A speedup method for solving the inverse kinematics problem of robotic manipulators

Shuxin Xie, Lining Sun, Zhenhua Wang and Guodong Chen

Abstract
The inverse kinematics problem involves the study that the inverse kinematics solver needs to calculate the values of the joint variables given the desired pose of the end-effector of a robot. However, to apply to seven-degree-of-freedom robots with arbitrary configuration, analytical methods need to fix one joint and set an increment when the current value fails to solve the inverse kinematics problem. Although numerical methods based on inverse differential kinematics are efficient in solving the inverse kinematics problem of seven-degree-of-freedom robots with arbitrary geometric parameters, they are deficient in numerical stability and time-consuming for convergence to one solution governed by the initial guess. In order to reduce the execution time of an inverse kinematics solver, this article introduces a speedup method for analytical and numerical methods, which can improve their performance.

Keywords
Robot kinematics, speedup method, Newton–Raphson method, Levenberg–Marquardt algorithm, sequential quadratic programming, inverse kinematics

Introduction
In order to place its end-effector exactly at a desired position and orientation in Cartesian space, a serial-chain robot needs to possess no less than six degrees of freedom (DOFs). A 7-DOF robot has one more redundant DOF than it requires for positioning the end-effector in Cartesian space. A 7-DOF robot has more practical significance in applications as it has some advantages. For example, it can provide greater flexibility and reliability when executing tasks and avoid obstacles in complex and unknown work environments. But unfortunately, it is arduous to find the solution of inverse kinematics (IK) for a redundant robot in real time. How to solve the inverse kinematics problem (IKP) of a redundant serial-chain robot has been and remains quite challenging in robotics research and applications.

Generally speaking, there exist analytical methods and numerical methods to solve the IKP of redundant robots. Because the IK solver must be pre-built, an analytical IK solution method cannot be generalized to the cases where a robot configuration changes and a tool is mounted at the end. Numerical methods are widely used to solve the IKP of various configurations. However, due to the limitation of the number of iterations and the setting of convergence
conditions, these IK solution methods are sometimes not feasible. Most numerical IK solution methods use the Newton–Raphson (NR) algorithm or similar to iterate until they find a solution. Compared with analytical methods, numerical methods are reasonable in theory, and they take a long time to find one IK solution. Sometimes, the precision of the found solution is low, and even the phenomenon occurs that the IKP cannot be solved correctly. Scholars have developed many methods to accelerate the calculation of Jacobian and matrix inversion and optimize numerical methods to improve solution efficiency. Many methods are developed according to the Denavit–Hartenberg (D-H) parameters of robots, but there are methods based on screw transforms and exponential rotational matrices, which arise from the explicit physical interpretation of the mechanism with Euler wrist and can be easily realized. The IKP becomes more complicated when obstacles in the workspace impose collision avoidance constraints. Convex Iteration for Distance-Geometric Inverse Kinematics algorithm can solve the convex feasibility problem whose low-rank feasible points provide exact IK solutions for complex workspace constraints.

For redundant robots, the analytical IK solution needs one joint fixed and then solves other joint rotation angles. Concerning the initial fixed joint angle value, the solution may fail. If it happens, an increment will add to the initial value until the solution succeeds within the time limit. Thus, more precisely to say, the fixed joint is free to change its value. For specific mechanisms, such as manipulators with Euler wrist, using dual quaternions can obtain a very compact formulation for closed-form solutions. As to the numerical IK solution, a set of initial joint angles is required to compute the first Jacobian or as a seed state to start an iterative search. Prior to finishing the solving process, the number of iterations cannot be determined. To reach a particular precision, this kind of approach is computationally expensive in many cases, which can be numerically unstable. Although state-of-the-art algorithms have reduced inverse solution computation time to milliseconds or even microseconds, the challenge of computational efficiency in real-time robot motion planning still exists, especially for complex robot configurations. In this article, we propose a speedup solution to the IKP of a 7-DOF redundant robot, which can reduce the computational time when applying analytical and numerical methods for the IKP and improve the success rate of the analytical method.

Methods

Solving the IKP of redundant robots is a problem of nonlinear mapping from robot operating space to joint space. It has to seek the numerical solution of nonlinear transcendental equations. At present, methods to solve the IKP of redundant DOF robots are mainly numerical, such as pseudo-inverse of Jacobian methods, extended Jacobian methods, task-space augmentation methods, gradient projection methods, damped least square methods, and weighted least-norm methods. Besides, there are analytical methods to solve the IKP of redundant robots with specific geometric properties, such as solution algorithms based on screw theory, geometric methods for robots with unique geometric features, and the analytical approach by fixing a certain DOF. We denote the configuration space of an n-joint, serial-link articulated robot as \( Q \), which consists of n-tuples element \( \hat{q} = (\theta_1, \theta_2, \ldots, \theta_n)^T \in \mathbb{R}^n \), each \( \theta_i \) representing the \( i \)th joint value of the robot, and its operating space as \( X \), which consists of 6-tuples \( \bar{x} \in \mathbb{R}^6 \) with three translation variables \( p_x, p_y, \) and \( p_z \) and three rotation angle variables \( \alpha, \beta, \) and \( \gamma \). Then, a forward kinematic equation of this robot is represented as

\[
\bar{x} = FK(\hat{q}), \bar{x} \in X, \hat{q} \in Q \tag{1}
\]

Function \( FK : Q \rightarrow X \) is differentiable. For each \( \bar{x} \) in operating space, according to the IK equation \( IK(\bar{x}) = \{ \hat{q} | \bar{x} = FK(\hat{q}) \} \), there exist infinite sets of joint values \( \hat{q} \) corresponding to one same point \( \bar{x} \) in operation space. In other words, the IK equation will have infinite solutions.

**IKfast approach**

We can use the D-H convention, kinematic chains, or unified robot description format (URDF) to represent the geometry of a robotic mechanism. An articulated joint has a revolute joint variable and other constant parameters. Regardless of how to represent the geometry of a robot, the coordinate system of one of its joints relative to the reference frame can be defined as \( A_i = T_{i-1}J_i \), where \( T_i \) is a constant affine transformation computed by the invariant geometrical parameter and \( J_i \) is a transformation depending on variable \( \theta_i \), which is a rotation angle around an arbitrary axis \( \overrightarrow{v_{13}}(v_{13}, v_{1y}, v_{1z}) \)

\[
J_i(\theta_i) = I_{4 	imes 4} + \sin \theta_i \hat{\omega} + (1 - \cos \theta_i)\hat{\omega}^2 \tag{2}
\]

In equation (2), \( \hat{\omega} \) is a \( 4 \times 4 \) skew matrix defined by

\[
\hat{\omega} = \begin{bmatrix}
0 & -v_{1z} & v_{1y} & 0 \\
v_{1z} & 0 & -v_{1x} & 0 \\
v_{1y} & v_{1x} & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \tag{3}
\]

Thus, the transformation of the end-effector can be formulated by continuous multiplication of the fundamental transformation: \( T_{ee} = A_1A_2 \cdots A_n \). With respect to the base, the position and orientation of the end-effector are represented as a \( 4 \times 4 \) transformation matrix, \( T_{ee} \), and it is represented as
As a result, the IKP corresponds to computing the unknown variables, \( \theta_i \)’s.

Diankov believed that there are four different analytical methods to solve the IK problem. The first method is the algebraic representation of computational formulas, where the main problem is how to solve higher-order univariate polynomials. Univariate higher-order equations are liable to become ill-conditioned when looking for their roots. The second method is to analyze and solve the structure of the solution set of the polynomial system, but there is a numerical ill-conditioning. The third method is to apply linear algebra and formulate the solution process as an eigendecomposition. Although the eigendecomposition method is the fastest of all the proposed methods, the eigendecomposition of the 48 \( \times \) 48 matrix still suffers from numerical instability. The fourth method we choose in this article, IKfast, is a solver plug-in developed by Diankov. The IKfast method emphasizes the numerical stability of the solution process rather than the general applicability of the IK solution. The IKfast method can find the most robust closed-form solution for 6-DOF robots. For a robot with one DOF redundancy, a joint is taken as the free joint, and it is discretized between its upper and lower limit so that a similar closed-form solution can be obtained.

The IKfast method fundamentally changes the way researchers solve the IKP when doing kinematic planning for serial-chain manipulators. All researchers can easily and safely integrate analytical algorithms into their motion planners to obtain multiple inverse solutions, rather than relying on the gradient-based methods, which are plagued by initial values, numerical errors, and slow computing time, and even worse, by which only one approximate solution can be obtained. The IKfast method analyzes the structure of the whole equations derived from the forward kinematics and divides the functional relationship into four cases. Each entry of the matrix in the left side equations is constructed by the sine and cosine functions of the joint variables.

In equation (4), it is easy to see that there are 12 equations and they have scalar components on the right side. In the matrix, the rotation part is equivalent to the product of three elementary rotation matrices generated from a set of Euler angles or fixed angles, which represents the orientation of the end-effector. And the first three rows and three columns of \( T \) are orthonormal, plus three-position equations, only six equations are independent. Thus, for a robot with \( n \) joints, the IKP corresponds to solving six equations with \( n \) unknowns. Usually, solving these equations involves complicated trigonometry and generally does not yield a simple solution. When joints number \( n \geq 6 \), the number of independent equations is less than the DOF, thus it is called underconstrained problem.

When considering \( \sin \theta \) and \( \cos \theta \) as unknown variables, combined with trigonometric constraints \( \sin^2 \theta + \cos^2 \theta = 1 \) for all joint rotation angles, we can obtain a set of equations with up to \( 2n \) unknowns. In order to build a stable IK solver, our first task is to take advantage of as many simple structures as possible, starting with all variables that can be solved with low-degree polynomials. The solving process becomes more complicated when coupled pairs of joint variables exist. In this case, joint variables can be solved by finding the intersection of the quadratics. If there are more joint variable couplings, we seek a fast and robust algorithm for the IK of a manipulator with \( n \) joints. The solutions of polynomial equations will be roughly divided into the following four types:

(1) The angle \( \theta \) can be solved from two simple linearly independent equations with the functions of sine and cosine. The equation \( \sin \theta = a \) has two solutions \( \theta = \arcsin(a) \) and \( \theta = -\arcsin(a) \). Likewise, as to the equation \( \cos \theta = b \), it has two solutions \( \theta = \arccos(b) \) and \( \theta = -\arccos(b) \). For the combination \( \sin \theta = a \) and \( \cos \theta = b \), we have the only solution \( \theta = \arcsin(a) \).

(2) The little more complicated equation, such as the transcendental equation \( \arccos \theta + \arcsin \theta = c \), has two solutions \( \theta = \arccos(a) \) and \( \theta = \arccos(-a) \). When the right hand is a nonzero constant, the equation becomes \( \arccos \theta + \arcsin \theta = c \), and it can be solved by \( \theta = \arctan(\sqrt{c^2 - a^2}) \). A final solution can be solved within \( \pi \) radians. The set of equations

\[
\begin{align*}
\text{acos}\theta + \text{bsin}\theta &= c \\
\text{ecos}\theta + \text{f}\sin\theta &= d
\end{align*}
\]

is solved by \( \theta = \arctan(\sqrt{(ad - ce)/((af - be)), \text{(ce - bd)/(af - be)}}) \).

(3) Also resolve to equations of the form \( \cos^2 \theta = a \), which can have up to four solutions: \( \theta = \pm \arccos(\pm \sqrt{a}) \).

(4) The most complex condition is to resolve into a high-degree multivariate polynomial. In this case, we have to solve the determinant \( Ax^2 + Bx + C = 0 \) problem. Choosing one joint as the free joint and assigning a value by increment, the method developed by Manocha and Canny can solve the solution to the IKP for the 7R serial manipulator of general geometry.

Failure may happen when the IKfast solver runs at the initial guess for the free joint. In this case, the value of the
free joint can be set according to Algorithm 1. In this way, the success rate will improve markedly.

Algorithm 1. To generate a new guess for the free joint.

Require:
q\text{ini}, the initial guess for the free joint
q\text{upper}, the upper limit of the free joint
q\text{lower}, the lower limit of the free joint
\Delta q, the search discretization increment

Ensure:
q\text{new}, a new guess;

1: Initializing: q\text{new} \leftarrow 0, counter \leftarrow 0
2: \eta\text{pos} \leftarrow \text{FLOOR}(\frac{(q\text{upper} - q\text{ini})}{\Delta q}), getting the step number from the initial guess to the upper limit
3: \eta\text{neg} \leftarrow \text{CEIL}(\frac{(q\text{lower} - q\text{ini})}{\Delta q}), getting the step number from the lower limit to the initial guess
4: counter \leftarrow \text{GETCOUNT}(\eta\text{pos}, \eta\text{neg}, \text{counter}), to generate a new counter between \eta\text{pos} and \eta\text{neg}
5: q\text{new} \leftarrow q\text{ini} + \text{counter} \times \Delta q, to generate a new guess for the free joint
6: return q\text{new}

Newton–Raphson method

The NR method is a root-finding algorithm that produces successively better approximations to the true root for real-valued functions. In robotics, the NR method can solve the IKP of robots as a numerical method, and it needs to compute the Jacobian matrix and its pseudo-inverse. In the NR method, the forward kinematics of the initial configuration q\text{temp} is computed, and the result is expressed as a twist T\text{temp}. Given the desired pose, its corresponding twist is expressed as T\text{d}. Thus, we have the difference of the two twists: \Delta T = T\text{d} - T\text{temp}. Applying the eigendecomposition and the Moore–Penrose pseudo-inverse, we can compute joint increment \Delta q. For the new joint configuration q\text{temp} = q\text{temp} + \Delta q, an iteration process is executed until \Delta T \rightarrow 0. However, there are two necessary conditions for the success of the NR method: one is that the joints of the robot do not pass through singularities during the movement from the initial configuration to the final configuration; the other is that the desired end-effector position is located in the working space of the robot. Due to these limitations, the NR method largely depends on the initial value and does not guarantee global convergence.

Levenberg–Marquardt algorithm

The Levenberg–Marquardt algorithm (LMA) can solve nonlinear least-squares problems. The LMA, also known as the damped least-squares method, generally converges faster than first-order backpropagation methods. In the LMA, the damping factor adopts the square norm of the residual of the original equation with a bit of deviation, which helps to prevent the algorithm from escaping local minima. By choosing an appropriate damping factor, the LMA becomes more robust. It can overcome both problems: singularity and solvability. As an alternative numerical method to solve the IKP, the LMA significantly improves the numerical stability and convergence performance.

For an n-DOF robot, all its joint variables form a vector \hat{q}. Then, the kinematics of the robot is mathematically represented by the constraint relationship between a pose (\hat{p}, R) and \hat{q}. Then, the positional constraint on the end-effector is represented as \hat{f}_p(\hat{q}) = \hat{p}, where \hat{f}_p(\hat{q}) \in \mathbb{R}^3 is the current position of the end-effector. Meanwhile, an orientational constraint on the end-effector is represented as \hat{f}_R(\hat{q}) = R, where \hat{f}_R(\hat{q}) \in SO(3) is the rotation matrix of the current orientation of the end-effector. The desired pose of the end-effector in the operating space is expressed as \hat{\rho}_d \in \mathbb{R}^3 and R_d \in SO(3). In general, the residual \tilde{e}(\hat{q}) can be expressed as

\begin{equation}
\tilde{e}(\hat{q}) = \begin{pmatrix}
\hat{\rho}_d - f_p(\hat{q}) \\
\nu(R_d f_R(\hat{q})^T)
\end{pmatrix}
\end{equation}

where \nu(R) \in \mathbb{R}^3, proposed by Luh et al. to evaluate the orientational error, then modified by Sugihara. For an arbitrary \hat{R} \in SO(3), \nu(R) is equivalent to an angle-axis vector. Furthermore, the whole residual vector \tilde{e}(\hat{q}) \in \mathbb{R}^m is defined as

\begin{equation}
\tilde{e}(\hat{q}) = \begin{bmatrix}
e_1(\hat{q}) \\
e_2(\hat{q}) \\
\vdots \\
e_m(\hat{q})
\end{bmatrix}
\end{equation}

where m is the total amount of constraints. The LMA focuses on the following minimization objective

\begin{equation}
E(\hat{q}) = \frac{1}{2} \tilde{e}^T W_e \tilde{e} \rightarrow \min
\end{equation}

where \( W_e = \text{diag} \{w_i\} \) \( w_i > 0 \) for \( i = 1 \sim m \) is a weighting matrix on the whole residual. NR is available to solve the above objective function and accepts minimization numerical results as approximate solutions. From the evaluation function \( E(\hat{q}) \), the following equations are acquired

\begin{equation}
\nabla E = \tilde{e}^T W_e \nabla \tilde{e}
\end{equation}

\begin{equation}
\nabla^2 E = \nabla \tilde{e}^T W_e \nabla \tilde{e} + \sum_{i=1}^{m} \frac{\partial^2 \tilde{e}}{\partial q_i^2} W_e 
\end{equation}

where \( \frac{\partial \tilde{e}}{\partial q_i} \approx -\frac{\partial}{\partial q_i} \) is related to the Hessian matrix of the robot. Then, for this minimization objective, NR operates according to the following update convention

\begin{equation}
q_{k+1} = q_{k} - (\nabla^2 E)^{-1} (\nabla E)^T
\end{equation}

However, the objective in IK solution is to find a descent direction of the evaluation function \( E(\hat{q}) \) rather than to
obtain its exact curvature. In order to simplify the computation, it is preferable to generate a positive-definite matrix to substitute $\nabla^2 E$ in equation (10). If the $\Sigma$ summation term of equation (9) is omitted, it becomes the same as the Gauss–Newton method, and equation (10) turns to

$$q_{k+1} = q_k + (J_k^T W_e J_k)^{-1}(J_k^T W_e e_k)$$  \hspace{1cm} (11)$$

where $J_k \equiv J(q_k)$, $e_k \equiv e(q_k)$. The iteration can continue in equation (11), as long as $J_k$ is non-singular. But, if $J_k$ is singular, the whole method becomes numerically ill-posed. However, the LMA can fix this problem by adding a damping factor and turning equation (11) to

$$q_{k+1} = q_k + H_k^{-1}(J_k^T W_e e_k)$$  \hspace{1cm} (12)$$

$$H_k \equiv J_k^T W_e J_k + W_n$$  \hspace{1cm} (13)$$

where $W_n$ is the damping factor, and $W_n = \text{diag}(w_{n,i})$ ($w_n > 0$ for all $i = 1-m$). In equation (13), $H_k$ must be guaranteed to be proper and positive-definite. Under this circumstance, the incremental term necessarily varies toward a descent direction. As Nakamura and Hanafusa\textsuperscript{24} pointed out, the iteration in equation (12) is also equivalent to solving the mixed minimization problem as shown as below

$$\frac{1}{2} J_k^T W_e r_k + \frac{1}{2} \Delta q_k^T W_n \Delta q_k \rightarrow \min$$  \hspace{1cm} (14)$$

where $\Delta q_k \equiv q_k+1 - q_k$ and $r_k \equiv e_k - J_k \Delta q_k$. At last, after a number of iterations, equation (14) converges to a certain configuration that has the minimum joint distance from the initial one.

**Sequential quadratic programming method**

If the initial guess is close to the right solution, the LMA optimization method can make the solution algorithm converge faster. However, it does not handle arbitrary initial guesses well. The third numerical method for IK solving is the sequential quadratic programming (SQP) method,\textsuperscript{25} which uses the quasi-Newton method and the Broyden–Fletcher–Goldfarb–Shanno (BFGS)\textsuperscript{26} gradient projection algorithm to find the IK solution. As an iterative algorithm for nonlinear optimization, the SQP method is adequate for solving the problems when the objective function with constraints is twice consecutively differentiable. The SQP uses the BFGS algorithm as its iterative search method. The BFGS is a Newtonian approximation method and has performed well even for non-smooth optimizations. Compared to the conjugate gradient method, this algorithm requires more computation and storage space per iteration, but it usually converges in fewer iterations. Although the BFGS improves the performance of non-smooth optimization to a limited extent, it can still get stuck in local minima. Concerning the above facts, in specific applications, measures such as local minimum detection and random restarts are employed to avoid the algorithm getting stuck.

In the SQP method, the Euclidean distance error between the Cartesian pose of the current configuration and desired Cartesian pose is computed. Moreover, its square is taken as the optimization objective function, subject to some bounding constraints. The 6-element twist vector $T_{cur}$ is used as the metric to measure the distance error and angular error in Cartesian space, and the sum of squares of each element in $T_{cur}$ is minimized in the SQP method. The objective function is minimized as

$$\arg \min (T_{cur} - T_d)^T (T_{cur} - T_d)$$  \hspace{1cm} (15)$$

where $T_{cur} = \text{FK}(q_{cur}), s.t. b_l \leq q_{ur} \leq b_u, b_l$ is the lower joint displacement limit, and $b_u$ is the upper joint displacement limit. For the optimization process, the SQP method uses a BFGS update of the B-matrix and an L1-test function in the step-length to iteratively search the IK solution.\textsuperscript{27}

The SQP method is more robust than the LMA in finding IK solutions and is more effective when the configurations are close to joint limits or when the initial guess is far from the solution. It utilizes the gradient of the cost function from previous iterations to compute approximate second-derivative values. This value determines the steps to take in the current iteration. The gradient projection method is used to deal with the boundary constraints of the cost function constructed from the upper and lower joint limits of the robot model. By constantly correcting the calculated direction, the search direction is always valid. If the initial guess is close to the solution, the SQP method can quickly find the IK solution.

**Speedup method based on sampled data**

For 7-DOF redundant robots, the IKfast approach needs one joint fixed and to be free to change value according to Algorithm 1 when failure happens, then to solve the displacements of the other joints. With respect to the initial joint value of the fixed joint, the IKfast approach may fail. If it happens, an increment will add to the initial value until the IK solution succeeds in the running time limit. As to the three numerical IK solutions, a set of initial joint values, such as current or nominal joint values, is taken as a seed to compute Jacobian or the pose error. IK solvers require several iterations before they find the IK solution or fail because of singular configurations or the arrival of end conditions. To reach a particular precision, this kind of approach is computationally expensive in many cases, which can be numerically unstable. In view of the situation mentioned above, we propose a method to speed up solving the IKP based on sampled data which are computed using the forward kinematics equation under certain conditions.

After computing by the forward kinematics formula, the end-effector has a unique position and orientation corresponding to a given robot configuration. By a kind of random number generation method, a large number of random joint coordinates can be generated in the joint space of the robot. After mapping them to the operating space, each
group of joint values and the corresponding pose of the end-effector are recorded according to a specific filtering mechanism. In this way, we can obtain a large amount of sampled data. Choosing data among them as a seed state can improve the analytical method or numerical methods for IK solutions. These sampled data can be obtained by following the steps below.

1. To set several parameters:
   - **Maximum of samples** — it will have an effect on the performance of this proposed method.
   - **Minimum distance of configuration** — it decides the sampling density; if a newly generated random configuration is very close to the one already existing in the sample data, it will be discarded.
   - **Minimum distance of pose** — it also decides the sampling density; if the distance between the computed pose of the end-effector and each one in the sample data is less than the parameter, it will be discarded.

2. Generating a random joint configuration, if it is self-collision-free and satisfies the conditions in step (1), its corresponding pose will be computed. If the result satisfies the conditions in step (1), the joint configuration and its corresponding pose will be recorded as an entry.

3. Repeating step (2) until the number of sampled data gets to the limit.

After obtaining enough sample data, the next move is how to employ them. No matter which IK solver is used, these sampled data are read at first. Then, according to the desired task pose, the speedup method searches for the nearest pose in the sampled data. When the nearest entry is found, the IK solver carries on using the joint values in the searched entry as the seed state. As a result, the IK solver can perform better than the conventional way. An example of employing the speedup method for a random joint state is shown in Appendix 1.

In rare cases, our speedup method would fail to solve the IKP. If it happens, one measure is to randomly generate a set of angle values within the range of all joints. Then taking them as a new initial guess for the four original methods, the IK solvers start a new iteration.

### Results and discussion

In order to evaluate the performance of our IK solution speedup method, a simulation test is designed to solve the IKP for the Baxter robot, which has two 7-DOF manipulators. The test is executed on an Ubuntu Linux PC with Intel® Core™ i7-4712MQ@2.3GHz and 16G memory.

After analyzing the URDF files in the robot’s control system, the D-H parameters are worked out. When taking the upper shoulder link as the base link, the left arm and right arm of the robot have the same D-H parameters, and their installation positions have the same coordinate transformation, \(\text{trans}(0.055695, 0, 0.011038)\text{rotz}(0)\text{roty}(0)\text{rotz}(0)\). In this test, the SI unit of angle is rad and the SI unit of length is m. The left arm is taken as the test object, and its D-H parameters by Craig’s D-H parameter convention\(^{28}\) are listed in Table 1, and Figure 1 shows its corresponding D-H parameter motion model.

#### Table 1. D-H parameters of Baxter kinematics model.

| Link | \(a_{i-1}\) (rad) | \(d_{i}\) (m) | \(\theta_{i}\) (rad) | Offset (rad) |
|------|-------------------|---------------|---------------------|--------------|
| 1    | 0                 | 0             | \(\theta_1\) 0.27035 | 0            |
| 2    | \(\pi/2\)         | 0.069         | \(\theta_2\) \(\pi/2\) |              |
| 3    | \(\pi/2\)         | 0             | \(\theta_3\) 0.36442 | 0            |
| 4    | \(\pi/2\)         | 0.069         | \(\theta_4\) 0      | 0            |
| 5    | \(\pi/2\)         | 0             | \(\theta_5\) 0.37429 | 0            |
| 6    | \(\pi/2\)         | 0.01          | \(\theta_6\) 0      | 0            |
| 7    | \(\pi/2\)         | 0             | \(\theta_7\) 0.115975| 0            |

D-H: Denavit–Hartenberg.

#### Figure 1. Kinematics model of Baxter’s left arm.
The primitive analytical and numerical IK solvers are IKfast, NR, LMA, and SQP. The speedup versions are IKfast-SP, NR-SP, LMA-SP, and SQP-SP. Between the joint limit ranges, 100,000 random configurations of the left arm are generated, and their corresponding poses are computed by forward kinematics. These IK solvers search the IK solution for the 100,000 poses. In the IKfast solver, the free joint is the sixth joint, and the initial value is zero. The initial value is zero for all joints in the three numerical solvers: NR, LMA, and SQP. The sampled data size used for the speedup method is 1000, 5000, 10,000, and 50,000 in the four speedup IK solvers: IKfast-SP, NR-SP, LMA-SP, and SQP-SP. The value of timeout is 0.01s for each IK solver. Average cost time and success rate per IK solver call are measured, and the results are shown in Tables 2 and 3. We visualize the summary statistics of the running time with a box plot, as shown in Figures 2 to 5. In Figures 2 to 5, when the sampled data size is zero, it means that the IK solvers use the zero configuration or a random joint state as the initial guess.

In Tables 2 and 3, the data size is the number of candidate seed states for the IK solvers. The last line is the average cost time of the primitive IK solvers in Table 2. The IKfast solver has the longest running time, up to 14.1271 ms. When the sampled data size is 10,000, the IKfast-SP, NR-SP, and LMA-SP solvers have the best performance. However, for the SQP-SP solver, the average cost time is the smallest when the sampled data size is 1000. In terms of the success rate, there is no significant

### Table 2. Average cost time per IK solver call (ms).

| Data size | IKfast-SP | NR-SP | LMA-SP | SQP-SP |
|-----------|-----------|-------|--------|--------|
| 1000      | 2.3028    | 0.8537| 0.8238 | 0.9989 |
| 5000      | 2.2739    | 0.8316| 0.8412 | 1.0888 |
| 10,000    | 2.0946    | 0.8196| 0.8370 | 1.1192 |
| 50,000    | 2.5508    | 0.8979| 0.9082 | 1.2912 |
| IKfast    | 14.1271   | 1.5437| 1.1671 | 1.2721 |

IK: inverse kinematics; LMA: Levenberg–Marquardt algorithm; SQP: sequential quadratic programming; NR: Newton–Raphson.

### Table 3. Success rate per IK solver call (%).

| Data size | IKfast-SP | NR-SP | LMA-SP | SQP-SP |
|-----------|-----------|-------|--------|--------|
| 1000      | 91.77     | 99.10 | 99.39  | 98.84  |
| 5000      | 92.72     | 99.19 | 99.32  | 98.73  |
| 10,000    | 94.92     | 99.20 | 99.42  | 98.79  |
| 50,000    | 94.43     | 99.23 | 99.40  | 98.56  |
| IKfast    | 74.53     | 98.92 | 99.28  | 98.96  |

IK: inverse kinematics; LMA: Levenberg–Marquardt algorithm; SQP: sequential quadratic programming; NR: Newton–Raphson.
change in the NR-SP, LMA-SP, and SQP-SP and their primitive versions. However, the IKfast-SP solver promotes the percentage compared to the IKfast.

The zero configuration is taken as the seed state for the four primitive IK solvers in the simulation. If the joint distance is far from the desired configuration for a random robot configuration, the numerical IK solvers will execute more iterations before they find the IK solutions. In addition, using the default initial guess, the NR and LMA solvers may fail, and in this case, a new joint state will be generated at random and taken as the seed state. However, the speedup method provides a closer configuration as the seed state and results in fewer iterations, as shown in Table 4. As for the IKfast solver, the initial value assigned to the free joint is not possibly suitable or is far from the theoretical value. The IKfast solver spends more time with several iterations to change the value of the free joint before solving the IKP, but in most cases, the speedup method chooses a better initial value. Due to the timeout being set to a short time, for example, 0.01 s, the primitive IK solvers will fail before the convergence. In Tables 2 and 4, it is obvious that the IKfast-SP, NR-SP, LMA-SP, and SQP-SP solvers are more efficient in solving the IKP than the primitive ones.

Conclusion

To improve analytical and numerical IK solutions, we propose a speedup method. The speedup method performs random sampling many times in the configuration space, saves those configurations that satisfy the given constraints and their corresponding pose data, and generates a database file with a designed data structure. For a desired end-effector pose, the method searches the pose data loaded into memory according to the proximity principle until the nearest pose is found. Then, it uses the corresponding joint values of the nearest pose as the initial guess.

We have three conclusions: (1) if the original IK solver is slow, significant improvements can be obtained with the speedup method; (2) if the original IK solver is fast, then the speedup method does not improve performance much and can make it worse; and (3) for the analytical method, IKfast, the speedup method can acquire better success rate. In addition, performance can potentially be improved by tuning sampled data sizes. The result in the third section verifies that the proposed method promotes the performance of the primitive IK solvers.

Our speedup method is to choose a better initial guess for four IK methods to solve the IKP of 7-DOF robots, and we do not consider the singular problem for the time being. Of course, the speedup method is applicable to modify the numerical approach29 for 6-DOF robots, but research on whether it has the advantage over the closed-form methods30 needs to be done in the future. In addition, we do not test whether the speedup method can be applied to parallel31 and hybrid robots.32,33 Much more work should be done before the method can solve the IKP of parallel and hybrid robots.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1310201.

Table 4. An IK example for performance comparison.

| IK solvers | Iteration | Time (ms) | Error | J0  | J1  | J2  | J3  | J4  | J5  | J6  |
|------------|-----------|-----------|-------|-----|-----|-----|-----|-----|-----|-----|
| IKfast     | 9         | 3.378     | 7.74E-06 | 0.014934 | -0.110798 | -2.138208 | 2.023885 | -0.120769 | -0.120000 | -0.209480 |
| IKfast-SP  | 1         | 0.496     | 7.74E-06 | -0.576637 | -0.490358 | -2.723317 | 2.116051 | -1.088432 | -0.767574 | -1.357421 |
| LMA        | 5         | 3.181     | 2.51E-05 | 0.197011 | -0.502955 | -1.991701 | 2.003437 | 1.043405 | -0.352746 | -1.699305 |
| LMA-SP     | 1         | 0.769     | 3.44E-05 | 0.262207 | -0.671714 | -1.935135 | 1.991724 | 1.085539 | -0.492178 | -1.879615 |
| NR         | 2         | 17.100    | 7.26E-07 | -0.968328 | -1.776231 | 0.044129 | 2.159070 | -2.203004 | 1.342319 | 0.573050 |
| NR-SP      | 1         | 2.732     | 1.76E-06 | 0.262249 | -0.671822 | -1.935098 | 1.991708 | 1.085634 | -0.492262 | -1.879798 |
| SQP        | 1         | 3.283     | 1.82E-06 | 0.035950 | -0.149763 | -2.124205 | 2.020219 | 0.132647 | -0.119594 | -0.497306 |
| SQP-SP     | 1         | 0.593     | 6.86E-07 | 0.263031 | -0.673925 | -1.934434 | 1.991556 | 1.085758 | -0.493992 | -1.881667 |

IK: inverse kinematics; LMA: Levenberg–Marquardt algorithm; SQP: sequential quadratic programming; NR: Newton–Raphson.
References

1. Beeson P and Ames B. TRAC-IK: an open-source library for improved solving of generic inverse kinematics. In: Proceedings of the IEEE RAS humanoids conference, Seoul, South Korea, 3–5 November 2015, pp. 928–935. DOI: 10.1109/HUMANOID.2015.7363472.

2. Mokhtari A, Eisen M, and Ribeiro A. An incremental quasi-Newton method with a local superlinear convergence rate. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017, pp. 4039–4043. DOI: 10.1109/ICASSP.2017.7952915.

3. Funda J and Paul R. A computational analysis of screw transformations in robotics. IEEE Trans Robot Autom 1990; (6(3)): 348–356.

4. Ayiz C and Kucuk S. The kinematics of industrial robot manipulators based on the exponential rotational matrices. In: 2009 IEEE international symposium on industrial electronics, Seoul, South Korea, 5–8 July 2009, pp. 977–982. IEEE.

5. Giannou M, Maric F, Rosen DM, et al. Convex iteration for distance-geometric inverse kinematics. IEEE Robot Autom Lett 2022; (7(2)): 1952–1959.

6. Aydin Y and Kucuk S. Quaternion based inverse kinematics for industrial robot manipulators with Euler wrist. In: 2006 IEEE international conference on mechatronics, Budapest, Hungary, 3–5 July 2006, pp. 581–586. IEEE.

7. Huang W, Tan S, and Li X. Inverse kinematics of compliant manipulator based on the immune genetic algorithm. In: 2008 fourth international conference on natural computation, volume 4, Jinan, China, 18–20 October 2008, pp. 390–394. DOI: 10.1109/ICNC.2008.631.

8. Duleba I and Opalka M. A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators. Int J Appl Math Comput Sci 2013; 23(2): 373–382.

9. Ratajczak J. Design of inverse kinematics algorithms: extended Jacobian approximation of the dynamically consistent Jacobian inverse. Arch Contr Sci 2015; 25(1): 35–50.

10. Chiachio P, Chiaverini S, Sciavicco L, et al. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. Int J Robot Res 1991; 10(4): 410–425.

11. Zhgal H, Dubey R, and Euler J. Efficient gradient projection optimization for manipulators with multiple degrees of redundancy. In: Proceedings, IEEE international conference on robotics and automation, volume 2, Cincinnati, OH, USA, 13–18 May 1990, pp. 1006–1011. DOI: 10.1109/ROBOT.1990.126123.

12. Buss SR. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. IEEE J Robot Automat 2004; 17(1–19): 16.

13. Chan TF and Dubey R. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. IEEE Trans Robot Automat 1995; 11(2): 286–292.

14. Dimovski I, Trompesa M, Samak S, et al. Algorithmic approach to geometric solution of generalized Paden–Kahan subproblem and its extension. Int J Adv Robot Syst 2018; 15(1): 1729881418755157.

15. Oh J, Bae H, and Oh JH. Analytic inverse kinematics considering the joint constraints and self-collision for redundant 7DOF manipulator. In: 2017 first IEEE international conference on robotic computing (IRC), Taichung, Taiwan, 10–12 April 2017, pp. 123–128. DOI: 10.1109/IRC.2017.46.

16. Gong M, Li X, and Zhang L. Analytical inverse kinematics and self-motion application for 7-DOF redundant manipulator. IEEE Access 2019; 7: 18662–18674.

17. Hemami A. A more general closed-form solution to the inverse kinematics of mechanical arms. Adv Robot 1987; 2(4): 315–325.

18. Diankov R. Automated construction of robotic manipulation programs. PhD Thesis, Carnegie Mellon University, 2010.

19. Manocha D and Zhu Y. A fast algorithm and system for the inverse kinematics of general serial manipulators. In: Proceedings of the 1994 IEEE international conference on robotics and automation, San Diego, CA, USA, 8–13 May 1994, pp. 3348–3353. IEEE.

20. Manocha D and Canny JF. Real time inverse kinematics for general 6R manipulators. In: Proceedings 1992 IEEE international conference on robotics and automation, volume 1, Nice, France, 12–14 May 1992, pp. 383–389. DOI: 10.1109/ROBOT.1992.220309.

21. Sugihara T. Solvability-unconcerned inverse kinematics based on Levenberg-Marquardt method with robust damping. In: 2009 9th IEEE-RAS international conference on humanoid robots, Paris, France, 7–10 December 2009, pp. 555–560. IEEE.

22. Wilamowski BM and Yu H. Improved computation for Levenberg–Marquardt training. IEEE Trans Neural Networks 2010; 21(6): 930–937.

23. Luh J, Walker M, and Paul R. Resolved-acceleration control of mechanical manipulators. IEEE Trans Autom Control 1980; 25(3): 468–474.

24. Nakamura Y and Hanafusa H. Inverse kinematic solutions with singularity robustness for robot manipulator control. J Dyn Syst Meas Contr 1986; 108(3): 163–171.

25. Badreddine H, Vandewalle S, and Meyers J. Sequential quadratic programming (SQP) for optimal control in direct numerical simulation of turbulent flow. J Comput Phys 2014; 256: 1–16.

26. Kou CX and Dai YH. A modified self-scaling memoryless Broyden–Fletcher–Goldfarb–Shanno method for unconstrained optimization. J Optim Theory Appl 2015; 165(1): 209–224.

27. Krafft D. A software package for sequential quadratic programming. Technical Report 88-28, Institut für Dynamik der Flugsysteme, Obersaffaihofen, Germany, 1988.

28. Craig JJ. Introduction to robotics: mechanics and control. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

29. Hayat AA, Sadanand ROM, and Saha SK. Robot manipulation through inverse kinematics. In: Proceedings of the 2015 conference on advances in robotics, Goa, India, July 2015, pp.1–6. DOI: 10.1145/2783449.2783497.
30. Tian X, Xu Q, and Zhan Q. An analytical inverse kinematics solution with joint limits avoidance of 7-DOF anthropomorphic manipulators without offset. *J Franklin Inst* 2021; 358(2): 1252–1272.

31. Kucuk S. Simulation and design tool for performance analysis of planar parallel manipulators. *Simulation* 2012; 88(5): 542–556.

32. Tanev TK. Kinematics of a hybrid (parallel–serial) robot manipulator. *Mech Mach Theory* 2000; 35(9): 1183–1196.

33. Kucuk S and Gungor BD. Inverse kinematics solution of a new hybrid robot manipulator proposed for medical purposes. In: *2016 medical technologies national congress (TIPTENKO)*, Antalya, Turkey, 27–29 October 2016, pp. 1–4. IEEE.

Appendix 1

An IK example for performance comparison

A random joint state is generated for the right arm of the Baxter robot, and the configuration is $(1.128003, -1.696832, -2.166943, 1.811817, 0.795480, -1.238112, -2.580727)$. The position of the end-effector is $(-0.092606, -0.703302, 0.679877)$, and the orientation of the end-effector is represented as Z-Y-X Euler angles $(1.938044, -0.007836, -1.098123)$. The quaternion representation according to the Euler angles is $[0.484579, -0.292686, -0.432097, 0.702005]$. Employing the four IK solvers and their speedup versions solvers to solve the IKP of the random pose, and solution results are listed in Table 4.

In Table 4, the iteration is the number of calls of an IK solver before it succeeds in solving the IKP, and the time is the total running cost of an IK solver. The seven joints are labeled as J0, J1, J2, J3, J4, J5, and J6, and the unit of angle is rad. The error is the Euclidean norm of differential motion corresponding to infinitesimal motion from the computed pose to the desired pose. The SQP and SQP-SP solvers use an optimization function to search the IK solution only once according to the seed state.