# Second-order and implicit methods in numerical integration improve tracking performance of the closed-loop inverse kinematics algorithm

Dániel András Drexler*, Levente Kovács*
*Obuda University, EKIK Physiological Controls Group
Bécsi út 96/b H-1034 Budapest, Hungary
Emails: drexler.daniel@gmail.com, kovacs.levente@nik.uni-obuda.hu

***Abstract*—A general approach to solve the inverse kinematics problem of series manipulators, i.e. finding the required joint motions for the desired end effector motions, is based on the linear approximation of the forward kinematics map and discretization of the continuous problem. Due to the linearization, first velocities are calculated, so numerical integration needs to be done to get the joint variables. This general solution is just a numerical approximation, thus improving the tracking performance of the inverse kinematics algorithm is of great importance. The application of several numerical integration techniques (implicit Euler, explicit trapezoid, implicit trapezoid) is analyzed, and a fix point iteration is given that can be used to calculate implicit solutions. The tracking performance of the spatial inverse positioning problem of a spatial manipulator is analyzed by checking the tracking error in the desired direction (i.e. along the derivative of the desired end effector path) and in the plane perpendicular to the desired direction. The application of the explicit and implicit trapezoid methods yielded much better tracking performance in the directions orthogonal to the desired direction when the end effector had to track a linear path, while the tracking performance in the desired direction was similar for all the methods. Simulations showed that the application of implicit and second-order methods in the numerical integration may greatly improve the tracking performance of the closed-loop inverse kinematics algorithm.**

***Index Terms*—differential inverse kinematics, numerical integration, explicit Euler, implicit Euler, explicit trapezoid, implicit trapezoid**

## I. Introduction

The inverse kinematics problem (i.e. finding the required joint motion for the desired end effector motion) of serial robot manipulators is a key problem in robot control that does not have an analytical solution in the general case (see e.g. [1], [2], [3]). A general solution to the inverse kinematics problem that can be used for real-time applications is based on the linear approximation of the forward kinematics map that maps joint motion to end effector motion [4]. The Jacobian of the forward kinematics map gives a point-wise linear relationship between the joint velocities and the end effector velocities, so calculation of the required joint velocities can be done by solving a linear system of equations. In the practical case, the problem is discretized in time, so the joint variables are acquired from the joint velocities by numerical integration. The widely used numerical integration technique is the explicit Euler method, i.e. the joint variable is updated each time by adding the joint difference (discrete-time velocity) multiplied by the sampling time (see e.g. [5], [6], [7]). This algorithm is usually refered to as the differential inverse kinematics algorithm.

The tracking performance of the differential inverse kinematics algorithm can be improved by adding the tracking error of the end effector to the desired end effector velocity after being multiplied by a feedback gain parameter. Since this results in a closed-loop system, this algorithm is called the closed-loop inverse kinematics (CLIK) algorithm in the literature (see e.g. [8]), being discussed in Section II. Increasing the feedback gain parameter leads to better tracking performance, however the system becomes unstable when the gain parameter exceeds a certain limit (for recent results on the bound on this limit the reader is referred to [8]). Thus the CLIK algorithm only has limited potential in tracking performance increase. Moreover, the stability margin of the system depends on the singular values of the Jacobian matrix, that depends on the joint configuration. This implies that in practice the gain parameter should be kept relatively low to avoid unstable operation, or it should be recalculated in each step, that would lead to unpredictable behavior of the algorithm.

The tracking performance of the differential inverse kinematics or CLIK algorithm can also be improved by using higher-order explicit numerical integration techniques, as it was shown in [9]. Introduction of implicit methods may also bring further improvement, as it was shown in [10]. However, implicit techniques require the knowledge of the joint variables in the next time instant, albeit the goal of the inverse kinematics problem is to locally calculate the joint variables in the next time instant, so these quantities are not known in advance. However, an iteration can be used to calculate the implicit solutions. This iteration was first introduced in [10], and is further developed and generalized in Section III where the existence of its unique fixed-point (that is the implicit solution) is proved.

The implicit Euler, explicit trapezoid and implicit trapezoid methods and their application in the CLIK problem are discussed in Section IV, and the modified CLIK algorithms using these integration techniques are given. The algorithms are described for the general inverse kinematics problem, i.e. it does not matter if e.g. only the position, the orientation, or both the position and orientation of the end effector of the manipulator is concerned.

The numerical integration schemes are tested on an example when the spatial position of the end effector is concerned in Section V. In order to gain more insight into the tracking performance, the tracking error (i.e. the difference of the desired and current end effector position) is transformed into a new base at each time instant, whose first basis vector is the tangent of the desired end effector path at each time instant (it will be called a regular path direction), and the other two mutually orthogonal basis vectors are orthogonal to the regular path direction (these will be called singular path directions, however they should not be confused with manipulator singularities). If the tracking error has great components in the singular path directions, then it means that the end effector leaves the predefined curve, however, if these components are small, then the end effector moves on the desired end effector path. For example, if the desired end effector path is a line segment, and the tracking error component is zero in the singular path directions, then it means that the end effector moves on a path with the shape of a straight line, even if the tracking error component in the regular path direction is not zero.

It turns out, that application of the explicit and implicit trapezoid methods greatly increase the tracking performance in the case if the desired path is a straight line by having significantly smaller tracking error in the singular path directions. The results are summarized in Section VI.

## II. Closed-loop inverse kinematics algorithm

Let the vector of joint variables be denoted by $\theta$, with its $i$th component $\theta_i$ being the joint variable of the $i$th joint of the manipulator. Note that $\theta$ is a function of (positive) time that assigns the value of the joint variables to each $t$ time instant such that $t \geq 0$. Let the forward kinematics mapping be denoted by $f$, i.e. $f(\theta)$ is the end effector pose (position and orientation). Suppose that the orientation is represented as a vector, e.g. the components of the vector are rotations around the basis vectors of a fixed spatial frame. Let the desired end effector pose be $x_d$, while the desired end effector velocity be $\dot{x}_d$. Denote the Jacobian of the mapping $f$ at the joint variable $\theta$ by $J(\theta)$, then the relationship between the $\dot{\theta}$ joint velocities and the $\dot{x}$ end effector velocities is given by

$$\dot{x} = J(\theta)\dot{\theta}. \tag{1}$$

The goal is to find one of the (generally not unique) functions $\theta_d$ that satisfy

$$\dot{x}_d = J(\theta_d)\dot{\theta}_d. \tag{2}$$

This is usually done by first discretizing the functions $\theta$ and $x_d$ by defining $\theta[k] := \theta(kT_s)$ and $x_d[k] := x_d(kT_s)$ with $T_s$ being the sampling time and $k$ being a nonnegative integer. Then the velocities become differences, i.e. $\Delta\theta[k] := \dot{\theta}(kT_s)$ and $\Delta x_d[k] := \dot{x}_d(kT_s)$. The discretized version of (2) is thus

$$\Delta x_d[k] = J(\theta_d[k])\Delta\theta_d[k], \quad k = 0, 1, 2, \ldots \tag{3}$$

Each iteration of the differential inverse kinematics algorithm consists of a solution of a linear system of equations

$$\Delta x_d[k] = J(\theta[k])\Delta\theta[k], \quad k = 0, 1, 2, \ldots \tag{4}$$

to acquire $\Delta\theta[k]$, and then the numerical integration step

$$\theta[k+1] = \theta[k] + \alpha\Delta\theta[k] \tag{5}$$

to update the joint variable vector. Note that this numerical integration is the explicit Euler method [11]. Given an initial value $\theta[0]$, if we substitute the series $\theta[1], \theta[2], \ldots$ acquired using the iteration of steps (4) and (5) into the function $f$, the values $f(\theta[0]), f(\theta[1]), f(\theta[2]), \ldots$ may be different from the values $x_d[0], x_d[1], x_d[2], \ldots$, especially if $f(\theta[0]) \neq x_d[0]$. In the latter case, the algorithm does not converge at all.

In order to overcome this problem, the feedback term $\alpha\left(x_d[k] - f(\theta[k])\right)$ needs to be added to the desired velocity, so the first equation to be solved in each iteration step becomes

$$\Delta x_d[k] + \alpha(x_d[k] - f(\theta[k])) = J(\theta[k])\Delta\theta[k], \quad k = 0, 1, 2, \ldots \tag{6}$$

so the joint variable difference $\Delta\theta[k]$ is calculated as

$$\Delta\theta[k] = J^{\#}(\theta[k])\left(\Delta\theta[k] + \alpha(x_d[k] - f(\theta[k]))\right) \tag{7}$$

where the $J^{\#}$ denotes the (generalized) inverse of the Jacobian. The algorithm whose first step is (7) and second step is the numerical integration (5) in each iteration is called the CLIK algorithm. This algorithm has better tracking performance than the differential inverse kinematics algorithm without the feedback term. Generally the distance between the elements of the series $f(\theta[0]), f(\theta[1]), f(\theta[2])$ and $x_d[0], x_d[1], x_d[2], \ldots$ becomes smaller as $\alpha$ is increased, until the stability margin is reached, at which point the algorithm becomes unstable. If the stability margin is reached, the tracking performance can not be increased further by increasing the parameter $\alpha$. However, further increase in the tracking performance can be achieved by replacing the numerical integration step (5) by a different technique that will be discussed in the upcoming Sections.

The CLIK algorithm can be generalized for the general inverse kinematics problem by introducing the terms task space, task Jacobian and the task vector. The task space is the subspace of the linear space where $f$ maps to, e.g. if only the positioning problem is concerned, than this subspace is the 3-dimensional Euclidean space describing end effector positions, if only the planar position of the end effector is concerned, then this subspace is a 2-dimensional Euclidean space, if only the end effector orientation is concerned, this is the 3-dimensional Euclidean space defining the orientation of the end effector in the representation given in the beginning of this Section, and so on. Given the task space, the task Jacobian is the projection of the analytical Jacobian to this task space, while the task vector is the sum of the desired end effector velocity given in the task space, and the tracking error given in the task space multiplied by the feedback term.

In the remainder of the paper we suppose that $J$ is the task Jacobian, $x_d$ and $\dot{x}_d$ are given in the task space, and $f$ maps to the task space, so the task vector in the discrete time step $k$ is given by

$$t(\theta[k],k) = \Delta x_d[k] + \alpha\left(x_d[k] - f(\theta[k])\right), \tag{8}$$

and the first step (7) of the CLIK algorithm is written as

$$\Delta\theta[k] = J^{\#}(\theta[k])t(\theta[k],k). \tag{9}$$

## III. Iteration to calculate the implicit solutions

The numerical integration in (5) can be replaced by other methods to increase the tracking performance of the inverse kinematics algorithm. For example, in [9], the authors considered the application of higher-order explicit methods. However, there are numerous numerical integration methods that are implicit, i.e. the update law depends on $\theta[k+1]$ as well. Let the general form of the update law be

$$\theta[k+1] = \phi(\theta[k],\theta[k+1]) \tag{10}$$

for some function $\phi$. For example, for the implicit Euler method, the function $\phi$ is

$$\phi : (\theta[k],\theta[k+1]) \mapsto \theta[k] + T_s\Delta\theta[k+1], \tag{11}$$

and since $\Delta\theta[k+1] = J^{\#}(\theta[k+1])t(\theta[k+1],k+1)$, this can be written as

$$\phi : (\theta[k],\theta[k+1]) \mapsto \theta[k] + T_s J^{\#}(\theta[k+1])t(\theta[k+1],k+1). \tag{12}$$

However, the goal of the inverse kinematics algorithm is to calculate $\theta[k+1]$ in each discrete time step $k$, so we do not know its value in advance. Further note that usually the symbolic expressions for the implicit update are so complex, that we can not express $\theta[k+1]$ from these expressions explicitly. In order to calculate the implicit solution, an iteration was proposed in [10] to calculate the implicit Euler solution.

**Algorithm 1.** Iteration for the implicit Euler solution.

1) First initialize the estimation $\tilde{\theta}[k+1]$ with the explicit Euler method:
$$\tilde{\theta}[k+1] = \theta[k] + T_s\Delta\theta[k]. \tag{13}$$
2) Calculate the difference $\Delta\tilde{\theta}[k+1]$ based on $\tilde{\theta}[k+1]$ as
$$\Delta\tilde{\theta}[k+1] = J^{\#}(\tilde{\theta}[k+1])t(\tilde{\theta}[k+1],k+1). \tag{14}$$
3) Recalculate $\tilde{\theta}[k+1]$ with the new $\Delta\tilde{\theta}[k+1]$:
$$\tilde{\theta}[k+1] = \theta[k] + T_s\Delta\tilde{\theta}[k+1]. \tag{15}$$
4) Repeat steps 2 and 3 until the alteration of $\tilde{\theta}[k+1]$ is small enough or a certain number of iterations is reached.

We further develop and generalize this algorithm for the general update law (10). Write the update law $\phi$ in the form

$$\phi = \theta[k] + T_s\psi(\Delta\theta[k],\Delta\theta[k+1]). \tag{16}$$

**Algorithm 2.** Iteration for implicit solution with update law (16).

1) First, calculate $\Delta\theta[k]$ using the expression
$$\Delta\theta[k] = J^{\#}(\theta[k])t(\theta[k],k) \tag{17}$$
and calculate $\Delta\theta[k+1]$ using the expression
$$\Delta\theta[k+1] = J^{\#}(\theta[k])t(\theta[k],k+1), \tag{18}$$
and compute $\tilde{\theta}[k+1]$ using the update law (16).
2) Calculate the difference
$$\Delta\tilde{\theta}[k+1] = J^{\#}(\tilde{\theta}[k+1])t(\tilde{\theta}[k+1],k+1). \tag{19}$$
3) Update $\tilde{\theta}[k+1]$ using the expression (16) as
$$\tilde{\theta}[k+1] = \theta[k] + T_s\psi(\Delta\theta[k],\Delta\tilde{\theta}[k+1]). \tag{20}$$
4) Repeat steps 2 and 3 until the alteration of $\tilde{\theta}[k+1]$ is small enough or a certain number of iterations is reached.

**Theorem 1.** *Suppose, that we are far from singular configurations, i.e. we are moving in the connected set $U$ of the joint space, such that there exists a positive number $\eta > 0$ such that $\left\|J^{\#}(\theta)\right\|_\infty \le \eta$ for all $\theta \in U$, moreover there exists a positive number $\nu$ such that*

$$\max_i \left\|\partial_{\theta_i} J(\theta)\right\|_\infty \le \nu \tag{21}$$

*for all $\theta \in U$. Let $\Delta\theta[k+1]$ be the implicit difference and $\Delta\theta[k]$ be the explicit difference of joint variables. Suppose that $\psi$ is a continuously differentiable function of $\Delta\theta[k+1]$. Then if $\alpha$ satisfies the inequality*

$$\alpha < \frac{1}{T_s\left\|\partial_{\Delta\theta[k+1]}\psi(\Delta\theta[k],\Delta\theta[k+1])\right\|_\infty} - n\nu\eta\left\|\Delta\theta[k+1]\right\|_\infty \tag{22}$$

*where $n$ is the number of the joints of the robot, then $\theta[k+1]$ is the fix point of the function (10) with $\Delta\theta[k+1] = J^{\#}(\theta[k+1])t(\theta[k+1],k+1)$ and Algorithm 2 converges to this fixed point.*

*Proof:* We will show that with the above conditions the mapping $\phi$ is a contraction mapping (in the $\infty$-norm), i.e. if the conditions of the theorem hold then there exists a number $0 \le q < 1$ such that for all $a,b \in U$

$$\left\|\phi(a) - \phi(b)\right\|_\infty \le q\left\|a-b\right\|_\infty. \tag{23}$$

Since $\psi$ in (16) is continuously differentiable, the mapping (10) is also continuously differentiable due to the conditions of the theorem (since $J$ is nonsingular thus $J^{\#}$ exists and is continuously differentiable), so condition (23) is equivalent to the existence of a Lipschitz-constant $q$ such that

$$\left\|\partial_{\theta[k+1]}\phi\right\|_\infty \le q. \tag{24}$$

We will show that $\left\|\partial_{\theta[k+1]}\phi\right\|_\infty < 1$ implies (22) if the other conditions of the theorem hold. Applying the chain rule, the differential $\partial_{\theta[k+1]}\phi$ can be written as

$$\partial_{\theta[k+1]}\phi = T_s\partial_{\Delta\theta[k+1]}\psi(\Delta\theta[k+1])\partial_{\theta[k+1]}\Delta\theta[k+1], \tag{25}$$

where we have omitted the argument $\Delta\theta[k]$ of $\psi$ for clarity.

The derivative $\partial_{\theta[k+1]}\Delta\theta[k+1]$ is a matrix with its $i$th column being $\partial_{\theta_i[k+1]}\Delta\theta[k+1]$. For the sake of simplicity, in the remainder of the proof we will omit the argument $[k+1]$ and use the notations $\theta := \theta[k+1]$ and $\Delta\theta := \Delta\theta[k+1]$. Since $\Delta\theta$ is calculated as $\Delta\theta = J^{\#}(\theta)t(\theta, k+1)$, the derivative of $\Delta\theta$ with respect to the scalar $\theta_i$ is

$$\begin{aligned}\partial_{\theta_i}\Delta\theta &= \partial_{\theta_i}\left(J^{\#}(\theta)t(\theta,k+1)\right)\\ &= \left(\partial_{\theta_i}\left(J^{\#}(\theta)\right)\right)t(\theta,k+1)\\ &\quad + J^{\#}(\theta)\partial_{\theta_i}\left(t(\theta,k+1)\right).\end{aligned} \tag{26}$$

The differential of the task vector is

$$\begin{aligned}\partial_{\theta_i}\left(t(\theta,k+1)\right) &= \partial_{\theta_i}\left(\Delta x_d[k+1] + \alpha\left(x_d[k+1] - f(\theta)\right)\right)\\ &= -\alpha\partial_{\theta_i}f(\theta)\\ &= -\alpha J(\theta)(\cdot,i),\end{aligned} \tag{27}$$

where $J(\theta)(\cdot,i)$ denotes the $i$th column of the matrix $J(\theta)$ with the notation used e.g. in [12], [13]. Substituting this into the second term in (26) yields

$$\begin{aligned}J^{\#}(\theta)\partial_{\theta_i}\left(t(\theta,k+1)\right) &= J^{\#}(\theta)\left(-\alpha J(\theta)(\cdot,i)\right)\\ &= -\alpha e_i,\end{aligned} \tag{28}$$

where $e_i$ is the $i$th unit vector of $\mathbb{R}^n$.

The differential of the Jacobian pseudoinverse is

$$\partial_{\theta_i}\left(J^{\#}(\theta)\right) = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)J^{\#}(\theta), \tag{29}$$

so the first term in (26) becomes

$$\left(\partial_{\theta_i}\left(J^{\#}(\theta)\right)\right)t(\theta,k+1) = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)\Delta\theta, \tag{30}$$

so (26) reduces to

$$\partial_{\theta_i}\Delta\theta = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)\Delta\theta - \alpha e_i. \tag{31}$$

The norm of the function $\partial_\theta\phi$ can be bounded from above by

$$\|\partial_\theta\phi\|_\infty \le T_s\|\partial_{\Delta\theta}\psi\|_\infty\|\partial_\theta\Delta\theta\|_\infty \tag{32}$$

provided that $T_s > 0$. Since the $\infty$-norm of a matrix is its maximal absolute column sum,

$$\|\partial_\theta\Delta\theta\|_\infty = \max_i\left\{1_n\left|-\alpha e_i - J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right|\right\} \tag{33}$$

where $1_n$ is a row vector of length $n$ whose each element is one and $|\cdot|$ is the element-wise absolute value function. Due to the triangle inequality

$$\|\partial_\theta\Delta\theta\|_\infty \le \alpha + \max_i\left\{1_n\left|-J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right|\right\}, \tag{34}$$

and since the absolute column sum is not greater than the product of the length of the column and the absolute value of the element of the column that has the greatest absolute value, the inequality becomes

$$\|\partial_\theta\Delta\theta\|_\infty \le \alpha + n\max_i\left\|J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right\|_\infty \tag{35}$$

from which we obtain

$$\|\partial_\theta\Delta\theta\|_\infty \le \alpha + n\left\|J^{\#}(\theta)\right\|_\infty\|\Delta\theta\|_\infty\max_i\partial_{\theta_i}J(\theta). \tag{36}$$

Substituting the bounds from the conditions of the theorem into (36), and substituting the result into (32) yields

$$\|\partial_\theta\phi\|_\infty \le T_s\|\partial_{\Delta\theta}\psi\|_\infty\left(\alpha + n\nu\eta\|\Delta\theta\|_\infty\right). \tag{37}$$

This derivative is smaller than one and thus $\phi$ is contractive and has a unique fixed-point $\theta$ if

$$\alpha < \frac{1}{T_s\|\partial_{\Delta\theta}\psi\|_\infty} - n\nu\eta\|\Delta\theta\|_\infty \tag{38}$$

that is the result we were looking for. ∎

Note that usually $\|\partial_{\Delta\theta}\psi\|_\infty$ is a constant as it will be shown in the next Section.

## IV. Implicit and Second Order Numerical Integration Algorithms

The application of implicit Euler, explicit trapezoid and implicit trapezoid methods is tested in order to check if an increase in the tracking performance can be achieved by using these methods instead of the explicit Euler integration method in step (5). The implicit Euler and implicit trapezoid methods are implicit methods, so they require the iteration described in Section III.

### A. Implicit Euler method

The update law (10) for the implicit Euler method is

$$\phi = \theta[k] + T_s\Delta\theta[k+1] \tag{39}$$

so the function $\psi$ in (16) is

$$\psi(\Delta\theta[k], \Delta\theta[k+1]) = \Delta\theta[k+1], \tag{40}$$

and the $\infty$-norm of the differential of this function with regard to $\Delta\theta[k+1]$ is

$$\left\|\partial_{\Delta\theta[k+1]}\psi\right\|_\infty = 1. \tag{41}$$

In each iteration, the implicit solution $\theta[k+1]$ can be calculated with the iteration described in Algorithm 2, with the update law (39).

### B. Explicit trapezoid method

In the explicit trapezoid method the difference $\Delta\theta[k+1]$ is approximated using the joint variable value $\theta[k]$, i.e. the joint difference $\Delta\theta[k+1]$ is acquired by first calculating

$$\Delta\theta[k] = J^{\#}(\theta[k])t(\theta[k], k), \tag{42}$$

then using this quantity to approximate $\theta[k+1]$ using the explicit Euler method, i.e.

$$\tilde{\theta}[k+1] = \theta[k] + T_s\Delta\theta[k]. \tag{43}$$

Finally, the difference $\Delta\theta[k+1]$ is calculated as

$$\Delta\theta[k+1] = J^{\#}(\tilde{\theta}[k+1])t(\tilde{\theta}[k+1], k+1). \tag{44}$$

The update law for the explicit trapezoid method is

$$\phi = \theta[k] + \frac{1}{2}T_s\left(\Delta\theta[k] + \Delta\theta[k+1]\right), \tag{45}$$

however it is only applied once, so there is no iteration required when using this method.

## C. Implicit trapezoid method

The update law for the implicit trapezoid method is the same as in the case of explicit trapezoid method, i.e.

$$\phi = \theta[k] + \frac{1}{2}T_s\left(\Delta\theta[k] + \Delta\theta[k+1]\right), \tag{46}$$

thus the function $\psi$ is

$$\psi(\Delta\theta[k], \Delta\theta[k+1]) = \frac{1}{2}\left(\Delta\theta[k] + \Delta\theta[k+1]\right) \tag{47}$$

and the $\infty$-norm of its derivative with respect to $\Delta\theta[k+1]$ is

$$\left\|\partial_{\Delta\theta[k+1]}\psi\right\|_\infty = \frac{1}{2}. \tag{48}$$

The implicit trapezoid method can be applied by using Algorithm 2 with the update law (46).

## V. Experimental results

The different numerical integration techniques were applied for the solution of the inverse positioning problem of an elbow manipulator, a manipulator architecture consisting of three revolute joints that is widely applied in the practice. The three joint axes of the manipulator in the home configuration (i.e. in the configuration where $\theta = 0$) defined in a fixed frame are

$$\omega_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \omega_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \omega_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tag{49}$$

while some points on the joint axes 1, 2 and 3 respectively are

$$q_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad q_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad q_3 = \begin{pmatrix} 0 \\ 0 \\ l_1 \end{pmatrix}, \tag{50}$$

with $l_1 = 1$ being the length of the second segment of the manipulator, while the position of the end effector in the home configuration is

$$p(0) = \begin{pmatrix} 0 \\ 0 \\ l_1 + l_2 \end{pmatrix} \tag{51}$$

with $l_2 = 1$ being the length of the third segment of the manipulator. Based on these design parameters one can calculate the forward kinematics map in the task space and the task Jacobian in every configuration using techniques from e.g. [1], [14].

The initial configuration of the robot arm was $\theta[0] = (0, 0, \pi/2)^\top$ that is a nonsingular configuration, i.e. $J(\theta[0])$ is regular. The desired end effector path and end effector velocity were

$$x_d[k] = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} + \frac{k-1}{N}\begin{pmatrix} 0 \\ 0.5 \\ -1 \end{pmatrix} \tag{52}$$

$$\Delta x_d[k] = \frac{1}{N}\begin{pmatrix} 0 \\ 0.5 \\ -1 \end{pmatrix}. \tag{53}$$

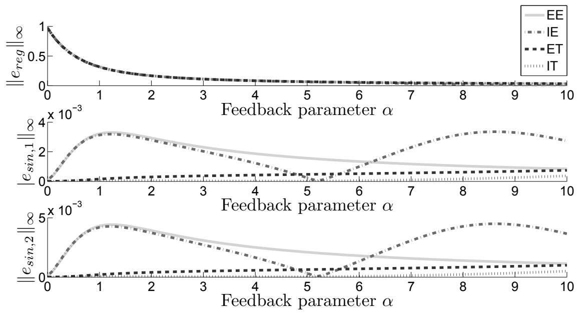

Fig. 1. The maximal absolute values of the tracking errors along the regular and the two singular path directions for different value of the feedback parameter $\alpha$

The number of iterations for the CLIK algorithm was $N = 30$, so $k = 0, 1, \ldots, N$, the sampling time was $T_s = 0.1sec$. The CLIK algorithm was solved using the explicit Euler, implicit Euler, explicit trapezoid and implicit trapezoid methods for different $\alpha$ feedback parameters. The initial value of $\alpha$ was zero, then it was increased by $0.1$ in each step until it reached the value $\alpha = 10$.

For every value of $\alpha$ the tracking error was calculated and transformed into a base for each discrete time $k = 0, 1, \ldots, N$ whose basis vectors are:

1) The regular path direction, that is $\Delta x_d[k]$ after normalization for each $k = 0, 1, \ldots, N$; components of this basis are denoted by the subscript $reg$.
2) The first singular path direction, that is a unit vector perpendicular to $\Delta x_d[k]$ for each $k = 0, 1, \ldots, N$; components of this basis are denoted by the subscript $sin1$.
3) The second singular path direction, that is a unit vector perpendicular to $\Delta x_d[k]$ and the first singular path direction for each $k = 0, 1, \ldots, N$; components of this basis are denoted by the subscript $sin2$.

Note that for each value of $\alpha$, the tracking error is a series in the three new components $\{e_{reg}[k]\}, \{e_{sin1}[k]\}, \{e_{sin2}[k]\}$ for $k = 0, 1, \ldots, N$, so along each component, the absolute value is taken and the maximal element is chosen. Thus for each $\alpha$ we take the $\infty$-norms of the series $\{e_{reg}[k]\}$, $\{e_{sin1}[k]\}$ and $\{e_{sin2}[k]\}$ that are the values $\max\|e_{reg}\|$, $\max\|e_{sin1}\|$ and $\max\|e_{sin2}\|$. These values are plotted for different values of $\alpha$ in Figure 1, where different curves correspond to different integration methods, the solid curve corresponds to the explicit Euler (EE) method, the dash-dot curve corresponds to the implicit Euler (IE) method, the dashed curve corresponds to the explicit trapezoid (ET) method, while the dotter curve corresponds to the implicit trapezoid (IT) method.

The maximal errors along the regular path direction are similar for all of the methods, however there is a significant improvement in the tracking error along the singular path directions if the second-order trapezoid methods are used. Table I shows the value of the errors for some specific $\alpha$ values in the first singular path direction.

The explicit trapezoid method gave slightly better results

| | $\alpha = 0$ | $\alpha = 5$ | $\alpha = 10$ |
|---|---|---|---|
| EE | $7.73 \cdot 10^{-5}$ | $1.56 \cdot 10^{-3}$ | $8.48 \cdot 10^{-4}$ |
| IE | $7.74 \cdot 10^{-5}$ | $2.42 \cdot 10^{-4}$ | $2.72 \cdot 10^{-3}$ |
| ET | $1.67 \cdot 10^{-8}$ | $4.74 \cdot 10^{-4}$ | $7.48 \cdot 10^{-4}$ |
| IT | $2.98 \cdot 10^{-8}$ | $1.72 \cdot 10^{-5}$ | $3.73 \cdot 10^{-4}$ |

TABLE I
THE $\infty$-NORMS OF THE TRACKING ERROR COMPONENTS IN THE FIRST SINGULAR PATH DIRECTION ($\|e_{sin1}\|_\infty$) FOR DIFFERENT VALUES OF FEEDBACK GAIN $\alpha$ AND FOR DIFFERENT NUMERICAL INTEGRATION METHODS

in the singular path directions for low values of $\alpha$, however for greater values of $\alpha$ the implicit trapezoid method gave better results in the singular path directions. Thus the tracking error of the end effector in the regular path direction was similar with all methods, however the errors in the singular path directions were much smaller with the trapezoid methods, which means that the end effector moved much closer to a linear path than in the case of the explicit Euler method. This can be very useful if preserving the shape of the path is an important goal, e.g. when approaching an object in a narrow corridor or putting an object into a hole.

## VI. Conclusion

An iterative algorithm was given to solve the CLIK algorithm using implicit methods in the numerical integration process with update laws of the form (16). It was proved, that this update law has a unique fixed point and the iteration in Algorithm 2 converges to this fixed point (that is the implicit solution) if certain conditions hold. These conditions require that the Jacobian is regular (thus the $\infty$-norm of its pseudoinverse is bounded from above), the $\infty$-norm of the change of the Jacobian is bounded from above, the functions in the update law are continuously differentiable, and that the feedback gain parameter is chosen such that it satisfies the inequality (22). This inequality depends on the $\infty$-norm of the differential of the function $\psi$ in the update law (16) with respect to $\Delta\theta[k+1]$ that was shown to be a constant for the typical numerical methods, and it also depends on the $\infty$-norm of the implicit difference $\Delta\theta[k+1]$. This difference can be estimated if we know the initial path tracking error, the desired velocity and the bound on the norm of the pseudoinvese of the Jacobian.

Simulations showed that the improvement in the path tracking caused by the application of the second-order methods appeared in the directions orthogonal to the desired end effector movement if the desired path is a straight line in space. This means that the resulting end effector motion will be much closer to a straight motion if the second-order methods are applied which can be useful in many applications. Application of the implicit trapezoid method even gave better results than the explicit trapezoid method for values of feedback gain parameters that are not close to zero.

## Acknowledgements

The work is supported by the European Research Council Starting Grant ERC-StG 679681. The first author is also supported by a Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme, FP7-PEOPLE-2012-IRSES-316338.

## References

[1] R. M. Murray, S. S. Sastry, and Z. Li, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[2] J. Selig, *Geometric Fundamentals of Robotics (Second Edition)*. Springer, 2005.

[3] L. Sciavacco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988.

[4] B. Siciliano, L. Sciavicco, S. Chiaverini, P. Chiacchio, L. Villani, and F. Caccavale, "Jacobian-based algorithms: A bridge between kinematics and control," in *Proceedings of the Special Celebratory Symposium In the honor of Professor Bernie Roth's 70th Birthday*, 2003, pp. 4–35.

[5] T. Sugihara, "Solvability-unconcerned inverse kinematics by the Levenberg-Marquardt method," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[6] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, pp. 163–171, 1986.

[7] F. Caccavale, S. Chiaverini, and B. Siciliano, "Second-order kinematic control of robot manipulators with Jacobian Damped Least-Squares inverse: Theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 2, no. 3, pp. 188–194, 1997.

[8] P. Falco and C. Natale, "On the stability of closed-loop inverse kinematics algorithms for redundant robots," *IEEE Transactions on Robotics*, vol. 27, pp. 780 –784, 2011.

[9] E. Sariyildiz and H. Temeltas, "Performance analysis of numerical integration methods in the trajectory tracking application of redundant robot manipulators," *International Journal of Advanced Robotic Systems*, vol. 8, no. 5, pp. 25–38, 2011.

[10] D. A. Drexler, "Solution of the closed-loop inverse kinematics algorithm using the Crank–Nicolson method," in *Proceedings of the 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlány, Slovakia, January 2016.

[11] U. M. Ascher and C. Greif, *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, 2011.

[12] P. Érdi and J. Tóth, *Mathematical Models of Chemical Reactions. Theory and Applications of Deterministic and Stochastic Models*. Princeton, New Jersey: Princeton University Press, 1989.

[13] D. A. Drexler and J. Tóth, "Global controllability of chemical reactions," *Journal of Mathematical Chemistry*, pp. 1–24, 2016.

[14] D. A. Drexler and I. Harmati, "Regularized Jacobian for the differential inverse positioning problem of serial revolute joint manipulators," in *Proceedings of the IEEE International Symposium on Intelligent Systems and Informatics*, Subotica, Serbia, September 2013.