ISSN 2075-8545 (online)

Nanotechnologies in construction
Нанотехнологии в строительстве

2026; 18 (5):
681–690

Nanobuild.ru

SYSTEM SOLUTIONS FOR TECHNOLOGICAL PROBLEMS

Original article
https://doi.org/10.15828/2075-8545-2026-18-5-681-690

CC BY 4.0

Inverse kinematics algorithm for trajectories with specified tool orientation

Petr A. Kulakov*, **Vitaly G. Afanasenko**
Ufa State Petroleum Technological University, Ufa, Kosmonavtov str. 1, 450064, Russian Federation
* Corresponding author: e-mail: kulakov.p.a@mail.ru

ABSTRACT

Introduction. During welding, coating application, and assembly or installation operations, the tool's position is specified in conjunction with its orientation. A single specified pose may correspond to various sets of manipulator joint angles, and the solutions obtained for individual points do not always form a continuous trajectory. In such cases, the transition from one point to another involves a sudden change in joint angles, a factor that must be taken into account when setting movement speeds. The trajectory is defined either by joint angles or by an array of Cartesian tool poses. In the latter case, the input data consists of coordinates and orientations derived from a CAD model, teach points, or an external planner. Calculating the trajectory from this data requires reconciling inverse kinematics solutions at adjacent points; in the algorithm under consideration, this is achieved by passing the calculated angles to the computation for the next point. **Methods and Materials.** The object of study is a six-link manipulator with six degrees of freedom. Joint angles were determined based on specified Tool Center Point (TCP) positions and orientations using the pseudoinverse of the numerical Jacobian matrix, while the orientation error component was calculated using ZYX Euler angles. The solution obtained for a given point served as the initial approximation for the subsequent point. Verification was performed using trajectories comprising varying numbers of points. **Results and Discussion.** The procedure for formulating the six-dimensional error, the Jacobian calculation, and the iteration termination criteria are presented. Position and orientation errors, iteration counts, and computation times were determined for trajectories consisting of 5, 10, and 20 points; trajectories with up to 50 points were also examined. Calculation results for an identical geometric trajectory showed average iteration counts per point of 28, 18, and 12, respectively. Utilizing the previous solution reduced the number of iterations and, in the tests conducted, enabled the maintenance of the selected solution branch. **Conclusion.** The developed algorithm enables the derivation of a sequence of manipulator joint angles from a set of tool positions and orientations. In the calculations performed, the transfer of the solution between adjacent points ensured smooth angular variation; however, this mechanism alone does not guarantee the avoidance of a solution branch switch for every possible trajectory. The algorithm's applicability is constrained by the convergence conditions of the numerical computation. When approaching a singular configuration or using sparse point spacing, it is necessary to additionally check the condition number of the Jacobian. Enhancing computational stability may require finer trajectory discretization and regularized pseudo-inversion.

KEYWORDS: six-link manipulator, inverse kinematics, trajectory tracking, Euler angles, numerical Jacobian, positioning error

SOURCES OF FUNDING FOR THE SCIENTIFIC WORK THAT RESULTED IN THE PUBLICATION: This work was conducted as part of the Strategic Project "New technologies, technical, and intelligent systems in mechanical engineering" of the Ufa State Petroleum Technological University Development Program for 2021–2030 in accordance with the Priority 2030 Strategic Academic Leadership Program.

ACKNOWLEDGEMENTS: The authors are grateful for the financial support of this project.

FOR CITATION:
Kulakov P.A., Afanasenko V.G. Inverse kinematics algorithm for trajectories with specified tool orientation. *Nanotechnologies in Construction.* 2026;18(5):681–690. https://doi.org/10.15828/2075-8545-2026-18-5-681-690. – EDN: BDBXER.

© Kulakov P.A., Afanasenko V.G., 2026

ISSN 2075-8545 (online)

Nanotechnologies in construction
Нанотехнологии в строительстве

2026; 18 (5):
681–690

Nanobuild.ru

SYSTEM SOLUTIONS FOR TECHNOLOGICAL PROBLEMS

Алгоритм обратной кинематики для траекторий с заданной ориентацией инструмента

Петр Алексеевич Кулаков*, Виталий Геннадьевич Афанасенко

Уфимский государственный нефтяной технический университет, 450064, Уфа, улица Космонавтов, 1, Российская Федерация

* Автор, ответственный за переписку: e-mail: kulakov.p.a@mail.ru

АННОТАЦИЯ

Введение. При сварке, нанесении покрытий и выполнении сборочных и монтажных операций положение инструмента задается совместно с его ориентацией. Одной заданной позе могут соответствовать различные наборы углов в суставах манипулятора, причем решения, полученные для отдельных точек, не всегда образуют непрерывную траекторию. В таком случае переход от одной точки к другой сопровождается скачком углов, что необходимо учитывать при задании скоростей движения. Траекторию задают углами в суставах либо массивом декартовых поз инструмента. Во втором случае исходными данными служат координаты и ориентации, полученные по CAD-модели, точкам обучения или от внешнего планировщика. Для расчета по этим данным требуется согласовать решения обратной кинематики в соседних точках; в рассматриваемом алгоритме это выполняется за счет передачи найденных углов в расчет следующей точки. **Материалы и методы.** Объектом исследования является шестизвенный манипулятор с шестью степенями свободы. По заданным положениям и ориентациям TCP определялись углы в суставах с использованием псевдообратной матрицы численного якобиана, а ориентационная часть ошибки рассчитывалась по углам Эйлера ZYX. Начальным приближением для каждой последующей точки служило решение, полученное на предыдущей. Проверка проведена для траекторий с различным числом точек. **Результаты и обсуждение.** Приведены порядок формирования шестимерной ошибки, расчет якобиана и условия завершения итераций. Для траекторий из 5, 10 и 20 точек определены ошибки положения и ориентации, число итераций и время расчета; дополнительно рассмотрены траектории до 50 точек. По данным расчетов для одинаковой геометрической траектории среднее число итераций на точку составило 28, 18 и 12 соответственно. Использование предыдущего решения уменьшало число итераций, а в рассмотренных тестах позволяло сохранять выбранную ветвь решения. **Выводы.** Разработанный алгоритм позволяет по массиву положений и ориентаций инструмента получить последовательность углов в суставах манипулятора. В проведенных расчетах передача решения между соседними точками обеспечивала плавное изменение углов, однако сама по себе не исключает смену ветви решения при любой траектории. Область применения алгоритма ограничивается условиями сходимости численного расчета. При приближении к сингулярной конфигурации или при редком задании точек необходима дополнительная проверка числа обусловленности якобиана. Для повышения устойчивости расчета могут потребоваться более плотная дискретизация траектории и регуляризованное псевдообращение.

КЛЮЧЕВЫЕ СЛОВА: шестизвенный манипулятор, обратная кинематика, движение по траектории, углы Эйлера, численный якобиан, ошибка позиционирования

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ НАУЧНОЙ РАБОТЫ, РЕЗУЛЬТАТОМ КОТОРОЙ СТАЛА ПУБЛИКАЦИЯ: Работа проводилась в рамках реализации Стратегического проекта «Новые технологии, технические и интеллектуальные системы в машиностроении» программы развития ФГБОУ ВО «Уфимский государственный нефтяной технический университет» на 2021–2030 годы в соответствии с программой стратегического академического лидерства «Приоритет 2030».

БЛАГОДАРНОСТИ: Авторы выражают благодарность за финансовую поддержку проекта.

ДЛЯ ЦИТИРОВАНИЯ:

Кулаков П.А., Афанасенко В.Г. Алгоритм обратной кинематики для траекторий с заданной ориентацией инструмента. *Нанотехнологии в строительстве.* 2026;18(5):681–690. https://doi.org/10.15828/2075-8545-2026-18-5-681-690. – EDN: BDBXER.

INTRODUCTION

Controlling a manipulator along a trajectory requires specifying the tool's position and orientation at each point. This approach is used in seam welding, coating application, assembly, and gripper positioning. For a single target pose, inverse kinematics yields one or more sets of joint angles. During continuous motion, it is necessary to select solutions that avoid configuration jumps and exceedance of permissible joint speeds [2–5].

Trajectory problems are solved in either joint space or cartesian space [2–4]. In the former case, joint angles are specified as functions of time, while the TCP position and orientation are determined via forward kinematics. In the latter case, a sequence of target poses is defined, and the inverse kinematics problem is solved for each pose.

ISSN 2075-8545 (online)

Nanotechnologies in construction
Нанотехнологии в строительстве

2026; 18 (5):
681–690

Nanobuild.ru

This approach is advantageous when the trajectory is derived from a CAD model, teach points, or data from an external planner. However, using an arbitrary initial guess can cause the algorithm to switch to a different solution branch. This results in discontinuities in the joint angle sequence, complicating time-based interpolation [3, 4, 11, 12]. In this study, the solution obtained for the previous point is used for the next point; with a sufficiently small step size, this technique preserves the selected solution branch [2–4].

For a given pose, inverse kinematics can be calculated using the Jacobian pseudoinverse, the Jacobian transpose, or the regularized least-squares method [5–9, 16]. Positional error is defined by the difference in Cartesian coordinates. Euler angles, axis-angle representation, quaternions, or a rotation vector can be used for orientation [7–10]. The algorithm's convergence and behavior near singular configurations depend on the chosen representation. In the algorithm under consideration, orientation error is defined using ZYX Euler angles, and the numerical Jacobian is calculated via the finite difference method.

The influence of the sampling step on the continuity of joint coordinates is discussed in [3, 4, 11–15]. When poses are specified at sparse intervals, the solution for a given point may not be sufficiently close to the solution for the subsequent point, causing the difference $q_{i+1} - q_i$ to increase. This constraint is particularly significant near singularities, during a switch between solution branches, and when there is a substantial change in tool orientation. Therefore, the step size between points is selected based on changes in position and orientation, rather than solely on the number of trajectory nodes. The initial trajectory is represented as an array of target poses obtained from teaching points or an external planner [3, 4, 11, 12].

The study utilized the kinematic model of the STP3 project [17], for which the forward kinematics, a numerical Jacobian, and an inverse kinematics algorithm for a single pose had previously been derived. The objective was to develop and validate a trajectory calculation algorithm based on specified TCP positions and orientations. The algorithm takes an array of target poses as input and generates a sequence of joint angles as output. To achieve this goal, continuity conditions were formulated, a method for specifying orientation was selected, convergence criteria were defined, and calculations were performed for 5, 10, 20, and 50 points.

A six-link manipulator is considered, the kinematics of which are defined by Denavit-Hartenberg parameters. The model corresponds to an anthropomorphic manipulator with revolute shoulder, elbow, and wrist joints. Parameter values are provided in [17]. Forward kinematics defines the mapping $q \to T_0^6(q)$, where $T_0^6(q)$ is a 4×4 homogeneous matrix of the form $[R(q)\ p(q);\ 0\ 1]$, $q \in \mathbb{R}^6$ is the vector of joint angles, $p(q) \in \mathbb{R}^3$ is the TCP position, and $R(q) \in SO(3)$ is the tool orientation matrix [1–5, 17]. For a given pose $T_d = (R_d, p_d)$, it is necessary to determine a vector q^* satisfying the conditions $\| p(q^*) - p_d \| \leqslant \varepsilon_p$ and $\rho(R(q^*), R_d) \leqslant \varepsilon_R$. The quantity $\rho(\cdot,\cdot)$ characterizes the orientation mismatch.

An array of positions $p_i \in \mathbb{R}^3$ and an array of orientations $R_i \in SO(3)$, $i = 1, \ldots, N$ are given. The objective is to determine a sequence of joint angles $q_i \in \mathbb{R}^6$, $i = 1, \ldots, N$, such that, for each i, the matrix $T_0^6(q_i)$ corresponds to the specified pose $T_i = (R_i | p_i)$ within a prescribed tolerance. The differences $\| q_{i+1} - q_i \|$ between consecutive points must not result in exceeding permissible joint velocities and accelerations during time-based interpolation [2–4, 11, 12]. The initial guess for point $i + 1$ is set to the solution obtained for the previous point: $q_0^{(i+1)} = q_i$.

The orientation is defined by Euler angles in the ZYX convention, which are subsequently used to construct a six-dimensional error vector:

$$R = R_z(\psi) R_y(\theta) R_x(\phi),\ \eta = (\phi, \theta, \psi)^T.$$

Taking into account the positional and orientational components, an error vector is obtained

$$e = \begin{pmatrix} p_d - p(q) \\ \eta_d - \eta(q) \end{pmatrix}.$$

The first three components characterize the position error, while the last three characterize the orientation error [7–10]. This representation is used when constructing the numerical Jacobian. Special handling of the angles is required in the vicinity of gimbal lock.

It is assumed that the target poses are reachable and not located in the immediate vicinity of singular configurations. If this condition is violated, the Jacobian becomes ill-conditioned, the number of iterations increases, and a small change in the target pose can cause a significant change in the angles [5–9, 16]. Therefore, alongside solving the inverse problem, it is necessary to monitor the stability of the calculation and select an appropriate trajectory step size.

METHODS

The calculation is performed sequentially for points $i = 1, \ldots, N$, with an internal iterative calculation carried out for each point. For $i = 1$, the initial approximation $q_0^{(1)}$ is set as either the zero configuration or a known configuration close to the first target pose. For the remaining points – that is, for $i \geqslant 2$ the previously obtained solution is used:

$$q_0^{(i)} = q^*_{i-1},$$

where the initial approximation is set equal to the result from the previous point. At iteration k, we first deter-

ISSN 2075-8545 (online)

Nanotechnologies in construction
Нанотехнологии в строительстве

2026; 18 (5):
681–690

mine the current pose $T_0^6(q_k)$ and the error e_k, then find the numerical Jacobian $J(q_k)$ and the increment

$$\Delta q = J^+(q_k)e_k.$$

After determining the increment, the angles are refined using the formula

$$q_{k+1} = q_k + \lambda \Delta q.$$

The iterative calculation terminates when $\| e_k \| < \varepsilon$ or after the specified number of iterations N_{max} is reached [5–9, 16].

The ZYX Euler angles (ϕ, θ, ψ) required to calculate the orientation error are determined from the rotation matrix $R \in SO(3)$. When writing

$$R = R_z(\psi)R_y(\theta)R_x(\phi)$$

the inverse transformation can be performed using the formulas

$$\theta = \operatorname{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right);$$

$$\phi = \operatorname{atan2}\left(\frac{r_{32}}{\cos\theta}, \frac{r_{33}}{\cos\theta}\right);$$

$$\psi = \operatorname{atan2}\left(\frac{r_{21}}{\cos\theta}, \frac{r_{11}}{\cos\theta}\right)$$

provided that $\cos\theta \neq 0$ [9, 10]. Special handling is applied in the vicinity of gimbal lock, and the resulting angles are mapped to the interval $[-\pi, \pi]$. These same angles are used in the six-dimensional error vector and in the calculation of the numerical Jacobian.

The Jacobian $J(q) \in \mathbb{R}^{6\times6}$ relates the increment in angles Δq to the increment of the "position + orientation" state vector:

$$\Delta x \approx J(q)\Delta q.$$

The i-th column of the matrix is calculated using the finite difference.

$$J_{:,i}(q) \approx \frac{f(q + \delta e_i) - f(q)}{\delta},$$

where $f(q)$ contains the three TCP coordinates and three Euler angles, e_i is a unit vector, and $\delta = 10^{-6}$ rad [5, 6]. Next, we compute the pseudoinverse matrix J^+ using scipy.linalg.pinv or an equivalent SVD procedure. Using this matrix, we determine the increment $\Delta q = J^+e$ that minimizes the residual norm $\| J\Delta q - e \|_2$.

In the implementation used, the condition number cond(J) is not automatically monitored. When cond(J) $> 10^5$, standard pseudoinversion should be replaced by a regularized damped least squares scheme.

$$J_\mu^+ = (J^T J + \mu I)^{-1} J^T,$$

where $\mu > 0$ is the regularization parameter [7–9, 16]. Regularization limits the increment Δq when the singular values of the Jacobian decrease.

Convergence is determined based on the error norm $\| e \|$. The threshold ε combines the positional component (in meters) and the orientational component (in radians). If the required accuracy is not achieved after N_{max} iterations, the calculation can be repeated using a different initial guess or a regularized pseudoinverse [5–9]. In the developed script, the parameter λ remains constant. For a point where convergence is not achieved, a warning is issued, and the value q_i is set equal to the previous solution q_{i-1}. This prevents a large increment from propagating to subsequent points, although it increases the error at the node in question.

At each step, forward kinematics, error generation, Jacobian calculation, increment determination, and stopping criterion check are performed. In Figure 1 shows the sequence of operations and shows the transfer of the result of the previous point as an initial approximation.

The step size determines what fraction of the calculated increment is used to refine the angles. If the matrix J is well-conditioned, the increment is fully utilized when $\lambda = 1$; however, in the presence of a large initial mismatch, such a step may lead to oscillations in the solution. To limit the step size, values of $\lambda \in [0.5, 1]$ are considered, or the coefficient is made dependent on the current error:

$$\lambda = \min\left(1, \frac{\lambda_0}{\| e \|}\right),$$

where λ_0 is the base coefficient for step-size selection [5–9]. A different approach was adopted for the calculations: a value of $\lambda = 0.7$ was selected based on preliminary

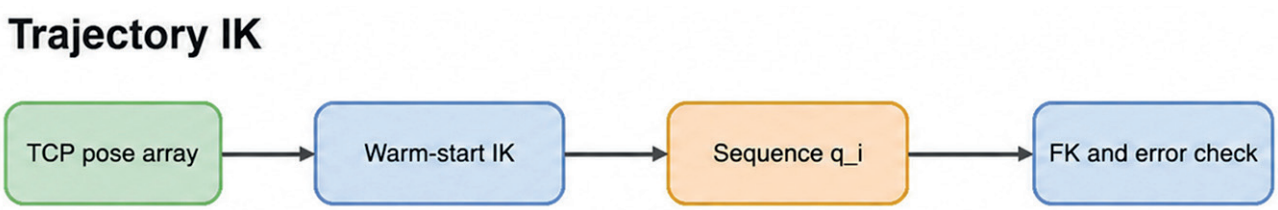

Fig. 1. Block diagram of the IK algorithm for the trajectory

ISSN 2075-8545 (online)

tests and used for all iterations. The relationship shown illustrates a possible method for varying λ; however, adaptive step-size selection was not implemented in the version tested.

When formulating the error vector $e \in \mathbb{R}^6$, there are three positional components (expressed in meters) and three orientation components (expressed in radians). In the calculation model, these are included in the combined vector without additional weighting coefficients:

$$e = \begin{pmatrix} e_p \\ e_\eta \end{pmatrix}.$$

With this calculation, the overall norm depends on the chosen units for position and orientation. If the permissible orientation mismatch differs from the positional requirement, the influence of the angular component can be adjusted by introducing a weighting factor:

$$e = \begin{pmatrix} e_p \\ w_\eta e_\eta \end{pmatrix},$$

where $w_\eta \approx 0.1 \ldots 0.2$ [7–10]. Separate tolerances can also be specified: ε_p for position and $\varepsilon_\eta \approx 10^{-4}$ rad for orientation, with $w_\eta = 1$. In this case, the convergence condition is expressed as $\| e \| \leq \sqrt{\varepsilon_p^2 + \varepsilon_\eta^2}$. The error on the order of 10^{-8} rad observed in experiments allows for the use of separate criteria without imposing stricter requirements on the solver.

Several iterations are performed for a single point. In each iteration, forward kinematics, a numerical Jacobian, and a 6×6 pseudoinverse matrix are calculated. The greatest computational cost is associated with the numerical Jacobian, as its construction requires six additional forward kinematics calculations. The total time can be estimated by the expression

$$T_{\text{total}} \sim N_{\text{точек}} \cdot \overline{N}_{\text{iter}} \cdot T_{\text{iter}},$$

where $\overline{N}_{\text{iter}}$ is the average number of iterations per point, and T_{iter} is the time for a single iteration [3–6, 14, 15]. To reduce computation time, an analytical Jacobian, the storage of intermediate matrices, and compiled linear algebra libraries can be employed. In the algorithm under consideration, the initial approximation obtained at the previous point reduces computational cost by decreasing the number of iterations.

Figure 1 shows the data flow from the array of TCP poses to the array of joint angles. This scheme corresponds to the sequence of operations in the algorithm's software implementation.

Calculations were performed in a Python 3 environment using the kinematic model described in [17]. The script `ik_trajectory.py` utilized functions for forward kinematics, the numerical Jacobian, and the inverse kinematics solution for a single pose. Coordinates were specified in meters, and angles in radians.

Three scenarios were considered. In the first case, the trajectory consisted of five points located along a straight line segment within the workspace; the tool orientation remained constant. In the second case, ten points were defined along an arc in the x–y plane at a constant height, with the orientation either remaining constant or changing to align with the direction of motion. The third scenario involved twenty points with a uniform change in position and a rotation of the orientation around the vertical axis.

For each scenario, the following parameters were determined: the number of iterations, the position error $\| p(q_i) - p_i \|$, the orientation error based on the difference in Euler angles, and the total computation time for the sequence $q_1, \ldots, q_N$ [3–9, 11, 12]. Trajectory smoothness was evaluated based on the change in angles between adjacent points.

The calculations utilized a convergence threshold of $\varepsilon = 10^{-6}$, a maximum iteration count of $N_{\max} = 100$, and a step coefficient of $\lambda = 0.7$. The value $\lambda = 0.7$ was selected based on preliminary tests; λ remained constant thereafter – meaning no adaptive step-size adjustment was applied. For the first point, the initial approximation was set to the zero configuration $q_0 = (0,0,0,0,0,0)$, while the solution from the previous point was used for each subsequent point. No additional weighting factors for orientation error were introduced.

Calculations were performed on Windows 10 using Python 3.8 and the NumPy, SciPy, and Matplotlib libraries. The total trajectory computation time (excluding visualization) was measured using the time.perf_counter function. Based on the results, the TCP trajectory and plots showing position error and joint angle changes were generated.

Additionally, points at the boundary of the workspace and small pose variations relative to a previously obtained solution were tested. With small pose variations, the number of iterations decreased. Calculations were performed for a single geometric trajectory using $N = 5, 10, 20, 50$. As the number of nodes increased, the average number of iterations per point decreased. At $N = 50$, the total time was 1–2 seconds, making the algorithm suitable for offline planning.

For a trajectory consisting of 15 points, a 30° orientation rotation around the z-axis was specified. The resulting sequence q_i was continuous, and the error did not exceed the established threshold. When the same extended segment was defined by only three points, a solution branch switch occurred in one of the calculations, resulting in a jump in joint angles. For trajectories of 10 and 20 points, the condition number cond(J) remained within the range of 8–12. At certain nodes of the 5-point trajectory, however, it reached higher values. Consequently, the density of pose specification significantly influences the algorithm's performance [2–4, 11, 12].

ISSN 2075-8545 (online)

RESULTS

The convergence condition was met for five points. The highest number of iterations was required for the first point, for which the zero configuration was used. According to Table 1, the average number of iterations for this trajectory was also the highest. The position error did not exceed 10^{-7} m or 10^{-4} mm (Fig. 2). The resulting TCP trajectory matches the target trajectory within the specified accuracy, and the joint angles vary smoothly [5–10] (Fig. 3).

As the number of points increased, the average number of iterations decreased from 28 at $N = 5$ to 18 at $N = 10$ and to 12 at $N = 20$ (Table 1). Position and orientation errors did not exceed the specified threshold. The computation time for 20 points allows the algorithm to be used for offline planning (Fig. 4). Operation at a frequency of 100–500 Hz would require an analytical Jacobian or

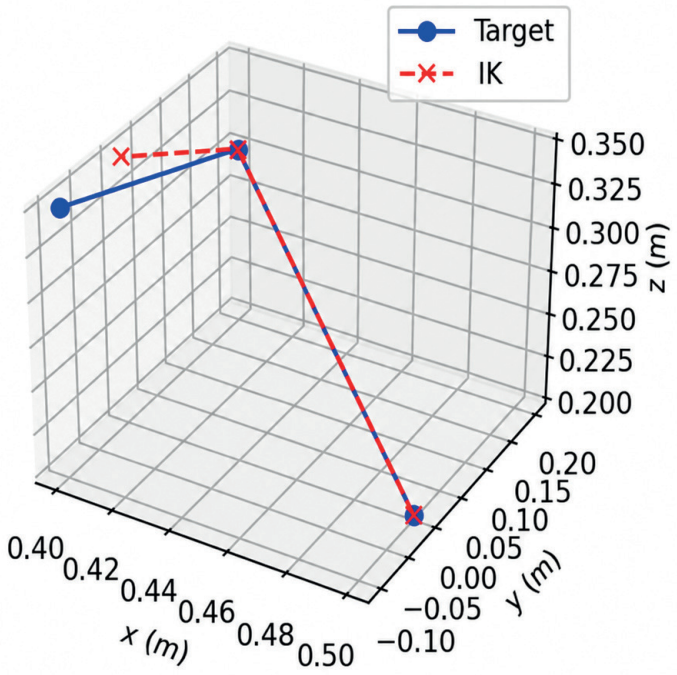

Fig. 2. Target and actual TCP trajectory in 3D (5 points)

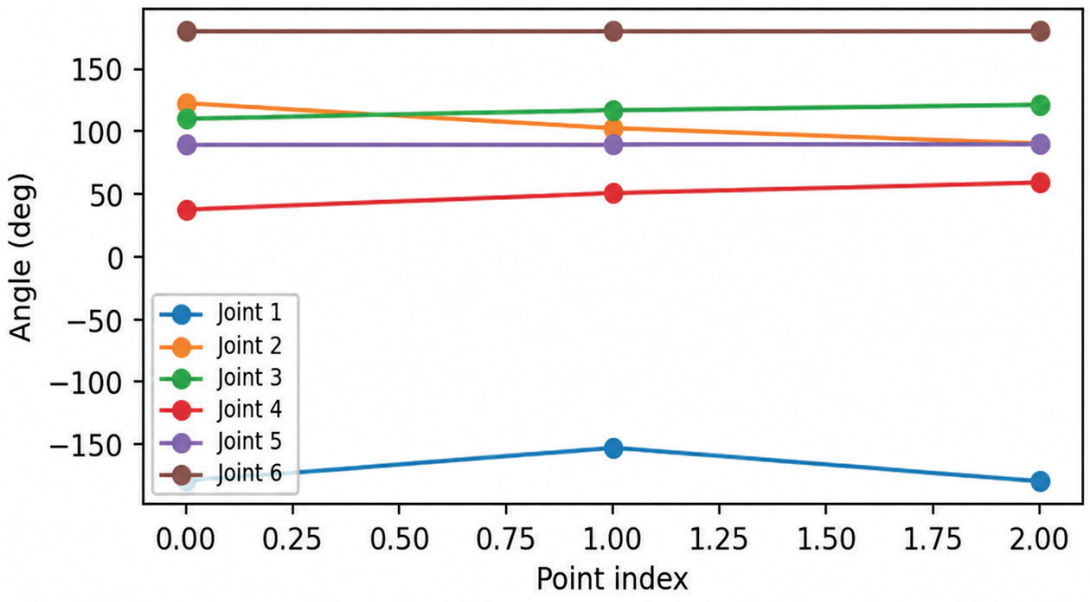

Fig. 3. Joint angles based on trajectory points

ISSN 2075-8545 (online)

Table 1. Number of iterations and inverse kinematics computation time for the trajectory

N points	Average number of iterations per point	Max. iterations per point	Total time (ms)
5	28	45	180
10	18	35	220
20	12	28	320

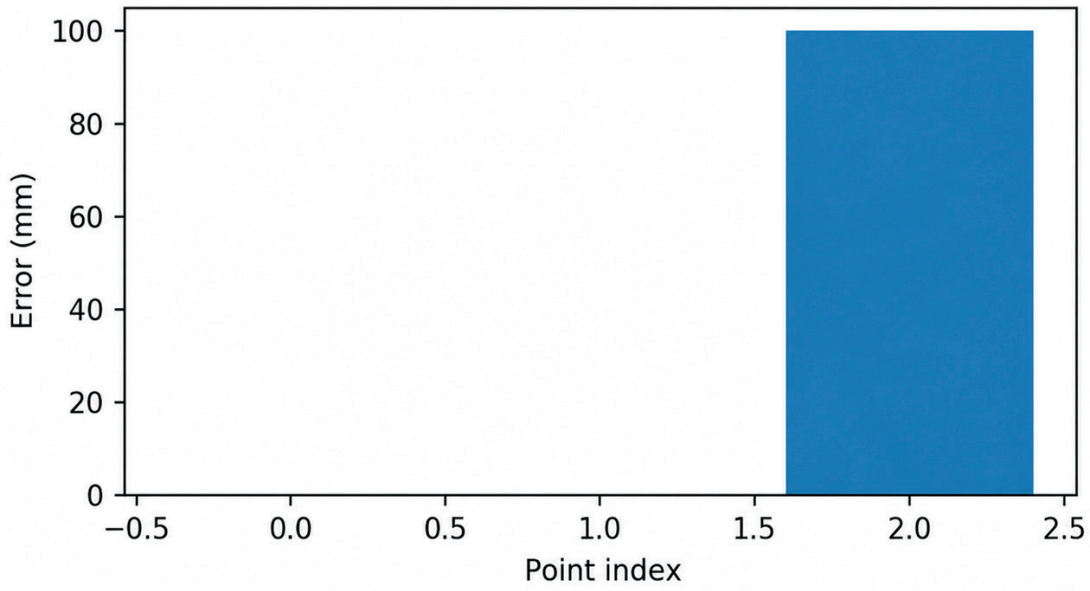

Fig. 4. Positioning error along trajectory points (mm)

a faster implementation of the computational operations [3–8, 14–16].

For a trajectory consisting of 10 points, the position error did not exceed 10^{-7} m or 10^{-4} mm. This value is smaller than the relevant manufacturing tolerances; therefore, in practical application, accuracy will be determined by the kinematic model, calibration, and specified tolerances.

A comparison was made with a calculation in which a zero initial guess was specified for each point (Fig. 5). In that case, the number of iterations increased, and a switch to a different solution branch could occur. Using the previous result reduced the average number of iterations by a factor of 2–3 and eliminated discontinuities in the trajectories under consideration (Table 1). Thus, the initial guess affects both the computation time and the continuity of joint coordinates [2–4, 7–9].

The number of iterations and the computation time for a single geometric trajectory were compared across different discretization levels (Table 1). As N increases, adjacent target poses become closer to one another; consequently, the previously found solution requires fewer refinements when transitioning to the next point.

Table 1 shows that the average time per point decreases from 36 ms for a 5-point trajectory to 16 ms for a 20-point trajectory. At the same time, the total computation time increases from 180 to 320 ms: the reduction in the number of iterations per point does not compensate for the increase in the number of points themselves.

Based on the calculation results, the discretization step should be selected by taking into account the changes in position and orientation between adjacent poses. When approaching a singularity, the condition number should be monitored, and a switch made to regularized pseudoinversion. For trajectories consisting of 10 and 20 points, cond(J) remained moderate, whereas poor conditioning was observed at specific nodes of the 5-point trajectory. If orientation requirements are less stringent than position requirements, one can maintain $w_\eta = 1$, set a tolerance of $\varepsilon_\eta \approx 10^{-4}$ rad, and use the criterion $\| e \| \leq \sqrt{\varepsilon_p^2 + \varepsilon_\eta^2}$. The algorithm is designed for offline trajectory generation and can be implemented as part of a planning module [3, 4, 7–9, 11–16].

The algorithm was developed using engineering control of solution continuity, orientation representation via Euler angles, monitoring of singular configurations, and regularized pseudoinversion. The combination of these methods aligns with the Russian and international approaches discussed previously [2–9, 16].

ISSN 2075-8545 (online)

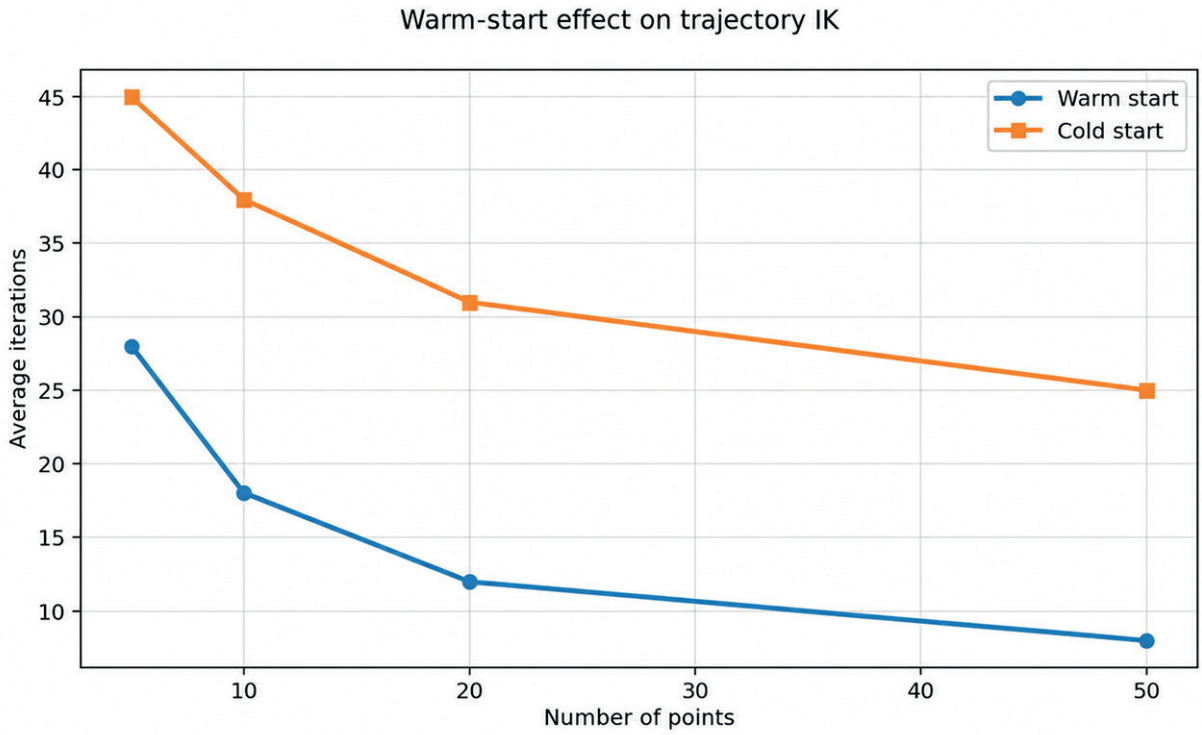

Fig. 5. Comparison of warm start and zero-order approximation

Table 2. Russian and foreign views on trajectory inverse kinematics

Criterion	Russian practice	International practice
Primary focus	Solution continuity and engineering resilience	Warm start, branch consistency, task-space pipelines
Representation of orientation	More often Euler angles and matrix description	Euler, quaternion, axis-angle
Primary risk	Switch solution branch	Loss of continuity near singularity
Typical sources	Robotics training courses and engineering manuals	Craig, Siciliano, Buss, Wampler

CONCLUSION

An inverse kinematics algorithm is proposed for trajectories with specified tool orientation, utilizing the solution from the preceding point as the initial approximation. For the trajectories analyzed, this computational approach ensured continuity of joint coordinates and reduced the number of iterations. By employing the pseudoinverse of the numerical Jacobian and ZYX Euler angles, a positional error on the order of 10^{-7} m was achieved with a threshold of $\varepsilon = 10^{-6}$ [5–10].

Convergence conditions deteriorate as the system approaches a singular configuration; furthermore, if poses are defined sparsely, a switch in the solution branch may occur even when using the previous result. For trajectories consisting of 10 and 20 points, the condition number remained moderate, whereas an increase was observed at specific nodes of a 5-point trajectory. These results highlight the need to monitor cond(J), refine the discretization step, and apply regularization when signs of instability arise [7–9, 16].

The algorithm enables the determination of joint coordinates based on Cartesian poses specified by an operator or a CAD system, thereby reducing the manual effort required for trajectory definition. Future research could focus on incorporating joint constraints, avoiding singular configurations, utilizing an analytical Jacobian, and validating the algorithm on a physical manipulator [3, 4, 11–17].

ISSN 2075-8545 (online)

REFERENCES

1. Denavit J., Hartenberg R. S. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*. 1955;22:215–221. https://doi.org/10.1115/1.4011045

2. Craig J. J. *Introduction to Robotics: Mechanics and Control*. 3rd ed. Boston: Pearson; 2005.

3. Siciliano B., Sciavicco L., Villani L., Oriolo G. Robotics: *Modelling, Planning and Control*. London: Springer; 2009. https://doi.org/10.1007/978-1-84628-642-1

4. Lynch K. M., Park F. C. *Modern Robotics: Mechanics, Planning, and Contro*l. Cambridge: Cambridge University Press; 2017. https://doi.org/10.1017/9781316661239

5. Spong M. W., Hutchinson S., Vidyasagar M. *Robot Modeling and Control*. New York: Wiley; 2006.

6. Murray R. M., Li Z., Sastry S. S. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC Press; 1994. https://doi.org/10.1201/9781315136370

7. Wampler C. W. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*. 1986;SMC-16(1):93–101. https://doi.org/10.1109/TSMC.1986.289285

8. Chiaverini S. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*. 1997;13(3)398–410. https://doi.org/10.1109/70.585902

9. Buss S. R. *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods*. 2004. URL: http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/ikmethods.pdf (accessed: 21.04.2026).

10. Kuipers J. B. *Quaternions and Rotation Sequences*. Princeton: Princeton University Press; 1999. https://doi.org/10.1515/9780691211701

11. LaValle S. M. *Planning Algorithms*. Cambridge: Cambridge University Press; 2006. https://doi.org/10.1017/CBO9780511546877

12. Latombe J.-C. *Robot Motion Planning*. Boston: Kluwer Academic Publishers; 1991. https://doi.org/10.1007/978-1-4615-4022-9

13. Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*. 1986;5(2):90–98. https://doi.org/10.1177/027836498600500106

14. Yoshikawa T. Dynamic and kinematic redundancy of robot manipulators. *Journal of Robotic Systems*. 1985;2(1):83–93. https://doi.org/10.1177/027836498500400201

15. Slotine J.-J. E., Li W. *Applied Nonlinear Control*. Englewood Cliffs: Prentice Hall; 1991.

16. Siciliano B. Handling robot inverse kinematics with redundancy and singularity robustness. *Advanced Robotics*. 2010;24(7-8):1021–1038. https://doi.org/10.1163/016918610X493592

17. Kinematic model of a 6-DOF manipulator based on Denavit-Hartenberg parameters: forward and inverse kinematics. *STP3 project internal preprint (paper St1)*. 2026.

ADDITIONAL INFORMATION

Automated language editing tools were used in the preparation of the manuscript. The scientific formulation, calculations, analysis of results, and final review of the text were performed by the authors.

INFORMATION ABOUT THE AUTHORS

Petr A. Kulakov – Cand. Sci. (Eng.), Associate Professor, Department of Machine Design and Mechanics, Ufa State Petroleum Technological University, Ufa, Kosmonavtov str. 1, 450064, Republic of Bashkortostan, Russian Federation, kulakov.p.a@mail.ru; https://orcid.org/0000-0003-1402-1366

Afanasenko Vitaly Gennadyevich – Cand. Sci. (Eng.), Head of the Department of Machine Design and Mechanics, Ufa State Petroleum Technological University, Ufa, Kosmonavtov str. 1, 450064, Republic of Bashkortostan, Russian Federation, afanasenko.v.g@yandex.ru; https://orcid.org/0000-0002-4349-1925

ISSN 2075-8545 (online)

CONTRIBUTION OF THE AUTHORS

P.A. Kulakov – scientific supervision, development of research concept and methodology, analysis of research results, writing the article, formulation of final conclusions.

V.G. Afanasenko – data editing and processing, preparation and drafting of the article's conclusions.

The authors declare no conflicts of interest.

The article was submitted 30.06.2026; approved after reviewing 02.10.2026; accepted for publication 07.10.2026.