ISSN: 3067-2384
Research Article

World Journal of Sensors Network Research

Algorithm of Intelligent Positional Contour Control of Objects Moving Along a Specified Trajectory

Jasur Sevinov*, Dildora Sevinova, Orifjon Zaripov & Husan Igamberdiev

Department of Information Processing and Management Systems, Tashkent State Technical University, Tashkent, 100095, Uzbekistan

*Corresponding author: Jasur Sevinov, Department of Information Processing and Management Systems, Tashkent State Technical Univer-sity, Tashkent, 100095, Uzbekistan.

Submitted: 14 July 2025 **Accepted:** 30 July 2025 **Published:** 20 August 2025

https://doi.org/10.63620/MKWJSNR.2025.

Citation: *Sevinov, J., Sevinova, D., Zaripov, O., & Igamberdiev, H (2025). Department of Information Processing and Man-agement Systems, Tashkent State Technical University, Tashkent, 100095, Uzbekistan. Wor Jour of Sens Net Res, 2(4), 01-10.*

Abstract
This article is devoted to the algorithm of intelligent positional-contour control of objects moving along a defined trajectory. In the process of positional-contour control of a multi-link industrial robot, the use of intelligent tech-nology methods for planning a sequence of actions according to the defined coordinate trajectory and position of the manipulator handle on the example of moving objects is described. According to the tactical level of management, the problems of solving the right and inverse problems of robot kinematics and ensuring the flexibility of the manipulator handle based on the dynamic properties of the object were considered. In addition, a synthesis based on a mathe-matical model, as well as an intelligent control structure and algorithm based on neural network technology, are proposed to ensure stability in the movement of the manipulator handle along the programmed trajectory in the control of a multi-link industrial robot. The proposed algorithm allows positional-contour control systems to store the sequence of movements at initial positions along a specified coordinate trajectory of an industrial robot in a database and determine the optimal trajectory.

Keywords: Moving Object, Multi-Link Robot, Adaptive-Positional Contour Control, Intelligent Algorithm, Neural Network Model.

Introduction

Intelligent robotic systems and robotic technologies are widely used in modern manufacturing enterprises. Their efficiency directly depends on the correct selection of control criteria [1]. In recent years, much research has been conducted in this area. However, these studies are aimed at calculating the values of the trajectory at discrete time positions along a given coordinate trajectory and controlling each link of the industrial robot separately [2, 3]. The motion control system of a multi-link industrial robot based on a fixed position is built depending on its adaptability to external influences and depending on its dynamic characteristics [4]. The motion control system of a multi-link industrial robot based on the set position is developed based on flexibility and dynamic characteristics to external influences [1, 5]. Also, the control systems of multi-link industrial robots are divided into two types according to the defined coordinate trajectory and position: positional and contour control systems. In position control systems, the values of the coordinate trajectory of the robot links executed at the previous position are stored in the control system's database, which ensures the sequence of actions [5, 6]. In contour control systems, the trajectory along a given coordinate is evaluated based on the value of . Often, in adaptive position control, when a program trajectory is given, the control system has the ability to operate both positional and contour systems. Ensuring the stability of the executive body moving along the program trajectory during the control process and synthesizing control algorithms are one of the urgent tasks [7]. To solve these tasks, it is important to develop intelligent control systems for multi-link industrial robot manipulators [1, 3, 8]. This control method requires forming the program trajectory of the multi-link industrial robot, performing kinematic synthesis based on the dynamic characteristics of the manipulator handle, planning the sequence of actions, and creating intelligent control algorithms.

Research Methodology

Usually, the solution of the inverse kinematic problem for multi-link manipulators is reduced to solving a system of nonlinear equations, that is, if the kinematic equation is given in the form $x = f(\theta)$, we can express it by the following relation:

$$F(q) = S \quad (1)$$

where q – is the generalized coordinate vector defining the configuration of the manipulator; $S = [S_1, \ldots, S_6]^T$ – is the vector defining the position and orientation of the manipulator handle in space.

As a rule, the solution to this problem is not unique, that is, each state of the gripper device corresponds to a certain set of configurations $G = \{G_k\}$, $q_j \in Q$, [1, 9]. Then, the sequence of states $\{S(k)\}$, $k = 1, \ldots, M$ of the gripper device at the workplace corresponds to the sequence of the set of permissible configurations $\{G_k\}$ $k = 1, \ldots, M$ of the manipulator as follows:

$$G_k = \{q_j(k) : S(k) = F(q_j(k))\}, \quad j \in g_k ,$$

where g_k – is the set of indices for the th allowed configurations at the th position of the trajectory.

$$K = \sum_{k=1}^{R=M-s} r_k[q(k)\ q(k+1)\ \ldots, q(k+s)] \to \min, \quad (2)$$

where $q(k) \in G_k,\ k = 1, \ldots, M;\ 1 \le s \le M-1$.

For the case $s = 1$, the objective function K represents the sum of losses r_k in the sections between two consecutive cases and is defined as follows:

$$K = \sum_{k=1}^{k=M-s} r_{jm}(k) \quad (3)$$

where $j \in g_k,\ m \in g_{k+1};\ r_k[q_j(k)\ \ q_m(k+1)],\ k = 1, \ldots, M$. Also, in the case of minimizing the total time, in expression (3), the following value can be obtained as a distance measure $r_{jm}(k)$:

$$r_{jm}(k) = \max_{i \in 1, \ldots, N} \left| q_{ij}(k) - q_{in}(k+1) \right| / v_{i\max},$$

where $v_{i\max}$ represents the maximum value of the velocity of the $i-$ th link of the manipulator moving along the adaptive position trajectory [10].

The node configurations corresponding to the minimum value of the objective function K are determined by the dynamic programming method based on the recurrent (returning) relations based on the criterion for creating a program trajectory as follows:

$$K[q(k)\ \ldots, q(k+s-1)] =$$
$$\min_{q(k+s-1) \in G_{k+s}} \{r_k[q(k)\ \ldots, q(k+s)] + K_{k+1}[q(k+1)\ \ldots, q(k+s)]\},$$

where $k = M-s,\ M-s-1, \ldots, 1$.

The problem of constructing an optimal basis for a program trajectory according to the distance measurement criterion in equation (3) is reduced to the problem of finding the shortest path in a certain sense among the allowed configurations, which path starts from the handle position , since the initial positions are usually given exactly [1, 11].

Denoting the minimum path length from q_l (1) to $q_j(k)$ by $K(k, j)$, we can write the connection based on the boundary condition as follows:

$$K(k+1, m) = \min_{j \in g_k} [r_{jm}(k) + K(k, j)], \quad k = 1, \ldots, M-1. \quad (4)$$

As a result of solving equation (4), the values of the function $K(k+1, m)$ are obtained (where $m \in g_k$, $k = 1, \ldots, M-1$) and the array $\{u(k+1)\}$ is formed [1, 12]. The elements of this array represent the numbers of nodes that belong to the shortest path to point $(k+1, m)$. The shortest path length $K(M)$ and the index of the last node corresponding to it are determined based on the following condition:

$$K(M) = K(M, j(M)) = \min_{j \in g_M} K(M, j)$$

Solving the problem of building the optimal basis, which is the initial stage of building the program trajectory, is usually carried out at the final stage of planning manipulator trajectories, which consists of constructing and choosing the optimal path in the plan graph describing the structure of the working space of the manipulator handle [1, 13].

Software trajectory construction. The task of software trajectory construction is to construct the law of change in time of the generalized coordinate vector $q = [q_1, \ldots, q_N]^T$ (i.e., the vector function $q(t) = q_1(t)\ \ldots, q_N(t)),\ \ t \in [t_0, t_T]$, which ensures the movement of the manipulator handle [14]. Several methods can be used to solve this problem, one of which is the method of constructing the program trajectory through a discrete approximation, called the basis of the program trajectory, represented by a sequence of positions $q^1, q^2, \ldots, q^M$ in a generalized coordinate space. A discrete approximation of the program trajectory can be obtained by solving the inverse kinematic problem, for example, using the method of feasible directions [15].

$$\begin{cases} t_{k+1} t_k + \max_{1 \le i \le N} \left(\dfrac{\left| q_i^{k+1} - q_i^k \right|}{v_{i\max}} \right), & k = 0, 1, \ldots, M \\ t_M = t_T \end{cases} \quad (5)$$

where $v_{i\max} = q_{i\max}$, the maximum allowed tracking speed of the $i-$ th link.

The following expression is true for the control interval:

$$T = t_T - t_0 = \sum_{k=1}^{M} |\Delta t_k| \quad (6)$$

where $\Delta t_k = [t_{k-1}, t_k] -$ represents the time taken to perform the movements from position q^{k-1} to position q^k of the programmed trajectory.

The problem of interpolation of the programmed trajectory can be solved on the basis of approximation using splines.

In the spline mode of order $S^m(q,t)$, the following function, consisting of a polynomial of degree , exists on each of the intersection segments Δt_k :

$$S^m(q,t) = P_{m,k}(t) = \sum_{i=0}^{m} a_k t^j, \quad t_{k-1} \le t \le t_k, \qquad (7)$$

this in turn must satisfy the following conditions — namely, the continuity of the function and the continuity of its derivatives up to order $(m-1)$ at the points of division t_k, $k = 1,\ldots,M$:

$$P_{m,k}^{(i)}(t_k) = P_{m,k+1}^{(i)}(t_k) \quad k = 1,\ldots,M-1; i = 0,\ldots,m-1 . \ (8)$$

In the spline mode, expression (7) in the form of a polynomial of degree m contains uncertainties $M(m+a_k)$ and the continuity relations (8) form a system of equations $(M-1)m$ [1, 16]. It is possible to construct a system of missing equations by adding additional conditions to the trajectory generation properties of the program. The simplest linear spline function that satisfies the continuity conditions (8) can be of the form:

$$S^1(q,t) = P_{1,k}(t) = \sum_{j=0}^{1} a_k t^j$$

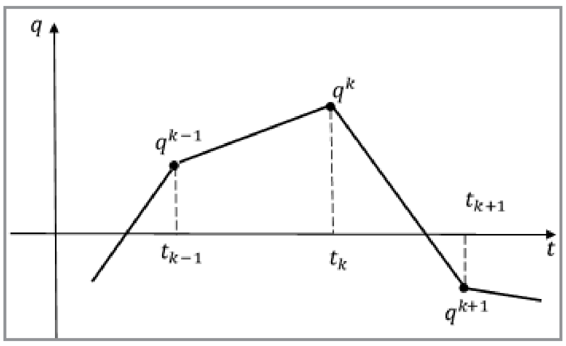
In this case, the total number of unknowns is $2M$, and the continuity conditions are $-(M-1)$. If the spline function values are required to correspond to $q(t)$ at M points $t_k, k = 0,\ldots,M$ then it becomes possible to fill in these $(m+1)$ missing conditions:

In this case, the first-order spline can be expressed as follows (Fig. 1, a):

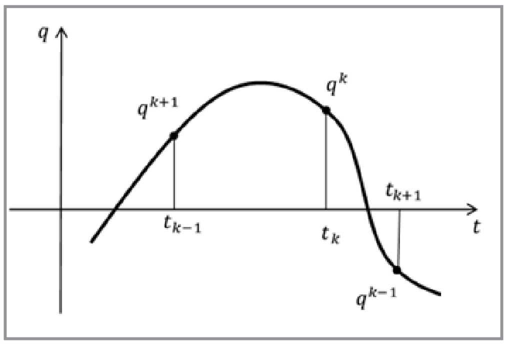
$$P_1(t_0) = q^0, P_{1k}(t_k) = q^k, \quad k = 1,\ldots,M. \qquad (9)$$

where

$$a_{0k} = q^{k-1} - a_{1k}t_{k-1}; \quad a_{1k} = \frac{q^k - q^{k-1}}{t_k + t_{k-1}}.$$



(a)



(b)

Figure 1: First and third order spline functions. a) first-order spline; b) third-order spline.

Given a variational problem aimed at finding a continuous function and its derivative $q(t)$ passing through the given points $q(t_k) = q^k$, $k = 0,1, ..., M$, and by corresponding functional minimization, we can determine a program trajectory of the same type:

$$I_1 = \int_0^T (\dot{q}(t))^2 d . \qquad (10)$$

Since the Euler equation for the program trajectory determination function given in equation (10) has the form $\ddot{q}(t) = 0$, the function $q(t)$ is linear in each section [1, 3, 15].

The implementation of the structured program trajectory does not provide the required control quality, so it is recommended to use splines of the third order (Fig. 1, b).

Therefore, the problem of determining the control signal can be expressed as a variational problem of finding a continuous first derivative function $q(t)$ that satisfies the conditions $q(t_k) = q^k$, $k = 0,1,\ldots,M$ and minimizes the mean square acceleration functional along the trajectory as follows:

$$I_2 = \int_0^T (\ddot{q}(t))^2 d . \qquad (11)$$

For equation (11), which minimizes the root-mean-square functional of the accelerations along the trajectory, the Euler equation has the form $q^I(t) = 0$, and the program trajectory is determined by the following expression:

$$q(t) = P_{3k}(t) = a_{0k} + a_{1k} + a_{2k}t^2 + a_{3k}t^3, \quad t_{k-1} \le t \le t_k \qquad (12)$$

In this case, the number of uncertain parameters a_{1k} is equal to $4M$, and the number of conditions is $(3M-1) : -2M$ is continuous and generates the approximation conditions:

$$\left.\begin{aligned} P_{3k}(t_k) &= P_{3,k+1}(t_k) \quad k=1,\dots,M-1; \\ P_{13}(t_0) &= q^0, P_{3k}(t_k) = q^k, \quad k=1,\dots,M; \end{aligned}\right\} \qquad (13)$$

The continuity condition of the first derivative is as follows:

$$\dot{P}_{3k}(t_k) = \dot{P}_{3,k+1}(t_k) \quad k=1,\dots,M-1 \qquad (14)$$

(14) the continuity conditions of accelerations in the expression are determined as follows:

$$\ddot{P}_{3k}(t_k) = \dot{P}_{3,k+1}(t_k) \quad k=1,\dots,M-1 \qquad (15)$$

The necessary boundary conditions for the movement of the manipulator in the specified position are derived from the necessary condition for the functional minimum in expression (11). In the absence of additional restrictions on positions along the trajectory, the boundary conditions will be as follows:

$$\ddot{P}_{13}(t_0) = \dot{P}_{3M}(t_M) = 0 \qquad (16)$$

The condition of steady state at the initial and final points of the trajectory, as a condition chosen based on practical considerations, imposes the following boundary conditions on the respective velocities:

$$\dot{P}(t_0) = \dot{P}_{3M}(t_M) = 0 \qquad (17)$$

However, the boundary conditions in equations (16) and (17) are not required to be satisfied simultaneously only when using third-order splines [1, 16]. To solve the problem with constraints of the form (17) on the velocities of the manipulator motion, fifth-order polynomials can be used as approximations of the initial and final positions of the trajectory to compensate for the non-zero accelerations that arise:

$$\left.\begin{aligned} P_{51}(t) &= P_{13}(t) + c_1(t-t_0)^2(t_1-t_0)^3(t_1-t_0)^3; \\ P_{5M}(t) &= P_{M3}(t) + c_M(t_M-t)^2(t-t_{M-1})^3(t_M-t_{M-1})^3. \end{aligned}\right\} \qquad (18)$$

where is $c_1 = -\frac{\ddot{P}_{13}(t_0)}{2}; c_M = -\frac{\ddot{P}_{3M}(t_M)}{2}$.
The functions in the form (18) ensure that the boundary conditions (16) are fulfilled without violating the approximation conditions (13), the continuity condition of the first derivative, (15) the acceleration conditions.

The parameters of the spline functions $P_{3k}(t) \quad k=1,\dots,M$ are determined from the boundary conditions imposed on the corresponding velocities (13), (14), (15) and (17) [1, 17].

The third-order spline functions $P_{3k}(t) \quad k=1,\dots,M$ can be expressed on the interval as follows:

$$P_{3k}(t), k = D_{k-1}\frac{(t_k-t)^2}{6h_k} + D_k\frac{(t-t_{k-1})^3}{6h_k} + \left(q^{k-1} - D_{k-1}\frac{h_k^2}{6}\right)\frac{t_k-t}{h_k} + \left(q^k - D_k\frac{h_k^2}{6}\right)\frac{t-t_{k-1}}{h_k} \qquad (19)$$

where is $h_k = t_k - t_{k-1}, \ k=1,\dots,M$.

It can be seen that the spline (19) satisfies the constraints (13) and (15). For any values of D_k, its second derivatives vary continuously and piecewise linearly:

$$\frac{h_k}{6}D_{k-1} + \frac{h_k+h_{k-1}}{3}D_k, \quad t \in [t_{k-1}, t_k]$$

The continuity conditions of the first derivative based on expression (14) allow us to obtain the following equations:

$$\ddot{P}_{3k}(t) = D_{k-1}\frac{t_k-t}{h_k} + D_k\frac{t-t_{k-1}}{h_k}, \quad t \in [t_{k-1}, t_k]$$

The continuity conditions of the first derivative based on expression (14) allow us to obtain the following equations:

The boundary conditions for the velocities in equation (17) allow us to obtain the following equations for determining the quantities $D_0,\dots,D_M$:

$$D_0\frac{h_1}{3} + D_1\frac{h_1}{6} = \frac{q_1-q_0}{h_1}, \qquad D_{M-1}\frac{h_M}{6} + D_M\frac{h_M}{3} = \frac{q_{M-1}-q_M}{h_M}.$$

In this case, the parameters $D_k = \ddot{q}(t_k) \quad k=1,\dots,M$ can be determined from the following equation:

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \cdots & 0 & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \mu_{M-1} & 2 & \lambda_{M-1} \\ 0 & 0 & 0 & \cdots & 0 & \mu_M & 2 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdots \\ D_{M-1} \\ D_M \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \cdots \\ d_{M-1} \\ d_M \end{bmatrix} \qquad (20)$$

where is $\lambda_0 = \mu_M = 1$.

$$d_0 = \frac{6(q^1-q^0)}{h_1^2}; d_m = \frac{6(q^{M-1}-q^M)}{h_M^2}; \quad \mu_k = \frac{h_k}{h_k+h_{k+1}};$$

$$\lambda_k = 1 - \mu_k\lambda_k;$$

$$d_k = 6\left[\frac{q^{k+1}-q^k}{h_{k+1}} - \frac{(q^k-q^{k-1})}{h_k}\right](h_{k+1}+h_k) \quad k=1,\dots,M-1.$$

Once the values of d_k are determined, the spline $S^3(q,t) = P_{3k}(t)$ can be calculated [2, 18] according to expression (19).

After calculating the spline functions of the signals controlling the actuator for each movement link, the control duration interval and the correctness of its division into Δt_k sections are checked. To do this, the maximum speed and acceleration of each function in each section are determined. If the speed or acceleration of the function in some time section exceeds the permissible value, the duration of this section is increased accordingly. Then the procedure for calculating the spline function is repeated for a new control interval.

Stabilization of program trajectories is one of the methods for eliminating the shortcomings of controlling the manipulator according to the program trajectory $q_p(t)$ values. This is control based on the principle of solving inverse kinematics and is expressed as follows:

$$\dot{v} = D_1\dot{u} + D_2u + D_3M_{NN}(q,\dot{q},\xi).$$

(21)

Using the inverse problem of the manipulator kinematics in terms of $q, \dot{q}, u$, we determine the control law of the actuator mechanisms in the following form:

$$\dot{v} = D_1\dot{u} + D_2\{A'(q,\xi')[\ddot{q}_p + \Gamma_1(\dot{q}) - \dot{q}_p + \Gamma_0(q - q_p) + B'(q,\dot{q},\xi)]\} + D_3M_{NN}(q,\dot{q},\xi),$$

(22)

where $q_p -$ is the desired/program trajectory; $\Gamma_0 = diag(\gamma_0, \dots, \gamma_{N0})$ $\Gamma_1 = diag(\gamma_1, \dots, \gamma_{N1})$ are $N \times N$, diagonal matrices; $\gamma_{il} < 0$, $i = 1,..,N$, $l = 0,1$; ξ' – is the value of the vector ξ, which represents the state of the manipulator and the mechanism in motion, and $\xi' = \xi$ is a complete information form.

The problem of stabilizing the program trajectory is expressed as a system of equations that includes the dynamics of the manipulator and the dynamic equations of the actuators that set it in motion:

$$\left.\begin{aligned} &A'(q,\xi)\ddot{q} + B'(q,\dot{q},\xi) = u; \\ &\dot{v} = D_0\ddot{u} + D_1\dot{u} + D_2u + D_3M_H(q,\dot{q},\xi), \end{aligned}\right\}$$

(23)

where

$$A'(q,\xi) = A(q,\xi) + J'_{execution\ \ engine};$$

$$J'_{execution\ \ engine} = diag(i^2_{derivatives1}J_{execution\ \ engine1}, \dots, i^2_{derivativesN}\ J_{execution\ \ engineN})$$

$$B'(q,\dot{q},\xi) = B'(q,\dot{q},\xi) - M_r - M_\Pi;$$

$M_r, M_\Pi -$ moments related to friction forces and external driving moments acting on the manipulator actuator;

$u = M_\Pi - N -$ dimensional vector of moments produced by the actuator mech-anisms in the robot links; $v - N -$ di-mensional vector of control signals; $M_N\ (q,\dot{q},\xi) -$ is an $N -$ is an dimensional vector of loads in the execution mech-anisms of robot links; $D_i = diag(d_{1j}, \dots, d_N) - N \times N$ dimensional diagonal matrices related to the parameters of the actuator mechanisms in the robot links.

Analysis and Results

Typically, transitions in the actuators of robot links occur much faster than transitions in the manipulator. According to the system of equations (21), e

suring the asymptotic stability of the program trajectory $q_p(t)$ $t \in [t_0, t_T]$ for the manipulator dynamics is achieved by synthesizing a control algorithm. The complexity of such tasks is associated with the analysis of the current situation, processing, generalization and synthesis of large volumes of sensor data. This process requires making appropriate management decisions, i.e., the task of minimizing computational resources and, most importantly, time consumption is of great importance [1, 13, 19]. In particular, it is very important to apply the tactical control level of neural network technologies based on a self-learning model, which provides high speed and error correction in the control of a multi-link industrial robot manipulator by parallel processing of input signals. In this case, the difference between the current x_{goal}, y_{goal}

and target x_{goal}, y_{goal} and target positions of the multi-link industrial robot manipulator is minimized. In this case, deviations from the function approximated to the initial and final points of the trajectory at a certain distance are taken into account (Fig. 2). At the same time, the presented scheme is based on the principles of tactical control level, and it involves solving direct and inverse problems related to the kinematics of the manipulator, as well as direct and inverse problems related to the dynamics of a multi-link manipulator. The minimization method used here is based on calculating the mismatch in the generalized coordinates of the manipulator and the initial and final positions of the trajectory and is expressed as follows:

$$\nabla q_i = \frac{\partial H(q_1, q_2, \dots, q_n)}{\partial q_i},$$

(24)

$$H(q_1, q_2, \dots, q_n) = \frac{(x_ц + x_T)^2 + (y_m - y_T)^2}{2}.$$

(25)

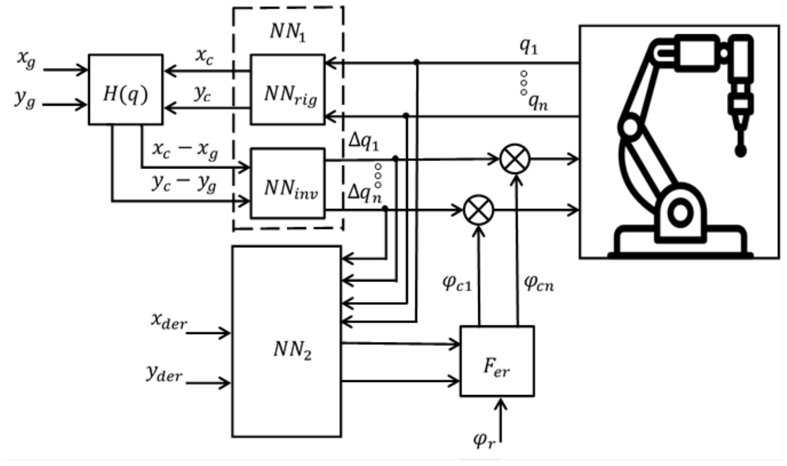


Figure 2: Generalized structure of multi-link industrial robot manipulator control by neural network model based on self-learning model.

To determine the vector of a multi-link industrial robot at the specified coordinates, it is necessary to calculate the current spatial position of the manipulator handle parameters by solving the correct kinematics problem for each moving link as follows:

$$x_{current} = F_x(q_1, q_2, \dots, q_n),$$
$$y_{current} = F_y(q_1, q_2, \dots, q_n), \quad (26)$$

where F_x, F_y, are the connection functions between the generalized and Cartesian coordinates of the multi-link industrial robot manipulator [1, 20]. This process (N_1) is performed by a neural network structure, which is based on the method of solving the inverse problem of kinematics by changing the weight coefficients of the links and the parameters of the neurons, and is convenient even for manipulators with complex kinematic schemes. The obtained values of the manipulator coordinates make it possible to form the function of errors (25). The determination of the magnitude of the change in the generalized coordinate vector for the movement of the industrial robot manipulator handle along the target direction vector according to equation (24) is carried out using the same N_{right} and $N_{reverse}$ neural network structure as the determination in the inverse kinematics mode (N_1). This proposed approach required special research to select the optimal method of systematic organization of the neural network designed to implement state changes of the manipulator lever according to the trajectory and position of the specified coordinates. We propose an approach for this based on the Jordan recurrent network (Figure 3). This approach is used to detect changes in each link in the control of a multi-link robot manipulator [1 ,2, 21]. It consists of a network with a recurrence and an output layer from the hidden layer, representing inverse kinematics in the hidden layer. Also, due to the memory capacity of the control system and the additional layer of nonlinear activation functions, the efficiency of approximation and prediction is increased. The Jordan network performs the tasks of recognizing time series, separating them into coordinates and positions, and classifying them into classes, calculating the values of the trajectory $q(t_k)$ at discrete time points $t_0, t_1, \dots, t_r$ along a given coordinate trajectory, and controlling each link of an industrial robot separately [1, 2, 22]. In addition, there are feedback loops formed through the context layer between the outputs of this network and additional inputs, which allow us to ensure the asymptotic stability of the program trajectory in the form $q_p(t)$ $t \in [t_0, t_T]$ in the dynamics of the manipulator and synthesize a control algorithm. The following Jordan network is described by the following system of recurrent equations:

$$v_j(n+1) = \sum_{i=1}^{p} w_{ji}^{(1)} u_i(n) + \sum_{i=1}^{N} w_{ji}^{c}(y_i(n) + \alpha y_i(n-1)) + b_j^{(1)}, \quad (27)$$

$$x_j(n+1) = F_1(v_j(n+1)), \quad j = 1,2,\dots,N, \quad (28)$$

$$y_j(n+1) = F_2\left(\sum_{i=1}^{N} w_{ji}^{(2)} x_i(n+1) + b_j^{(2)}\right), \quad j = 1,2,\dots,M, \quad (29)$$

,
or the equations can be expressed in matrix form as follows:

$$X(n+1) = F_1(W^{(1)}U(n) + W^c(Y(n) - \alpha Y(n-1)) + B^{(1)}), \quad (30)$$

$$Y(n+1) = F_2(W^{(2)}X(n+1) + B^{(2)}), \quad (31)$$

where $U(n)-$ is the vector of external input signals at time n in the control process; $p-$ is the number of external input channels of the network; $X(n)-$ output signals of the hidden layer at $(n+1)$ instants; $N-$ signal discretization step in the context layer; $W^{(1)}, W^c, W^{(2)}-$ external input signals, matrices of synaptic weights of the context layer and output layer; $B^{(1)}, B^{(2)}-$ vectors of displacement weights of neurons in the hidden and output layers; F_1, F_2- vectors of activation functions in the hidden and output layers; $Y(n+1)-$ vector of output signals of the network at $(n+1)$ instants; $M-$ output channels of the network.

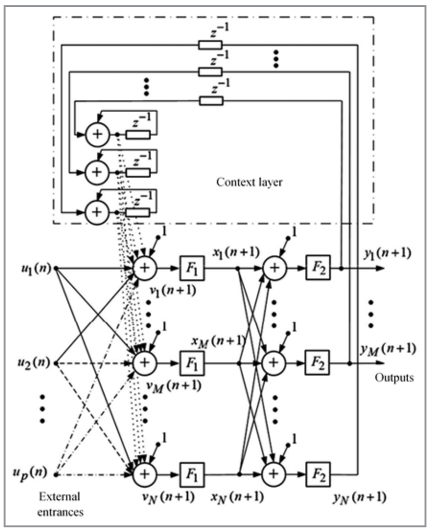


Figure 3: The structure of the Jordan network, which forms the basis of the self-learning model.

Based on these mathematical models and the model in the robot manipulator control structure described in Figure 2, seven equally spaced sample points in the range of link angles $[\pi/2, 3\pi/2]$ were selected for training the network. For three multi-link manipulators, this makes a total of $7^3 = 343$ combinations or patterns. A larger, higher resolution training set was also prepared. It selected 13 equally spaced points in the above interval and formed a "typical" sample set of $\text{ß}^3 = 2197$ elements. In the following sections, these two sets are referred to as the "small" and "normal" sets, respectively. The network's interpolation performance and test results show that the network should be able to interpolate correctly not only on the sample set used during training, but also on points in between. Therefore, it was tested on the test set according to the system of formation equations (26) for training based on the Jordan recurrent neural network based on the self-learning model described in Figure 3. In this, 1000 randomly selected samples were generated in the interval $[\pi/2, 3\pi/2]$, and the generalization interpolation property of the network was selected for testing (Table 1). All neurons have a logical activation function applied, so their output values are only in the range [0, 1]. Therefore, all input and output data are scaled to this interval [1, 2, 23].

Table 1: Results of Jordan Recurrent Neural Network Structure Training Based on Self-learning Model

Number of neurons in intermediate layers	After 150,000 cycles		After 250,000 cycles		After 450,000 cycles	
	10 neurons	12 neurons	10 neurons	12 neurons	10 neurons	12 neurons
Training						
Small	0.01266	0.01461	0.01090	0.01185	0.01059	0.01032
Regular	0.01309	0.01213	0.00903	0.00819	0.00905	0.007548
Testing						
Small	0.01242	0.01313	0.01070	0.01049	0.01047	0.00907

To calculate the vector of a multi-link industrial robot along the specified coordinates, the Jordan recurrent neural network was tuned according to the training equation for each movement link, using the current parameters of the current spatial position $x_{current}, y_{current}$ of the manipulator handle, using the following general evaluation function:

$$H_\Sigma = \sum_{j=1}^{J} H_j,$$

where $J -$ is the total number of examples in the intermediate layers; $j -$ is the index indicating the number of examples; H_j is the estimate of the amount of inconsistency of the $j -$th example, calculated according to expression (25).

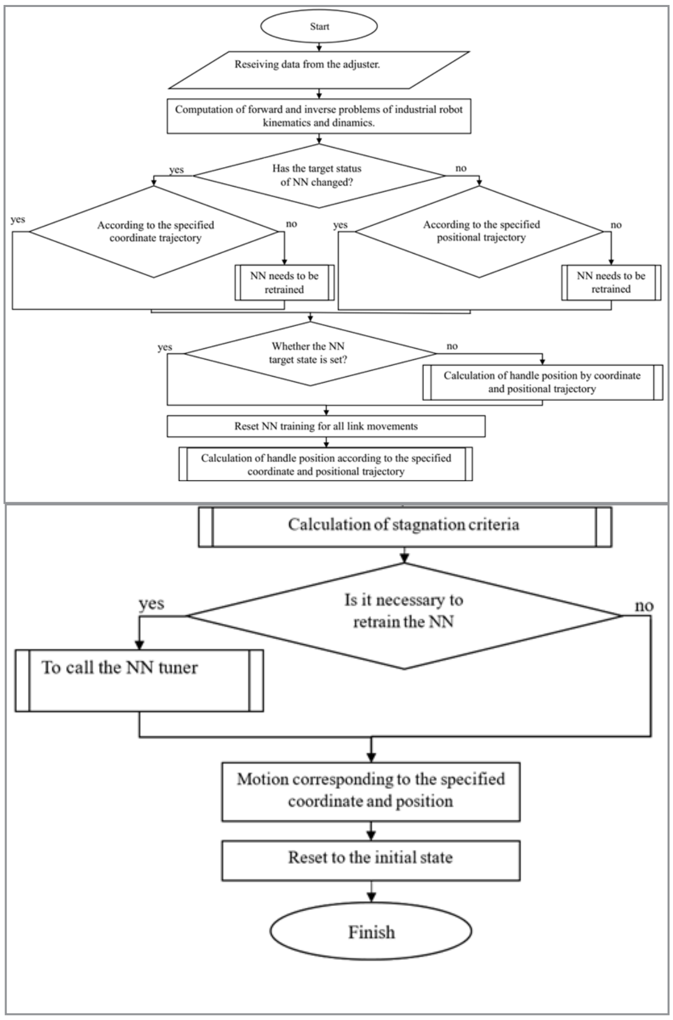
Analysis of the data in Table 1 shows that the nature of the neural network training process largely depends on its structure [1, 2, 17, 23]. Therefore, the state changes along the trajectory and position of the manipulator handle along the specified coordinates are determined by the adjustment of the synaptic connections between the layers of the neural network and their total number, organized in separate layers:

$$M = \sum_{i=0I}^{I} m_i m_{i+1},$$

where $M -$ is the total number of synaptic connections; $I -$ is the number of intermediate layers; $i -$ is the number of layers (links) corresponding to the links of a multi-link industrial robot;

m_0- is the number of inputs; m_i- is the number of neurons in the $i-1$ layer corresponding to the links of a multi-link industrial robot; m_I- is the number of outputs.

In order to eliminate the shortcomings of controlling the manipulator according to the program trajectory m_I- values in the control algorithm of a multi-link industrial robot manipulator through a neural network system based on a self-learning model, the introduction of an additional intermediate layer while maintaining the total number of neurons allows achieving high accuracy with the appropriate duration at the end of the network training process (Fig. 4) [24, 25]. Using the previous training example, such a neural network-based controller allowed us to carefully reduce the value of the total error correction $H_{\sum}$ to 0.0414757, which is sufficient to ensure the accuracy of the position of the manipulator handle [26].



Start
Reseiving data from the adjuster.
Computation of forward and inverse problems of industrial robot kinematics and dinamics.
Has the target status of NN changed?
yes
no
According to the specified coordinate trajectory
yes
no
NN needs to be retrained
According to the specified positional trajectory
yes
no
NN needs to be retrained
Whether the NN target state is set?
yes
no
Calculation of handle position by coordinate and positional trajectory
Reset NN training for all link movements
Calculation of handle position according to the specified coordinate and positional trajectory
Calculation of stagnation criteria
Is it necessary to retrain the NN
yes
no
To call the NN tuner
Motion corresponding to the specified coordinate and position
Reset to the initial state
Finish

Figure 4: Control Algorithm of Multi-Link Industrial Robot Manipulator by Neural Network System Based on Self-Learning Model.

The formation of an error function that allows the multi-link industrial robot manipulator to prevent collisions with external objects during the targeted movement of the handle is performed by another N_2 neural network structure. In this case, to unify the base of links, the layer-by-layer organization of the neural network structure N_2 is chosen to be the same as H_1. To simplify the training process, (x_{der}, y_{der}) is taken as an input as a measure of the distance between the obstacle according to the specified coordinate and position. In this case, the value of the allowed approximation angle φ_{der} corresponding to the obstacle of the manipulator along each link of the generalized coordinates of the robot kinematic chain $\varphi_{rigt1}, \varphi_{rigt2}, \ldots, \varphi_{rigt\,n}$ is given as the basis to the output neuron.

The advantages of this algorithm are that it is possible to adjust the control parameters and ensure stability even when the learning rate is higher than the permissible limit in NN [27- 30].

Conclusion

The article deals with the development of an intelligent position-contour control algorithm for objects moving along a defined trajectory. In the case of moving objects, in the process of positional-contour control of a multi-link industrial robot, it is proposed to use intelligent technology methods in order to plan the sequence of actions according to the specified coordinate trajectory and position of the manipulator handle, to ensure the accuracy of the stop positions. In the case of moving objects, in the process of position-contour control of a multi-link industrial robot, it is proposed to use intelligent technologies in order to plan a sequence of actions corresponding to the specified coordinate trajectory and position of the manipulator handle and to ensure accuracy in stopping positions. Depending on the decision-making criteria, the methods for solving the correct and inverse problems related to the kinematics of a multi-link industrial robot in accordance with the tactical level of control and ensuring the adaptability of the manipulator grip taking into account the dynamic characteristics of the object are considered. Also, a mathematical model aimed at ensuring the stability of the manipulator handle moving along the programmed trajectory, as well as an intelligent control structure and algorithm based on the neural network technology, which implements kinematic synthesis based on the dynamic characteristics of the handle, have been developed. The developed algorithm serves to determine the optimal trajectory by storing the sequence of actions in the initial positions along the specified coordinate trajectory of the industrial robot in the positional-contour control systems directly into the database, processing the input commands.

References

1. Makarov, I. M., Rahmankulov, V. Z., Nazaretov, V. M., & others. (1986). Robototehnika i gibkie avtomatizirovannye proizvodstva. Tom 3 iz 9. Upravlenie robototehnicheskimi sistemami i gibkimi avtomatizirovannymi proizvodstvami [Robotics and flexible automated manufacturing. Volume 3 of 9. Control of robotic systems and flexible automated manufacturing]. Moscow: Vysshaja shkola. (In Russian)
2. Yurevich, E. I. (2007). Intellektual'nye roboty [Intelligent robots]. Moscow: Mashinostroenie. (In Russian)
3. Lewis, F. L., Jagannathan, S., & Yesildirek, A. (1999). Neural network control of robot manipulators and nonlinear systems. London: Taylor & Francis Ltd.
4. Ioannou, P. A., & Sun, J. (1996). Robust adaptive control. New Jersey: Prentice Hall.
5. Ivanov, V. A., & Yushhenko, A. S. (1983). Teoriya diskretnyh sistem avtomaticheskogo upravleniya [Theory of discrete automatic control systems]. Moscow: Nauka. (In Russian)
6. Kobrinskiy, A. A., & Kobrinskiy, A. E. (1988). Manipuljacionnye sistemy robotov [Robotic manipulation systems]. Moscow: Nauka. (In Russian)
7. Kozlov, Yu. M. (1990). Adaptaciya i obuchenie v robototehnike [Adaptation and learning in robotics]. Moscow: Nauka. (In Russian)
8. Lazerev, V. G., Piyl, E. I., & Turuga, E. N. (1984). Postroenie programmiruemyh upravlennih ustroystv [Construction of programmable control devices]. Moscow: Energoatomizdat. (In Russian)
9. Glazunov, V. A. (2018). Novye mehanizmy v sovremennoj robototehnike [New mechanisms in modern robotics]. Moscow: TEHANOSFERA. (In Russian)
10. Zenkivich, S. L., & Jushhenko, A. S. (2000). Upravlenija robotami. Osnovy upravlenija manipuljacionnymi robotami [Robot control. Basics of manipulating robots control]. Moscow: MGTU im. N.E. Baumana. (In Russian)
11. Vukobratovich, M., & Stokich, D. (1989). Upravlenie manipuljacionnymi robotami [Control of manipulation robots]. Moscow: Nauka. (In Russian)
12. Gavrish, A. P., & Jampol'skiy, L. S. (1989). Gibkie robototehnicheskie sistemy [Flexible robotic systems]. Kiev: Vysshaja shkola. (In Russian)
13. Galiullin, A. S. (1986). Metody resheniya obratnyh zadach dinamiki [Methods for solving inverse problems of dynamics]. Moscow: Nauka. (In Russian)
14. Makarov, I. M. (2001). Intellektualnye sistemy avtomaticheskogo upravleniya [Intelligent automatic control systems]. Moscow: FIZMATLIT. (In Russian)
15. Yusupbekov, N. R., & others. (2015). Intellektualnie sistemi upravleniya i prinyatie resheniy [Intelligent management and decision-making systems]. Tashkent: Izdatelstvo Natsionalnoy ensiklopedii Uzbekistana. (In Russian)
16. Makarov, I. M., Lohin, V. M., & others. (2006). Iskusstvenniy intellekt i intellektualniy sistemy upravleniya [Artificial intelligence and intelligent control systems]. Moscow: Nauka. (In Russian)
17. Nazarov, H. N. (2019). Intellektual'nye mnogokoordinatnye mehatronnye moduli robototehnicheskih sistem [Intelligent multi-coordinate mechatronic modules of robotic systems]. Tashkent: Mashhur-Press. (In Russian)
18. Korendjasev, A. I., Salamadra, B. L., & Tysves, L. I. (2006). Teoreticheskie osnovy robototehniki [Theoretical foundations of robotics]. Moscow. (In Russian)
19. Zaripov, O. O., Sevinova, D. U., & Bobojanov, S. G. (2024). Adaptive position-determination and dynamic model properties synthesis of moving objects with trajectory control system (in the case of multi-link manipulators). Transactions of the Korean Institute of Electrical Engineers, 73(3), 576–584. https://doi.org/10.5370/KIEE.2024.73.3.576
20. Zaripov, O. O., & Sevinova, D. U. (2023). Structural and kinematic synthesis algorithms of adaptive position-trajectory control systems (in the case of assembly industrial robots). In ICoRSE 2023, LNNS 762 (pp. 1–16). Springer. https://doi.org/10.1007/978-3-031-40628-7_50
21. Matyokubov, N. R., & Rakhimov, T. O. (2024). Group control of functional linear actuation elements of mechatronic modules. Transactions of the Korean Institute of Electrical Engineers, 73(6), 995–1004. https://doi.org/10.5370/KIEE.2024.73.6.995
22. Sevinov, J. U., Boborayimov, O. K., & Bobomurodov, N. H. (2024). Algorithms for synthesis of adaptive neural network control systems based on the velocity gradient method. In R. A. Aliev, J. Kacprzyk, W. Pedrycz, M. Jamshidi, M. Babanli, & F. M. Sadikoglu (Eds.), 16th International Conference on Applications of Fuzzy Systems, Soft Computing and Artificial Intelligence Tools – ICAFS-2023 (Vol. 1141, pp. 377–387). Springer. https://doi.org/10.1007/978-3-031-76283-3_34
23. Li, Q., Schurmann, C., Haschke, R., & Ritter, H. (2013). A control framework for tactile servoing. In Robotics: Science and Systems. Princeton, NJ: Citeseer.
24. Yusupbekov, A. N., Sevinov, J. U., Mamirov, U. F., & Botirov, T. V. (2021). Synthesis algorithms for neural network regulator of dynamic system control. In R. A. Aliev, J. Kacprzyk, W. Pedrycz, M. Jamshidi, M. Babanli, & F. M. Sadikoglu (Eds.), 14th International Conference on Theory and Application of Fuzzy Systems and Soft Computing – ICAFS-2020 (Vol. 1306, pp. 973–982). Springer. https://doi.org/10.1007/978-3-030-64058-3_90
25. Matyokubov, N. R., & Rakhimov, T. O. (2022). Structural-mode graphs of electromagnetic and mechatronic modules of intelligent robots. In 12th World Conference “Intelligent System for Industrial Automation” (WCIS). Springer.

https://doi.org/10.1007/978-3-031-53488-1_30
26. Macfarlane, S., & Croft, E. (2003). Jerk-bounded manipulator trajectory planning design for real-time applications. IEEE Transactions on Robotics and Automation, 19(1), 42–52.
27. Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2010). Robotics: Modelling, planning and control. Springer-Verlag London Limited.
28. Lewis, F. L., Liu, K., & Yesildirek, A. (1993). Neural net robot controller with guaranteed tracking performance. In Proceedings of the International Symposium on Intelligent Control (pp. 225–231). Chicago, IL.
29. Vukobratovich, M., Stokich, D., & Kirchiski, N. (1989). Neadaptivnoe i adaptivnoe upravlenie manipuljacionnymi robotami [Non-adaptive and adaptive control of manipulation robots]. Moscow: Mir. (In Russian)
30. Zaripov, O. O., Sevinova, D. U., & Bobojanov, S. G. (2024). Adaptive position-determination and dynamic model properties synthesis of moving objects with trajectory control system (in the case of multi-link manipulators). Transactions of the Korean Institute of Electrical Engineers, 73(3), 576–584. https://doi.org/10.5370/KIEE.2024.73.3.576