Int. J. Appl. Math. Comput. Sci., 2013, Vol. 23, No. 2, 373–382
DOI: 10.2478/amcs-2013-0028

A COMPARISON OF JACOBIAN–BASED METHODS OF INVERSE KINEMATICS FOR SERIAL ROBOT MANIPULATORS

IGNACY DULĘBA, MICHAŁ OPAŁKA

Institute of Computer Engineering, Control and Robotics
Wrocław University of Technology, ul. Janiszewskiego 11/17, 50-372 Wrocław, Poland
e-mail: {ignacy.duleba,michal.opalka}@pwr.wroc.pl

The objective of this paper is to present and make a comparative study of several inverse kinematics methods for serial manipulators, based on the Jacobian matrix. Besides the well-known Jacobian transpose and Jacobian pseudo-inverse methods, three others, borrowed from numerical analysis, are presented. Among them, two approximation methods avoid the explicit manipulability matrix inversion, while the third one is a slightly modified version of the Levenberg–Marquardt method (mLM). Their comparison is based on the evaluation of a short distance approaching the goal point and on their computational complexity. As the reference method, the Jacobian pseudo-inverse is utilized. Simulation results reveal that the modified Levenberg–Marquardt method is promising, while the first order approximation method is reliable and requires mild computational costs. Some hints are formulated concerning the application of Jacobian-based methods in practice.

Keywords: serial manipulator, inverse kinematics, Jacobian methods, comparison, applications.

1. Introduction

Inverse kinematics are the most frequently solved robotic tasks (in fact, forward and inverse tasks are encountered in many domains of science (Hunek and Latawiec, 2011) and various robotic sub-domains (Dulęba and Jagodziński, 2011)). For open-chain serial manipulators, considered in this paper, whose forward kinematics are given analytically, the inverse kinematics task is to find such a configuration at which the end-effector of the robot reaches a given point in the taskspace.

Due to the non-linearity of forward kinematics, direct and analytic computations of inverse kinematics are hardly ever possible. Therefore, some numerical approaches have been developed in robotics to solve the task. The most prominent among them are based on the Jacobian matrix, which describes the transformation between velocities in the configuration and the taskspaces.

There are many versions of Jacobian-based methods (Nakamura, 1991), formulated as Newton algorithms, which differ in computational complexity, the speed of convergence, the way of passing through singular configurations (Nakamura, 1991; Dulęba and Sasiadek, 2002), and other features (repeatability (Tchoń *et al*., 2009), optimality (Nakamura, 1991)). Some researchers considered the inverse kinematic task even in the case when forward kinematics are not available and the inverse transformation has to be learned (D'Souza *et al*., 2001). Inverse kinematics can be also viewed as an optimization task solved with general-purpose methods (neural networks (Tejomurtula and Kak, 1999), genetic algorithms (Nearchou, 1998), etc.), but those approaches are usually computationally ineffective.

The Jacobian-based algorithms search for a configuration that solves the task by means of an iterative process initialized at an assumed initial configuration. While solving a given inverse kinematics task, various Jacobian-based methods can be compared with each other by counting the number of iterations needed to complete the inverse task multiplied by the computational cost of a single iteration. However, this comparison is not reliable because different methods, although initialized at the same configuration, may generate trajectories significantly different from each other as the computations progress. Moreover, the number of iterations required to finish a given algorithm may be strongly influenced by consecutive configurations of the generated trajectory because the Jacobian matrix is a function of the current configuration. Therefore, in this paper, Jacobian-based methods will be evaluated locally, around selected configurations.

The main problem addressed in this paper concerns

the local convergence of selected Jacobian-based methods. In particular, a convergence measure will be defined. Based on this measure, a question concerning the convergence of the tested method for a given manipulator at a given configuration will be answered. As the reference method, the Jacobian pseudo-inverse method is selected because, locally, it generates a straightforward motion towards the goal with a minimal configuration effort. A natural angle-measure will be defined to relate the motion towards the goal provided by the Jacobian pseudo-inverse method with the motion offered by a method being evaluated.

The paper is organized as follows. In Section 2, Jacobian-based methods are presented, and a measure used to evaluate their speed of convergence is defined. Roboticians are quite familiar with two of them, while the remaining three methods have been transferred to the inverse kinematic task from numerical analysis. The first one was borrowed from the robust-inverse kinematic task solved by the Levenberg–Marquardt method (Marquardt, 1963). Exploiting the analogy between the Jacobian transposed method and the singularity robust inverse method in its basic form (Nakamura, 1991), a modification of the Levenberg–Marquardt method is designed to avoid the computationally involved matrix inversion. The idea of two other methods is rooted in numerical procedures of matrix inversion. Some time ago, researchers (Ben-Isreal and Cohen, 1966) noticed that the inversion can be computed as a limit of an appropriately defined iteration process that avoids explicit matrix inversion. Although the process may require many iterations to converge, we propose to use either one or two initial iterations to get a rough matrix inverse estimation. In Section 3, simulations of Jacobian-based methods of inverse kinematics carried out on four models of redundant manipulators (including two industrial robots) are presented. Section 4 concludes the paper.

2. Jacobian-based methods of inverse kinematics and their evaluation

The forward kinematics for serial robot manipulators are a mapping

$$\begin{aligned} &\mathbb{Q} \ni \boldsymbol{q} \rightarrow \boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{x} \in \mathbb{X} \subset \mathbb{SE}(3), \\ &\dim(\boldsymbol{q}) = n \geq m = \dim(\boldsymbol{x}), \end{aligned} \tag{1}$$

where $\boldsymbol{q}$ is a configuration living in the configuration space $\mathbb{Q}$, $\boldsymbol{x}$ is a generalized location in the taskspace $\mathbb{X}$ and $\mathbb{SE}(3)$ denotes the special Euclidean group. For a given point $\boldsymbol{x}_f$ in the taskspace, the inverse kinematics algorithms search for such a configuration $\boldsymbol{q}^\star$ that $\boldsymbol{k}(\boldsymbol{q}^\star) = \boldsymbol{x}_f$. Jacobian-based methods of solving inverse kinematics are derived by differentiating Eqn. (1) with respect to time,

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q}) \dot{\boldsymbol{q}}, \tag{2}$$

and approximating infinitesimal $\dot{\boldsymbol{q}}$, $\dot{\boldsymbol{x}}$ by their finite equivalents, $(\boldsymbol{q}_{i+1} - \boldsymbol{q}_i)/\Delta t$, $\xi_i(\boldsymbol{x}_f - k(\boldsymbol{q}_i))/\Delta t$, respectively. It is assumed that an initial configuration $\boldsymbol{q}_0$ is known.

To get $\boldsymbol{q}_{i+1}$ as a function of the data from the i-th iteration, some kind of inversion of the Jacobian matrix $\boldsymbol{J}$ has to be applied. The most frequently encountered in practice is the pseudo-inverse matrix (Moore–Penrose (Nakamura, 1991)) defining a process whose iterations are counted by the variable i,

$$\begin{aligned} \boldsymbol{q}_{i+1} &= \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i) \boldsymbol{M}(\boldsymbol{q}_i)^{-1} (\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)) \\ &= \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^{\#}(\boldsymbol{q}_i) (\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)), \end{aligned} \tag{3}$$

where the superscript 'T' introduces the matrix transposition, ξ_i is a given (or appropriately varied from one iteration to another) positive coefficient,

$$\boldsymbol{M}(\boldsymbol{q}_i) = \boldsymbol{J}(\boldsymbol{q}_i) \boldsymbol{J}^T(\boldsymbol{q}_i) \tag{4}$$

denotes an $(m \times m)$ non-negative definite, symmetric manipulability matrix, and the initial value of $\boldsymbol{q}_0$ is given.

The pseudo-inverse $\boldsymbol{J}^{\#} = \boldsymbol{Y}$ displays all four axioms of generalized matrix inverses for a real-valued matrix $\boldsymbol{J}$ (Ben-Isreal and Greville, 2003),

$$\begin{aligned} \boldsymbol{JYJ} &= \boldsymbol{J}, & \boldsymbol{YJY} &= \boldsymbol{Y}, \\ (\boldsymbol{JY})^T &= \boldsymbol{JY}, & (\boldsymbol{YJ})^T &= \boldsymbol{YJ}. \end{aligned} \tag{5}$$

Very similar to the pseudo-inverse is the adjoint pseudo-inversion $\mathrm{adj}(\boldsymbol{M}) = \det(\boldsymbol{M})(\boldsymbol{M})^{-1}$ that excludes the determinant of the manipulability matrix $\boldsymbol{M}$ (Tchoń and Dulęba, 1993) to partially avoid the problem of singular configurations. The Jacobian pseudo-inverse matrix inversion $\boldsymbol{J}^{\#}$ is usually quite complex to compute, as it requires the manipulability matrix inversion. Therefore, other methods of inversion have been invented.

Singularity robust inverses (Nakamura, 1991) are used to cope with singular configurations at which $\det(\boldsymbol{M})$ takes the value of zero, and Eqn. (3) fails to generate consecutive configurations. In order to retrieve invertibility in a close vicinity of singular configurations, a diagonal matrix $\boldsymbol{A}$ scaled with a small positive coefficient λ is usually added to the manipulability matrix $\boldsymbol{M}$,

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i) (\boldsymbol{M}(\boldsymbol{q}_i) + \lambda \boldsymbol{A})^{-1} (\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)). \tag{6}$$

In the Levenberg version (Levenberg, 1944), $\boldsymbol{A}$ is equal to the $(m \times m)$ identity matrix $\boldsymbol{I}_m$, while in the Levenberg–Marquardt (Marquardt, 1963) version $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{M})$, where $\mathrm{diag}(\boldsymbol{M})$ denotes an $(m \times m)$ matrix composed of main diagonal elements of $\boldsymbol{M}$.

Two computationally cheap algorithms can be obtained when the manipulability matrix is omitted from Eqn. (6) and coefficient λ is merged into new $\xi_i = \xi_i^{\mathrm{old}} \cdot \lambda$: a well-known Jacobian transpose inverse method

introduced into robotics by Chiacchio and Siciliano (1989),

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)), \tag{7}$$

and the modified Levenberg–Marquardt one (mLM),

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)(\mathrm{diag}(\boldsymbol{M}(\boldsymbol{q}_i))^{-1}(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)). \tag{8}$$

The last two Jacobian-based methods of inverse kinematics studied in this paper result from an approximation of the Jacobian pseudo-inverse matrix $\boldsymbol{J}^{\#}$ proposed by Ben-Isreal and Cohen (1966), computed by the recursive formula

$$\begin{gathered}\boldsymbol{Y}_0 = \alpha \boldsymbol{J}^T, \qquad \boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k(2\boldsymbol{I}_m - \boldsymbol{J}\boldsymbol{Y}_k), \\ \alpha \in \left(0, \frac{2}{\lambda_{\max}(\boldsymbol{M})}\right),\end{gathered} \tag{9}$$

where $\lambda_{\max}(\boldsymbol{M})$ is the largest singular value of the manipulability matrix $\boldsymbol{M}$ and the parameter α influences the speed of convergence of the scheme (9). It was proven (Ben-Isreal and Cohen, 1966) that $\boldsymbol{Y}_k \to \boldsymbol{J}^{\#}$ as $k \to \infty$.

There exist other numerical methods of matrix inversion based on a factorization of the matrix (for example, singular value decomposition (Maciejewski and Klein, 1989) or LU factorization (Golub and Van Loan, 1996)). As a rule they provide an easy way to get matrix inversion, if only factorization has been performed. However, matrix factorization is almost as computationally complicated as inverting the matrix itself. Some specialized methods of matrix inversion can be applied to matrices in a special form (block, sparse, with a significant rank deficit), but Jacobian matrices of inverse kinematics do not belong to those matrix sub-classes.

Now, let us turn back to Eqn. (9). The singular value decomposition method allows computing all singular values of the manipulability matrix. However, the decomposition is computationally involved, so the explicit derivation of $\lambda_{\max}$ is not computationally justified as a preliminary step to set α in Eqn. (9). Therefore, $\lambda_{\max}$ will be estimated, rather than determined explicitly. To speed up the convergence of the process (9), the estimated value of $\lambda_{\max}$ should be as close to its real value as possible, and α as large as possible.

Fortunately, a reliable estimation of $\lambda_{\max}$ is possible with little computational effort. Singular values of a given symmetric and non-negative defined matrix $\boldsymbol{M}$ are non-negative and satisfy the equality (Horn and Johnson, 1986)

$$\sum_{i=1}^{m} \lambda_i(\boldsymbol{M}) = \mathrm{tr}(\boldsymbol{M}) = \sum_{i=1}^{m} M_{ii}. \tag{10}$$

Equation (10) allows setting the range for possible values of $\lambda_{\max}(\boldsymbol{M})$:

$$\frac{1}{m}\mathrm{tr}(\boldsymbol{M}) \le \lambda_{\max}(\boldsymbol{M}) \le \mathrm{tr}(\boldsymbol{M}). \tag{11}$$

The left equality in Eqn. (11) covers the case when all singular values are identical, while the right equality corresponds to the case when there is only one non-zero singular value.

To keep a mild computational complexity, only the first and the second (iterations $k = 1, 2$) order approximation of $\boldsymbol{J}^{\#}$ given by (9) will be used later on. The first order approximation of $\boldsymbol{J}^{\#}$ is defined by the following iterative scheme:

$$\begin{aligned}\boldsymbol{q}_{i+1} &= \boldsymbol{q}_i \\ &+ \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)\left(\boldsymbol{I}_m - \frac{\alpha(\boldsymbol{q}_i)}{2}\boldsymbol{M}(\boldsymbol{q}_i)\right)(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)),\end{aligned} \tag{12}$$

when $\boldsymbol{Y}_1$ from Eqn. (9) substitutes $\boldsymbol{J}^{\#}$ in Eqn. (3) and ξ_i includes also the $2 \cdot \alpha(\boldsymbol{q}_i)$ term. The second order approximation of $\boldsymbol{J}^{\#}$ generates the inversion algorithm

$$\begin{aligned}&\boldsymbol{q}_{i+1} \\ &= \boldsymbol{q}_i + \xi_i \cdot \boldsymbol{J}^T(\boldsymbol{q}_i)(\boldsymbol{I}_m - \frac{\alpha(\boldsymbol{q}_i)}{2}\boldsymbol{M}(\boldsymbol{q}_i)) \\ &\quad \cdot (\boldsymbol{I}_m - \alpha(\boldsymbol{q}_i)\boldsymbol{M}(\boldsymbol{q}_i)(\boldsymbol{I}_m - \frac{\alpha(\boldsymbol{q}_i)}{2}\boldsymbol{M}(\boldsymbol{q}_i))) \\ &\quad \cdot (\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)),\end{aligned} \tag{13}$$

where $\boldsymbol{Y}_2$ replaces $\boldsymbol{J}^{\#}$, and ξ_i includes also the $4 \cdot \alpha(\boldsymbol{q}_i)$ term.

Jacobian-based versions of the Newton algorithm described by Eqns. (7), (8), (12), (13) are computationally cheaper than the original algorithm (3) based on the pseudo-inverse. It should be stressed that the aforementioned algorithms do not display all the properties of the generalized inverse (5) (in fact (7), (12), (13) satisfy the last two properties in (5), while (8) satisfies none of them).

2.1. Evaluation measure of Jacobian-based methods. It is easy to observe a formal similarity of all the Jacobian-based methods given by Eqns. (3), (7), (8), (12), (13). In all of them, $\boldsymbol{J}^T$ is multiplied by a square $(m \times m)$ symmetric matrix and post-multiplied by a vector $(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i))$. Therefore, a quite natural measure serving to compare these methods could be based on any matrix metrics applied to matrices $\boldsymbol{J}^{\#}, \boldsymbol{J}^T, \boldsymbol{J}^T\mathrm{diag}(\boldsymbol{M})^{-1}, \boldsymbol{J}^T(\boldsymbol{I} - \frac{\alpha}{2}\boldsymbol{M})$, $\boldsymbol{J}^T\left(\boldsymbol{I} - \frac{\alpha}{2}\boldsymbol{M}\right)\left(\boldsymbol{I} - \alpha\boldsymbol{M}\left\{\boldsymbol{I} - \frac{\alpha}{2}\boldsymbol{M}\right\}\right)$ or to their variants without trailing $\boldsymbol{J}^T$. However, these metrics do not take into account the real direction to the goal $(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i))$.

As the Jacobian pseudo-inverse method (3) generates a short-time straightforward motion towards the goal in the taskspace, it is a natural candidate to be chosen as the reference method for other tested methods. For the pseudo-inverse method and outside singular

configurations, we define a virtual direction to the goal as

$$\boldsymbol{x}_{\mathrm{ref}} = \boldsymbol{M}(\boldsymbol{q}_i)^{-1}(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)). \tag{14}$$

The virtual direction lives in the same $\mathbb{R}^m$ space as the real direction to the goal $\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)$. The other Jacobian methods will be related to the Jacobian pseudo-inverse one by measuring the angle between $\boldsymbol{x}_{\mathrm{ref}}$ and $\boldsymbol{x}_{\mathrm{met}}$,

$$e(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q}), \boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q})) = \arccos\left(\frac{\langle \boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q}), \boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q})\rangle}{||\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q})|| \cdot ||\boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q})||}\right), \tag{15}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product in $\mathbb{R}^m$, $||\cdot||$ is the Euclidean metrics. $\boldsymbol{x}_{\mathrm{met}}$ derived from Eqns. (7), (8), (12), (13) is given by

$$\boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q}) = \boldsymbol{B} \cdot (\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i)),$$

where

$$\boldsymbol{B} = \begin{cases} \boldsymbol{I}_m \\ (\mathrm{diag}(\boldsymbol{M}(\boldsymbol{q}_i))^{-1} \\ \left(\boldsymbol{I}_m - \frac{1}{2}\boldsymbol{M}_\alpha\right) \\ \left(\boldsymbol{I}_m + \boldsymbol{M}_\alpha\left(-\frac{3}{2}\boldsymbol{I}_m + \boldsymbol{M}_\alpha(\boldsymbol{I}_m - \frac{1}{4}\boldsymbol{M}_\alpha)\right)\right) \end{cases} \tag{16}$$

and $\boldsymbol{M}_\alpha = \alpha(\boldsymbol{q}_i)\boldsymbol{M}(\boldsymbol{q}_i)$. In the last row of Eqn. (16), an equivalent version of (13) is presented. Obviously, $e(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q}), \boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q})) = 0$, $e(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q}), \boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q})) \in [0, \pi]$, and if only $\forall \boldsymbol{q} \in \mathbb{Q} : e(\boldsymbol{x}_{\mathrm{ref}}(\boldsymbol{q}), \boldsymbol{x}_{\mathrm{met}}(\boldsymbol{q})) \in [0, \pi/2)$ the tested method is at least locally convergent, i.e., it generates a motion decreasing the distance to the goal. Intentionally, the angular measure (15) neglects the information about the length of the vector $\boldsymbol{x}_{\mathrm{met}}$. This length is not important as its influence on the local motion can be adjusted by appropriately varying the ξ_i coefficient.

Having recalled Jacobian-based methods of inverse kinematics, and defined a measure allowing to compare them, it is necessary to introduce a methodology of performing the tests. For a given manipulator with given kinematics, let us fix a non-singular configuration $\boldsymbol{q}_0$ and compare its image $\boldsymbol{x}_0 = \boldsymbol{k}(\boldsymbol{q}_0)$ in the taskspace. Around $\boldsymbol{x}_0$ a small radius R ball is fixed that collects all possible infinitesimal directions of motion $\Delta\boldsymbol{x} = \boldsymbol{x}_f - \boldsymbol{x}_0$.

Notice that the short distance of R is not particularly important as the measure (15) is insensitive to the length of $\Delta\boldsymbol{x}$. On the surface of the ball, select a set of uniformly distributed goal points $\boldsymbol{x}_f$. For each of the targets $\boldsymbol{x}_f$ and each of the tested methods, Eqns. (14), (16) determine the measure (15). Physically, this simulates a single step motion towards $\boldsymbol{x}_f$ of the tested method. Results are collected and processed statistically to get reliable characteristics at $\boldsymbol{q}_0$. The following characteristics are used (N is the number of goal points, $i = 1, \ldots, N$):

- the histogram of $e(i)$ (the range $[0, \pi] \ni e(i)$ is split into K equi-length sub-intervals and frequencies of entering each sub-interval are found);
- the mean angular error value,

$$\overline{e} = \frac{1}{N}\sum_{i=1}^{N} e(i);$$

- the standard deviation,

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(e(i) - \overline{e})^2};$$

- the maximal angular value,

$$e_{\max} = \max_{i=1}^{N} e(i);$$

- the percentage of trails that generate angular error larger than $90°$.

Computer simulations carried out on some models of manipulators should answer two basic questions: how the characteristics depend on the kinematics $\boldsymbol{k}$ and the configuration $\boldsymbol{q}_0$ under evaluation, and whether there exist configurations at which the measure (15) attains values higher than $90°$, which means that the convergence is locally lost.

2.2. Complexity. The presented Jacobian-based methods of inverse kinematics can also be evaluated according to their computational complexity. It is natural to measure the complexity by the number of elementary operations, i.e., the number of additions (subtractions) and multiplications (divisions), and to express these numbers as a function of the dimensionality of the taskspace m. We assume that the complexity of a given method is the same as that of computing the matrix $\boldsymbol{B}$ in Eqn. (16) because other transformations (a pre-multiplication by $\boldsymbol{J}^T$ and a post-multiplication by $(\boldsymbol{x}_f - \boldsymbol{k}(\boldsymbol{q}_i))$ are the same for all methods while computing an inverse kinematic task.

The complexity of the Jacobian pseudo-inverse method is measured by the complexity of computing $\boldsymbol{M}^{-1}$. It was assumed that the computation of the matrix $\boldsymbol{M}$ does not contribute to the overall complexity. Taking into account a straightforward calculation of the matrix determinant and matrix inversion (only for the pseudo-inverse method), multiplication of symmetric matrices, multiplication of a matrix by a scalar and matrix additions, the following complexity (additions $\oplus$ and multiplications $\odot$) has been obtained for the presented

Table 1. Number of elementary operations to compute a crucial part of Jacobian-based methods of inverse kinematics as a function of m: additions (a), multiplications (b).

(a)

method (Eqn.)	m				
	2	3	4	5	6
pseudo-inverse (3)	1	11	73	464	3218
Jacobian transpose (7)	0	0	0	0	0
L–M modified (8)	0	0	0	0	0
1st approximation (12)	2	3	4	5	6
2nd approximation (13)	12	33	72	135	228

(b)

method (Eqn.)	m				
	2	3	4	5	6
pseudo-inverse (3)	5	30	202	1575	13701
Jacobian transp. (7)	0	0	0	0	0
L–M modified (8)	2	3	4	5	6
1st approx. (12)	3	6	10	15	21
2nd approx. (13)	15	42	90	165	273

methods:

- Jacobian pseudo-inverse,
$$\left(\frac{(m+1)!}{2} - \frac{m(m+1)}{2} + m! - 1\right) \oplus + \left(\frac{(m+1)!(m-2)}{2} + \frac{m(m+1)}{2} + m!(m-1)\right) \odot;$$
- Jacobian transpose, $0 \oplus +0 \odot$;
- modified Levenberg–Marquardt, $0 \oplus +m \odot$,
- 1st approx., $m \oplus +\dfrac{m(m+1)}{2} \odot$;
- 2nd approx., $m(m^2+2) \oplus +\dfrac{m(m+1)(2m+1)}{2} \odot$. (17)

Numerical values of complexity of Jacobian-based methods given by Eqn. (17) for some taskspaces encountered in practice are presented in Table 1.

3. Simulations

Simulations were carried out on models of three robotic manipulators. If possible, their geometric parameters correspond to manufactured robots, while their taskspaces were selected to get redundant kinematics ($\dim \mathbb{Q} > \dim \mathbb{X}$). The following manipulators were tested:

- a 3D(4D) planar pendulum with a two(three)-dimensional taskspace $(x, y)^T$ ($(x, y, \Theta)^T$) and three (four)-dimensional configuration space $\boldsymbol{q} = (q_1, q_2, q_3)^T$ ($\boldsymbol{q} = (q_1, q_2, q_3, q_4)^T$), described by its kinematics
$$\begin{bmatrix} x \\ y \\ \Theta \end{bmatrix} = \boldsymbol{k}(\boldsymbol{q}) = \begin{bmatrix} l_1 c_1 + l_2 c_{12} + l_3 c_{123} (+ l_4 c_{1234}) \\ l_1 s_1 + l_2 s_{12} + l_3 s_{123} (+ l_4 s_{1234}) \\ q_1 + q_2 + q_3 + q_4 \end{bmatrix}, \quad (18)$$
and link-lengths equal to $l_1 = 1$, $l_2 = 2$, $l_3 = 3$, $l_4 = 3$. In Eqn. (18), and later on, a standard robotic convention to denote sine/cosine functions was utilized, $c_1 = \cos(q_1)$, $s_{12} = \sin(q_1 + q_2)$;
- SCARA (Tchoń *et al.*, 2000) with the kinematics
$$\begin{bmatrix} x \\ z \\ \phi \end{bmatrix} = \boldsymbol{k}(\boldsymbol{q}) = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ d_1 + q_3 \\ q_1 + q_2 + q_4 \end{bmatrix}, \quad (19)$$
where $\dim \mathbb{Q} = 4$, $\boldsymbol{q} = (q_1, q_2, q_3, q_4)^T$, $\dim \mathbb{X} = 3$ and real geometric parameters equal $d_1 = 0.8$ [m], $a_1 = 0.445$ [m], $a_2 = 0.355$ [m]. The taskspace includes two positional coordinates (x, z) and one angle coordinate ϕ describing the orientation of the end-effector in the xy plane. The motion of the prismatic joint q_3 was restricted to the range $(0, 0.8)$ [m];
- PUMA's positional kinematics $(x, y, z)^T$ (Lee, 1982),
$$\boldsymbol{k}(\boldsymbol{q}) = \begin{bmatrix} c_1 A - s_1(d_2 + d_6 s_4 s_5) \\ s_1 A + c_1(d_2 + d_6 s_4 s_5) \\ -a_2 s_2 + d_4 c_{23} + d_6(c_5 c_{23} - c_4 s_5 s_{23}) \end{bmatrix},$$
$$A = a_2 c_2 + d_4 s_{23} + d_6(c_4 s_5 c_{23} + c_5 s_{23}), \quad (20)$$
where $\dim \mathbb{Q} = 5$, $\boldsymbol{q} = (q_1, q_2, q_3, q_4, q_5)^T$, $\dim(\mathbb{X}) = 3$ with the following geometric parameters $a_2 = 0.432$ [m], $d_2 = 0.0745$ [m], $d_4 = 0.432$ [m], $d_6 = 0.056$ [m]. To preserve all technical data, the motion ranges for all coordinates were restricted: $q_1 \in [-160°, 160°]$, $q_2 \in [-225°, 45°]$, $q_3 \in [-45°, 225°]$, $q_4 \in [-100°, 100°]$, $q_5 \in [-266°, 266°]$ (Lee, 1982).

In order to get statistically reliable data, in all simulations a set of 1000 non-singular configurations $\boldsymbol{q}_0$ was generated randomly within the configuration space (at a non-singular configuration the determinant of the manipulability matrix should exceed a given threshold). For each configuration a corresponding point in the taskspace was computed, $\boldsymbol{x}_0 = \boldsymbol{k}(\boldsymbol{q}_0)$. Centered at this point, a ball was defined in the taskspace. A set of goal points $\boldsymbol{x}_f$ placed on the surface of the ball was selected.

For each pair $(\boldsymbol{q}_0, \boldsymbol{x}_f)$ and all Jacobian-based methods subject to the tests (coded by data given in Eqn. (16)), the measure (15) was computed and results gathered and processed to get statistical characteristics. Moreover, for the first (12) and the second (13) order approximation of the inverse Jacobian matrix, five values of the parameter α were involved,

$$\begin{cases} \alpha_1(\boldsymbol{q}) = 2/\mathrm{tr}(\boldsymbol{M}(\boldsymbol{q})), \\ \alpha_2(\boldsymbol{q}) = 2m/\mathrm{tr}(\boldsymbol{M}(\boldsymbol{q})), \\ \alpha_3(\boldsymbol{q}) = (m+1)/\mathrm{tr}(\boldsymbol{M}(\boldsymbol{q})), \\ \alpha_4(\boldsymbol{q}) = (m+1)/2(\mathrm{tr}(\boldsymbol{M}(\boldsymbol{q}))), \\ \alpha_5(\boldsymbol{q}) = 2/(\lambda_{\max}(\boldsymbol{M})). \end{cases} \tag{21}$$

The value α_1 corresponds to the right-hand side estimate of $\lambda_{\max}$ in Eqn. (11) and covers the most pessimistic case. α_2 results from the left-hand side estimate of $\lambda_{\max}$ in Eqn. (11) and describes the most optimistic case. α_3 is the average of α_1 and α_2, while α_4 is a dumped version of α_3 with the dumping factor equal to 2. Here α_5 corresponds to the ideal case when the maximal singular value is known (in fact, it was computed numerically for a given manipulator and a given point in the configuration space using the SVD algorithm (Maciejewski and Klein, 1989)). Although α_5 does not have got any practical significance (exact $\lambda_{\max}$ computation is too costly) it may certainly be useful in a comparative study with other values (α_{1-4}). Evaluation of inverse tasks using α_5 allows answering the question how good approximations could be if real values of $\lambda_{\max}$ instead of estimated ones were used.

Because the manipulators have different taskspaces, the distribution of goal points $\boldsymbol{x}_f$ should be adjusted accordingly. For a 3D planar pendulum, the polar coordinates are used,

$$x_f = x_0 + R\cos(\psi), \quad y_f = y_0 + R\sin(\psi), \quad \psi \in [-\pi, \pi), \tag{22}$$

where, here and later on, R denotes a small radius of the ball. In a similar fashion, goal points were generated for the SCARA robot

$$\begin{aligned} &x_f = x_0 + R\cos(\psi), \quad y_f = y_0 + R\sin(\psi), \\ &\phi_f = \phi_0 + \phi, \quad \psi \in [-\pi, \pi), \quad \phi \in [-\pi, \pi), \end{aligned} \tag{23}$$

while for the PUMA manipulator the spherical coordinates are applied,

$$\begin{aligned} &x = R\cos(\varphi)\cos(\psi), \quad y = R\cos(\varphi)\sin(\psi), \\ &z = R\sin(\varphi), \qquad \varphi \in [-\pi, \pi), \quad \psi \in [0, 2\pi). \end{aligned} \tag{24}$$

In all the cases, $R = 0.1$, and mesh points in taskspaces were generated by dividing the ranges of variables ψ, φ with the step $\pi/18$. Statistical data gathered from simulations for 3D-pendulum, SCARA and PUMA robots are collected in Tables 2–4, respectively. Graphics are provided only for the SCARA robot, in Figs. 1–3.

Table 2. Statistical data obtained from 1000 random simulations for the 3D planar pendulum.

method	$e_{\max}$ [°]	$\overline{e}$ [°]	σ [°]	$e > 90^o$ [%]
Jacobian transp.	88.7	36.7	21.1	0.0
L–M modified	153.0	24.9	23.8	2.1
1st approx. α_1	0.0	0.0	0.0	0.0
1st approx. α_2	179.9	53.2	36.7	14.1
1st approx. α_3	179.9	38.8	35.8	9.3
1st approx. α_4	84.8	16.9	14.9	0.0
1st approx. α_5	89.9	8.2	14.9	0.0
2nd approx. α_1	0.0	0.0	0.0	0.0
2nd approx. α_2	179.9	80.0	29.5	2.9
2nd approx. α_3	179.9	55.3	36.3	14.5
2nd approx. α_4	82.6	10.9	11.9	0.0
2nd approx. α_5	89.9	8.2	14.9	0.0

Table 3. Statistical data obtained from 1000 random simulations for the SCARA robot.

method	$e_{\max}$ [°]	$\overline{e}$ [°]	σ [°]	$e > 90^\circ$ [%]
Jacobian transp.	84.5	57.6	15.1	0.0
L–M modified	159.4	28.0	33.7	8.4
1st approx. α_1	79.2	37.5	15.2	0.0
1st approx. α_2	179.9	81.7	26.9	34.0
1st approx. α_3	179.9	66.6	29.3	19.2
1st approx. α_4	79.2	37.5	15.2	0.0
1st approx. α_5	89.9	29.0	18.3	0.0
2nd approx. α_1	76.3	28.7	14.2	0.0
2nd approx. α_2	179.9	96.1	18.1	72.7
2nd approx. α_3	179.9	82.0	26.8	33.9
2nd approx. α_4	76.3	28.7	14.2	0.0
2nd approx. α_5	89.9	21.6	16.3	0.0

It appears that for all manipulators the computationally cheapest Jacobian transpose method always guarantees the convergence ($e_{max} < 90°$), but it is really slow (relatively large values of $\overline{e}$). Only a little bit more computationally expensive is the modified Levenberg–Marquardt method. Statistically, this method is much better that the Jacobian transposed method, but it occasionally loses the convergence property. Therefore, when this method is applied in motion planning, it is advised to check the convergence. When the convergence is lost, for a single iteration, the Jacobian transpose method or any other method preserving convergence should be used. The performance of the first and second order approximation methods strongly depends on the α coefficient used. Its pessimistic value (α_1) always

Table 4. Statistical data obtained from 1000 random simulations for the PUMA robot.

method	$e_{\max}$ [°]	$\overline{e}$ [°]	σ [°]	$e > 90°$ [%]
Jacobian transp.	67.5	44.5	14.9	0.0
L–M modified	106.2	32.0	19.6	0.3
1st approx. α_1	57.2	32.2	13.3	0.0
1st approx. α_2	179.9	67.0	39.3	2.7
1st approx. α_3	179.9	29.2	30.5	6.1
1st approx. α_4	57.2	32.2	13.3	0.0
1st approx. α_5	89.9	18.2	15.1	0.0
2nd approx. α_1	46.0	20.5	10.4	0.0
2nd approx. α_2	179.9	86.0	31.1	45.1
2nd approx. α_3	179.8	33.0	33.6	8.1
2nd approx. α_4	46.0	20.5	10.4	0.0
2nd approx. α_5	89.9	15.2	14.6	0.0

preserves the convergence property (α_1 is always within the range $(0, 1/(2 \cdot \lambda_{\max}))$), while the optimistic value frequently loses this property ($\alpha_2 \geq 1/2 \cdot \lambda_{\max})$)).

Averaging the optimistic and pessimistic values (α_3) sometimes also brings bad results. Only if α_3 is dumped is the convergence restored. Generally, the second order approximation provides better results than the first order one. However, this method is more computationally costly. Unexpected results were obtained for the 3D planar pendulum. It appeared that the first order approximation method is ideal for this particular manipulator and gives even better results than the second approximation one.

3.1. Time complexity. In Section 2.2 the theoretical complexity of Jacobian-based methods was analyzed. It is instructive to evaluate also the time required to compute the transformations. To perform the task, 10000 positive definite matrices $\boldsymbol{M}$ and coefficients α were generated randomly for some values of m. Using the Mathematica package, for each matrix-coefficient pair, the matrix $\boldsymbol{B}$ in Eqn. (16) was computed and the total computation time was recorded. Results are presented in Table 5. Maybe the absolute values of the run-time are not so informative (they depend on particular hardware and other circumstances), but their relative values are quite meaningful. It appears that the Jacobian transpose, the modified Levenberg–Marquardt and the first order approximation methods are significantly better than the Jacobian pseudo-inverse method and better than the second order approximation method. It appeared that in practice the Jacobian pseudo-inverse method is not as time consuming as predicted by the theoretical analysis. Likely, the matrix inversion in the Mathematica package is computed by means of optimized algorithms.

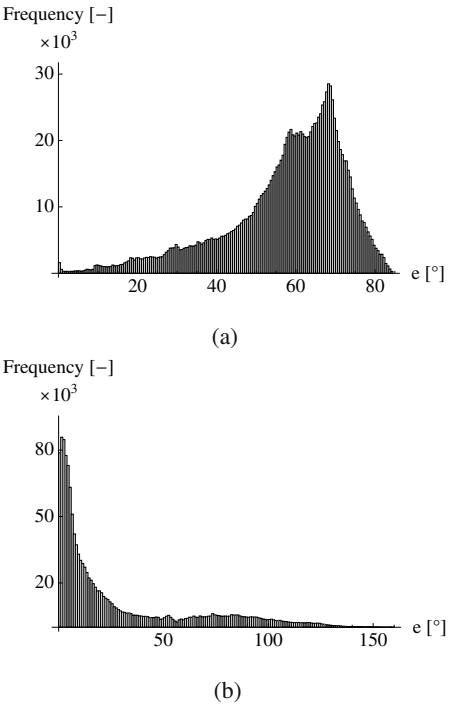


Fig. 1. Frequency distribution of the error e for the SCARA robot collected from 1000 random simulations while applying the Jacobian transposed and the modified Levenberg–Marquardt methods of inverse kinematics: Jacobian transposed (a), modified Levenberg–Marquardt (b).

3.2. Examples of real inverse kinematics tasks. Evaluation of Jacobian-based methods of inverse kinematics has been performed based on objective criteria (efficiency in a short motion (15) and the computational complexity, cf. Section 2.2), valid for short-distance (single step of the algorithms) motions. In this subsection, complete inverse kinematics tasks will be solved with these methods. Two tasks for the 4D planar pendulum (18) are run with the following initial data (initial configuration and the goal point):

Table 5. Time (in $[s]$) to compute many instances of the matrix $\boldsymbol{B}$ as a function of m.

method	m			
	3	4	5	6
pseudo-inverse	1.672	5.125	14.828	39.98
Jacobian transpose	0	0	0	0
L–M modified	0.265	0.312	0.36	0.406
1st approximation	0.344	0.422	0.515	0.609
2nd approximation	2.094	2.969	4.016	5.313

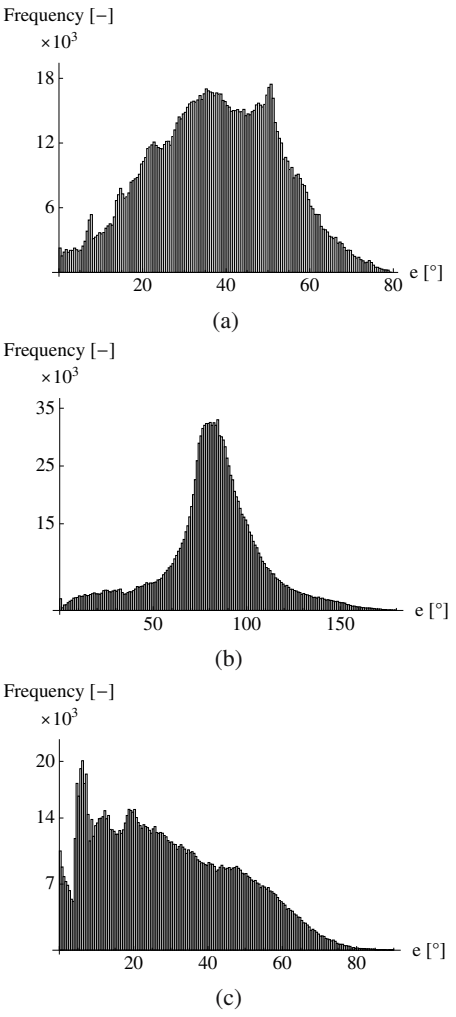


Fig. 2. Frequency distribution of the error e for the SCARA robot gathered from 1000 random simulations while applying three variants (a)–(c) of the Jacobian-based method (first order approximation) of inverse kinematics: coefficient α_1 (a), coefficient α_2 (b), coefficient α_5 (c).

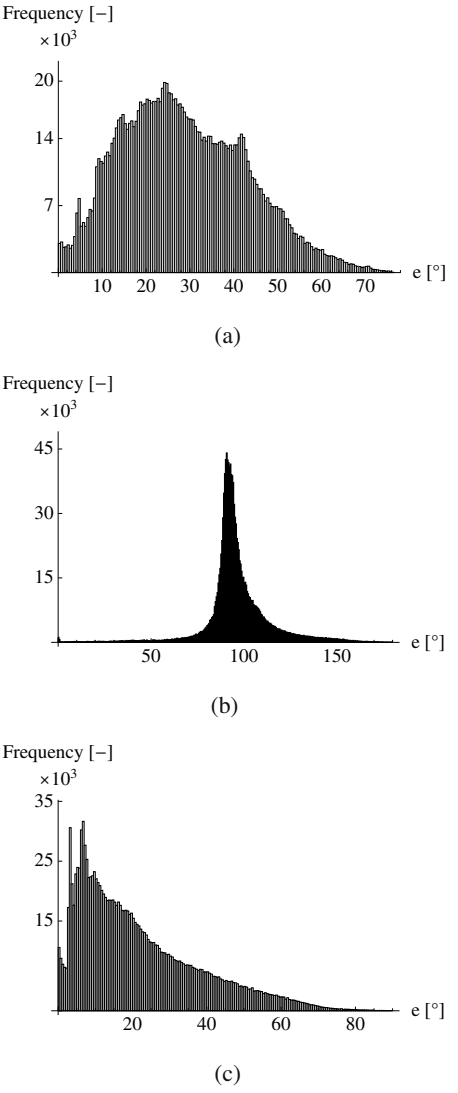


Fig. 3. Frequency distribution of the error e for the SCARA robot collected from 1000 random simulations while applying three variants (a)–(c) of the Jacobian-based method (second order approximation) of inverse kinematics: coefficient α_1 (a), coefficient α_2 (b), coefficient α_5 (c).

Task 1:

$$q_0 := (15^\circ, -30^\circ, 45^\circ, 10^\circ)^T, \quad x_f = (-2, -2, 90^\circ)^T,$$

Task 2:

$$q_0 = (0^\circ, 20^\circ, -25^\circ, 0^\circ)^T, \quad x_f = (3, -1, -45^\circ)^T,$$

and one task for the PUMA robot (20):
Task 3:

$$q_0 = (-45^\circ, -15^\circ, 25^\circ, 50^\circ, -10^\circ, -97^\circ)^T,$$

$$x_f = (0.803, -0.06, 0.018)^T.$$

For all the tasks the accuracy of reaching the goal x_f was set to $\epsilon = 0.1$ while the length of a single step motion was restricted to 0.0005. Trajectories in the taskspace generated with these methods are visualized in Fig. 4, while the number of iterations and the computation time (algorithms implemented in Mathematica and run on a 3.2 GHz computer) are presented in Table 6.

It appears that, for some tasks, trajectories produced by the modified Levenberg–Marquardt and the Jacobian transpose methods can be quite far from the straight-line motion towards the goal (Figs. 4(a) and (b)). Approximation methods more exactly reflect properties of the Jacobian pseudo-inverse algorithm, thus trajectories generated with those methods are close to straight-line

Table 6. Number of iterations and time of computations (in [s]) to complete Tasks 1–3 with Jacobian-based methods.

		$J^{\#}$	J^{T}	mLM	appr1	appr2
1	time	5.85	6.31	4.44	5.08	5.21
	iter	11372	14323	9371	8982	8845
2	time	3.92	2.96	3.22	4.14	4.00
	iter	6955	7693	7043	7379	7324
3	time	2.26	1.02	1.39	1.78	1.87
	iter	2298	1759	1773	1809	1870

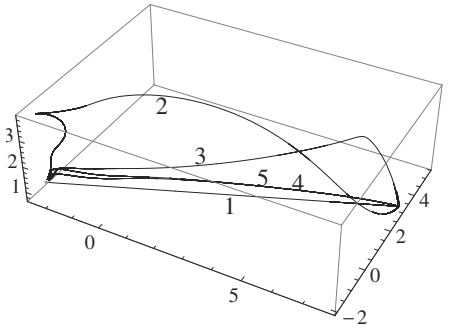
motions. Usually, the modified Levenberg–Marquardt method is among the fastest methods, while the Jacobian transpose can be significantly faster than the Jacobian pseudo-inverse method, but sometimes it can also be slower. As expected, the first order approximation method is faster than the second order method.

Two final remarks are of a quite general nature. First, advantages of using methods alternative to Jacobian pseudo-inverse ones become more evident as the dimensionality of the taskspace, m, increases, cf. (17) and Table 1. In the simplest and slightly unrealistic case $m = 1$ all the methods are virtually equivalent. Second, the alternative methods can be useful when the initial point and the final point in the taskspace are relatively far from each other. When they are close, the computational time advantage of the Jacobian transpose and mLM methods can disappear easily as trajectories generated resemble rather a spiral around the final point than a straight-line segment leading towards the goal.

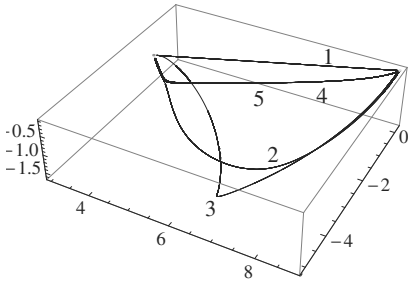
4. Conclusions

In this paper four Jacobian-based methods of solving the inverse kinematics task have been evaluated. The most natural measure, the number of iterations required to complete a given inverse kinematics task, was rejected as a comparative criterion of the methods because this characteristic is not reliable and path-dependent. To avoid these drawbacks, a new criterion based on the evaluation of a motion produced by a given method related to the motion generated with the Jacobian pseudo-inverse method has been proposed.

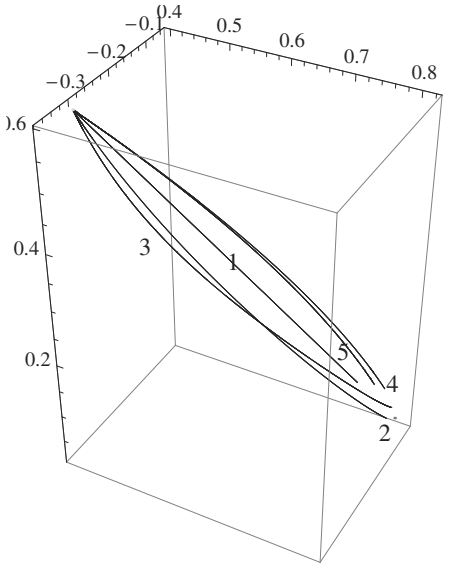
Besides the well-known Jacobian transpose method, the modified Levenberg–Marquardt method and a number of methods based on the first and the second order approximations of the inverse Jacobian matrix have been tested. For three manipulators, simulations carried out on a quite large set of random configurations revealed that the modified Levenberg–Marquardt method is much better than the Jacobian transpose one, although occasionally it may lose convergence. Simulation experiments showed that approximation methods are very sensitive to the estimated maximal singular value.



(a)



(b)



(c)

Fig. 4. Trajectories in the taskspace generated with the different Jacobian-based methods of inverse kinematics: Jacobian pseudo-inverse (1), Jacobian transpose (2), modified Levenberg-Marquardt (3), the first approximation of $J^{\#}$ (4), the second approximation of $J^{\#}$ (5): Task 1 (a), Task 2 (b), Task 3 (c).

Estimation of the maximal singular value is necessary because the computation of its real value is demanding. It appeared that it is better to underestimate

than to overestimate the maximal singular value. As expected, the second order approximation is better than the first order approximation, but it is also computationally more involved. The latter method is comparable to the modified Levenberg–Marquardt one, which is more reliable (the convergence has been always preserved by the first order approximation method).

Results of the paper seem to be important for the implementation of inverse kinematic algorithms when computational resources are restricted and fast solutions are desired. This situation is frequently encountered in practice for mobile robots carrying a manipulator on the board.

References

Ben-Isreal, A. and Cohen, D. (1966). On iterative computation of generalized inverses and associated projections, *SIAM Journal on Numerical Analysis* **3**(3): 410–419.

Ben-Isreal, A. and Greville, T. (2003). *Generalized Inverses: Theory and Applications*, CMS Books in Mathematics, 2nd Edn., Springer, New York, NY.

Chiacchio, P. and Siciliano, B. (1989). A closed-loop Jacobian transpose scheme for solving the inverse kinematics of nonredundant and redundant wrists, *Journal of Robotic Systems* **6**(5): 601–630.

D'Souza, A., Vijaykumar, S. and Schaal, S. (2001). Learning inverse kinematics, *International Conference on Intelligent Robots and Systems, Maui, HI, USA*, pp. 298–303.

Dulęba, I. and Jagodziński, J. (2011). Motion representations for the Lafferriere–Sussmann algorithm for nilpotent control systems, *International Journal of Applied Mathematics and Computer Science* **21**(3): 525–534, DOI: 10.2478/v10006-011-0041-y.

Dulęba, I. and Sasiadek, J. (2002). Modified Jacobian method of transversal passing through the smallest deficiency singularities for robot manipulators, *Robotica* **20**(4): 405–415.

Golub, G. and Van Loan, C. (1996). *Matrix Computations*, 3rd Edn., Johns Hopkins, Baltimore, MD.

Horn, R. and Johnson, C. (1986). *Matrix Analysis*, Cambridge University Press, New York, NY.

Hunek, W. and Latawiec, K.J. (2011). A study on new right/left inverses of nonsquare polynomial matrices, *International Journal of Applied Mathematics and Computer Science* **21**(2): 331–348, DOI: 10.2478/v10006-011-0025-y.

Lee, C. (1982). Robot arm kinematics, dynamics, and control, *Computer* **15**(12): 62–80.

Levenberg, K. (1944). A method for the solution of certain problems in least squares, *Quarterly of Applied Mathematics* **2**: 164–168.

Maciejewski, A. and Klein, C. (1989). The singular value decomposition: Computation and applications to robotics, *International Journal of Robotics Research* **8**(6): 63–79.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics* **11**(2): 431–441.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*, Addison Wesley, New York, NY.

Nearchou, A. (1998). Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm, *Mechanism and Machine Theory* **33**(3): 273–292.

Tchoń, K. and Dulęba, I. (1993). On inverting singular kinematics and geodesic trajectory generation for robot manipulators, *Journal of Intelligent and Robotic Systems* **8**(3): 325–359.

Tchoń, K., Dulęba, I., Muszyński, R., Mazur, A. and Hossa, R. (2000). *Manipulators and Mobile Robots: Models, Motion Planning, Control*, PLJ, Warsaw, (in Polish).

Tchoń, K., Karpińska, J. and Janiak, M. (2009). Approximation of Jacobian inverse kinematics algorithms, *International Journal of Applied Mathematics and Computer Science* **19**(4): 519–531, DOI: 10.2478/v10006-009-0041-3.

Tejomurtula, S. and Kak, S. (1999). Inverse kinematics in robotics using neural networks, *Information Sciences* **116**(2–4): 147–164.

Ignacy Dulęba was born in 1961. He received his M.Sc. and Ph.D. degrees in robotics from the Wrocław University of Technology in 1986 and 1991, respectively. In 1999 he received a D.Sc. degree from the Warsaw University of Technology. Currently, he is a professor at the Wrocław University of Technology. His scientific activities have been concerned with the modelling of robots, motion planning of nonholonomic systems, and control.

Michał Opałka was born in Wrocław, Poland. He obtained his M.Sc. degree in control engineering and robotics from the Wrocław University of Technology in 2007. Currently, he is working towards his Ph.D. thesis devoted to methods of motion planning. His research interests cover the area of control theory applied to nonholonomic robotic systems.

Received: 14 March 2012
Revised: 5 September 2012