# Closed-Loop Inverse Kinematics Algorithm with Implicit Numerical Integration

**Dániel András Drexler**

Physiological Controls Research Center, Research and Innovation Center of Óbuda University, Óbuda University, Hungary
drexler.daniel@nik.uni-obuda.hu

*Abstract: The closed-loop inverse kinematics algorithm is a numerical approximation of the solution of the inverse kinematics problem, which is a central problem of robotics. The accuracy of this approximation, i.e. the convergence of the numerical solution to the real solution can be increased by increasing the value of a feedback gain parameter. However, this can lead to unstable operation if the stability margin is reached. The accuracy of the closed-loop inverse kinematics algorithm is increased here by replacing the numerical integration with second-order and implicit numerical integration techniques. The application of implicit Euler, explicit trapezoid, implicit trapezoid and the weighted average method is considered, and an iteration is presented to calculate the implicit solutions. Simulation results show that implicit second-order methods give the best results. However, they decrease the stability margin due to the iteration required to calculate the implicit solution. The stability margin of the algorithms with different numerical integration techniques is analyzed, and it turns out that the implicit trapezoid method has the most desirable properties.*

*Keywords: differential inverse kinematics; numerical integration; explicit Euler; implicit Euler; explicit trapezoid; implicit trapezoid; theta method; weighted average method*

## 1 Introduction

In robotics, finding the motion of the robot joints that results in the predefined motion of the end effector of the robot is a central problem, called the inverse kinematics problem. The solution of this problem lies in the inversion of nonlinear geometric transformations (that are nonlinear in the joint variables of the robot), and we can only find symbolic solution in special cases (for special robot architectures), see e.g. [1–3]. Thus in general, we need to use numerical techniques to solve the inverse kinematics problem; the most widely used approaches that can be used in real-time are based on the Jacobian of the robot [4]. This Jacobian defines a linear relationship for fixed joint variables between the joint velocities and the end effector velocities, thus given the desired end effector velocities, we can calculate the necessary joint velocities by solving a system of linear equations defined by the Jacobian.

In the implementation of the Jacobian-based inverse kinematics algorithm, first we discretize the problem in time, solve the linear system of equations, and finally integrate the joint variables using a numerical integration technique. The most common numerical integration technique used in the literature is the explicit Euler method (see e.g. [5–7]). The Jacobian-based solution of the inverse kinematics problem is usually called the differential inverse kinematics algorithm.

Since the differential inverse kinematics algorithm is a numerical approximation of the solution, it has a tracking error. The tracking performance can be improved by adding a constant multiple of the tracking error to the desired end effector velocity, this constant will be called the feedback gain, and the algorithm with nonzero feedback gain will be referred to as the closed-loop inverse kinematics (CLIK) algorithm (see e.g. [8]), being presented in Section 2. The tracking error can be decreased by increasing the value of the feedback gain. However, there is an upper limit for that depending on the sampling time (for the proof see [8]); if the feedback gain exceeds this limit, then the algorithm becomes unstable. The maximal value of this limit is $2/T_s$ with $T_s$ being the sampling time used at the discretization, if the initial tracking error is small enough, which implies that this also gives an upper limit for the tracking performance.

The tracking performance of the CLIK algorithm can be further increased if we replace the numerical integration with second-order methods, as it was shown, e.g. in [9–11]. Second-order, explicit trapezoid method was proposed in [9], while implicit methods with an algorithm to calculate the implicit solutions were proposed in [10, 11]. In [11], explicit Euler, implicit Euler, explicit trapezoid and implicit trapezoid methods were implemented and compared to each other, and it turned out that the trapezoid methods yield better performance, and the implicit trapezoid method has the best performance properties.

The implicit trapezoid method does the integration using the average of the velocities from the explicit and implicit Euler methods. This can be generalized further by using the convex combination of these velocities, this is called the $\vartheta$-method (or weighted average method) in the literature [12]. Note that these methods are usually used to solve partial differential equations, and for specific problems it was shown that among all convex combinations, the average gives the best tracking performance; we show the same here with simulations in Section 4.

The implicit Euler, explicit trapezoid, implicit trapezoid and $\vartheta$-methods and their application in the CLIK problem are explained in Section 3. The iterative algorithm to calculate the implicit solutions is also expounded in Section 3, where the conditions on the convergence of the iteration are also given. The application of the algorithms is demonstrated using simulations in Section 4, where the results from the known explicit and the proposed implicit methods are compared.

The maximal value of the feedback gain parameter in the CLIK algorithm is examined using simulations with different numerical integration techniques. It turns out that the stability margin is close to $2/T_s$ for the explicit and implicit trapezoid methods. However, the CLIK algorithm becomes unstable for smaller feedback gain if implicit Euler method is used for numerical integration. For the weighted average

method, the stability margin depends on the weighting parameter $\vartheta$. Theoretical upper limits for the stability margins are also given symbolically for these algorithms in Section 3 that are verified in Section 4 with the simulations. The simulation results show that the implicit trapezoid method has the best performance, and the stability margin for the feedback gain, if the implicit trapezoid method is used, is very close to the stability margin of the CLIK algorithm using the explicit Euler method that was used in [8].

# 2 Closed-Loop Inverse Kinematics Algorithm

We will denote the vector of joint variable functions with $\theta$, whose $i$th component $\theta_i$ is the joint variable of the $i$the joint of the manipulator. $\theta$ is a function that maps the value of the joint variables (angles or displacements depending on the type of the joints) for each positive time instant. We denote the forward kinematics mapping 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 $J(\theta)$, i.e. $\dot{x} = J(\theta)\dot{\theta}$.

We are looking for the joint variable function $\theta_d$ such that

$$f(\theta_d) = x_d \tag{1}$$

holds. This implies that for the velocities the expression

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

holds as well.

First, we discretize the problem in time, i.e. consider the functions $\theta$ and $x_d$ in discrete time instants $t = kT_s$ with $k$ being a nonnegative integer, while $T_s$ being the sampling time. Define the discretized functions as $\theta[k] := \theta(kT_s)$ and $x_d[k] := x_d(kT_s)$. After the discretization, 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

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

The differential inverse kinematics algorithm is based on solving 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 $\Delta\theta[k]$, followed by a numerical integration

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

to update the joint variable vector. The numerical integration used in (5) is the explicit Euler method [13]. Since (4) only describes velocities, but the goal is to track the desired path, i.e. to minimize the difference between the elements of the series $f(\theta[0]), f(\theta[1]), f(\theta[2]), \ldots$ and $x_d[0], x_d[1], x_d[2], \ldots$, it is necessary to take the tracking error $x_d[k] - f(\theta[k])$ into consideration as well. We add this feedback term after multiplication with the feedback parameter $\alpha$ to the desired velocities, so that (4) becomes

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

thus 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. This algorithm is called the CLIK algorithm that has better tracking performance than the differential inverse kinematics algorithm without the feedback term. The performance of the algorithm increases, i.e. the speed of the convergence of the series $f(\theta[0]), f(\theta[1]), f(\theta[2])$ to the series $x_d[0], x_d[1], x_d[2], \ldots$ becomes faster as $\alpha$ is increased, until the stability margin is reached, at which point the algorithm becomes unstable, thus the increase of the performance by changing the feedback parameter is limited. However, further increase in the tracking performance can be achieved by replacing the numerical integration step (5) by a different technique [9–11] that will be discussed in the upcoming sections.

We supposed that the task is to achieve the desired position and orientation of the end effector of the manipulator. However, the task can be more specific, for example only the position or the orientation of the end effector is considered, or the robot moves only in a plane (i.e. it is a planar manipulator). In this case, the functions $x_d$ and $f$ can be described such that they map to the space relevant to the specific task, we will call this space the task space. Similarly, we can define the Jacobian of the new function $f$ that maps to the task space, we will call this Jacobian the task Jacobian. The sum of the desired velocity and the feedback term defined in the task space will be called the task vector and denoted by $t(\theta[k], k)$, where the first argument means that the task vector is considered at joint variable $\theta[k]$, and the second argument shows that the desired $x_d$ and $\Delta x_d$ values are considered at the discrete time instant $k$, i.e.

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

The first step (7) of the CLIK algorithm is written with this terminology as

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

where $J$ is the task Jacobian. In the remaining sections we will suppose that the corresponding functions map to the task space, thus the application of the discussed methods does not depend on the task space.

# 3 Implicit and Second-Order Numerical Integration Algorithms

The tracking performance of the numerical solution of the inverse kinematics algorithm can be increased by using implicit or second-order numerical integration techniques instead of the explicit Euler method in (5). The authors in [9] considered the application of higher-order explicit methods. However, implicit methods may yield better tracking performance [10, 11], thus we consider mostly implicit methods here, i.e. the update law that replaces (5) 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 $\phi$. Some of the results (iteration to calculate the implicit solution, bounds for the convergence of the iteration) will be given for the general update law. However we will consider specific $\phi$ functions (standing for specific numerical integration techniques) in the simulations.

Note that an implicit update law depends on $\theta[k+1]$ that is the solution we are looking for (that makes the method implicit), and the underlying expressions are usually complex and nonlinear, so the update law can not be rearranged to express $\theta[k+1]$ explicitly, thus we need an iteration to calculate the implicit solution $\theta[k+1]$.

Suppose that the update law $\phi$ can be written in the form

$$\phi = \theta[k] + T_s \psi(\Delta\theta[k], \Delta\theta[k+1]). \tag{11}$$

Then an iteration to calculate the update law is shown in Algorithm 1 [11].

**Algorithm 1.** *Iteration for implicit solution with update law (11).*

1. *First, calculate $\Delta\theta[k]$ using the expression*

   $$\Delta\theta[k] = J^{\#}(\theta[k]) t(\theta[k], k) \tag{12}$$

   *and calculate $\Delta\theta[k+1]$ using the expression*

   $$\Delta\theta[k+1] = J^{\#}(\theta[k]) t(\theta[k], k+1), \tag{13}$$

   *and compute $\tilde{\theta}[k+1]$ using the update law (11).*

2. *Calculate the difference*

   $$\Delta\tilde{\theta}[k+1] = J^{\#}(\tilde{\theta}[k+1]) t(\tilde{\theta}[k+1], k+1). \tag{14}$$

3. *Update $\tilde{\theta}[k+1]$ using the expression (11) as*

   $$\tilde{\theta}[k+1] = \theta[k] + T_s \psi(\Delta\theta[k], \Delta\tilde{\theta}[k+1]). \tag{15}$$

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.*

The following theorem gives an upper bound for the feedback gain $\alpha$ so that Algorithm 1 is convergent [11], i.e. the alteration of $\tilde{\theta}[k+1]$ becomes arbitrarily small after the appropriate number of iterations.

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

$$\max_i \left\|\partial_{\theta_i} J(\theta)\right\|_{\infty} \leq \nu \tag{16}$$

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

$$\alpha < \frac{1}{T_s \left\|\partial_{\Delta\theta[k+1]} \psi(\Delta\theta[k], \Delta\theta[k+1])\right\|_{\infty}} - n\nu\eta \left\|\Delta\theta[k+1]\right\|_{\infty} \tag{17}$$

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 1 converges to this fixed point.

*Proof.* We will show that with the above conditions the mapping $\phi$ in (10) is a contraction mapping (in the $\infty$-norm), i.e. if the conditions of the theorem hold then there exists a number $0 \leq q < 1$ such that for all $a, b \in U$

$$\left\|\phi(a) - \phi(b)\right\|_{\infty} \leq q \left\|a - b\right\|_{\infty}. \tag{18}$$

Since $\psi$ in (11) 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 (18) is equivalent to the existence of a Lipschitz-constant $q$ such that

$$\left\|\partial_{\theta[k+1]} \phi\right\|_{\infty} \leq q. \tag{19}$$

We will show that $\left\|\partial_{\theta[k+1]} \phi\right\|_{\infty} < 1$ implies (17) if the other conditions of the theorem hold. Applying the chain rule, the differential $\partial_{\theta[k+1]} \phi$ can be written as

$$\partial_{\theta[k+1]} \phi = T_s \partial_{\Delta\theta[k+1]} \psi(\Delta\theta[k+1]) \partial_{\theta[k+1]} \Delta\theta[k+1], \tag{20}$$

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

$$\begin{aligned} \partial_{\theta_i} \Delta\theta &= \partial_{\theta_i} \left(J^{\#}(\theta) t(\theta, k+1)\right) \\ &= \left(\partial_{\theta_i} \left(J^{\#}(\theta)\right)\right) t(\theta, k+1) \\ &\quad + J^{\#}(\theta) \partial_{\theta_i} \left(t(\theta, k+1)\right). \end{aligned} \tag{21}$$

The differential of the task vector is

$$\begin{aligned}\partial_{\theta_i}\left(t(\theta,k+1)\right) &= \partial_{\theta_i}\left(\Delta x_d[k+1]+\alpha\left(x_d[k+1]-f(\theta)\right)\right)\\ &= -\alpha\partial_{\theta_i}f(\theta)\\ &= -\alpha J(\theta)(\cdot,i),\end{aligned} \tag{22}$$

where $J(\theta)(\cdot,i)$ denotes the $i$th column of the matrix $J(\theta)$ with the notation used, e.g. in [14, 15]. Substituting this into the second term in (21) yields

$$\begin{aligned}J^{\#}(\theta)\partial_{\theta_i}\left(t(\theta,k+1)\right) &= J^{\#}(\theta)\left(-\alpha J(\theta)(\cdot,i)\right)\\ &= -\alpha e_i,\end{aligned} \tag{23}$$

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

The differential of the Jacobian pseudoinverse is

$$\partial_{\theta_i}\left(J^{\#}(\theta)\right) = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)J^{\#}(\theta), \tag{24}$$

so the first term in (21) becomes

$$\left(\partial_{\theta_i}\left(J^{\#}(\theta)\right)\right)t(\theta,k+1) = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)\Delta\theta, \tag{25}$$

so (21) reduces to

$$\partial_{\theta_i}\Delta\theta = -J^{\#}(\theta)\left(\partial_{\theta_i}J(\theta)\right)\Delta\theta - \alpha e_i. \tag{26}$$

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

$$\|\partial_\theta\phi\|_\infty \leq T_s\,\|\partial_{\Delta\theta}\psi\|_\infty\,\|\partial_\theta\Delta\theta\|_\infty \tag{27}$$

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

$$\|\partial_\theta\Delta\theta\|_\infty = \max_i\left\{1_n\left|-\alpha e_i - J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right|\right\} \tag{28}$$

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

$$\|\partial_\theta\Delta\theta\|_\infty \leq \alpha + \max_i\left\{1_n\left|-J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right|\right\}, \tag{29}$$

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

$$\|\partial_\theta\Delta\theta\|_\infty \leq \alpha + n\max_i\left\|J^{\#}(\theta)\partial_{\theta_i}J(\theta)\Delta\theta\right\|_\infty \tag{30}$$

from which we obtain

$$\|\partial_\theta\Delta\theta\|_\infty \leq \alpha + n\left\|J^{\#}(\theta)\right\|_\infty\|\Delta\theta\|_\infty\max_i\partial_{\theta_i}J(\theta). \tag{31}$$

Substituting the bounds from the conditions of the theorem into (31), and substituting the result into (27) yields

$$\|\partial_\theta \phi\|_\infty \leq T_s \|\partial_{\Delta\theta} \psi\|_\infty (\alpha + n\nu\eta \|\Delta\theta\|_\infty). \tag{32}$$

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

$$\alpha < \frac{1}{T_s \|\partial_{\Delta\theta} \psi\|_\infty} - n\nu\eta \|\Delta\theta\|_\infty \tag{33}$$

that is the result we were looking for. □

The parameter $q$ in (18) characterizes the speed of convergence, since the distance of the implicit solution and the approximate solution in the $n$th iteration is

$$\|\Delta\theta[k+1] - \Delta\tilde{\theta}[k+1]^{(n)}\| \leq \frac{q^n}{1-q} \|\Delta\tilde{\theta}[k+1]^{(0)} - \Delta\tilde{\theta}[k+1]^{(1)}\| \tag{34}$$

where $\Delta\theta[k+1]$ is the implicit solution and $\Delta\tilde{\theta}[k+1]^{(n)}$ is the solution resulting from Algorithm 1 after $n$ number of iterations. Thus, the required number of iterations in Algorithm 1 depends on the value of $q$ that depends on the value of $\alpha$: if $\alpha$ is closer to the limit defined by Theorem 1, then $q$ is closer to 1, so more number of iterations are required to get solutions that are sufficiently close to the real solution.

Condition (17) of Theorem 1 contains the norm of the partial derivative of the function $\Psi$ in the update law that depends on the joint velocities. However, for specific numerical integration techniques $\left\|\partial_{\Delta\theta[k+1]} \psi\right\|_\infty$ is usually a constant as it will be shown in the upcoming subsections.

## 3.1 Implicit Euler Method

The update law in (10) in the case of implicit Euler integration becomes

$$\phi = \theta[k] + T_s \Delta\theta[k+1] \tag{35}$$

so the function $\psi$ in (11) is

$$\psi(\Delta\theta[k], \Delta\theta[k+1]) = \Delta\theta[k+1], \tag{36}$$

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

$$\left\|\partial_{\Delta\theta[k+1]} \psi\right\|_\infty = 1. \tag{37}$$

Thus, the iteration described in Algorithm 1 converges for feedback gain $\alpha$ that satisfies

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

that is much smaller than the limit $2/T_s$ for the explicit Euler integration (since the quantities in the second term on the right-hand side of this inequality are all positive, so the limit is less than $1/T_s$), so the application of the implicit Euler method makes the CLIK algorithm unstable for smaller $\alpha$ values than in the case of explicit Euler integration. Note that if the CLIK algorithm becomes unstable for $1/T_s < \alpha < 2/T_s$ (and it does not become unstable if explicit Euler integration is used) it is because the iteration for the implicit solution becomes unstable since the conditions of Theorem 1 does not hold.

## 3.2 Trapezoid Methods

The update law in the case of trapezoid methods is the average of the explicit and implicit difference, i.e.

$$\phi = \theta[k] + \frac{1}{2} T_s \left(\Delta\theta[k] + \Delta\theta[k+1]\right). \tag{39}$$

The trapezoid method is called the explicit trapezoid method if the solution is approximated without iteration, so only the first step of Algorithm 1 is carried out.

If the iteration in Algorithm 1 is used, then the trapezoid method is called the implicit trapezoid method, and if the iteration is convergent, then the result of the algorithm converges to the implicit solution. The function $\psi$ in the update law is

$$\psi(\Delta\theta[k], \Delta\theta[k+1]) = \frac{1}{2}\left(\Delta\theta[k] + \Delta\theta[k+1]\right) \tag{40}$$

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

$$\left\| \partial_{\Delta\theta[k+1]} \psi \right\|_\infty = \frac{1}{2}. \tag{41}$$

Thus the upper bound for the feedback gain $\alpha$ from the condition of Theorem 1 is

$$\alpha < \frac{2}{T_s} - n\nu\eta \left\|\Delta\theta\right\|_\infty \tag{42}$$

that is smaller than the limit $2/T_s$. However, it can be close to that limit if the second term on the right-hand side of the inequality is small enough.

## 3.3 Weighted Average or $\vartheta$-Method

The update law in the case of the weighted average method is

$$\phi = \theta[k] + T_s\left((1-\vartheta)\Delta\theta[k] + \vartheta\Delta\theta[k+1]\right) \tag{43}$$

with $\vartheta \in [0,1]$, thus, the function $\psi$ becomes

$$\psi(\Delta\theta[k], \Delta\theta[k+1]) = (1-\vartheta)\Delta\theta[k] + \vartheta\Delta\theta[k+1]. \tag{44}$$

Note that the three special cases of the weighted average method are

- $\vartheta = 0$ that corresponds to the explicit Euler method;
- $\vartheta = 1/2$ that corresponds to the implicit trapezoid method;
- $\vartheta = 1$ that corresponds to the implicit Euler method.

The $\infty$-norm of the derivative of (44) with respect to $\Delta\theta[k+1]$ is

$$\left\|\partial_{\Delta\theta[k+1]}\psi\right\|_\infty = \vartheta. \tag{45}$$

Thus, the upper bound for the feedback gain $\alpha$ from the condition of Theorem 1 is

$$\alpha < \frac{1}{T_s\vartheta} - n\nu\eta\,\|\Delta\theta\|_\infty \tag{46}$$

that depends on the value of $\vartheta$. Note that as $\vartheta \to 0$, the result converges to the explicit Euler solution, and for the explicit Euler solution no iteration is needed, thus, the range of the convergence for the iteration tends to infinity. This inequality also shows that choosing $\vartheta < 0.5$ may give a bound for $\alpha$ that is higher than $2/T_s$, so the stability of the CLIK algorithm is not harmed. However, as we will see in the following section, $\vartheta = 0.5$ (which special case corresponds to the implicit trapezoid method) gives the best tracking performance, and different $\vartheta$ values result in worse tracking performance.

# 4 Simulation Results

The numerical integration techniques are tested on a benchmark problem where the solution of the inverse positioning problem of an elbow manipulator is considered; the manipulator consists of three revolute joints, the first two joint axes intersect each other and are perpendicular, while the second and third joint axes are parallel. This architecture is widely used 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

$$\omega_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \omega_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \omega_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tag{47}$$

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

$$q_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad q_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad q_3 = \begin{pmatrix} 0 \\ 0 \\ l_1 \end{pmatrix}, \tag{48}$$

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

$$p(0) = \begin{pmatrix} 0 \\ 0 \\ l_1 + l_2 \end{pmatrix} \tag{49}$$

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, 16].

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

$$x_d[k] = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} + \frac{k}{N} \begin{pmatrix} 0 \\ 0.5 \\ -1 \end{pmatrix} \tag{50}$$

$$\Delta x_d[k] = \frac{1}{N} \begin{pmatrix} 0 \\ 0.5 \\ -1 \end{pmatrix}. \tag{51}$$

The number of iterations for the CLIK algorithm was $N = 30$, so in the simulations the discrete time steps $k = 0, 1, \ldots, N$ were considered, while the sampling time was $T_s = 0.1$ sec. The CLIK algorithm was solved for different $\alpha$ feedback gain parameters. The number of iterations in Algorithm 1 used for the implicit methods was chosen to depend on $\alpha$ as

$$M = \lfloor 5(1+\alpha) \rfloor \tag{52}$$

in order to ensure good convergence (note that as $\alpha$ increases and gets closer to the limit given in Theorem 1 the parameter $q$ in (18) gets close to 1, so the speed of convergence decreases, and more iterations are required). However, this influences the computation time, since the implicit methods require approximately $1+M$ times more computation than the explicit methods.

For every value of $\alpha$ the tracking error was calculated and transformed into a specific coordinate system 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 *sin*1.

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 *sin*2.

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 $\max \|e_{sin2}\|$. This way, we can characterize each simulation (i.e. the result of the CLIK algorithm for the desired path tracking task) with three numbers: the maximum absolute values of the tracking errors in the direction

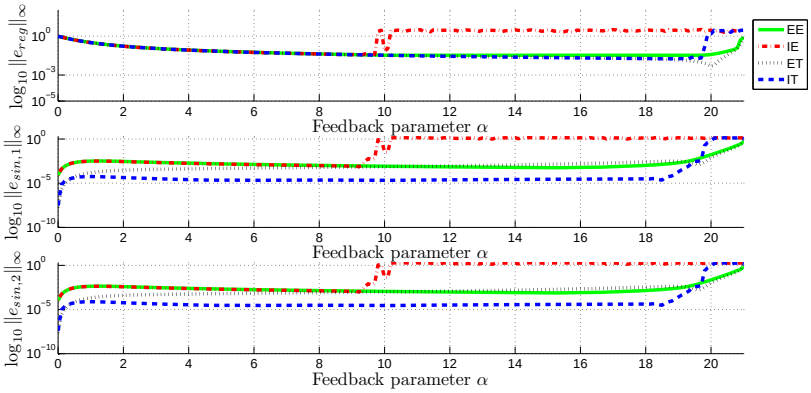


Figure 1
The logarithm of the maximal absolute values of the tracking errors along the regular and the two singular path directions for different values of the feedback gain parameter $\alpha$, with the application of the explicit Euler (EE), implicit Euler (IE), explicit trapezoid (ET) and implicit trapezoid methods (IT)

of the desired movement ( the regular path direction) and the two mutually orthogonal directions perpendicular to the direction of the desired movement (the singular path directions).

In the first case, the CLIK algorithm was solved using the explicit Euler [8], implicit Euler [10, 11], explicit trapezoid [9] and implicit trapezoid [10, 11] 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 $\alpha = 21$. Note that since the sampling time was $T_s = 0.1$ sec, the stability limit for $\alpha$ in the case of the explicit Euler method is $\alpha = 2/T_s = 20$, so the explicit Euler method must become unstable if $\alpha \geq 20$.

The logarithm of the $\infty$-norms of the different error components for different feedback parameters are in Fig. 1. The tracking errors start to increase at $\alpha = 9.3$ for the implicit Euler method, since the iteration in Algorithm 1 becomes unstable for that value of $\alpha$. The tracking errors start to increase for the implicit trapezoid method at $\alpha = 18.5$ for the same reason. The explicit methods provide stable operation for $\alpha < 20$ as expected.

All the four methods have similar performance in their stable region in the regular path direction, however there are differences in the singular path directions. The explicit and implicit Euler methods have same performance in the singular path directions as well until the implicit Euler method becomes unstable. For low values of $\alpha$, the explicit trapezoid method has better performance in the singular path directions than the Euler methods, however for $\alpha > 6$ its performance becomes similar to the performance of the Euler methods. The implicit trapezoid method outperforms all the other methods, and has better performance with at least two orders of magnitude than the other methods (except for $\alpha$ close to zero, in this case the

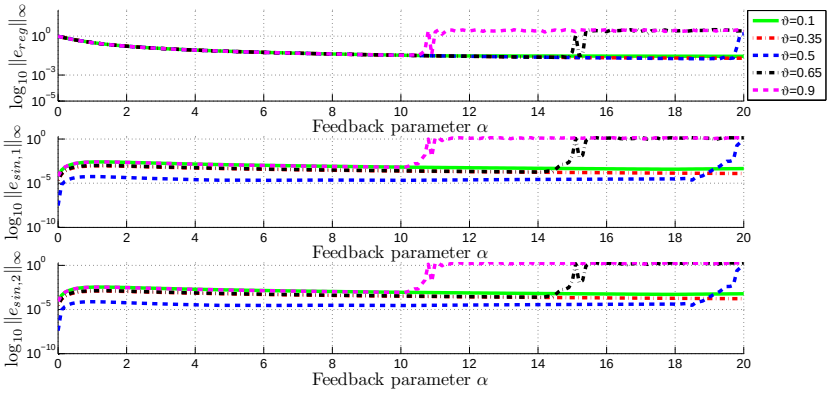


Figure 2
The logarithm of the maximal absolute values of the tracking errors along the regular and the two singular path directions for different values of the feedback gain parameter $\alpha$, with the application of the weighted average method with $\vartheta \in \{0.1, 0.35, 0.5, 0.65, 0.9\}$

explicit trapezoid method has similar performance) until it becomes unstable for $\alpha > 18.5$. Thus, the implicit trapezoid method has the best tracking performance in the singular path directions for a wide range of $\alpha$.

The CLIK algorithm was simulated with the application of the weighted average method as well, in order to find the optimal value of $\vartheta$ with which the algorithm has the best performance. The simulation results in this current situation showed that $\vartheta = 0.5$ has the best performance that corresponds to the implicit trapezoid method. The simulation results with $\alpha = 0, 0.1, 0.2, 0.3, \ldots, 20$ with five different $\vartheta$ values, i.e. $\vartheta \in \{0.1, 0.35, 0.5, 0.65, 0.9\}$ are in Fig. 2. Note that the results for the other values of $\vartheta$ are not depicted so that the data on the figure remain interpretable.

The figure shows that the $\vartheta = 0.5$ choice (i.e. the implicit trapezoid method) gave the best tracking performance. The differences are not relevant in the regular path direction, however the $\vartheta = 0.5$ solution has much better performance in the singular path directions. As the distance of the parameter $\vartheta$ from 0.5 increases, the tracking error in the singular path directions increases as well. For the pairs $\vartheta = \{0.1, 0.9\}$ and $\vartheta = \{0.35, 0.65\}$ (i.e. whose difference from 0.5 is the same), the performance is same in their stable region. For $\vartheta < 0.5$ the iteration in Algorithm 1 did not become unstable as it can be observed from (46) and since the tracking errors did not increase as the value $\alpha = 20$ was approached. However, for $\vartheta > 0.5$ the tracking errors started to increase for $\alpha < 20$ because the iteration for the implicit solution became unstable, i.e. for $\vartheta = 0.65$ the tracking error starts to increase at $\alpha = 14.5$ and for $\vartheta = 0.9$ the tracking error starts to increase at $\alpha = 10.2$. Note that as $\vartheta \to 1$, the $\vartheta$-method tends to the implicit Euler method, and the value of $\alpha$ where the iteration becomes unstable tends to $\alpha = 9.3$, the stability margin for the implicit Euler method. Thus, the simulation results showed that the $\vartheta = 0.5$ choice gives the best results.

## Conclusions

Application of second-order and implicit numerical integration methods in the CLIK algorithm were presented and the effect of the different integration methods on the tracking performance were examined. It turned out that the second-order methods give better tracking performance than the first order methods, and the implicit trapezoid method has the best performance. In order to explore if the implicit solution can be used more efficiently by taking its convex combination with the explicit solution, the application of the weighted average method has been considered as well and it turned out that the implicit trapezoid method (that is a special case of the weighted average method) has the best performance among all the possible choices.

An iteration was presented to calculate the implicit solutions, and the region of convergence for the iteration was given. Symbolic calculations showed that the implicit methods decrease the range of stability of the CLIK algorithm which was verified by simulations as well. Simulation results showed that the decrease of the stability margin in the case of implicit trapezoid method is small. However, the implicit Euler method greatly decreases the stability margin, while the decrease of the stability margin for the weighted average method depends on the parameter $\vartheta$.

The results clearly show that the tracking performance of the CLIK algorithm can be increased by replacing the first-order numerical integration technique with a second-order one. Moreover, simulation results showed that implicit second-order methods give the best performance. The drawback of the implicit methods is that they require iteration to calculate the implicit solution that decreases the stability margin. As $\alpha$ tends to the stability margin for the iteration required to calculate the implicit solution, the number of required iterations increases as well. However, the results demonstrated that the decrease in the stability margin is relatively small, i.e. it does not affect the utility of the results, moreover, the increase in the computation time is only linear, thus do not harm real-time implementation criteria, while the increase in the tracking performance is significantly larger.

## References

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

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

[3] Lorenzo Sciavacco and Bruno Siciliano. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 4(4):403–410, 1988.

[4] Bruno Siciliano, Lorenzo Sciavicco, Stefano Chiaverini, Pasquale Chiacchio, Luigi Villani, and Fabrizio 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*, pages 4–35, 2003.

[5] Tomomichi Sugihara. Solvability-unconcerned inverse kinematics by the Levenberg-Marquardt method. *IEEE Transactions on Robotics*, 27(5):984–991, 2011.

[6] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, 1986.

[7] Fabrizio Caccavale, Stefano Chiaverini, and Bruno Siciliano. Second-order kinematic control of robot manipulators with Jacobian Damped Least-Squares inverse: Theory and experiments. *IEEE/ASME Transactions on Mechatronics*, 2(3):188–194, 1997.

[8] Pietro Falco and Ciro Natale. On the stability of closed-loop inverse kinematics algorithms for redundant robots. *IEEE Transactions on Robotics*, 27:780 –784, 2011.

[9] Emre Sariyildiz and Hakan Temeltas. Performance analysis of numerical integration methods in the trajectory tracking application of redundant robot manipulators. *International Journal of Advanced Robotic Systems*, 8(5):25–38, 2011.

[10] Dániel 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] Dániel A. Drexler and Levente Kovács. Second-order and implicit methods in numerical integration improve tracking performance of the closed-loop inverse kinematics algorithm. In *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3362–3367, 2016.

[12] Gordon D. Smith. *Numerical solution of partial differential equations: finite difference methods, Oxford applied mathematics and computing science series*. Oxford University Press, 1985.

[13] Uri M. Ascher and Chen Greif. *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, 2011.

[14] Péter Érdi and János Tóth. *Mathematical Models of Chemical Reactions. Theory and Applications of Deterministic and Stochastic Models*. Princeton University Press, Princeton, New Jersey, 1989.

[15] Dániel A. Drexler and János Tóth. Global controllability of chemical reactions. *Journal of Mathematical Chemistry*, 54:1327–1350, 2016.

[16] Dániel A. Drexler and István 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.