In this paper performance of Puma 560 manipulator is being compared for hybrid gradient descent and least square method learning based ANFIS controller with hybrid Genetic Algorithm and Generalized Pattern Search tuned radial basis function based Neuro-Fuzzy controller. ANFIS which is based on Takagi Sugeno type Fuzzy controller needs prior knowledge of rule base while in radial basis function based Neuro-Fuzzy rule base knowledge is not required. Hybrid Genetic Algorithm with generalized Pattern Search is used for tuning weights of radial basis function based Neuro- fuzzy controller. All the controllers are checked for butterfly trajectory tracking and results in the form of Cartesian and joint space errors are being compared. ANFIS based controller is showing better performance compared to Radial Basis Function based Neuro-Fuzzy Controller but rule base independency of RBF based Neuro-Fuzzy gives it an edge over ANFIS.

**Keywords**—Neuro-Fuzzy, Robotic Control, RBFNF, ANFIS, Hybrid GA.

I. INTRODUCTION

The Integrated Neuro-fuzzy system combines the advantages of ANN and FIS. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be an advantage from the viewpoint of ANN. Integrated Neuro-fuzzy systems share data structures and knowledge representations. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special ANN like architecture. However the conventional ANN learning algorithms (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm.

FALCON [1] uses a five-layered architecture with hybrid-learning algorithm comprising of unsupervised learning to locate initial membership functions/rule base and a gradient descent learning to optimally adjust the parameters of the membership function to produce the desired outputs. GARIC [2] implements a Neuro-Fuzzy controller by using two neural network modules, the ASN (Action Selection Network) and the AEN (Action State Evaluation Network). ANFIS [3] implements a Takagi Sugeno FIS and has a five layered architecture. ANFIS uses hybrid backpropagation and least square method for learning. NEFCON [4] is designed to implement Mamdani type FIS and uses a mixture of reinforcement and backpropagation learning. A Neuro-Fuzzy methodology based on radial basis function and tuned with Genetic Algorithm is implemented in [5].

Robot manipulator faces uncertainties in their dynamics, such as payload mass, friction, and disturbance. Therefore, it is difficult to obtain an accurate model for manipulators. Thus, model based control systems may not be easily implemented in manipulators control. A new hybrid direct/indirect adaptive FNN controller with state observer and supervisory controller for a class of uncertain nonlinear dynamic systems is implemented in [6]. A robust adaptive fuzzy neural controller (AFNC) for identification and control of a class of uncertain multiple-input–multiple-output (MIMO) nonlinear systems is developed in [7]. A robust adaptive fuzzy neural controller (AFNC) suitable for motion control of multilink robot manipulators is implemented in [8]. A fast online structure and parameter learning algorithm, which can add or delete fuzzy control rules or neural network nodes automatically and systematically without predefinition is proposed in [9-10]. In this paper a systematic approach for designing Neuro-Fuzzy controller is developed. Starting with PID controller, Takagi-Sugeno (TS) type Fuzzy PID controller is designed. From TS type Fuzzy PID, ANFIS and Radial basis Function based Neuro-Fuzzy (RBFNF) is designed. In section II, modeling of dynamics and kinematics is discussed. Section III deals details of ANFIS based controller design. Hybrid Genetic Algorithm tuned RBF based Neuro-Fuzzy is presented in Section IV. Section V and Section VI deal with results and conclusion.

II. MODELLING OF MANIPULATOR DYNAMIC AND KINEMATICS

The dynamics of an n-link robotic manipulator is characterized by a set of highly nonlinear and strongly coupled second order differential equation.

$$\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) + F(\theta) = \tau$$

where \(\ddot{\theta}\) is the nxn inertial matrix, \(C(\theta, \dot{\theta})\) is the nx1 vector of centrifugal forces, \(G(\theta)\) is the nx1 vector of gravity loading, \(F(\theta)\) is nx1 vector of friction term, \(\theta, \dot{\theta}\) and \(\ddot{\theta}\) are nx1 vector for
joint angular position, velocity and acceleration, \( \tau \) is nx1 joint torque vector. \( D, C, G, F \) are very complicated function of \( T \) and \( T' \). The dynamic parameters of Puma 560 have been taken from [11]. Puma 560 joint actuators are DC servo motors with armature voltage as control input. The motor is connected to manipulator links through gear where the Robot dynamics appears as dynamic load. The dynamics of DC motor can be represented as (2-5)

\[
E_a = E_b + L \frac{dI}{dt} + RI \tag{2}
\]

\[
E_i = K_e N \Omega \tag{3}
\]

\[
I = E_a - K_e N \Omega / L_s + R \tag{4}
\]

\[
\tau = K_m I \tag{5}
\]

Where \( E_a \) is the armature voltage, \( E_b \) the Back e.m.f, \( L \) and \( R \) are inductance and reactance of armature windings respectively, \( I \) is the armature current, \( N \) is gear ratio, \( K_e \) is the back e.m.f constant, \( K_m \) is motor constant and \( \Omega \) is load angular velocity. Actuator data of puma 560 Robot is taken from [12]. The transformation between the joint space and the Cartesian space of the robot is very important since robots are controlled in the joint space, whereas tasks are defined and object manipulated in the Cartesian space. The kinematics problem deals with the analytical study of the relation between these two spaces. The direct kinematics defined as the transformation from the joint space to the Cartesian space and the inverse kinematics defined as the transformation from the Cartesian space to the joint space. While modeling the kinematics of manipulator, arm singularity and configuration must be checked. Many methods have been proposed for better and feasible solution of manipulator kinematic problems [13-20]. The Forward and Inverse kinematic equations have been modeled [16] and are given in appendix A. Control system diagram of Puma 560 is shown in Fig.1 which consists of desired Cartesian space trajectory T, inverse kinematics block I, PID controller, servo motor M, dynamics D and forward kinematics Block F. In this paper PID controller in Fig.1 is being replaced with Neuro-Fuzzy Controller.

![Fig. 1 Block diagram representation of Puma 560 control](image)

Simulink model of Forward dynamics is shown in Fig.2 and simulink model of complete system is shown in Fig.3.

**III. ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM BASED CONTROLLER**

ANFIS implements a Takagi Sugeno FIS and has a five layered architecture. The first hidden layer is for Fuzzification of the input variables and T-norm operators are deployed in the second hidden layer to compute the rule antecedent part. The third hidden layer normalizes the rule strengths followed by the fourth hidden layer where the consequent parameters of the rules are determined. Output layer computes the overall input as the summation of all incoming signals. ANFIS uses back propagation learning to determine premise parameters (to learn the parameters related to membership functions) and least square estimation to determine the consequent parameters. The first step involved in designing ANFIS based controller is creation of Takagi-Sugeno FIS. A TS type Fuzzy PD+I controller is designed and data is collected from fuzzy controller for training of ANFIS. Fuzzy controller can be designed using parameters of crisp PID controller. Fuzzy PID controller is implemented as Fuzzy PD+I controller. The inputs to fuzzy controllers are error and error change. Matlab simulation diagram of fuzzy PD+I Controller is shown in Fig.4.
Important steps involved in designing fuzzy controller are rule base generation and input output gains setting. If $\theta (k)$ is desired joint angle and $\theta (k)$ is actual output angle at any sampling instant $k$, error $e(k)$, change in error $\dot{e}(k)$ and integral error $ie(k)$ are given as

$$e(k) = \theta_d(k) - \theta(k)$$  

(6)

$$\dot{e}(k) = \frac{e(k) - e(k-1)}{T_s}$$  

(7)

$$ie(k) = \sum_{i=1}^{k} e(i)T_s$$  

(8)

For classical PD controller the controller output is given as

$$u(k) = k_p(e(k) + T_s \dot{e}(k))$$  

(9)

Where $k_p$ is gain of classical PD controller, $T_s$ is derivative time constant and $u(k)$ is control signal.

When actuating signal $u(k)$ is equal to zero

$$k_p(e(k) + T_s \dot{e}(k)) = 0$$  

(10)

$$\dot{e}(k) = -\frac{1}{T_s} e(k)$$  

(11)

From (10) it is clear that $\dot{e}(t)$ directly depends upon $T_s$. If state trajectory of the closed loop controlled system with PD controller for some constant PD value is plotted, it draws a sharp boundary between positive and negative control signals. This can be used to map rule base in discrete state space by taking the diagonal element of rule base as ZE. Rule base for fuzzy controller is given in Table I.

Crisp PID controller parameters are used to initially set fuzzy input output gains. Input error scaling factor is $S_{in}$, error change scaling factor is $S_{ic}$ and output scaling factor is $S_{out}$.

$$u_j = \left( S_{in}e(k) + S_{ic}\dot{e}(k) + S_{out}ie(k) \right) S_{rel}$$

(12)

Comparing (11) with the crisp PID controller output, values of scaling factors come out to be

$$S_{in} = k_p / T_s$$

$$S_{ic} = T_s / e_{max}$$

$$S_{out} = 1 / e_{max} T_s$$

If maximum probable error for any joint is $e_{max}$ and input/output membership function universe is taken as [-1 1], the error scaling factor $S_{in}$ can be set to $1 / e_{max}$. Error change and output scaling factor will be

$$S_{ic} = k_p e_{max} / T_s$$

$$S_{out} = 1 / e_{max} T_s$$

Since better trajectory always starts from the current position of joint angle, initial tracking angle is zero. Taking a worst condition error of 10 radians, value of error scaling factor $S_{in}$ is set to 0.1 and all other initial scaling factors $S_{ic}$, $S_{out}$ are calculated using values of classical PID parameters taken from [20] as given in Table II. For all six joints structure of Fuzzy controller is same as that of Fig.4 but with different scaling factors as given in Table II.

Data set collected from fuzzy controller is used for training ANFIS. Structure of ANFIS used is shown in Fig.5. Membership function after training is shown in Fig.6 and Fig.7. The designed ANFIS is substituted in place of Fuzzy block in Fig.4 keeping gains as it is. The Control system diagram with ANFIS is shown in Fig.8.

### Table I

<table>
<thead>
<tr>
<th>Joint</th>
<th>$S_{in}$</th>
<th>$S_{ic}$</th>
<th>$S_{out}$</th>
<th>$S_{rel}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.009</td>
<td>0.4</td>
<td>2109.375</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.0084</td>
<td>0.4267</td>
<td>2400</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.0084</td>
<td>0.4267</td>
<td>2400</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Joint</th>
<th>$S_{in}$</th>
<th>$S_{ic}$</th>
<th>$S_{out}$</th>
<th>$S_{rel}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.009</td>
<td>0.4</td>
<td>2109.375</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>0.0084</td>
<td>0.4267</td>
<td>2400</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.0084</td>
<td>0.4267</td>
<td>2400</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>0.0056</td>
<td>0.64</td>
<td>5400</td>
</tr>
</tbody>
</table>

Fig. 4 Matlab simulation diagram of Fuzzy PD+I

Fig. 5 structure of ANFIS

Fig. 6 Membership function after training

Fig. 7 The designed ANFIS is substituted in place of Fuzzy block in Fig.4 keeping gains as it is
IV. HYBRID GENETIC ALGORITHM TUNED RADIAL BASIS FUNCTION BASED NEURO FUZZY CONTROLLER

The RBF neural network (RBFNN) is usually used to approximate a continuous linear or nonlinear function mapping. The structure of the two-input and single output RBFNN is shown as in Fig.9. The input layer accepts the system state feedback \((e, \dot{e})\) and the fuzzy inferencing is processed at the hidden layer. The strength of the control action for each of the fuzzy rules is given by the interconnected weights between the hidden and the output layers. The output layer implements the normalization operation to produce the control signals \(u_{cf}\). Basically, fuzzy logic control involves three main stages: Fuzzification, inferencing, and defuzzification. This fuzzy inference mechanism can be further simplified to as only pattern matching and weights averaging, thereby, eliminating the procedures of Fuzzification and defuzzification. The first operation deals with the IF part of the fuzzy control rules; it determines the matching degree of the current input to the condition of each of the fuzzy control rules. By characterizing the fuzzy input membership functions with only two parameters \((C_x, D_x)\), and using the Gaussian membership functions, the matching formula can be written as follows:

\[
h_i = \exp\left(\frac{C_x - x_n}{D_x}\right)
\]

(12)

For \(i = 1\) to \(T\).

Here \(T\) is the total number of fuzzy rules \(C_{x,n}^i\) and \(D_{x,n}^i\) denotes the center and the width of \(n^{th}\) input variable’s membership assigned to the \(i^{th}\) control rules, respectively. While \(\parallel \cdot \parallel\) is the norm operator presented as Euclidean distance. The matching degree process is simply an operation that returns the matching level between the inputs and the rule pattern for the \(i^{th}\) rule. A matching degree of ‘1’ means that a full match occurs to that rule, while a small \(h_i\) indicates poor matching between the input pattern and the particular rule pattern. The weights are then averaged to obtain the control action of each output variable. Thus controller output \(u_{cf}\) can be computed by normalizing the weights

\[
u_{cf} = \frac{\sum_{i=1}^{T} (h_i w_i)}{\sum_{i=1}^{T} (h_i)}
\]

(14)

From ANFIS block diagram shown in Fig.8 only fuzzy block is replaced with Radial basis function based Neuro- Fuzzy block keeping all the gains same. For each of 49 rules a hidden layer neuron is taken. Each of the node consists of two Gaussian functions with there centre and width as that of error and error change in antecedent part of rule base to implement (6) as shown in Fig.10.
Final control output $u_{nf}$ is calculated using (7) as shown in Fig.11. For 49 hidden layer neurons there are 49 weights between hidden layer and output. Seven membership Gaussian functions for error and error change give 28 variables as each of them have there mean and variance. So total of 77 variables comes in action. GA was used initially to search the optimum search space using all these 77 variables. Neuro-Fuzzy block is attached with first joint of fuzzy controller and goal of GA was to minimize integral absolute error IAE between fuzzy output and Neuro-Fuzzy output. Fitness curve of GA shown in Fig.12. After tuning process of GA, control using Neuro-Fuzzy started and GPS was used to search mean and variance (28 variables keeping weight same as that returned by GA) to minimize ITSE of system. After 30 iterations of GPS, variance is kept same and now only mean is varied using GPS. Tuning curve of GPS is shown in Fig.13 and Fig.14. Tuned membership error and error change membership function are shown in Fig.15 and Fig.16.
V. RESULTS

Results of both types of Neuro-Fuzzy controllers are tested in terms of ITSE in Cartesian space and ISE in joint space. The desired Cartesian space trajectories taken for testing controller is butterfly. Parametric equation of Butterfly trajectory is

\[
x(t) = 200 \cos(5200 \cdot t)(e^{\cos(40t)} - 2\cos(4t) - \sin^2(\pi / 12)) + 350
\]

\[
y(t) = 200 \sin(5200 \cdot t)(e^{\cos(40t)} - 2\cos(4t) - \sin^2(\pi / 12)) + 200
\]

For checking the robustness of controller a disturbance torque \( D \) is applied

\[D = 1.5 \sin(4.3575t) + \sin(9.825t) + \sin(2.7075) + 1\]

The sampling time of system is 1ms. Fig.17 and Fig.18 show desired and actual output butterfly trajectory with RBFNF based controller with and without disturbance. Cartesian space error \( dx, dy, dz \) for tracking butterfly trajectory with RBFNF based controller with and without disturbance are shown in Fig.19.a and Fig.19.b and corresponding joint space ITSE and Cartesian space Integral square errors; \( ISE_x, ISE_y, ISE_z \) are shown in Table III and Table IV. Fig.20 to Fig.22.b shows trajectory and error for ANFIS based controller. From Table III and Table IV of joint space ITSE and Cartesian space ISE, it is clear that performance of ANFIS in joint space is better than RBFNF.

### TABLE III

<table>
<thead>
<tr>
<th>JOINT SPACE ITSE USING NEURO FUZZY CONTROLLER</th>
<th>Butterfly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without D</td>
</tr>
<tr>
<td>ANFIS</td>
<td>19.6763</td>
</tr>
<tr>
<td>RBF-NF</td>
<td>42.7020</td>
</tr>
</tbody>
</table>

### TABLE IV

<table>
<thead>
<tr>
<th>CARTESIAN SPACE ISE USING NEURO FUZZY CONTROLLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ISE_x ) (mm)</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>ANFIS</td>
</tr>
<tr>
<td>RBF-NF</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Fig. 17 Butterfly trajectory with RBFNF without disturbance

Fig. 18 Butterfly trajectory with RBFNF with disturbance
Fig. 19.a Cartesian space error for butterfly trajectory tracking using RBFNF without disturbance

Fig. 19.b Cartesian space error for butterfly trajectory tracking using RBFNF with disturbance

Fig. 22.a Cartesian space error for butterfly trajectory tracking using ANFIS without disturbance

Fig. 22.b Cartesian space error for butterfly trajectory tracking using ANFIS with disturbance

Fig. 20 Butterfly trajectory with ANFIS without disturbance

Fig. 21 Butterfly trajectory with ANFIS with disturbance
VI. CONCLUSIONS

ANFIS and Hybrid Genetic Algorithm, Generalized Pattern Search tuned RBFN is implemented for Puma 560 manipulator control. Both of controllers are being compared for butterfly trajectory tracking in Cartesian space. Performances in terms of joint space ITSE and Cartesian space ISE is being compared. The proposed RBFN methodology in [5] is modified here by using hybrid GA and Generalized Pattern Search technique and successfully implemented for robust manipulator control applications. GA tuned RBFN is not very effective because of increase in dimensionality of search space with increase in number of antecedent in Fuzzy rule base. ANFIS is found to be slightly better than RBF-NF. But the designing of RBF-NF without rule base gives it an edge over ANFIS.

REFERENCES


Sufian Ashraf Mazhari was born in India and received his Bachelor degree (Electrical Engineering) from Aligarh Muslim University, Aligarh India, in 2005 and M.Tech degree in Electrical Engineering from Indian Institute of Technology, Roorkee India.Currently is working with GS E & C ,Gurgaon ,India. His area of research includes Robotic Control, Optimization and Computer vision system.

Surendra Kumar (M’07 ) received B.E (Electrical), M.(E)System Engineering and Operations Research) and Ph.D. Electrical in 1969,1971 and 1982 respectively in India. He joined the Department of Electrical Engineering, Indian Institute of Technology Roorkee, India as lecturer in 1972 .He has 36 year of teaching and research experience. Presently he is Assistant Professor .He has been on teaching assignment to University of Technology, Baghdad, IRAQ during 1987-1989.He is member IEEE, Fellow of Institution of Engineers India and member of ISTE India. His area of research interest is mainly Control, Optimization, AI application to Robotic Control and Fuzzy Reliability.

APPENDIX.

Abbreviation used:

\( c_i = \cos(\theta_i), s_i = \sin(\theta_i), c_{ij} = \cos(\theta_i + \theta_j) \)
\( s_{ij} = \sin(\theta_i + \theta_j), s_{i-j} = \sin(\theta_i - \theta_j) \)

The arm configuration parameters of Puma 560
\( k_1, k_2 \) and \( k_3 \) are defined as

\[ k_1 = \begin{cases} +1, & \text{lefty} \smallskip \end{cases} \]
\[ k_2 = \begin{cases} +1, & \text{elbow up} \smallskip \end{cases} \]
\[ k_3 = \begin{cases} -1, & \text{elbow down} \smallskip \end{cases} \]
The parameters are \( k_1, k_2 \) and \( k_3 \) used to find inverse kinematics solution. However in the case of a known set of joint angles, as in the case of the direct kinematics, these parameters can be computed.

**Forward kinematics:** The problem is defined as given the joint angles vector, find the Cartesian position/orientation vector \( R \), and the arm configuration parameters \( k_1, k_2, k_3 \).

The orientation angles \( r_\rho, r_\varphi \) and \( r_\varphi \) are defined as:

\[
\begin{align*}
\cos(r_\rho) &= c_{23} c_{5} - s_{23} s_{5} c_{3} \\
r_\rho &= \theta_1 + a \tan \left[ 2 \left( s_{23} s_{4} s_{c} + s_{23} c_{3} c_{5} \right) \right] \\
r_\varphi &= \theta_2 + a \tan \left[ 2 \left( s_{5} s_{4} c_{3} + s_{23} s_{5} c_{4} \right) \right]
\end{align*}
\]

Where \( \tan^{-1} \) is the four-quadrant version of \( \tan \) as \( \tan^{-1} (x, y) \) is four-quadrant version of \( \tan \).

The accuracy of equations deteriorates because \( \cos(\theta_2) \approx 1 \) is a singular point. If \( \sin(\theta_2) \approx 0 \), \( r_\varphi \) is set to zero or \( \pi \) depending upon sign of \( \cos(\theta_2) \). Value of \( r_\rho \) is set to zero and \( r_\varphi \) is calculated using:

\[
r_\rho = \theta_1 + a \tan \left[ 2 \left( s_{23} s_{4} c_{5} + c_{23} \right) \right]
\]

Position vector \( R \) is defined as:

\[
\begin{align*}
x &= -w_{3} s_{3} - d_{c1} - l_{4} s_{4} c_{r_\rho} \\
y &= w_{3} c_{3} - d_{s1} + l_{4} s_{4} c_{r_\varphi} \\
z &= w_{4} + l_{4} c_{r_\rho}
\end{align*}
\]

The arm configuration is determined by evaluating parameters \( k_1, k_2, k_3 \). If \( w_3 \geq 0 \) then the arm is lefty and \( k_1 = +1 \), but if \( w_3 < 0 \), then the arm is righty and \( k_1 = -1 \). If \( k_2 \theta_2 \geq 0 \) then the arm is elbow up and \( k_2 = +1 \) else \( k_2 = -1 \). If \( \theta_2 \geq 0 \) then a no-flip solution exists and \( k_3 = +1 \) but if \( \theta_2 < 0 \) then a flip solution exist and \( k_3 = -1 \).

**Inverse kinematics:** The problem is defined as given Cartesian position/orientation vector \( R \), and the arm configuration parameters \( k_1, k_2, k_3 \) find the joint angles vector.

Joint angles \( \theta_1, \theta_2, ... \) are given as:

\[
\theta_1 = a \tan \left[ 2 \left( -k_1 w_{11}, k_1 w_{12}, k_1 \right) - k_0 \right] \tan (d/l)
\]

Where:

\[
w_{1i} = (r_\rho + l_{4} s_{4} c_{r_{\rho}}) j + (r_\varphi - l_{4} s_{4} c_{r_{\varphi}}) j + (r_\varphi - l_{4} c_{r_{\rho}}) k
\]

A singular point exists if \( w_{1i} = 0 \). However considering the arm geometry this condition is never satisfied:

\[
\cos(\theta_i') = \frac{n^2 - l^2_4 - l^2_3}{2 l_3 l_4}
\]

The parameters are \( k_1, k_2 \) and \( k_3 \) are used to find inverse kinematics solution. However in the case of a known set of joint angles, as in the case of the direct kinematics, these parameters can be computed.

Forward kinematics: The problem is defined as given the joint angles vector, find the Cartesian position/orientation vector \( R \), and the arm configuration parameters \( k_1, k_2, k_3 \).

The orientation angles \( r_\rho, r_\varphi \) and \( r_\varphi \) are defined as:

\[
\begin{align*}
\cos(r_\rho) &= c_{23} c_{5} - s_{23} s_{5} c_{3} \\
r_\rho &= \theta_1 + a \tan \left[ 2 \left( s_{23} s_{4} s_{c} + s_{23} c_{3} c_{5} \right) \right] \\
r_\varphi &= \theta_2 + a \tan \left[ 2 \left( s_{5} s_{4} c_{3} + s_{23} s_{5} c_{4} \right) \right]
\end{align*}
\]

Where \( \tan^{-1} \) is the four-quadrant version of \( \tan \) as \( \tan^{-1} (x, y) \) is four-quadrant version of \( \tan \).

The accuracy of equations deteriorates because \( \cos(\theta_2) \approx 1 \) is a singular point. If \( \sin(\theta_2) \approx 0 \), \( r_\varphi \) is set to zero or \( \pi \) depending upon sign of \( \cos(\theta_2) \). Value of \( r_\rho \) is set to zero and \( r_\varphi \) is calculated using:

\[
r_\rho = \theta_1 + a \tan \left[ 2 \left( s_{23} s_{4} c_{5} + c_{23} \right) \right]
\]

Position vector \( R \) is defined as:

\[
\begin{align*}
x &= -w_{3} s_{3} - d_{c1} - l_{4} s_{4} c_{r_\rho} \\
y &= w_{3} c_{3} - d_{s1} + l_{4} s_{4} c_{r_\varphi} \\
z &= w_{4} + l_{4} c_{r_\rho}
\end{align*}
\]

The arm configuration is determined by evaluating parameters \( k_1, k_2, k_3 \). If \( w_3 \geq 0 \) then the arm is lefty and \( k_1 = +1 \), but if \( w_3 < 0 \), then the arm is righty and \( k_1 = -1 \). If \( k_2 \theta_2 \geq 0 \) then the arm is elbow up and \( k_2 = +1 \) else \( k_2 = -1 \). If \( \theta_2 \geq 0 \) then a no-flip solution exists and \( k_3 = +1 \) but if \( \theta_2 < 0 \) then a flip solution exist and \( k_3 = -1 \).

**Inverse kinematics:** The problem is defined as given Cartesian position/orientation vector \( R \), and the arm configuration parameters \( k_1, k_2, k_3 \) find the joint angles vector.

Joint angles \( \theta_1, \theta_2, ... \) are given as:

\[
\theta_1 = a \tan \left[ 2 \left( -k_1 w_{11}, k_1 w_{12}, k_1 \right) - k_0 \right] \tan (d/l)
\]

Where:

\[
w_{1i} = (r_\rho + l_{4} s_{4} c_{r_{\rho}}) j + (r_\varphi - l_{4} s_{4} c_{r_{\varphi}}) j + (r_\varphi - l_{4} c_{r_{\rho}}) k
\]

Singular point exists if \( w_{1i} = 0 \). However considering the arm geometry this condition is never satisfied:

\[
\cos(\theta_i') = \frac{n^2 - l^2_4 - l^2_3}{2 l_3 l_4}
\]