Mechanism and Machine Theory 74 (2014) 310–318

Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmt

Solving inverse kinematics by fully automated planar curves intersecting

Tomasz Rudny *

Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland

ARTICLE INFO

Article history:
Received 3 October 2012
Received in revised form 22 December 2013
Accepted 28 December 2013
Available online xxxx

Keywords:
Inverse kinematics
Serial manipulators
Planar curves
Bernstein elimination

ABSTRACT

This paper presents a new method for solving the Inverse Kinematics Problem of general 6-DOF serial manipulators. This problem has been a fundamental research area for the last 4 decades as it has numerous applications. Granted, many methods have been developed, but none of them is at the same time fast, accurate, numerically stable for an arbitrary end-effect or pose and capable of finding all solutions to the problem. The inverse kinematics problem can be reduced to finding intersections between planar curves, as it was shown by Jorge Angeles. Here we present a new way of transforming those curves into bivariate polynomials without the application of half-tan substitution and then of finding their intersections numerically using Bernstein elimination. It is achieved without any human interaction, so the method is fully automated, thus suitable for applications.

1. Introduction

1.1. Background

Serial manipulators play an important part in applications — NC milling, robot control, computer graphics and animation. Kinematic chains – the mathematical notation of serial manipulators – can be used not only in industry, but also to describe various objects of nature e.g. skeletons. Manipulators consist of links which are connected with joints. Here, we consider joints of only two types — revolute and prismatic, allowing to rotate or slide correspondingly a link against a given axis. All other types of joints can be described as a composition of (possibly many) joints of those two simple types.

Following a Denavit–Hartenberg (D–H) notation, each joint can be described by four parameters. Their values fully determine the transformation needed to transform a local Cartesian coordinate frame associated with the link L_i into the frame associated with the next link L_{i+1}. For a revolute joint one D–H parameter, the angle θ_i is the joint variable, while for a prismatic joint it is the distance between some joints axes — d_i.

When the joints variables are given for a kinematic chain, then the pose of its last link, the end-effector can be easily found by simple matrix multiplication. However, the inverse is not true: finding all joints variables when knowing the pose of the end-effector is a difficult computational problem.

Practical applications require that a method for solving the inverse kinematics problem is:

1 capable of finding all solutions to the IKP for the given pose of the end effector,
2 efficient i.e. with solve time acceptable in CAD/CAM and possibly in robot control,
3 accurate i.e. the error of the found solutions approximations should not be greater than 0.001 rad or 0.001 mm,
4 numerically stable i.e. should converge for any pose of the end effector (or return error message for singular configurations).

* Tel./fax: +48 61 278 64 18.
E-mail address: rudnyt@mini.pw.edu.pl.

0094-114X/$ – see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mechmachtheory.2013.12.020

In this paper we present such method.

1.2. Existing methods

Inverse kinematics problem has been studied by many great researches for the last 40 years. The first discoveries were made by Pieper ([13]) and later by Freudenstein. In 1972 Roth et al. proved that a 6R open manipulator can admit at most 32 poses. In 1980 Duffy and Crane found a way to transform the IKP into the problem of finding roots of a polynomial of degree 32. In 1985 Tsai proved that for a general 6R manipulator the number of solution is at most 16. In 1989 Raghavan and Roth found a polynomial of degree 16 describing the IKP [14]. In 1992 Manocha and Canny ([9]) modified this method to look for eigenvalues of a companion matrix instead, which improved stability and accuracy of the method.

Since then many methods have been designed and tested, including optimization methods ([16,19]), neural networks ([8]), interval algebra ([4]), genetic algorithms ([17]) to name just a few. Many of the new methods focus on deriving the polynomials in a more efficient way or one that reveals the geometric structure of the problem ([7]).

Jorge Angeles noticed that instead of deriving a univariate polynomial, a set of 4 bivariate equations in variables θ_4, θ_5 may be obtained. These define 4 contours on the $\theta_4 - \theta_5$ plane.[1] The common intersection points of these correspond to the solutions of the IKP. Angeles developed a GUI in Matlab that allowed plotting these contours. A user would then localize intersections with a mouse. These approximations were then improved using a Newton method.

A comprehensive discussion of planar curves in computations can be found in [6]. The comparison of the efficiency of different implicit curve plotting methods based on interval arithmetics, Bernstein coefficient, continuation and others is given in [10]. These methods can be also applied to curves intersecting. However, none of these general methods provide needed accuracy within the needed time.

One key observation is that curves intersecting is much faster for polynomial curves.

2. Inverse kinematics as planar curves intersecting

2.1. Transforming equations

Let the elements of the matrix describing the pose of the end-effector be given by:

$$\mathbf{A}_{hand} = \begin{bmatrix} l_x & m_x & n_x & p_x \\ l_y & m_y & n_y & p_y \\ l_z & m_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

This matrix subjects to the matrix equation:

$$\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\mathbf{A}_6 = \mathbf{A}_{hand} \tag{2}$$

or equivalently:

$$\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5 = \mathbf{A}_2^{-1}\mathbf{A}_1^{-1}\mathbf{A}_{hand}\mathbf{A}_6^{-1}. \tag{3}$$

Now taking the 3rd and 4th columns of the matrix Eq. (3) we can write new equations ([14,1]):

$$\overline{\mathbf{f}} = \overline{\mathbf{g}} \tag{4}$$

$$\overline{\mathbf{h}} = \overline{\mathbf{i}} \tag{5}$$

where:

$$\overline{\mathbf{f}} \equiv \begin{bmatrix} 1 & 0 & 0(6) \\ 0 & -\lambda_2 & \mu_2(7) \\ 0 & \mu_2 & \lambda_2 \end{bmatrix} \left(\begin{bmatrix} \mathbf{c}_3 & \mathbf{s}_3 & 0(8) \\ \mathbf{s}_3 & -\mathbf{c}_3 & 0(9) \\ 0 & 0 & 1 \end{bmatrix} \widetilde{\mathbf{f}} + \begin{bmatrix} \mathbf{a}_2(10) \\ \mathbf{d}_2\mu_2(11) \\ \mathbf{d}_2\lambda_2 \end{bmatrix} \right) \tag{6}$$

$$\overline{\mathbf{g}} \equiv \begin{bmatrix} \mathbf{c}_2 & \mathbf{s}_2 & 0(13) \\ \mathbf{s}_2 & -\mathbf{c}_2 & 0(14) \\ 0 & 0 & 1 \end{bmatrix} \widetilde{\mathbf{h}} \tag{7}$$

$$\overline{\mathbf{h}} \equiv \begin{bmatrix} 1 & 0 & 0(16) \\ 0 & -\lambda_2 & \mu_2(17) \\ 0 & \mu_2 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{c}_3 & \mathbf{s}_3 & 0(18) \\ \mathbf{s}_3 & -\mathbf{c}_3 & 0(19) \\ 0 & 0 & 1 \end{bmatrix} \widetilde{\mathbf{r}} \tag{8}$$

$$\bar{\mathbf{i}} \equiv \begin{bmatrix} \mathbf{c}_2 & \mathbf{s}_2 & 0(21) \\ \mathbf{s}_2 & -\mathbf{c}_2 & 0(22) \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{n}} \tag{9}$$

and the components of vectors $\tilde{\mathbf{f}}$, $\tilde{\mathbf{h}}$, $\tilde{\mathbf{r}}$ and $\tilde{\mathbf{n}}$ are given in Table 1.

Using special properties of vectors $\bar{\mathbf{f}}$, $\bar{\mathbf{g}}$, $\bar{\mathbf{h}}$ and $\bar{\mathbf{i}}$ which were rigorously proven by [14] four new vector equations can be written:

$$\bar{\mathbf{f}} \cdot \bar{\mathbf{f}} = \bar{\mathbf{g}} \cdot \bar{\mathbf{g}} \tag{10}$$

$$\bar{\mathbf{f}} \cdot \bar{\mathbf{h}} = \bar{\mathbf{g}} \cdot \bar{\mathbf{i}} \tag{11}$$

$$\bar{\mathbf{f}} \times \bar{\mathbf{h}} = \bar{\mathbf{g}} \times \bar{\mathbf{i}} \tag{12}$$

$$\left(\bar{\mathbf{f}} \cdot \bar{\mathbf{f}}\right)\bar{\mathbf{h}} - 2\left(\bar{\mathbf{f}} \cdot \bar{\mathbf{h}}\right)\bar{\mathbf{f}} = (\bar{\mathbf{g}} \cdot \bar{\mathbf{g}})\bar{\mathbf{i}} - 2\left(\bar{\mathbf{g}} \cdot \bar{\mathbf{i}}\right)\bar{\mathbf{g}} \tag{13}$$

Together with Eq. (4) we obtain 14 scalar equations.

The special structure of these equations allows to cast them in a matrix form:

$$\bar{\mathbf{P}}\mathbf{x}_{45} = \bar{\mathbf{R}}\mathbf{x}_{12} \tag{14}$$

where:

$$\mathbf{x}_{45} = [\, s_4 s_5 \quad s_4 c_5 \quad c_4 s_5 \quad c_4 c_5 \quad s_4 \quad c_4 \quad s_5 \quad c_5 \quad 1 \,]^T \tag{15}$$

$$\mathbf{x}_{12} = [\, s_1 s_2 \quad s_1 c_2 \quad c_1 s_2 \quad c_1 c_2 \quad s_1 \quad c_1 \quad s_2 \quad c_2 \,]^T. \tag{16}$$

Thanks to the special structure of the equations the 14×9 matrix $\bar{\mathbf{P}}$ has all elements linear in $\mathbf{x}_3 = \begin{bmatrix} c_3 \\ s_3 \end{bmatrix}$, while the 14×8 matrix $\bar{\mathbf{R}}$ has all elements independent of the unknown variables.

This allows to rewrite Eq. (14) into two groups of six and eight equations respectively:

$$\mathbf{P}_\mathbf{u}\mathbf{x}_{45} = \mathbf{C}\mathbf{x}_1 \tag{17}$$

$$\mathbf{P}_\mathbf{l}\mathbf{x}_{45} = \mathbf{A}\tilde{\mathbf{x}}_{12} \tag{18}$$

Table 1
Components of vectors used in equations.

Component	Expression
f_1	$c_4 l_1 + s_4 l_2 + a_3$
f_2	$-\lambda_3(s_4 l_1 - c_4 l_2) + \mu_3 l_3 + d_3$
f_3	$\mu_3(s_4 l_1 - c_4 l_2) + \lambda_3 l_3 + d_3$
l_1	$c_5 a_5 + a_4$
l_2	$-s_5\lambda_4 a_5 + \mu_4 d_5$
l_3	$s_5\mu_4 a_5 + \lambda_4 d_5 + d_4$
h_1	$c_1 p + s_1 q - a_1$
h_2	$-\lambda_1(s_1 p - c_1 q) + \mu_1(r - d_1)$
h_3	$\mu_1(s_1 p - c_1 q) + \lambda_1(r - d_1)$
p	$-l_x a_6 - (m_x\mu_6 + n_x\lambda_6)d_6 + p_x$
q	$-l_y a_6 - (m_y\mu_6 + n_y\lambda_6)d_6 + p_y$
r	$-l_z a_6 - (m_z\mu_6 + n_z\lambda_6)d_6 + p_z$
r_1	$c_4 m_1 + s_4 m_2$
r_2	$-\lambda_3(s_4 m_1 - c_4 m_2) + \mu_3 m_3$
r_3	$\mu_3(s_4 m_1 - c_4 m_2) + \lambda_3 m_3$
m_1	$s_5\mu_5$
m_2	$c_5\lambda_4\mu_5 + \mu_4\lambda_5$
m_3	$-c_5\mu_4\mu_5 + \lambda_4\lambda_5$
n_1	$c_1 u + s_1 v$
n_2	$-\lambda_1(s_1 u - c_1 v) + \mu_1 w$
n_3	$\mu_1(s_1 u - c_1 v) + \lambda_1 w$
u	$m_x\mu_6 + n_x\lambda_6$
v	$m_y\mu_6 + n_y\lambda_6$
w	$m_z\mu_6 + n_z\lambda_6$

where C is a 6 × 2 constant matrix formed by the nonzero elements in rows 3, 6, 7, 8, 11 and 14 of the matrix $\overline{\mathbf{R}}$, and similarly $\mathbf{P_u}$ is made of rows 3, 6, 7, 8, 11 and 14 of the $\overline{\mathbf{P}}$ matrix.

Now the proposed approach is to take any two from the first six equations, solve them for $\mathbf{x}_1 = \begin{bmatrix} c_1 \\ s_1 \end{bmatrix}$, then substitute it back to the remaining four equations. To accomplish this we first partition the six scalar equations into:

$$\mathbf{C_u}\mathbf{x}_1 = \mathbf{d_u} \tag{19}$$

$$\mathbf{C_l}\mathbf{x}_1 = \mathbf{d_l} \tag{20}$$

where $\mathbf{C_u}$, $\mathbf{C_l}$ are the 2 × 2 and 4 × 2 upper and lower parts of the C matrix, and $\mathbf{d_u}$, $\mathbf{d_l}$ the corresponding vectors made from the appropriate entries of $\overline{\mathbf{P}}_\mathbf{u}\mathbf{x}_{45}$.

Having solved for $\mathbf{x}_1$ the remaining four equations, free from θ_1 and θ_2 are given:

$$\mathbf{C_l}\mathbf{C_u}^{-1}\mathbf{d_u} - \mathbf{d_l} = 0_4. \tag{21}$$

Finally, the above equations can be rewritten as:

$$\mathbf{D}_1\mathbf{y}_3 = 0_4 \tag{22}$$

where $\mathbf{D}_1$ being a 4 × 3 matrix with all elements bilinear in s_4, c_4, s_5, c_5, and $\mathbf{y}_3$ being 3-dimensional vector defined as:

$$\mathbf{y}_3 \equiv [\, c_3 \quad s_3 \quad 1 \,]^T. \tag{23}$$

Here Angeles suggested to quit following Raghavan and Roth procedure which later leads to a univariate polynomial of degree 16, but rather to notice that for the solutions to exist the $\mathbf{D}_1$ matrix must be rank-deficient. This means that every of its four determinants must be zero. Thus, four equations in the unknown variables θ_4, θ_5 are obtained. They define four contours on a $\theta_4 - \theta_5$ plane.

$$F_i(\theta_4, \theta_5) = 0, i = 1, 2, 3, 4. \tag{24}$$

In order to obtain polynomial curves we now propose to substitute for any k, j:

$$c_4^{2k+1} = c_4 c_4^{2k}$$

$$c_5^{2j+1} = c_5 c_5^{2j}$$

then using the identities:

$$c_4^2 = 1 - s_4^2$$

$$c_5^2 = 1 - s_5^2$$

we obtain equations that are linear in c_4, c_5. From any of the 4 equations we calculate c_4 and put it into the remaining 3 equations. Now, since the expression for c_4 is in the form $\frac{\varsigma}{\xi}$, where ς, ξ are functions of c_5, s_4, s_5, hence the resulting equations will also be in the fraction form. However, assuming $\xi \neq 0$ the equations are reduced to comparing their numerators with 0.

Now in a similar way c_5 is calculated from any of the 3 equations and put into the remaining 2 equations. After expanding and simplifying a set of 2 polynomial equations in 2 variables s_4, s_5 is obtained. These are the new contours on the $s_4 - s_5$ plane.

$$\hat{F}_i(s_4, s_5) = 0, \quad i = 1, 2. \tag{25}$$

Theoretically in the above bivariate polynomials the degree of variable s_4 is 20 and s_5 is 24. This is because we start with (38) where the highest power of s_4 and s_5 is 3, since these are generated as determinants of a 3 × 3 matrix which is linear in those variables. Applying trigonometrical identities to 4, Eq. (24) elevates the degree of s_4 to 5. Calculating c_4 and putting it into the remaining 3 equations elevates the degree of s_4 to 10 and of s_5 and c_5 to 6. Applying trigonometrical identities to c_5 elevates its degree to 12. Finally, calculating c_5 and putting it into the remaining 2 equations may yield the powers of 20 and 24 of variables s_4 and s_5.

However, these theoretical estimations have never been seen in the numerical experiments. Even for manipulators with all non-zero terms in Eq. (14) the degree of polynomials (Eq. (25)) was at most 11. Most likely the specific structure of the matrix $\mathbf{D}_1$ prevents the degree from reaching the limits given above.

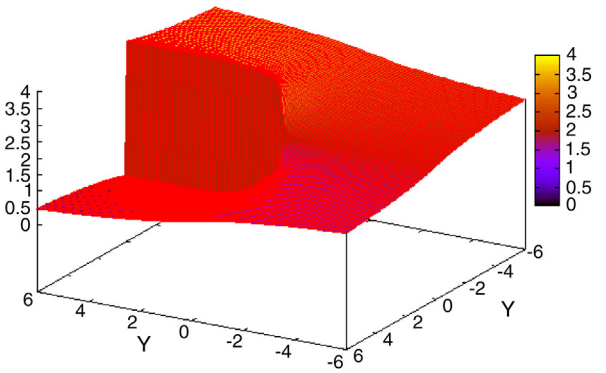

Fig. 1. α_{FAM} – the angle between a vector $\mathbf{v} = [x,y]$ and the X axis.

2.2. Finding the intersections

Now the 2 Eq. (25) can be expressed in the Bernstein basis (also in the matrix form):

$$\hat{F}(s_4, s_5) = \mathbf{B}_{\mathbf{s}_4}\mathbf{M}\mathbf{B}_{\mathbf{s}_5},$$

where the domain is:

$$s_4 \in \left[s_4^{min}, s_4^{max}\right], \quad s_5 \in \left[s_5^{min}, s_5^{max}\right].$$

Of course we start with: $[s_4^{min}, s_4^{max}] = [-1, 1]$, $[s_5^{min}, s_5^{max}] = [-1, 1]$.
$\mathbf{B}_{\mathbf{s}_4}, \mathbf{B}_{\mathbf{s}_5}$ are vectors of Bernstein basis functions:

$$\mathbf{B}_{\mathbf{s}_4} = \begin{bmatrix} B_0^n(s_4) & \dots & B_n^n(s_4) \end{bmatrix}$$
$$\mathbf{B}_{\mathbf{s}_5} = \begin{bmatrix} B_0^n(s_5) & \dots & B_n^n(s_5) \end{bmatrix}$$

where:

$$B_k^n(s_4) = \binom{n}{k}\left(\frac{s_4 - s_4^{min}}{s_4^{max} - s_4^{min}}\right)^k \left(1 - \left(\frac{s_4 - s_4^{min}}{s_4^{max} - s_4^{min}}\right)\right)^{n-k}, \quad \forall s_4 \in [s_4^{min}, s_4^{max}]$$
$$B_k^n(s_5) = \binom{n}{k}\left(\frac{s_5 - s_5^{min}}{s_5^{max} - s_5^{min}}\right)^k \left(1 - \left(\frac{s_5 - s_5^{min}}{s_5^{max} - s_5^{min}}\right)\right)^{n-k}, \quad \forall s_5 \in [s_5^{min}, s_5^{max}].$$

The matrix of coefficients in Bernstein basis $\mathbf{M}$ can be calculated from the matrix of coefficients in power basis $\mathbf{P}$ as (see [2] for proof):

$$\mathbf{M} = \left(\mathbf{U}_{s_4}\right)^{-1}\left(\mathbf{V}_{s_4}\right)^{-1}\left(\mathbf{W}_{s_4}\right)^{-1}\mathbf{P}\left(\mathbf{W}_{s_5}\right)^{-1}\left(\mathbf{V}_{s_5}\right)^{-1}\left(\mathbf{U}_{s_5}\right)^{-1}, \tag{26}$$

where:

$$\mathbf{U}_{s_4} = \begin{bmatrix} 1 & & & 0 \\ \binom{n}{0}\binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & & \\ \vdots & & \ddots & \\ \binom{n}{0}\binom{n}{n}(-1)^1 & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \dots & \binom{n}{n}\binom{n-n}{0}(-1)^0 \end{bmatrix}$$
$$\mathbf{V}_{s_4} = Diag\left(\frac{1}{s_4^{max} - s_4^{min}} \dots \frac{1}{(s_4^{max} - s_4^{min})^n}\right)$$
$$\mathbf{W}_{s_4} = \begin{bmatrix} 1 & \binom{1}{0}\left(-s_4^{min}\right)^1 & \dots & \binom{n}{0}\left(-s_4^{min}\right)^n \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \binom{n}{n}\left(-s_4^{min}\right)^{n-n} \end{bmatrix}.$$

Table 2
D–H parameters of Fanuc Arc Mate of 1990 manipulator.

i	a_i [mm]	d_i [mm]	α_i	θ_i
1	200	810	90°	θ_1
2	600	0	0°	θ_2
3	130	−30	90°	θ_3
4	0	550	90°	θ_4
5	0	100	90°	θ_5
6	0	100	0°	θ_6

Table 3
Valid solutions of the IKP for Fanuc Arc Mate of 1990.

θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
83.447917°	87.898526°	9.2685354°	−137.36737°	170.30092°	−42.221849°
85.417924°	16.156475°	153.21251°	−175.33825°	100.59664°	−0.85909878°
70.781671°	15.151453°	151.07728°	19.743721°	−102.98997°	175.38756°
83.366157°	90.974913°	−8.0041961°	136.45778°	−170.34612°	43.134322°

Matrices $\mathbf{U}_{s_5}, \mathbf{V}_{s_5}, \mathbf{W}_{s_5}$ over s_5 are defined similarly.

Bernstein basis has some useful properties (well described e.g. in [5]):

- The value of a polynomial in Bernstein basis is a convex combination of the basic functions and the coefficients of the combination are the elements of matrix **M** of the polynomial.
- Basis functions are non-negative within the domain and they sum up to 1.

A simple conclusion from the above properties is that *the value of a bivariate polynomial can be* $0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ *if and only if* 0 *lies inside the convex hull of points* $\mathbf{p}_{ij}$*, which are defined by the elements of matrices* **M** *for both polynomials*. Hence, if 0 does not lie within the convex hull of points $\mathbf{p}_{ij}$ it cannot be the value of the polynomial and there is no solution.

This observation leads to a numerical procedure known as *Bernstein elimination* or *Bezier clipping* ([3,15,12,18]).[1] The domain of the problem is recursively divided by de Casteljau subdivision (see e.g. [5]). Each new rectangular region is tested whether it contains solutions and depending on the result of the test it is discarded or further subdivided. The procedure stops when the size of the region bounding the solution is smaller than the set tolerance. The procedure quickly converges (for spaces of dimension 2 which was the reason to derive 2 equations in 2 variables instead of approaching 4 equations of Angeles directly) and localizes all solutions.

Moreover, the method is numerically stable. This is evident when trying to plot curves (39). Most of the algorithms for implicit curves plotting either fails or gives a very inaccurate approximation. In turn, Bernstein elimination finds the intersection points with an arbitrary accuracy.

In order to find out if a rectangular region $R = [s_4^{min}, s_4^{max}] \times [s_5^{min}, s_5^{max}]$ contains solutions we test if there exists an angle $\phi \leq 180^\circ$ based in 0 which contains all points $\mathbf{p}_{ij}$. The angle is given by angles between its edges and one of the axes of the coordinates system. The cost of this test is proportional to the number of points and thus is $O(N)$, which is much faster than e.g. Graham algorithm for finding the convex hull. The performance is increased also by using our own angle measure α_{FAM} which does not use trigonometric functions:

$$\alpha_{FAM}(x,y) = \begin{cases} \dfrac{y}{x+y}, & x>0, y \geq 0 \\ 3.0 + \dfrac{x}{x-y}, & x>0, y<0 \\ 1.0 + \dfrac{x}{x-y}, & x<0, y \geq 0 \\ 2.0 + \dfrac{y}{x+y}, & x<0, y<0 \\ 1.0, & x=0, y>0 \\ 3.0, & x=0, y<0 \\ ERROR, & x=0, y=0 \end{cases} \tag{27}$$

$\alpha_{FAM} \in [0,4]$. The shape of α_{FAM} resembles greatly arctan (see Fig. 1).

Having found the intersections of the curves one needs to calculate the remaining angles of the kinematic chain and verify if all intersection points correspond to solutions of the IKP. To do it well-known methods are applied (see ([1], chapter 9)).

[1] In [3] a different way of employing Bernstein elimination was used.

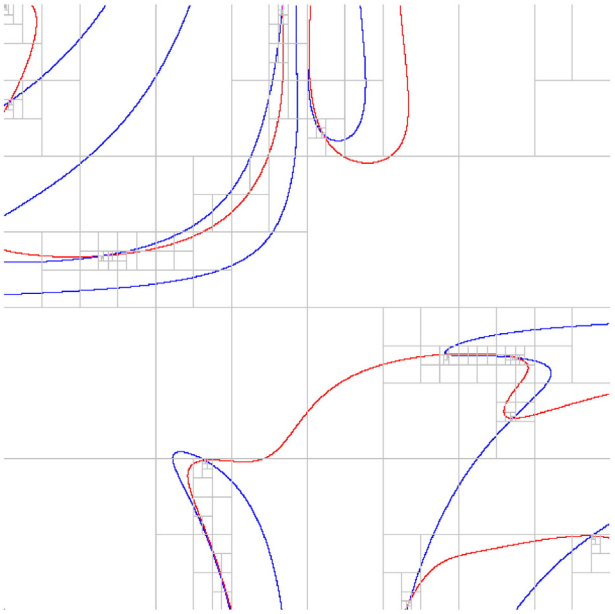

Fig. 2. Contours and Bernstein subdivisions for Fanuc Arc Mate of 1990 manipulator. On the horizontal axis there is s_4, while s_5 is on the vertical axis.

3. Implementation and results

3.1. Symbolic computations

Symbolic computations were implemented directly in C++ using the GiNaC library. The advantage of it over traditional approach of using Computer Algebra Systems (CAS) stems from the ability to use:

- complex data types,
- templates,
- collections like the C++ Standard Template Library,
- namespaces,
- Integrated Developer Environment (IDE), debugger,
- error handling mechanisms and exceptions.

Here we abandon the standard path of performing symbolic computations in preprocessing, then substituting the numerical values for symbols, because it proved to be too slow — the evaluation of equations coefficients takes approximately 3 s on an IBM PC 2 GHz CPU machine. Instead, substituting numerical values as soon as possible and then performing symbolic computations anew at each step takes on average 10 ms, which is acceptable in most applications.

3.2. Numerical examples

We tested our method against 2 classical examples well described in the literature. These were also used by Jorge Angeles ([1]) in his method which we here extended.

The first example is the IKP of Fanuc Arc Mate of 1990 manipulator. Its D–H parameters are presented in Table 2. The kinematics of this manipulator has non-trivial solutions and therefore is well suited as a benchmark of algorithms.

Table 4
D–H parameters of Li manipulator.

i	a_i [mm]	d_i [mm]	α_i	θ_i
1	120	0	−57°	θ_1
2	1760	890	35°	θ_2
3	70	250	95°	θ_3
4	880	−430	79°	θ_4
5	390	500	−75°	θ_5
6	930	−1340	−90°	θ_6

Table 5
Valid solutions of the IKP for Li manipulator.

θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
−46.013977°	−19.25665°	−46.988378°	−120.21832°	−145.86487°	−144.76892°
177.53854°	−148.17858°	159.42905°	−148.64748°	−129.27823°	110.98433°
174.083°	−163.30255°	−164.79177°	−107.81872°	−155.73819°	141.28129°
1.2264585°	−7.353213°	142.69698°	−123.87886°	−29.214528°	149.20796°
−22.26026°	−22.430884°	−32.024764°	−32.411449°	−172.61695°	−17.155638°
−173.92894°	150.69717°	47.811462°	−21.000605°	−40.438674°	−92.284209°
−22.602843°	28.094558°	98.631082°	−176.24585°	12.454973°	169.87873°
−8.0891076°	25.6552°	97.500427°	−175.83556°	15.920819°	166.27629°
164.80001°	−154.29075°	−85.341393°	4.7799298°	−127.8091°	−101.35935°
−41.684874°	−29.130121°	52.360732°	6.559396°	−129.12405°	25.0915°
−139.05934°	128.11274°	96.052116°	25.440661°	−7.3458003°	−119.83766°
−137.19515°	−156.92034°	68.306816°	135.68583°	−51.347828°	147.44646°
−83.094606°	57.022858°	130.97635°	67.570014°	−10.827516°	−110.98141°
−148.77535°	−179.71756°	−78.505632°	158.08613°	148.25398°	55.710955°
−159.84407°	−159.33594°	−111.34731°	120.27011°	176.59828°	21.67557°
−53.173674°	26.161489°	9.1052424°	145.86826°	136.35133°	127.98152°

We put the end-effector in the same pose as Angeles did:

$$\mathbf{A}_{hand} = \begin{bmatrix} 0 & 1 & 0 & 130 \\ 0 & 0 & 1 & 850 \\ 1 & 0 & 0 & 1540 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

After symbolic processing which took 8 ms (on an IBM PC, 2 GHz CPU machine), Bernstein elimination was launched. The tolerance was set to 9 digits. Bernstein elimination required 12 ms and 1465 subdivisions. 11 intersections were found, out of which only4 proved to be valid solutions of the IKP (see Table 3). The 2 contours and the subdivisions are shown in Fig. 2. It is interesting to see that in some areas the curves seem to overlap, but in fact Bernstein elimination proves otherwise. It is yet another proof of the excellent numerical stability of the method.

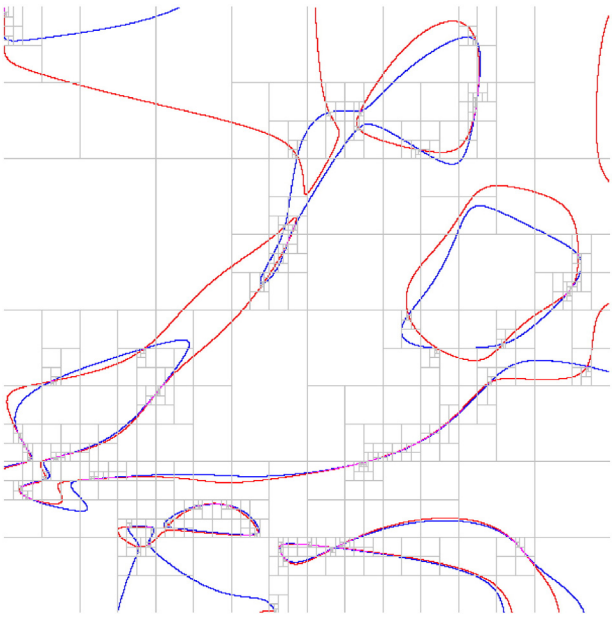

Fig. 3. Contours and Bernstein subdivisions for Li manipulator. On the horizontal axis there is s_4, while s_5 is on the vertical axis.

Next, we run the method against the Li manipulator which is designed to achieve the theoretical limit of 16 solutions (its D–H parameters are shown in Table 4). We put the end-effector at:

$$\mathbf{A}_{hand} = \begin{bmatrix} -0.357276 & -0.85000 & 0.387106 & -798.840 \\ 0.915644 & -0.237000 & 0.324694 & -0.331 \\ -01.84246 & 0.470458 & 0.862973 & 1200.658 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Bernstein elimination required 6113 subdivisions which took 26 ms and the tolerance was set to 9 digits. 51 intersections were found out, of which 16 were valid solutions of the IKP; they are presented in Table 5. The history of Bernstein elimination and the contours are shown in Fig. 3.

4. Conclusions

The presented method solves the inverse kinematics problem of serial 6-DOF manipulators for any non-singular pose of the end-effector. The efficiency and accuracy are high enough for applications in CAD/CAM and even in robot control. The hallmark of the method is its numerical stability through the properties of Bernstein basis of polynomials and not introducing the half-tan substitution.

References

[1] J. Angeles, Fundamentals of robotic mechanical systems: theory, Methods and Algorithms, Springer Books, 2005.
[2] J. Berchtold, I. Voiculescu, A. Bowyer, Multivariate Bernstein form polynomials, Technical Report 31, School of Mechanical Engineering, University of Bath, 1998.
[3] Bombin, C., Ros, L., Thomas, F. A Concise Bezier Clipping Technique for Solving Inverse Kinematics Problems, 53–60
[4] A. Castellet, F. Thomas, An algorithm for the solution of inverse kinematics problems based on an interval method, Advances in Robot Kinematics: Analysis and Control, 393–4021998.
[5] G. Farin, Curves and Surfaces for CADG, A Practical Guide, Fifth edition, Academic Press, 2002.
[6] A.J. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, C. Galbraith, Implicit curves and surfaces: mathematics, Data Structures and Algorithms, Springer-Verlag, 2009.
[7] M. Husty, M. Pfurner, H. Schrocker, A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator, Mech. Mach. Theory 42 (2007) 66–81.
[8] J. de Lope, T. Zarraonandia, R. Gonzalez-Careaga, D. Maravall, Solving the inverse kinematics in humanoid robots: a neural approach, Artificial Neural Nets Problem Solving Methods, LNCS 2687, 177–1842003.
[9] D. Manocha, J. Canny, Real time inverse kinematics for general 6R manipulators, Proceedings of the 1992 IEEE Conference on Robotics and Automation, Nice, France. pp. 383–389, 1992.
[10] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, G. Wang, Comparison of interval methods for plotting algebraic curves, Comput. Aided. Geom. Des. 552–582 (2002).
[11] C. Mavroidis, F. Ouezdou, P. Bidaud, Inverse kinematics of six-degree of freedom general and special manipulators using symbolic computation, Robotica 12 (1994) 421–430.
[12] C. Muñoz, A. Narkawicz, Formalization of a representation of Bernstein polynomials and applications to global optimization, J. Autom. Reason. (2012)(Accepted for publication).
[13] D. Pieper, The Kinematics of Manipulators Under Computer Control, (Ph.D. thesis) Stanford University, Stanford, 1968.
[14] M. Raghavan, B. Roth, Kinematic analysis of the 6R manipulator of general geometry, International Symposium on Robotics, Tokyo, 1989, pp. 314–320.
[15] M. Reuter, T.S. Mikkelsen, E.C. Sherbrooke, T. Maekawa, N.M. Patrikalakis, Solving nonlinear polynomial systems in the barycentric bernstein basis, Vis. Comput. 24 (2008) 187–200.
[16] T. Rudny, Universal inverse kinematics problem solver, Proceedings of the 16th International Conference on Systems Science, Wroclaw, Poland, 2007, pp. 161–166.
[17] I.L. Salcedo, M.S. Dutra, P., O.L., New technique for inverse kinematics problem using GA, 11th International Conference on Mechatronics Technology, 2007, Korea, 2007, pp. 318–320.
[18] M.R. Spencer, Polynomial real root finding in Bernstein form, (Ph.D. thesis) Provo, UT, USA. UMI Order No. GAX94–23360, 1994.
[19] C.G. Uzctegui, D.B. Rojas, A memetic differential evolution algorithm for the inverse kinematics problem of robot manipulators, In. J. Mechatron. Autom. 3 (2013) 118–131.