999999	1.0000000000000	1.0000000000000
q_2	0.9999999999999992	0.9999999999999971	0.999999999999995	0.9999999999999768
q_3	1.000004023116677	1.000001598632977	9.999948946771762	0.9999959643548794
Error (10^{-6})	4.0231	1.5986	5.1053	4.03568

Table 7 The six solutions for a manipulator for the case of four non degenerate roots, one degenerate

P_1	#1	#2	#3	#4	#5	#6
φ (degrés)	– 43.8047	– 6.6275	0	0	23.6387	58.4875
x	– 0.3395	– 0.9849	– 09487	– 0.1393	0.6631	0.9768
y	0.9405	0.1728	0.3126	– 0.9902	– 0.7484	– 0.5141

Table 8 Pi positions for a manipulator for the case of four non-degenerates roots, one degenerate

P_i	#1		#2		#3		#4°		#5°		#6	
P_1	– 0.3395	0.9406	– 0.9850	0.1728	– 0.9499	– 0.3126	– 0.1394	– 0.9902	0.9768	– 0.2141	0.6632	– 0.7485
P_2	1.1039	– 0.4838	1.0017	– 0.0580	1.0501	– 0.3126	1.8606	– 0.9902	2.8090	0.5878	– 1.7085	0.9566
P_3	1.1009	1.3590	– 0.0900	1.3766	– 0.1999	0.9864	0.6106	0.3088	1.1430	1.2766	– 0.0523	0.5699

Table 9 articular vector and overall resultant error for a manipulator for the case of four non-degenerate roots, one degenerate

P_1	#1	#2	#3	#4°	#5°	#6
q_1	1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000
q_2	0.999999999999998	0.999999999999997	1.000000000000034	1.0000000000001	0.999999999999974	0.699991607557378
q_3	0.699993182215874	0.699992614668121	0.700005720554232	0.699993461795613	0.999999999999997	0.699997896382708
Error (10^{-5})	0.6817	0.7385	0.5720	0.6538	0.8392	0.2103

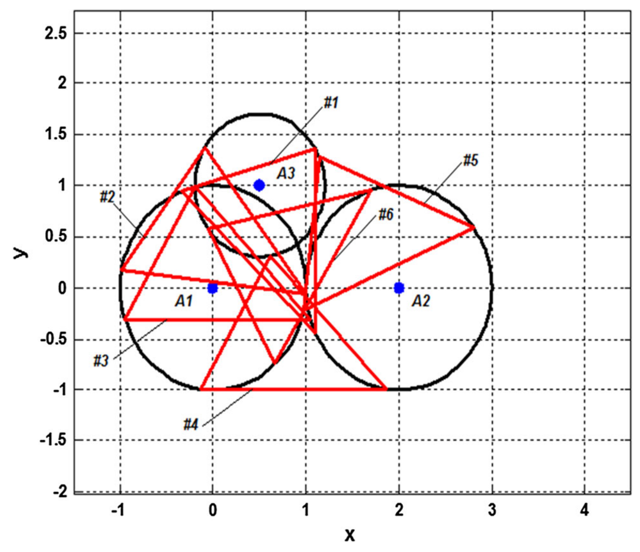


Fig. 6 The six assembly modes corresponding to Table 5 for four non-degenerate solutions, one degenerate

method with the same precision. It also makes it possible to generate other solutions that satisfy the method precision test conditions (ten solutions instead of six in the last example). The precision of the solution can be improved by reducing the sweep pitch of the circle centered on the point A_1 and of radius q_1.

We have resumed the four examples previously studied by the graphical method.

The geometric method that allows finding all the solutions generated by the Graphic method.

6 Polynomial solution

The direct geometric problem can be solved using the analytical method proposed by [15]. The loop closure equations are expressed by the expression of equation.

$$A_i P_i = q_i^2 i = (1, 3) \tag{24}$$

From the system of Eqs. (5), the equations of the inverse geometric problem can be written as follows:

$$q_1^2 = x^2 + y^2 \tag{25}$$

$$q_2^2 = (x + l_2 \cos\varphi - x_2)^2 + (y + l_2 sin\varphi)^2 \tag{26}$$

$$q_3^2 = (x + l_3 \cos(\varphi + \beta) - x_3)^2 + (y + l_3 \sin(\varphi + \beta) - y_3)^2 \tag{27}$$

The system of Eqs. (9–11) can be written in the simplified forms:

$$q_1^2 = x^2 + y^2 \tag{28}$$

$$q_2^2 = x^2 + y^2 + Qx + Ry + C \tag{29}$$

$$q_3^2 = x^2 + y^2 + Ux + Vy + F \tag{30}$$

Table 10 The six solutions for a manipulator for the case of degeneracy of order three

P_1	#1	#2	#3	#4	#5	#6
φ (*degrés*)	− 155.6267	− 90	− 90	− 48.6331	48.6331	53.6099
x	0.4335	− 0.4597	0.6547	0.2826	− 0.7484	0.3963
y	0.6724	0.6547	− 0.4597	− 0.7484	0.2826	0.6950
P_1	#7	#8	#9	#10		
φ (*degrés*)	53.6099	126.3898	126.3898	155.6268		
x	− 0.7945	0.0933	0.6950	0.6724		
y	0.0933	− 0.7945	0.3963	0.4335		

Table 11 Positions P_i, for a manipulator for the case of the degeneration of order three

P_i	#1		#2		#3		#4		#5		#6	
P_1	0.4335	0.6724	− 0.4597	0.6547	0.6547	− 0.4597	0.2826	− 0.7484	− 0.7484	0.2826	0.3963	0.6950
P_2	− 0.4774	0.2597	− 0.4597	− 0.3453	0.6547	− 1.4597	0.9434	− 1.4989	− 0.0876	1.0331	0.9895	1.500
P_3	0.8462	− 0.2385	− 1.4597	0.6547	− 0.3453	− 0.4597	1.0331	− 0.0876	− 1.4989	0.9434	1.2013	0.1017
P_i	#7		#8		#9		#10					
P_1	− 0.7945	0.0933	0.0933	− 0.7945	0.6950	0.3963	0.6724	0.4335				
P_2	− 0.2013	0.8983	− 0.5000	0.0105	0.1017	1.2013	− 0.2385	0.8462				
P_3	0.0105	− 0.500	0.8983	− 0.2013	1.5000	0.9896	0.2597	− 0.4774				

Table 12 Articular vector and overall resultant error for a manipulator for the case of degeneracy of order three

P_1	#1	#2	#3	#4°	#5°	#6
q_1	0.800000000000000	0.8000000000000002	0.8000000000000000e	8.000000000000000	8.000000000000000e	8.000000000000002
q_2	1.500000000000000	1.500000000000000	1.500000000000000	1.500000000000000	1.500000000000000e	1.500000000000000
q_3	1.500001310578265	1.499998227294674	1.499998618717735e	1.499997131024234	1.499994545565069e	1.499992739634409
Error (10^{-6})	1.315	1.7727	1.38183	2.8689	5.4544	7.2603
P_1	#7	#8	#9	#10°		
q_1	0.8000000000000000	0.8000000000000000	0.80000000000008	8.000000000000000		
q_2	1.499999999999998	1.500000000000000	1.500000000000000	1.500000000000000e		
q_3	1.500001037586281	1.499998508919643	1.500000440110932	1.500001649812265		
Error (10^{-5})	1.0375	1.4910	9.9541	1.6498		

With intermediate variables:

$$Q = 2l_2 \cos\varphi - 2x_2 \tag{31}$$

$$R = 2l_2 \sin\varphi \tag{32}$$

$$C = x_2^2 + l_2^2 - 2l_2x_2\cos\varphi \tag{33}$$

$$U = 2l_3 \cos(\varphi + \beta) - 2x_3 \tag{34}$$

$$V = 2l_3 \sin(\varphi + \beta) - 2y_3 \tag{35}$$

$$F = x_3^2 + y_3^2 + l_3^2 - 2l_3x_3 \cos(\varphi + \beta) - 2l_3y_3 \sin(\varphi + \beta) \tag{36}$$

It has already been established that, at most for a given combination of articular variables, there may be six possible solutions. The solutions are always even: two or four or six. In the case where the solution exists it is always matched. In the case of non-existence all solutions are complex.

(a) Case of non-degeneration

A numerical example of the polynomial method is now presented to show the case of non-degeneracy (two solutions) given assembly. The geometric parameters are

$$x_2 = 4,\ x_3 = 2,\quad y_3 = 4,\ l_2 = 1,\ l_3 = 1,\quad \beta = \pi/3,$$
$$q_1 = 1.4142,\quad q_2 = 3.1196,\quad q_3 = 2.6084$$

The dimensions of the platform are defined by the vector $T = [l_2\ l_3\ \beta]$

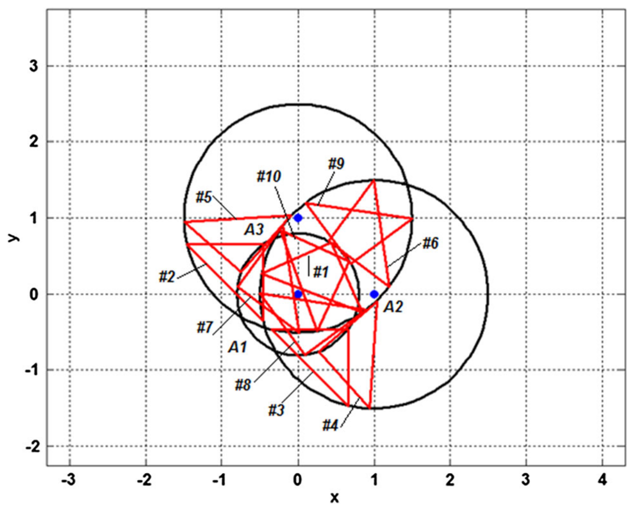


Fig. 7 The ten assembly modes corresponding to Table 7 for degeneration of order three

The passive points Ai are defined by the matrix:

$$A = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}$$

This combination was obtained by calculating the inverse geometry which starting from the position data of the three fixed points A, whose coordinates are represented by the matrix A. The configuration of the triangle representing the platform (matrix T [l_2 l_3 opening angle β (In degrees)]. The coordinates of the point M = P_1 (x, y, φ) calculates the articular vector Q = [q_1, q_2, q_3] and the remaining points P (P_2 and P_3).

The call of the program function implementing the polynomial method and displaying the platform poses with the indication of the angle and coordinates of the vertex P_1 gives the values of the other points Pi in the structured variable P and the corresponding angles in the φ vector. The input arguments are the matrix as already defined, the vector T and the vector Q (Fig. 8).

The two possible solutions are shown in the following Table 13.

The positioning error for each pose is calculated by the Matlab function, Which, starting from the positions Pi (i = 1:3), the matrix A, and the vector T, Addresses the Matlab function which returns the actual articular coordinates and compares it to the desired joint position using a quadratic criterion (Table 14).

We find an error: Er $= [0.0888 \;\; 0.1110] \cdot 10^{-14}$.

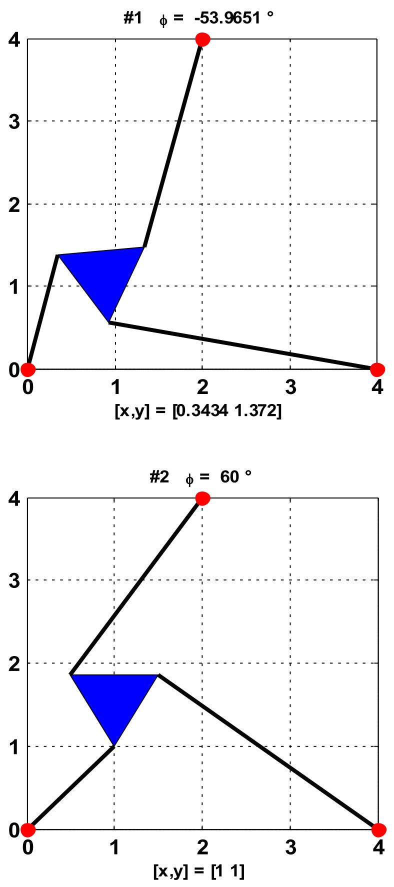


Fig. 8 Two assembly modes corresponding to Table 13 correspond to non-degeneracy with two solutions

Table 13 The two sets of solutions for a manipulator correspond non-degeneration has two solutions

P_1	#1	#2
φ (degrés)	– 53.9651	60.0
x	0.3434	1
y	1.3719	1

Table 14 Positions Pi for a corresponding manipulator to no degeneration with two solutions

P_i	#1		#2	
P_1	0.3434	1.3719	1	1
P_2	0.9316	0.5632	1.500	1.866
P_3	1.3378	1.4770	0.500	1.866

(b) Case of degeneration of order 1: four solutions

A numerical example of the polynomial method is now presented to show the case of degeneracy of order 1 (four solutions) given assembly. The geometric parameters are

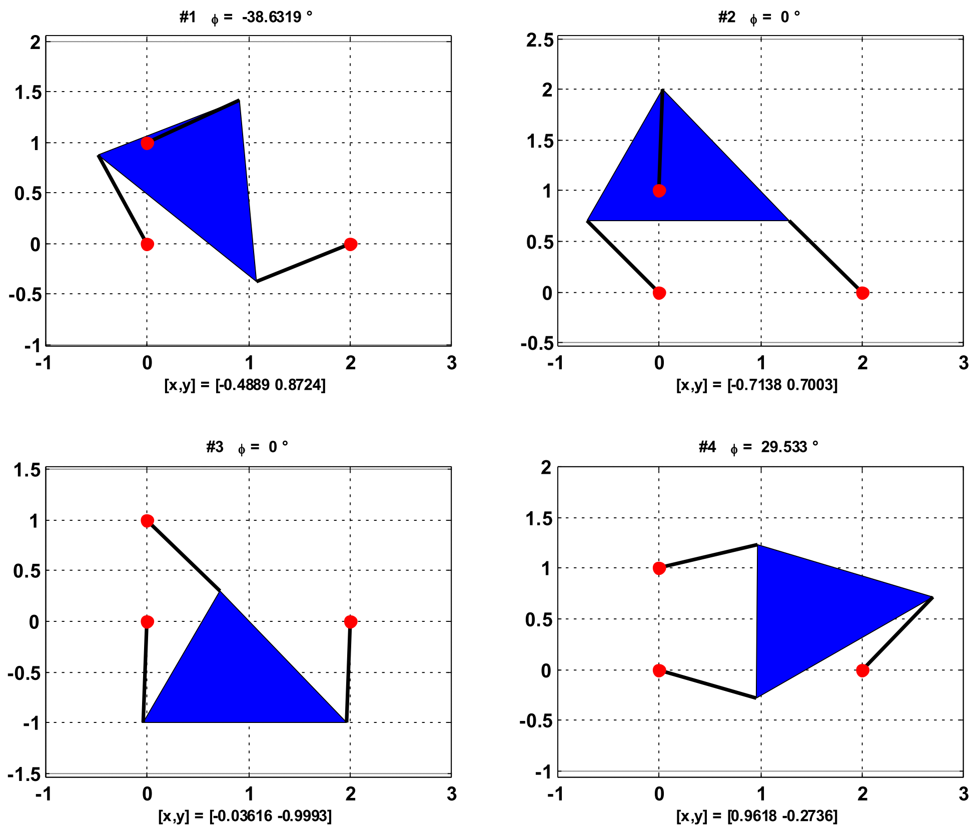


Fig. 9 The four assembly modes corresponding to the Table 15 corresponding to the degeneration of order 1 with four solutions

Table 15 The four sets of solutions for a manipulator correspond to degeneration of order 1 has four solutions

P_1	#1	#2	#3	#4
φ (*degrés*)	0	0	− 38.6319	29.5330
x	0.7138	− 0.0362	− 0.4889	0.9618
y	0.7003	− 0.9993	0.8724	− 0.2736

$$x_2 = 2,\ x_3 = 1,\quad y_3 = 1,\quad l_2 = 1,\ l_3 = 1,$$
$$\beta = -\pi/2,\quad q_1 = 0.8,\ q_2 = q_3 = 1.5$$

We are in the case where $x_2 = l_2$ and $q_1 = q_2$. We shall therefore be faced with degeneration of the first order. The angle phi equal to zero admits two positions

The degeneracy conditions for the three solutions are verified for these data (Fig. 9).

The four sets of solutions are reported in Table 15.

The six sets of positions Pi are reported in Table 16.

We find an error: Er = $[0.0089\ \ 0.0011\ \ 0.0022\ \ 0.1132] \cdot 10^{-13}$.

(c) Case degeneration of order 1: six solutions

A numerical example of the polynomial method is now presented to show the case of degeneracy of order 1 (six solutions) given assembly. The geometric parameters are defined by $x_2 = l_2 = 2$, $x_3 = 1/2$, $y_3 = 1$, $l_3 = 3/2$ and $\beta = \pi/3$.

$$q_1 = q_2 = 1 \text{ and } q_3 = 7/10,$$

As before, $x_2 = l_2$ and $q_1 = q_2$. The angle phi equal to zero admits two positions. The four other solutions will be non-degenerate (Fig. 10).

The six sets of solutions are reported in Table 17.

The six sets of positions Pi are reported in Table 18.

Table 16 Positions Pi for a manipulator the last two correspond to degeneration of order 1 has four solutions

P_i	#1		#2		#3		#4	
P_1	− 0.7138	0.7003	− 0.0362	− 0.9993	− 0.4889	0.8724	0.9618	− 0.2736
P_2	1.2862	0.7003	1.9638	− 0.9993	1.0735	− 0.3763	2.7020	0.7122
P_3	0.0362	1.9993	0.7138	0.2997	0.9080	1.4189	0.9741	1.2263

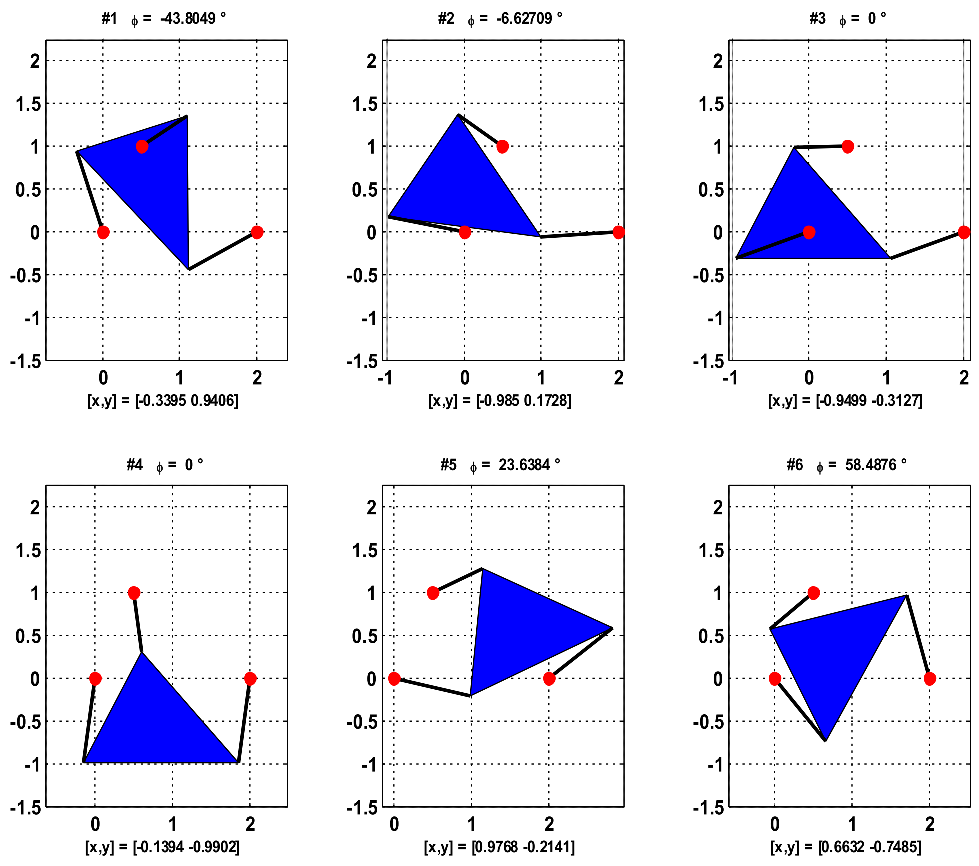


Fig. 10 The six assembly modes corresponding to Table 14 with the last two corresponding to the degenerate root

Table 17 The six sets of solutions for a manipulator with the last two corresponding to the degenerate root

P_2	#1	#2	#3	#4	#5	#6
φ (degrés)	– 43.8049	– 6.6271	0	0	23.6384	58.4876
x	– 0.3395	– 0.9850	– 0.9499	– 0.1394	0.9768	0.6632
y	0.9406	0.1728	– 0.3127	– 0.9902 –	– 0.2141	– 0.7485

Table 18 Positions Pi, for a manipulator with the last two corresponding to the degenerate root

P_i	#1		#2		#3		#4		#5		#6	
P_1	– 0.3395	0.9406	– 0.9850	0.1728	– 0.9499	– 0.3127	– 0.1394	– 0.9902	0.9768	– 0.2141	0.6632	– 0.7485
P_2	1.1039	0.4438	1.0017	0.0580	1.0501	– 0.3127	1.8606	– 0.9902	2.8090	0.5878	1.7085	0.9566
P_3	1.1010	1.3590	– 0.0900	1.3768	– 0.1999	0.9864	0.6106	0.3088	1.1430	1.2766	– 0.0523	0.5699

We find an error: Er $=$ [0.0067 0.5473 0.0033 0.0100 0.0100 0.0344] $\cdot 10^{-13}$.

(d) Case of degeneration over the whole space

A numerical example is now presented to show the different modes of degenerate manipulation given assembly. The geometric parameters are $x_2 = l_2 = 1, x_3 = 0, y_3 = 1, l_3 = 1$ and $\beta = -\pi/2$ these parameters fulfill the geometric conditions for manipulation to be degenerate. The direct kinematics is now a calculator for $q_3 = 4/5$, $q_2 = q_3 = 3/2$.

The six sets of solutions are reported in Table 19.

The six sets of positions Pi are reported in Table 20

The conditions of degeneracy for the set of solutions are verified for these data (Fig. 11).

We find an error:

$$\mathrm{Er} = \left[2.2210 \cdot 10^{-15} \quad 1.7763 \cdot 10^{-15} \quad 4.4408 \cdot 10^{-16} \quad 2.2204 \cdot 10^{-16} \quad 1.1738 \cdot 10^{-08} \quad 1.1738 \cdot 10^{-08}\right]$$

Table 19 The six sets of solutions for a degenerate manipulator

P_1	#1	#2	#3	#4°	#5°	#6
φ (*degrés*)	− 90	− 90	53.6102	53.6102	126.389	126.389
x	− 0.4597	0.6547	− 0.7945	0.3963	0.6950	0.0933
y	0.654745	− 0.4597	0.0933	0.6950	0.3963	− 0.7945

Table 20 Positions Pi for a degenerate manipulator

P_i	#1		#2		#3		#4°		#5°		#6	
P_1	− 0.4597	0.6547	0.6547	− 0.4597	− 0.7945	− 0.0933	0.3963	0.6950	0.6950	0.3963	0.0933	− 0.7945
P_2	− 0.4597	− 0.3453	0.6547	− 1.4597	− 0.2013	0.8983	0.9895	1.500	0.1017	1.2013	− 0.500	0.0105
P_3	− 1.4597	0.6547	− 0.3453	− 0.4597	0.0105	− 0.5000	1.2013	0.1017	1.500	0.9895	0.8983	− 0.2013

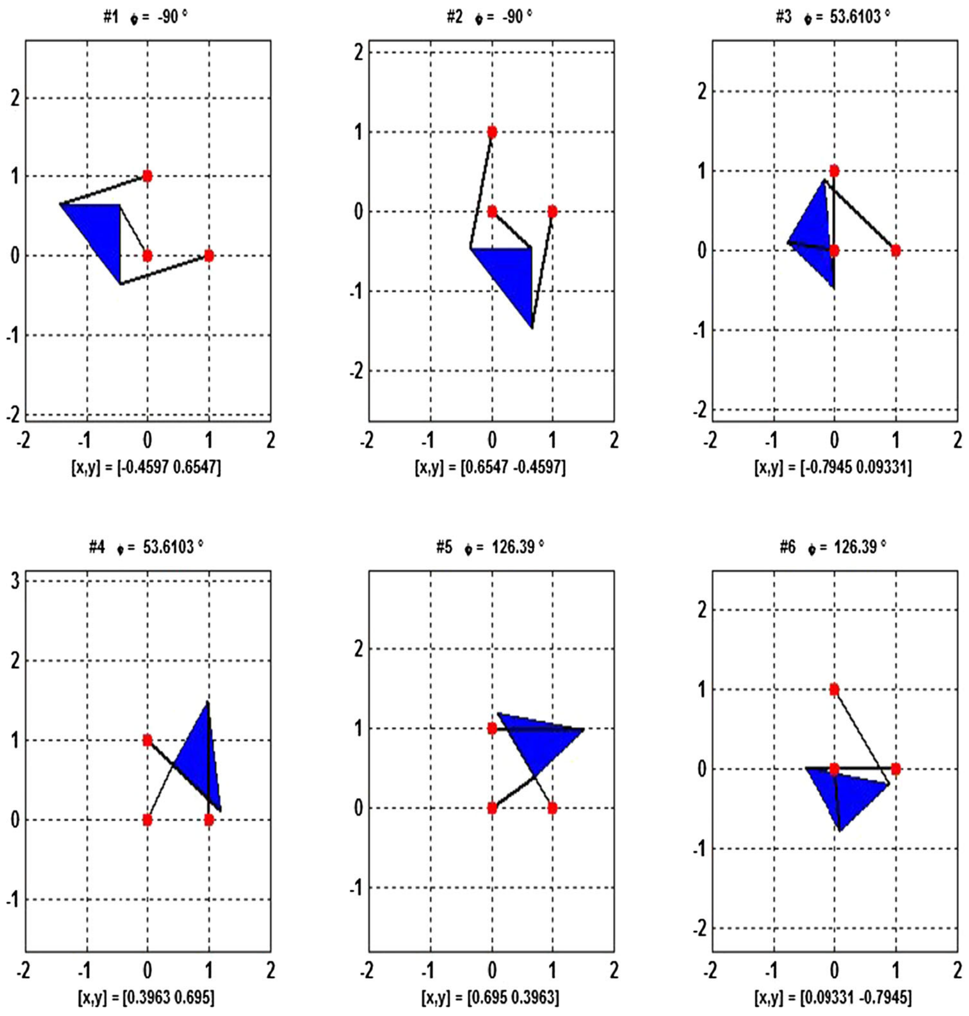


Fig. 11 The six assembly modes corresponding to Table 16

The polynomial method is systematic. It always gives even solutions in the case where the polynomial of order six admits solutions. The problematic case is that where the solution is degenerate which corresponds to a singularity that is to say that for the same articular coordinates. There correspond two distinct solutions which must be calculated according to the procedure. This type of singularity is distinct from that encountered when calculating the direct kinematics where the calculation of the operational speeds may prove impossible for certain configurations of the robot.

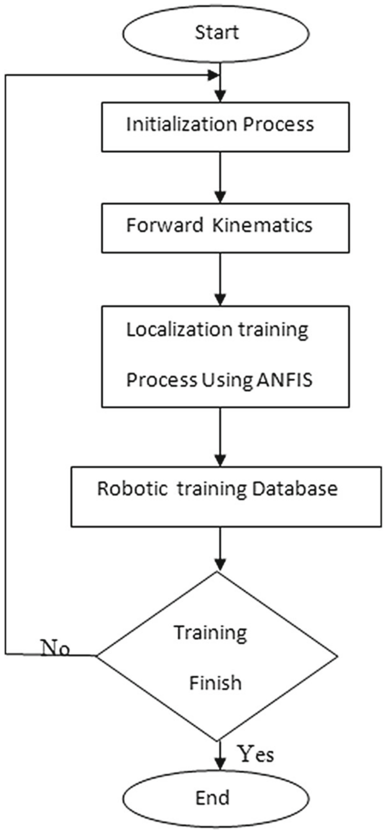


Fig. 12 Flowchart of training process

7 Method using ANFIS

7.1 Introduction to the adaptive neuro fuzzy inference system method (ANFIS)

Neuro- Fuzzy systems combine the advantages of two complementary techniques. Fuzzy systems provide a good representation of knowledge. The integration of neural networks within these systems improves their performance through the ability to learn neural networks. Conversely, the injection of fuzzy rules into neural networks, which are often critical for their lack of readability, clarifies the meaning of the network parameters and facilitates their initialization, which represents a considerable saving of time for their identification [16–19].

In this paper, a supervised ANFIS approach is developed to control the movement of 3RPR planar parallel manipulators. The hybrid approach (Neuro-Fuzzy) is used to solve the problem of direct kinematic solutions. This approach also overcomes the problems of singularities and uncertainties that occur during trajectory planning because it has, like any ANFIS algorithm, a generalization capacity [20–22] (Fig. 12).

We have illustrated this method on the first example analyzed by the two previous methods and which admits two solutions. The Matlab function geo_directe_fuzzy that implements this method is given in the “Appendix”. It calls for the following remarks:

1. It uses the Matlab anfis tool to automatically generate the inference rules using the neural networks of the three structures with three inputs q_1, q_2 and q_3 and one output, x, y and phi, respectively anfis1, anfis2 and anfis3.
2. The number of channels that can be taken by the output depends on the number of membership functions adopted for each input. If we call n1, n2 and n3 the number of these membership functions (MF: Membership function), then the number of channels of the neural structure will be equal to n1 × n2 × n3
3. Firstly, two candidate functions will be taken by input of the bell function type (gbellMF), which will generate 8 channels at the output, then thereafter one will take three functions per input which will generate 27 channels. The fuzzy structures will be called fismat1, fismat2 and fismat3.
4. The values of the variation intervals for each joint value, that of the parameters of the membership functions, as well as the linear and non-linear coefficients of the outputs are given in “Appendix”.
5. The matrix of the input data for each structure will have a size of n^3 if n is the grid pitch adopted for each of the three dimensions (meshgrid). One notices that this size becomes rapidly voluminous and can lead to an overflow of the memory signaled by Matlab. The execution times for the two examples studied are on average two (02) and nine (09) minutes for each structure of the two examples. The compilation times quickly become exhibitory as soon as the grid density (mesh) increases.
6. In order to apply this method, we have been led to reduce the search domain of the solution and to split the grid size by two by assigning half of the data to the initiation step (the odd indices), and the remaining half (even indices) to learning which will generate the final fuzzy structure. The definition files are given in the “Appendix” for both examples.

7.2 First case: case where two membership functions per input are used

The analysis research domain that allows generating the three dimensional grid is as follows:

$$0.8 \le x \le 1.2, \quad 0.8 \le y \le 1.2, \quad 45^\circ \le \varphi \le 90^\circ$$

Calling the Matlab function geo_directe_fuzzy with these parameters uses the anfis function which defaults to two bell

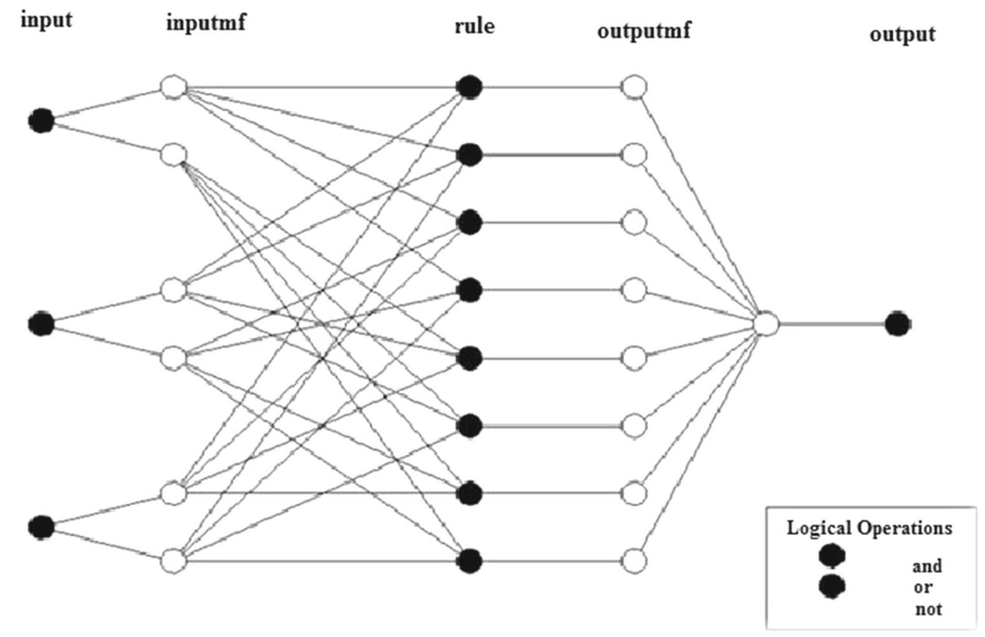


Fig. 13 Structure output

Table 21 coefficients a, b, c of the bell functions

	a	b	c
q_1	0.2926122351942617	1.999575554910495	1.137526254458919
	0.2921104456450778	1.999508511833954	1.691752052184750
q_2	0.3309800878348759	1.998893941500382	2.879697690925664
	0.3315245202873079	1.998958418473497	3.359159191615123
q_3	0.6411531208672442	1.999966155038961	2.117796859719111
	0.6412853049205336	1.999975382640182	3.399587220647379

membership functions for initiating the fuzzy structure and automatically generating inference rules.

The schematic of the structure obtained by the Matlab rule edit or rule view commands followed by the selection of the “Edit” and then “Structure” menus is given below (only the AND operator is used): Fig. 13.

The $2^3 = 8$ inference rules are as follows: (Rule Edit command)

1. (if q1 is inmf1) and (q2 is inmf1) and (q3 is inmf1) then (x is outmf1)
2. (if q1 is inmf1) and (q2 is inmf1) and (q3 is inmf2) then (x is outmf2)
3. (if q1 is inmf1) and (q2 is inmf2) and (q3 is inmf1) then (x is outmf3)
4. (if q1 is inmf1) and (q2 is inmf2) and (q3 is inmf2) then (x is outmf4)
5. (if q1 is inmf2) and (q2 is inmf1) and (q3 is inmf1) then (x is outmf5)
6. (if q1 is inmf2) and (q2 is inmf1) and (q3 is inmf2) then (x is outmf6)
7. (if q1 is inmf2) and (q2 is inmf2) and (q3 is inmf1) then (x is outmf7)
8. (if q1 is inmf2) and (q2 is inmf2) and (q3 is inmf2) then (x is outmf8)

The bell membership functions for the three inputs can be visualized by the Matlab mfedit command or calculated directly from the data of their parameters a, b, c. We can also address the function Matlab gbellmf (Table 21; Figs. 14, 15, 16, 17):

$$f(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}$$

The curves of the three outputs according to the three inputs will have the following appearance (Fig. 15).

We can then evaluate the predicted outputs for the three structures that correspond to the solution of the problem of direct geometry by the Matlab evalfis

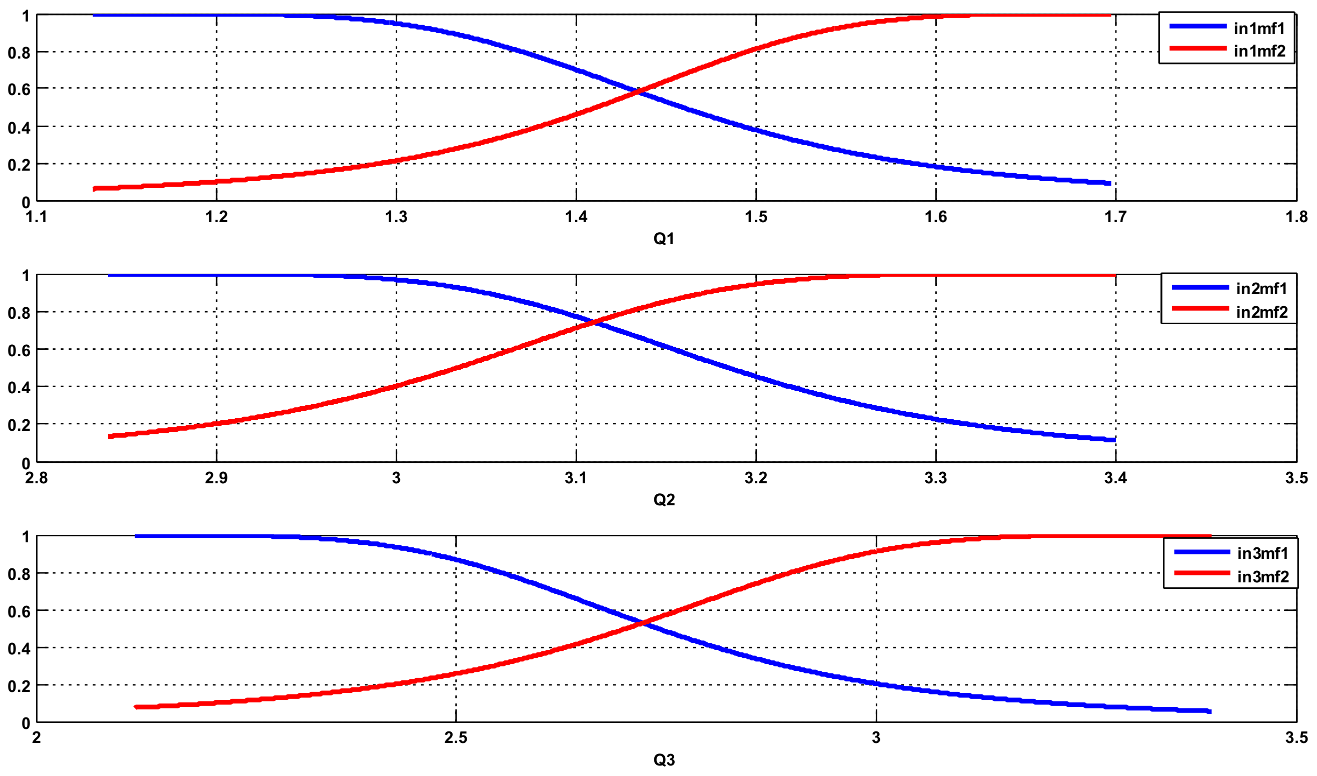


Fig. 14 Plot of the membership functions

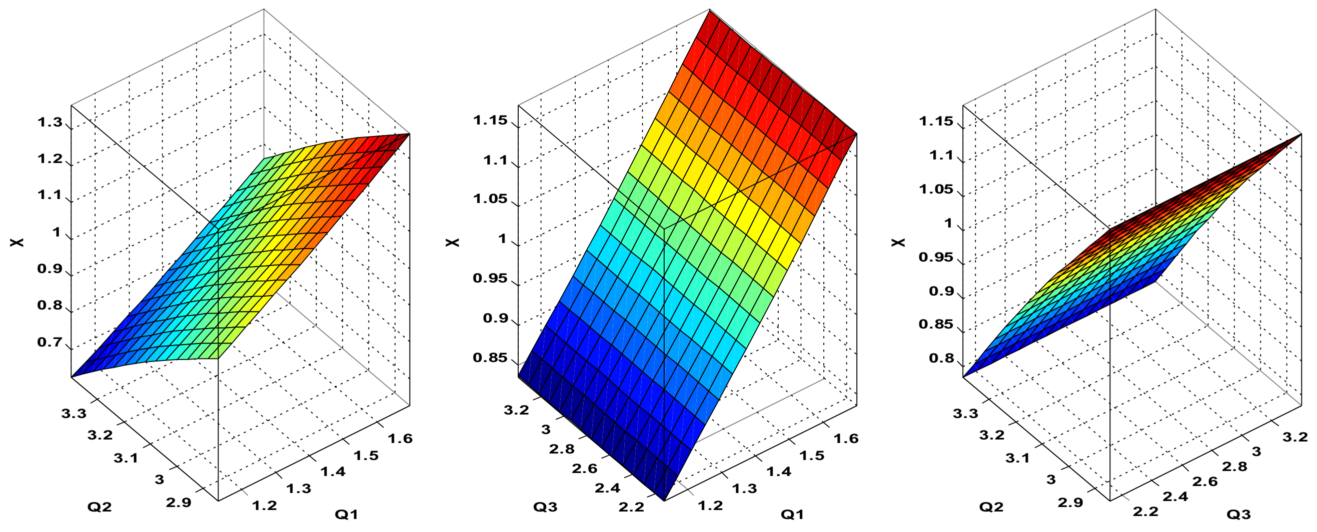


Fig. 15 Output curves X = f (q1, q2, q3)

Input Articular Vector:

$$Q = [1.414213562373095 \quad 3.119623504137779 \quad 2.608418597022603]$$
$$\text{Sortie } x = \text{evalfis}(Q, \text{anfis1}) = 1.000027167658954$$
$$\text{Sortie } y = \text{evalfis}(Q, \text{anfis2}) = 1.000670581045958$$
$$\text{Sortie } \varphi = \text{evalfis}(Q, \text{anfis3}) = 1.049580733914649$$
$$\text{Theoretical value}[1, \ 1, \ \pi/4]$$

Let an absolute squared error be equal to: $2.47{\cdot}10^{-3}$ to improve this precision, one can either decrease the value of the pitch of the meshgrid, or increase the number of membership functions, thus the number of rules of inferences.

We have adopted this approach for the second example.

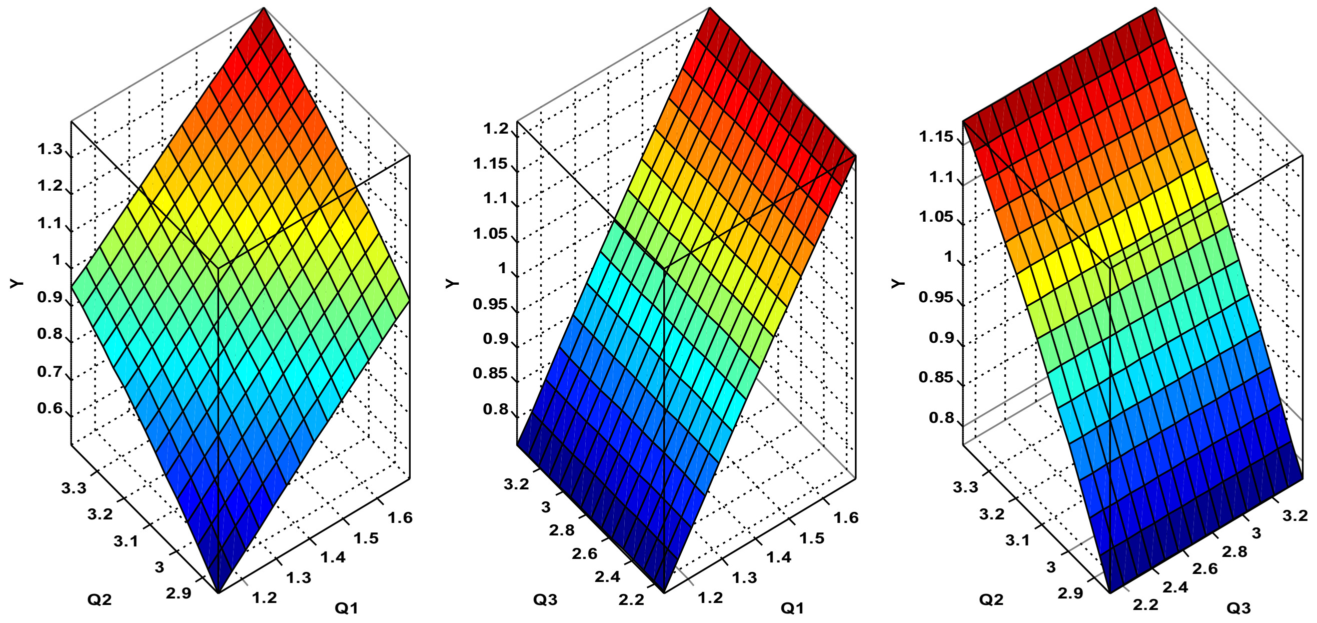


Fig. 16 Output curves Y = g (q1, q2, q3)

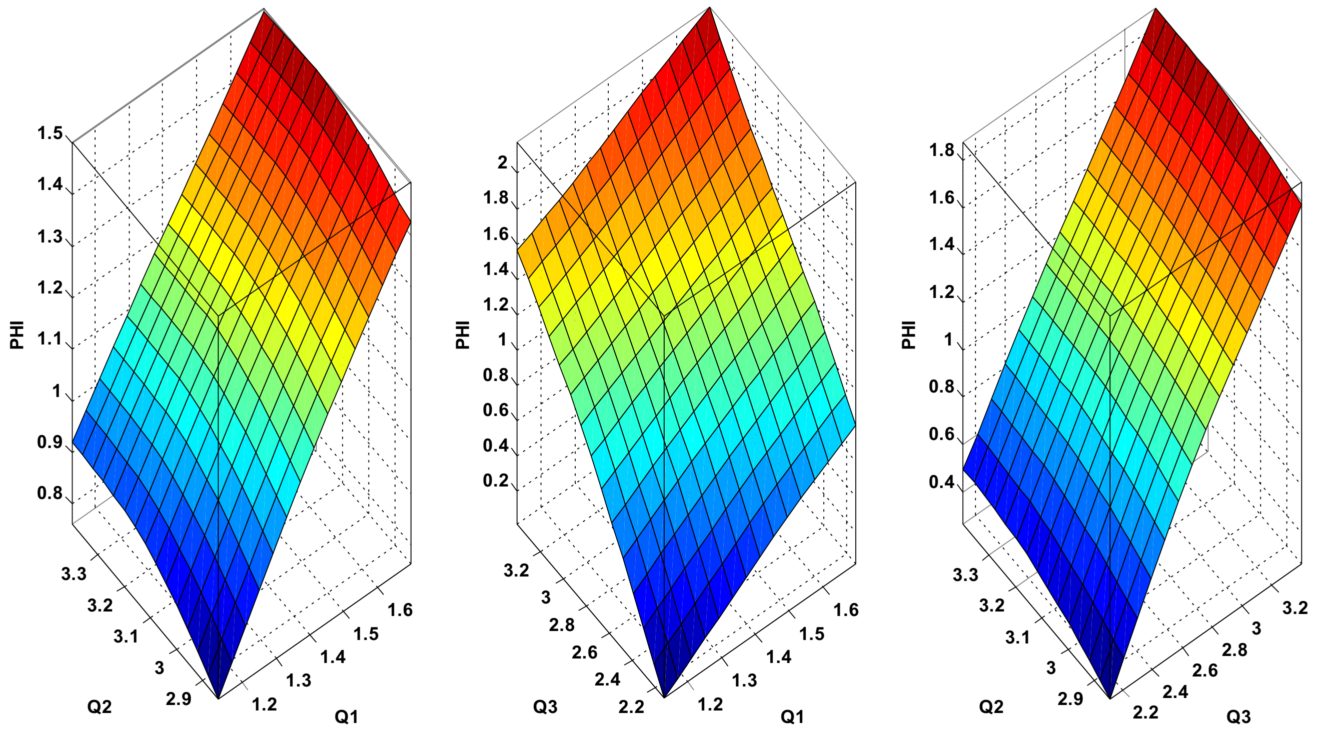


Fig. 17 Output curves φ = h (q1, q2, q3)

7.3 Second case: use of three MF membership functions per entry

The three functions induce twenty-seven 27 rules obtained by factorial combination of 3 × 3 × 3 values. The resulting neural structure (fismat1, fismat2 and fismat3) for the three outputs will be as follows, only the AND connector is used (Fig. 18).

The coefficients a, b, c for the three inputs is shown in the following Table 22.

As before, only the mapping of the membership functions for output X is given. The variation of these functions is given by (Fig. 19).

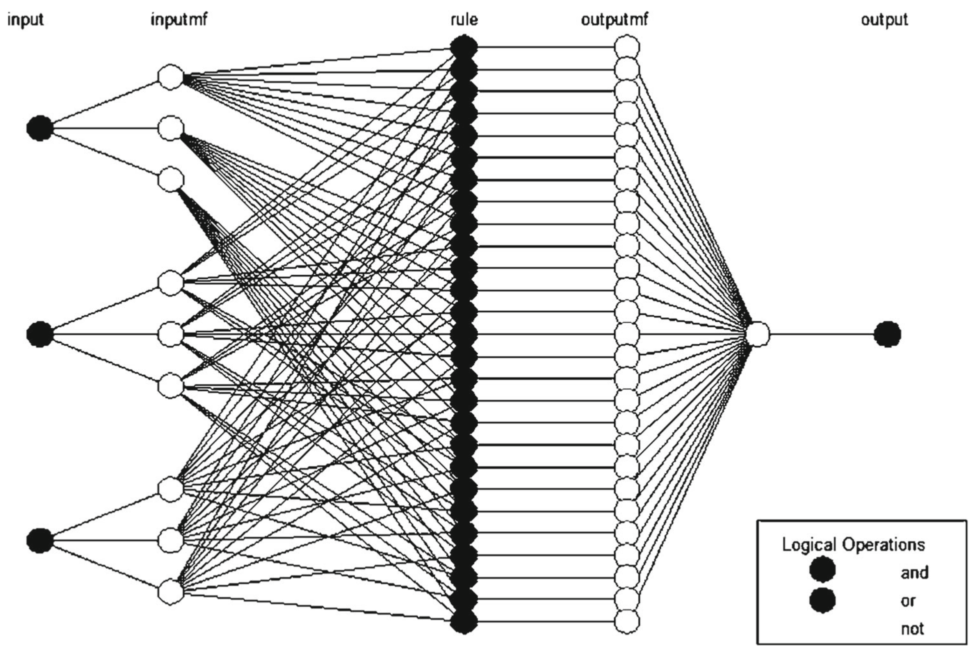


Fig. 18 Structure of the output

Table 22 Coefficients a, b, c of candidate bell functions

	a	b	c
q_1	0.1869765621511652	1.998865172765118	1.162004288131832
	0.1957437245602229	2.000463685625106	1.412000420106174
	0.1853454687358375	1.999372490381056	1.658321693403733
q_2	0.1964474545565682	1.997127645803953	1.154088654049545
	0.1881717541684498	2.000496920186938	1.412078678112379
	1.842899913945532	1.999333777721351	1.660393593238623
q_3	0.2028818362109726	1.995342301460311	1.144793336155356
	0.1828246535062499	2.001584855884983	1.424752650303841
	0.1598563243030001	2.000327527063421	1.675333465729023

The graphs of the outputs as a function of the inputs are represented by the two-dimensional curves (Figs. 20, 21, 22).

The solution found is obtained by following the same approach will be:

x = 1.000000956587682
y = 0.9999828841351883

Let an absolute squared error equal to: $1.57{\cdot}10^{-4}$.

A 25% step reduction (n = 75) will require an average execution time of 20 min (20 min) per structure and gives the following results

x = 1.000000956587682
y = 1.000000956587682
φ = 1.049506257922610

The model of the PC used is Intel type dual core, 2.4 GHz, and 1 Gb of RAM.

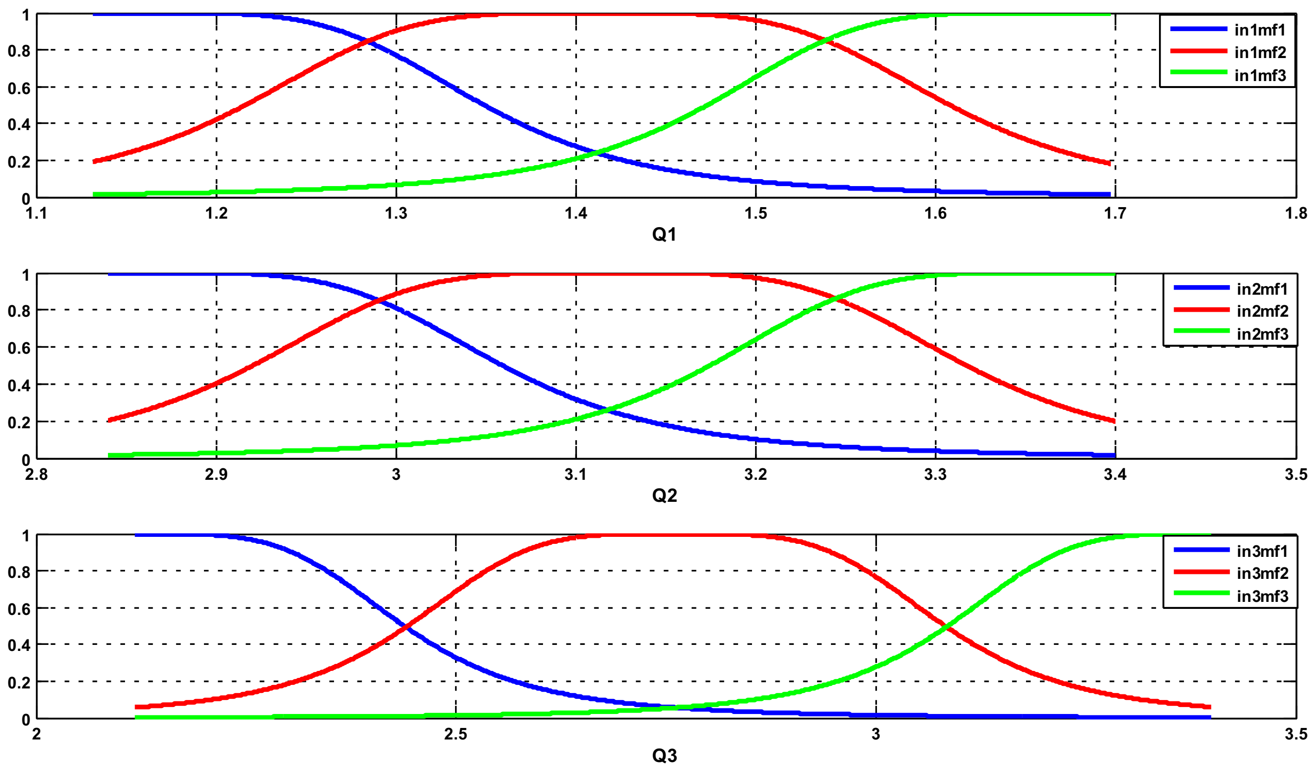


Fig. 19 Plot of the membership functions

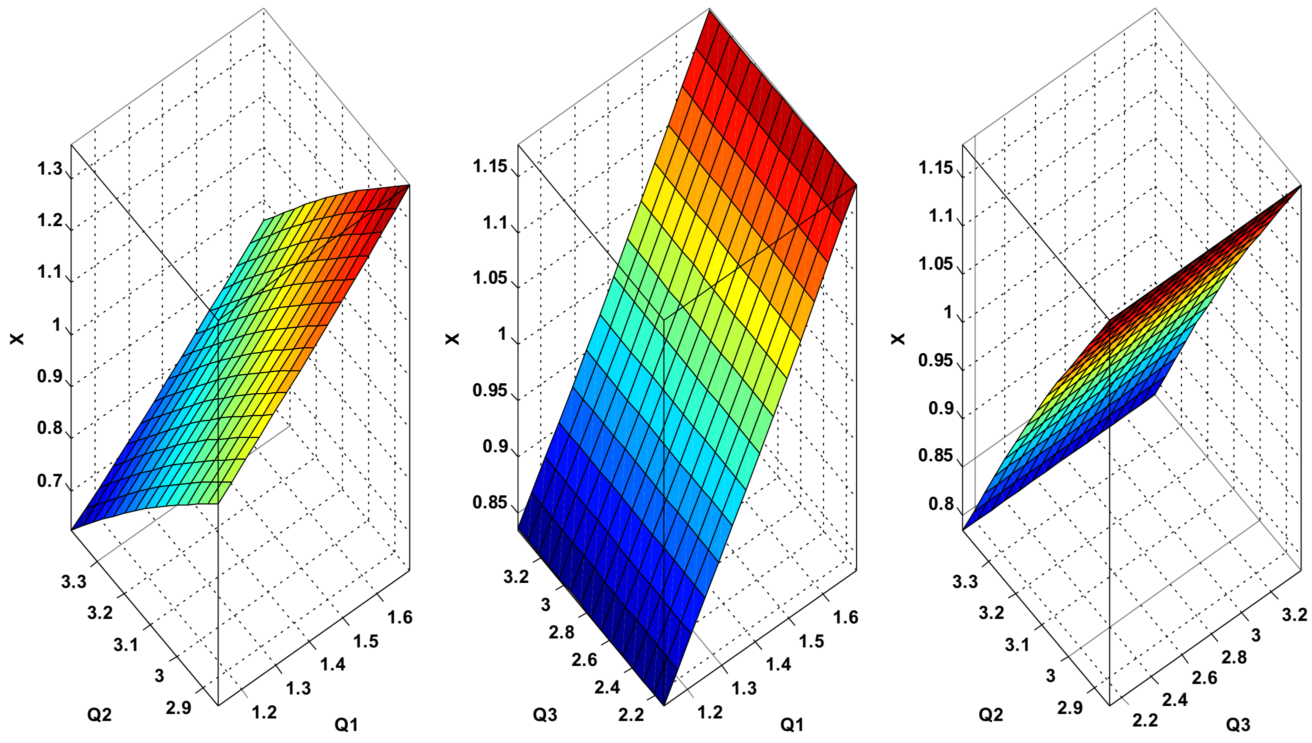


Fig. 20 Plot of the Output X Membership Functions

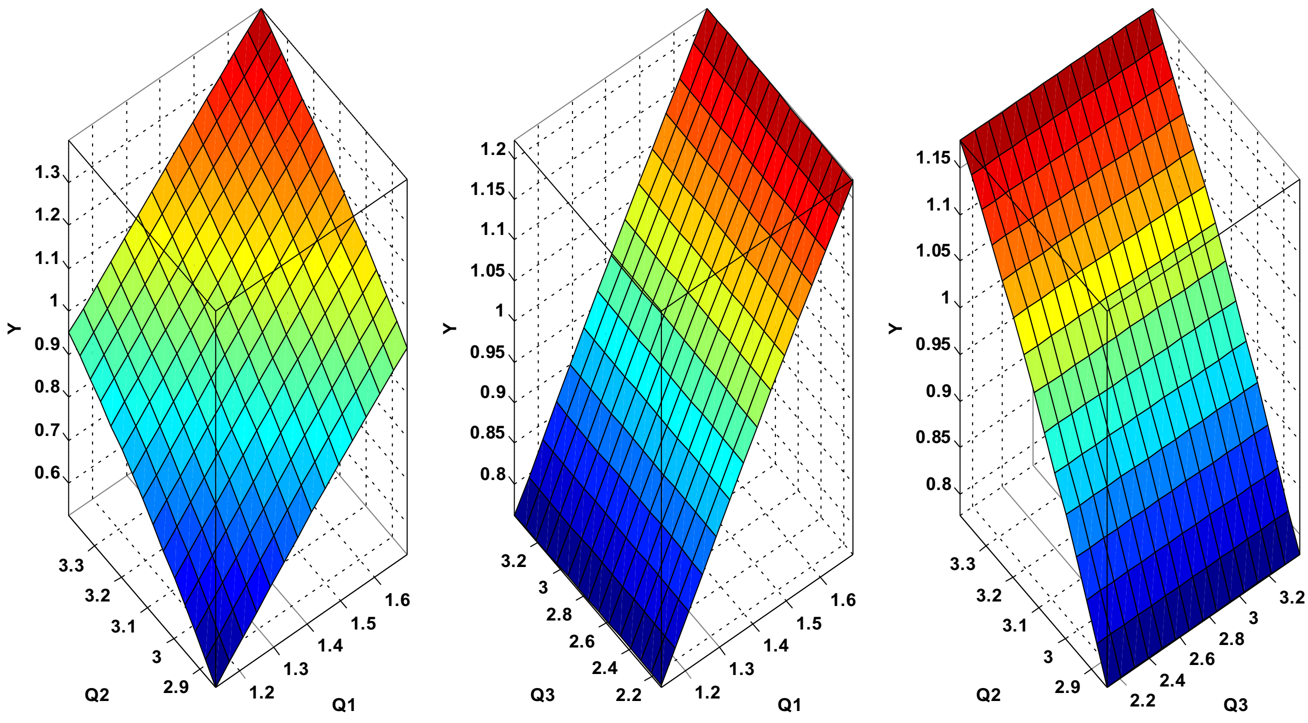


Fig. 21 Plot of the Output Y Membership Functions

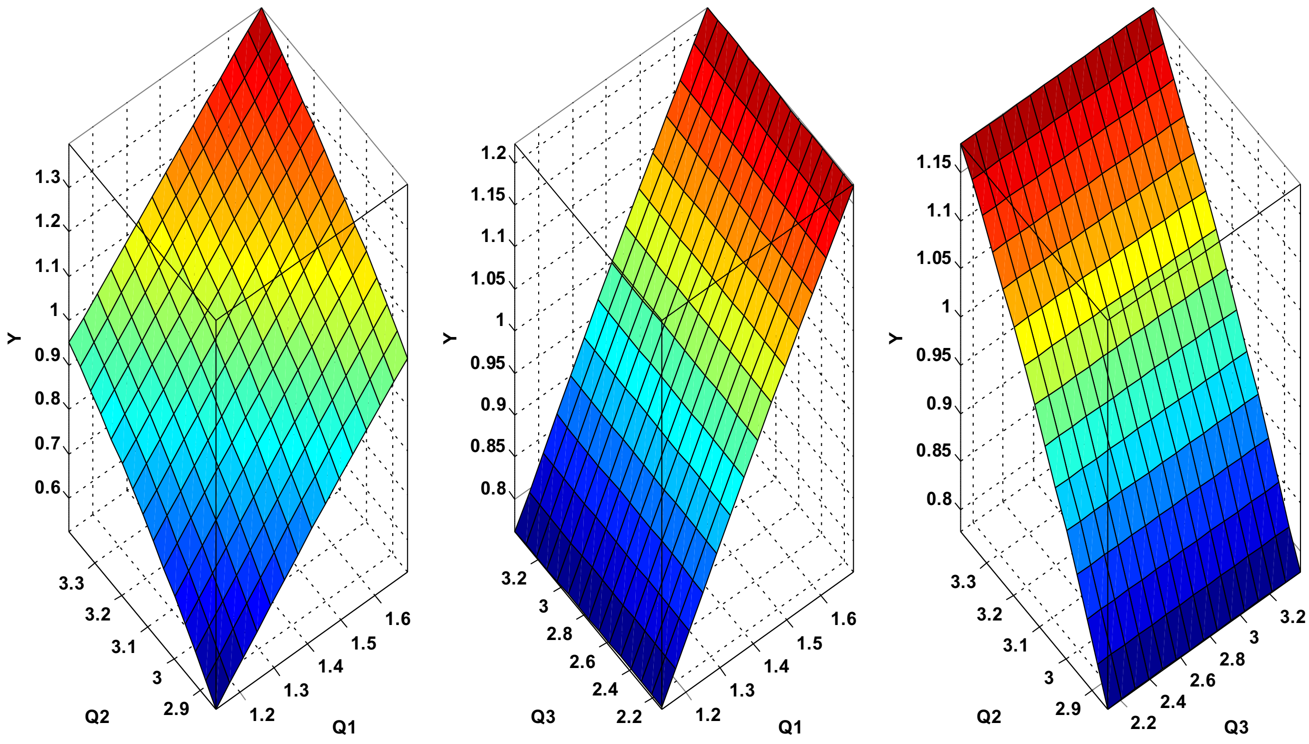


Fig. 22 Plot of the Membership Functions φ Output

Let us consider an absolute squared error equal to: $1.56 \cdot 10^{-4}$ of the same order as that found above but with three entries. An increase in the number of inputs with the grid pitch being maintained ($n = 5$), improves the absolute precision on the abscissa X) and the gate at $0.9 \cdot 10^{-5}$. However, at the cost of a run time for A single structure (output X) of two hours and fifty minutes (2 h 50 min).

The calculation of the direct geometry by the neural network-fuzzy logic approach requires relatively long execution times for the determination of the structures that establish the input–output correspondence. But this step necessary to determine the fuzzy structure once realized, will be able to determine the solution of the problem for any input articular vector insofar as it belongs to the intervals of variations.

The accuracy is greatly improved by reducing the grid pitch value of the variation range. This is also the case if the number of membership functions per entry is increased.

However, this method cannot determine all the solutions corresponding to a given articular vector or a fortiori degenerate solutions.

Moreover, the grid of the whole domain for a satisfactory precision leads to prohibitive matrix sizes and will inevitably lead to memory overruns.

The solution is either to work on a high-performance workstation with computational power and enough RAM or to partition the domain into subdomains on which so many structures will be defined, look for the optimal solution, checks the conditions of belonging to the domain and presents the minimum error if several solutions are found.

8 Conclusion

The work presented in this paper concerns the modeling of nonlinear systems using advanced techniques such as ANFIS, the geometric method and the polynomial method.

The proposed approaches rely on the optimization of the error of position as the objective function as well as to exploit the advantages of each of them to develop a better method.

An analysis and comparison of the three used methods are made for determining the best optimal solution concerning the minimum error with minimum execution time. The analysis shows that the optimal solution is obtained by the polynomial method.

Appendix: Fuzzy neural method

C1 : Fuzzy Neural Method

```
function M=geo_direct_fuzzy(A,T,dx,dy,dphi,n,m,Q)

% A : POSITIONS ANCHOR POINTSA=[x1 y1 ;x2 y2;x3 y3]
% T : TRIANGLE DIMENSIONS: T = [l2 L3 phi(deg)]
% Q : VECTOR JOINT POSITIONSQ=[q1 q2 q3]
% n :size mesh
% m :Number of entriesMF
% dX :Next Search Interval,
% dY :Interval following dy and
% dphi  :Angular range
% M : POSITION [x,y,phi(deg)]
x1=A(1,1);
y1=A(1,2);
x2=A(2,1);
y2=A(2,2);
x3=A(3,1);
y3=A(3,2);
l2=T(1);
l3=T(2);
BETA=T(3)*pi/180;
l1=l2^2+l3^2-2*l3*l2*cos(BETA);
xx=linspace(dx(1),dx(2),n);
yy=linspace(dy(1),dy(2),n);
phi=linspace(dphi(1),dphi(2),n);
[XX,YY,PHI]=meshgrid(xx,yy,phi);
X2=x2*ones(n^3,1);X3=x3*ones(n^3,1);
Y2=y2*ones(n^3,1);Y3=y3*ones(n^3,1);
BET=BETA*ones(n^3,1);
Q1=sqrt(XX(:).^2+YY(:).^2);
Q2=sqrt((X2-XX(:)-l2*cos(BET)).^2+(YY(:)+l2*sin(BET)).^2);
Q3=sqrt((X3-XX(:)-l3*cos(BET+PHI(:))).^2+(Y3-YY(:)-l3*sin(BET+PHI(:))).^2);

%data1 = [Q1 Q2 Q3 XX(:)]; % création  Q1-Q2-Q3-XX  dataset
%data2 = [Q1 Q2 Q3 YY(:)]; % création  Q1-Q2-Q3-YY  dataset
%data3 = [Q1 Q2 Q3 PHI(:)]; % création  Q1-Q2-Q3-PHI dataset

data11 = [Q1(1:2:end,:) Q2(1:2:end,:) Q3(1:2:end,:) (XX(1:2:end))']; % création  Q1-Q2-Q3-XX
dataset   training
data12 = [Q1(2:2:end,:) Q2(2:2:end,:) Q3(2:2:end,:) (XX(2:2:end))']; % création  Q1-Q2-Q3-XX
dataset   checking
data21 = [Q1(1:2:end,:) Q2(1:2:end,:) Q3(1:2:end,:) (YY(1:2:end))']; % création  Q1-Q2-Q3-YY
dataset   training
data22 = [Q1(2:2:end,:) Q2(2:2:end,:) Q3(2:2:end,:) (YY(2:2:end))']; % création  Q1-Q2-Q3-YY
dataset   checking
data31 = [Q1(1:2:end,:) Q2(1:2:end,:) Q3(1:2:end,:) (PHI(1:2:end))']; % création  Q1-Q2-Q3-PHI
dataset  training
data32 = [Q1(2:2:end,:) Q2(2:2:end,:) Q3(2:2:end,:) (PHI(2:2:end))']; % création  Q1-Q2-Q3-PHI
dataset   checking
initfit1= genfis1(data11,m,'gbellmf');
initfit2= genfis1(data21,m,'gbellmf');
initfit3= genfis1(data31,m,'gbellmf');
epochs_n = 30;
[fismat1,trnerr1,ss1,fismat1c,chkerr1]=anfis(data11,initfit1,epochs_n,[0 0 0 0],data12);
[fismat2,trnerr2,ss2,fismat2c,chkerr2]=anfis(data21,initfit2,epochs_n,[0 0 0 0],data22);
tic
```

```
[fismat3,trnerr3,ss3,fismat3c,chkerr3]=anfis(data31,initfit3,epochs_n,[0 0 0 0],data32);
toc
for ii=1:3
fismat1.input(ii).name=['Q',num2str(ii)];
fismat2.input(ii).name=['Q',num2str(ii)];
fismat3.input(ii).name=['Q',num2str(ii)];
end
fismat1.output.name='X';
fismat2.output.name='Y';
fismat3.output.name='PHI';
vect=[1 2;1 3;3 2;1 2;1 3;3 2;1 2;1 3;3 2];
nn=[1,2,3,1,2,3,1,2,3];
mm=[1,1,1,2,2,2,3,3,3];

for ii=1:3
figure(ii)
forjj=1:3
subplot(1,3,jj);
gensurf(eval(['fismat',num2str(ii)]),vect(jj,:),1);
set(ii,'color','w');
set(gca,'box','on','fontSize',10,'Fontweight','bold');
set(get(gca,'Xlabel'),'Fontweight','bold');
set(get(gca,'Ylabel'),'Fontweight','bold');
set(get(gca,'Zlabel'),'Fontweight','bold');
end
end

%*************************************************************************************
%anfis1 = anfis(data1); % train first ANFIS network
%anfis2 = anfis(data2); % train second ANFIS network
%tic
%anfis3 = anfis(data3); % train second ANFIS network
%toc
%for ii=1:3
%anfis1.input(ii).name=['Q',num2str(ii)];
%anfis2.input(ii).name=['Q',num2str(ii)];
%anfis3.input(ii).name=['Q',num2str(ii)];
%end
%anfis1.output.name='X';
%anfis2.output.name='Y';
%anfis3.output.name='PHI';
Nmf=m;
%vect=[1 2;1 3;3 2;1 2;1 3;3 2;1 2;1 3;3 2];
%nn=[1,2,3,1,2,3,1,2,3];
%mm=[1,1,1,2,2,2,3,3,3];
%TRACE FONCTIONS MF
for ii=1:3
figure(ii+3)
forjj=1:3
subplot(3,1,jj);
                    range=eval(['AFIS',num2str(ii),'.input(',num2str(jj),').range']);
xx=linspace(range(1),range(2),1000);
forkk=1:nMF
ifkk==1

par=eval(['AFIS',num2str(ii),'.input(',num2str(jj),').mf(',num2str(kk),').params']);
a=par(1);
                         b=par(2);
                         c =par(3);
yy=1./(1.+(abs(xx-c)/a).^(2*b));
```

```
                          h=plot(xx,yy,'b');
set(h,'linewidth',3)
set(gca,'fontSize',8,'Fontweight','bold','box','on')
hold on
grid on
hx=xlabel('Q1')
else

par=eval(['AFIS',num2str(ii),'.input(',num2str(jj),').mf(',num2str(kk),').params']);
a=par(1);
                          b=par(2);
                          c =par(3);
yy=1./(1.+(abs(xx-c)/a).^(2*b));
subplot(3,1,jj);
                          h=plot(xx,yy,'r');
set(h,'linewidth',3)
set(gca,'fontSize',8,'Fontweight','bold','box','on')
grid on
end
end
hx=xlabel(['Q',num2str(jj)]);
set(hx,'fontweight','bold');
legend(['in',num2str(jj),'mf1'],['in',num2str(jj),'mf2']);
set(hx,'Fontweight','bold');
end
end
set(gcf,'color','w');
% CALCUL SOLUTION POUR ENTREE Q
M(1)=evalfis(Q,fismat1);
M(2)=evalfis(Q,fismat2);
M(3)=evalfis(Q,fismat3);
```

C2 : First structure anfis1 output x: : 2MF

```
[System]
Name='anfis1'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=8
AndMethod='prod'
OrMethod='max'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'
[Input1]
Name='Q1'
Range=[1.13137084989848 1.69705627484771]
NumMFs=2
MF1='in1mf1':'gbellmf',[0.292612235194262 1.9995755549105 1.13752625445892]
MF2='in1mf2':'gbellmf',[0.292110445645078 1.99950851183395 1.69175205218475]

[Input2]
Name='Q2'
Range=[2.8400071559866 3.39977366439042]
NumMFs=2
MF1='in2mf1':'gbellmf',[0.330980087834876 1.99889394150038 2.87969769092566]
MF2='in2mf2':'gbellmf',[0.331524520287308 1.9989584184735 3.35915919161512]
```

```
[Input3]
Name='Q3'
Range=[2.11776435066446 3.39977366439042]
NumMFs=2
MF1='in3mf1':'gbellmf',[0.641153120867244 1.99996615503896 2.11779685971911]
MF2='in3mf2':'gbellmf',[0.641285304920534 1.99997538264018 3.39958722064738]

[Output1]
Name='X'
Range=[0.8 1.2]
NumMFs=8
MF1='out1mf1':'linear',[0.635774434944807 -0.617934672772152 -0.00208082800835871
2.04604909683858]
MF2='out1mf2':'linear',[0.595716185161632 -0.638176644964861 -0.00117402434237554
2.15630570802109]
MF3='out1mf3':'linear',[0.547364391098737 -0.782933381253647 -0.00100703881143561
2.66493974319152]
MF4='out1mf4':'linear',[0.528686085487732 -0.773315616877531 -0.00121126253785542
2.65796428193019]
MF5='out1mf5':'linear',[0.67388688981788 -0.663959232664152 0.000935325731505734
2.10371121751462]
MF6='out1mf6':'linear',[0.653312378057267 -0.634666104254941 0.000867949029006901
2.05169626464253]
MF7='out1mf7':'linear',[0.621754901549701 -0.792611640907116 0.00120847735912372
2.59634849199036]
MF8='out1mf8':'linear',[0.58351370991761 -0.794911255805113 0.00171078643620215
2.6576746038131]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 2 1, 3 (1) : 1
1 2 2, 4 (1) : 1
2 1 1, 5 (1) : 1
2 1 2, 6 (1) : 1
2 2 1, 7 (1) : 1
2 2 2, 8 (1) : 1
```

C3: Second structure anfis 2 output y: : 2MF

```
[System]
Name='anfis2'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=8
AndMethod='prod'
OrMethod='max'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name='Q1'
Range=[1.13137084989848 1.69705627484771]
NumMFs=2
MF1='in1mf1':'gbellmf',[0.299141279460712 1.99954235830746 1.14338406164624]
```

```
MF2='in1mf2':'gbellmf',[0.292328333774036 1.99924215754455 1.69358051089566]

[Input2]
Name='Q2'
Range=[2.8400071559866 3.39977366439042]
NumMFs=2
MF1='in2mf1':'gbellmf',[0.331117225991325 1.99877835857616 2.87883585635876]
MF2='in2mf2':'gbellmf',[0.330126855549728 1.99913128621097 3.35965191973663]

[Input3]
Name='Q3'
Range=[2.11776435066446 3.39977366439042]
NumMFs=2
MF1='in3mf1':'gbellmf',[0.641243102546115 1.99996070383854 2.1178615846668]
MF2='in3mf2':'gbellmf',[0.641337768816867 1.99996143399596 3.39957820193566]

[Output1]
Name='Y'
Range=[0.8 1.2]
NumMFs=8
MF1='out1mf1':'linear',[0.8702729353648 0.885547941916096 0.0052337450561803 -
2.99779317162771]
MF2='out1mf2':'linear',[0.908074927929557 0.870183755325938 0.000890499807789886 -
2.98723762380546]
MF3='out1mf3':'linear',[0.793444920049502 0.63227254619426 -0.000463364516373136 -
2.10523319514498]
MF4='out1mf4':'linear',[0.818032224470001 0.639700133517917 0.0024323671880743 -
2.16538661467184]
MF5='out1mf5':'linear',[0.774382421094582 0.762286851181414 -0.00291999434762494 -
2.46904929378847]
MF6='out1mf6':'linear',[0.810466889696643 0.802629369082266 0.000551996421030527 -
2.65335452842892]
MF7='out1mf7':'linear',[0.750971584405537 0.607230787097182 -0.000756650347913061 -
1.94514787401024]
MF8='out1mf8':'linear',[0.783266842730383 0.625109581081353 -0.00467508670101739 -
2.04905408906011]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 2 1, 3 (1) : 1
1 2 2, 4 (1) : 1
2 1 1, 5 (1) : 1
2 1 2, 6 (1) : 1
2 2 1, 7 (1) : 1
2 2 2, 8 (1) : 1

 [System]
Name='anfis3'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=8
AndMethod='prod'
OrMethod='max'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'
```

```
[Input1]
Name='Q1'
Range=[1.13137084989848 1.69705627484771]
NumMFs=2
MF1='in1mf1':'gbellmf',[0.307027665589648 1.99966652697967 1.15172908699893]
MF2='in1mf2':'gbellmf',[0.28825725346597 1.99896666854045 1.69970514003119]

[Input2]
Name='Q2'
Range=[2.8400071559866 3.39977366439042]
NumMFs=2
MF1='in2mf1':'gbellmf',[0.319728617084349 1.99819314268019 2.86457327191487]
MF2='in2mf2':'gbellmf',[0.326170463720999 1.99959199729879 3.36016885240972]

[Input3]
Name='Q3'
Range=[2.11776435066446 3.39977366439042]
NumMFs=2
MF1='in3mf1':'gbellmf',[0.637655234766923 1.9956685413686 2.10181051698125]
MF2='in3mf2':'gbellmf',[0.669562016896446 2.00049376797142 3.3726715663907]

[Output1]
Name='PHI'
Range=[0.785398163397448 1.5707963267949]
NumMFs=8
MF1='out1mf1':'linear',[1.73698896340592 0.775382399895264 1.58907260379332 -
7.87295168221017]
MF2='out1mf2':'linear',[0.961345405434135 0.306580604215183 1.030390346428 -
4.0078354380659]
MF3='out1mf3':'linear',[1.21084272413243 0.0917389815191804 1.20725250665163 -
4.12334854307864]
MF4='out1mf4':'linear',[0.921097121372049 -0.00202381018336596 0.993206675880491 -
2.85016581939552]
MF5='out1mf5':'linear',[1.25192270970879 0.377703615136204 1.24817438014747 -
5.15541073893423]
MF6='out1mf6':'linear',[0.94869061628918 0.221068323488853 1.11971301260128 -
3.97899648195236]
MF7='out1mf7':'linear',[1.06112155052438 0.0105607692146305 1.13184581753682 -
3.40387406811268]
MF8='out1mf8':'linear',[1.01510984879869 -0.0792142718530147 1.21160088963345 -
3.37066370331802]
[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 2 1, 3 (1) : 1
1 2 2, 4 (1) : 1
2 1 1, 5 (1) : 1
2 1 2, 6 (1) : 1
2 2 1, 7 (1) : 1
2 2 2, 8 (1) : 1
```

C4 : First structure anfis1 output x: :32 MF

```
[System]
Name='fismat1'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
```

```
AndMethod='prod'
OrMethod='max'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name='Q1'
Range=[1.13137084989848 1.69705627484771]
NumMFs=3
MF1='in1mf1':'gbellmf',[0.186976562151165 1.99886517276512 1.16200428813183]
MF2='in1mf2':'gbellmf',[0.195743724560223 2.00046368562511 1.41200042010617]
MF3='in1mf3':'gbellmf',[0.185345468735838 1.99937249038106 1.65832169340373]

[Input2]
Name='Q2'
Range=[2.8400071559866 3.39977366439042]
NumMFs=3
MF1='in2mf1':'gbellmf',[0.194193085703225 1.99715293331261 2.86502532140668]
MF2='in2mf2':'gbellmf',[0.198338033489649 1.99913295192428 3.11877560403719]
MF3='in2mf3':'gbellmf',[0.189897433605816 1.99814006419994 3.36428462029541]

[Input3]
Name='Q3'
Range=[2.11776435066446 3.39977366439042]
NumMFs=3
MF1='in3mf1':'gbellmf',[0.315703588753658 2.00017287795872 2.12271820139806]
MF2='in3mf2':'gbellmf',[0.320100650937797 1.99999431305839 2.76238845304297]
MF3='in3mf3':'gbellmf',[0.31611321455864 2.00015720487923 3.40159687415189]

[Output1]
Name='X'
Range=[0.8 1.2]
NumMFs=27
MF1='out1mf1':'linear',[0.669841615342268 -0.344280812156547 0.099728236063055
1.05479195218452]
MF2='out1mf2':'linear',[0.61374634793602 -0.550843352623667 0.00421583191100562
1.87329095075893]
MF3='out1mf3':'linear',[0.88099754871791 -0.360432464907213 0.00558735250549338
1.07699594570637]
MF4='out1mf4':'linear',[0.47730832836782 -0.103475775491586 -0.0393400215345871
0.711095572624436]
MF5='out1mf5':'linear',[0.523264825365292 -0.657562309884702 -0.00123481075467052
2.29423772998155]
MF6='out1mf6':'linear',[0.370447504533672 -0.292051991460374 -0.00317090720605022
1.31665071898132]
MF7='out1mf7':'linear',[0.703705115305846 -0.524135341740326 0.0461065379368025
1.46396775466058]
MF8='out1mf8':'linear',[0.579656308057527 -0.803713070638916 0.00116781998984474
2.69342760386061]
MF9='out1mf9':'linear',[0.81501793017261 -0.682874130296881 0.00460489975565211
1.99548815236859]
MF10='out1mf10':'linear',[0.882087998090871 -0.673143643249547 -0.0122246090877736
1.86679090298946]
MF11='out1mf11':'linear',[0.66507546487466 -0.626279060491136 -6.83395108171561e-006
2.01614554023163]
MF12='out1mf12':'linear',[1.09048829825821 -0.726754497821782 -0.00248299933105583
1.69176191126254]
MF13='out1mf13':'linear',[0.467347934088891 -0.88863017080748 0.00494478191443154
3.09380121754422]
```

```
MF14='out1mf14':'linear',[0.558343447014696 -0.748033030225354 -0.000191104050728682
2.54197367405859]
MF15='out1mf15':'linear',[0.323506791495486 -0.917613927519138 0.00363387147748432
3.3964700086634]
MF16='out1mf16':'linear',[0.871654563044804 -0.896926508037543 -0.00718608500669454
2.6306280051406]
MF17='out1mf17':'linear',[0.659608472072915 -0.83400485785865 0.000433317624901611
2.6852071991579]
MF18='out1mf18':'linear',[1.02843042474336 -0.87987781379666 -0.00665852734212626
2.35381169671434]
MF19='out1mf19':'linear',[0.838353631146505 -0.511970151671547 0.0053148191296121
1.38864300399458]
MF20='out1mf20':'linear',[0.677536815271381 -0.602146420265932 0.000257954821556303
1.92299370837664]
MF21='out1mf21':'linear',[0.851665318661971 -0.286456934335862 0.0145708515068411
0.689394382027974]
MF22='out1mf22':'linear',[0.576144337547922 -0.456573378709201 -0.00309603077972637
1.63963295170364]
MF23='out1mf23':'linear',[0.636462565696805 -0.678037988296858 -0.000528906976091689
2.21650578859684]
MF24='out1mf24':'linear',[0.531753697225722 -0.175747808051429 -0.0216535630921103
0.899879063641966]
MF25='out1mf25':'linear',[0.802958517810701 -0.705311681581462 0.00593659758935524
1.94568264131331]
MF26='out1mf26':'linear',[0.677134185667875 -0.797493599577687 0.00107343040716776
2.51379153307385]
MF27='out1mf27':'linear',[0.872706225015486 -0.629262049988767 0.0460965449325135
1.41159803509907]
[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1
```

References

1. Elsheikh, A.H., Showaib, E.A., Asar, A.E.M.: Artificial neural network based forward kinematics solution for planar parallel manipulators passing through singular configuration. Adv. Robot. Autom. (2013). https://doi.org/10.4172/2168-9695.1000106
2. Duka, A.V.: Neural network based inverse kinematics solution for trajectory traking of a robotic arm. In: The 7 International conference Interdisciplinarity in Engineering, INTER-ENG 2013, Tirgu-Mures, Romania (2013)
3. Duka, A.V.: ANFIS based Solution to inverse kinematics of a 3 DOF planar manipulator. In: The 8 International conference Interdisciplinarity in Engineering, INTER-ENG 2014, 9–10 October 2014,Tirgu-Mures, Romania (2014)
4. Ali, T.H., Ismail, N., Hamouda, A.M.S., Aris, I., Marhaban, M.H., et al.: Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations. Adv. Eng. Softw. **41**, 359–367 (2010)
5. Pozinor, B.P., Dieulot, J. Dubois, L.: « Introduction à la commande floue » Edition Technique 1998 Paris
6. Gosselin, C.M., Jean, M.: Determination of the workspace of planar parallel manipulators with joint limits. Robot. Auton. Syst. **17**, 129–138 (1996)
7. Cheng, H., Liu, G.F., Yiu, Y.K., Xiong, Z.H., Li, Z.X.: Advantages and dynamics of parallel manipulators with redundant actuation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems Proceedings, HI (2001)
8. Gosselin, C.: Kinematic analyse optimization and programming of parallel robotic manipulators. PhD thesis, McGill University, Montréal, 15 June 1988
9. Gosselin, C., Angeles, J.: The optimum kinematic design of a planar three degree of freedom parallel manipulator. J. Mech. Trans. Autom. **110**(1), 35–41 (1988). https://doi.org/10.1115/1.3258901
10. Gosselin, C.M., Merlet, J.P.: The direct kinematics of planar parallel manipulators: special architectures and number of solutions. Mech. Mach. Theory **29**(8), 1083–1097 (1994). https://doi.org/10.1016/0094-114X(94)90001-9
11. Dehghani, M, Ahmadi, M., Khayatian, A., Eghtesad, M.: Wavelet based neural network solution for forward kinematics problem of HEXA parallel robot. In: INES International Conference on Intelligent Engineering Systems, Iran (2008)
12. Gosselin, C.M., Wang, J.: Singularity loci of planar parallel manipulators with revolute actuators. Robot. Auton. Syst. **21**, 377–398 (1997)
13. Bonev, I.: Geometric analysis of parallel. Ph.D., University Laval, Québec (2002)
14. Sefrioui, J., Gosselin, C.M.: Singularity analysis and representation of planar parallel manipulators. Robot. Auton. Syst. **10**(4), 209–224 (1992). https://doi.org/10.1016/0921-8890(92)90001-F
15. Choi, K.B.: Kinematic analysis and optimal design of 3 PPR planar parallel manipulator. KSME Int. J. **17**(4), 528–537 (2003)
16. Slimane, A., Kebdani, S., Bouchouicha, B., et al.: An interactive method for predicting industrial equipment defects. Int. J. Adv. Manuf. Technol. **95**(9–12), 4341–4351 (2018)
17. Slimane, A., Bouchouicha, B., Benguediab, M., et al.: Contribution to the study of fatigue and rupture of welded structures in carbon steel-a48: ap experimental and numerical study. Trans. Indian Inst. Metals **68**(3), 465–477 (2015)
18. Assal, Samy F.M.: Self organizing approach for learning the forward kinematic multiple solutions of parallel manipulators. Robotica **30**, 951–961 (2012)
19. Tsai, L.-W.: Robot Analysis: The Mechanics of Serial and Parallel Manipulators. Wiley (1999)
20. Wenger, P., Chablat, D.: The kinematic analysis of a symmetrical three degree of freedom planar parallel manipulator. In: CISM IFTOMM Symposium on Robot Design, Dynamics and Control Montreal (2004)
21. Kong, X., Gosselin, C.M.: Forward displacement analysis of third class analytic 3 RPR planar parallel manipulators. Mech. Mach. Theory **36**(9), 1009–1018 (2001). https://doi.org/10.1016/S0094-114X(01)00038-6
22. Yee, C.S., Lim, K.B.: Neural network for the forward kinematics problem in parallel manipulator. In: IEEE International Joint Conference on Neural Networks (1991)