INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 12(1):76-89, February 2017.

Speed Computation for Industrial Robot Motion by Accurate Positioning

L.M. Matica, H. Oros

Liliana Marilena Matica*
Faculty of Electrical Engineering and Information Technology
University of Oradea
Oradea, Romania
*Corresponding author:lmatica@uoradea.ro

Horea Oros
Faculty of Sciences, Department of Mathematics and Computer Science
University of Oradea
Oradea, Romania
horos@uoradea.ro

Abstract: In this paper we define a new method for speed (velocity) computation, named mixt profile. The mixt profile of speed variation assures an accurate positioning at the end of motion (movement), in a well determinate time lapse. The method is linked with computation of location (position) matrix, about an industrial robot. Mixt profile of speed may be applied about motion on linear or circular trajectories. The paper continues the explanation from [6][a] regarding this method.
Keywords: kinematics of industrial robots; linear or circular trajectory; acceleration and deceleration stage of movement

[a]Reprinted and extended, with permission based on License Number 3979260943291 [2016] IEEE, from "Computers Communications and Control (ICCCC), 2016 6th International Conference on"

1 Introduction

This paper contains others explanation regarding mixt profile of speed, published in [6], those are: more detailed explanations about the computation method named mixt profile of speed; the adaptation of mixt profile method of computation for a circular trajectory and a concrete example of computation; a graphic about deceleration stage explaining software implementation of this computation method; formulas of reverse kinematics about RRRRRR robotic arm, which was done for illustrate the method of computation, figure 1; formulas of direct kinematics about this robotic arm; explanations about the solid kinematics principles that establish the direct kinematics; enunciation of Or algorithm about determining the direct kinematics of an robotic arm (a simple algorithm defined in this paper; it analyses only those two relative positions: parallel or perpendicular).

Concerning movement command of an industrial robot, [1]- [6] it is necessary to define the next location of it. The industrial robot location is defined by the location matrix. Starting with the values of location matrix, the values of kinematics joints parameters of the industrial robot may be computed. Those computation formulas are named, reverse kinematics. The reverse kinematics is the solution of an equation system formed by forward kinematics. Forward kinematics of an industrial robot is the result of a kinematics analysis. An example of kinematics analysis about an industrial robot, (more precise, about a robotic arm type RRRRRR) is illustrated in figure 1 [3].

In figure 1, the notations defines several Cartesian coordinate systems with its axles: X_i, Y_i, Z_i and its origins: O_i (index i goes from 1 to 6; $i = 1..6$); then six rotation driving kinematics

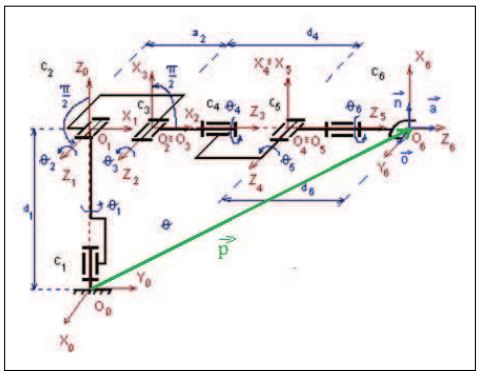

Figure 1: Kinematics analysis of robotic arm, type RRRRRR

couples (d.c.c.) of robotic arm: C_i $(i = 1..6)$; variable parameters of d.c.c.: Θ_i $(i = 1..6)$; constant parameters of the robotic arm: d_1; a_2; d_4; d_6; the versors: $\overrightarrow{n}$; $\overrightarrow{o}$; $\overrightarrow{a}$ (about sense and direction of axles OX_6; OY_6 and OZ_6), the Cartesian coordinate system of index 6 has the origin in the tool centre point (TCP) of the robotic arm. All Cartesian coordinate systems have the position of axles according with Denavit-Hartenberg convention. [3], [6]

About industrial robot movement (motion), if the motion time is defined, the positioning precision at the end of the motion is not very good. A very good positioning precision, at the motion end, can't be obtained, in a defined motion time. Both conditions are very hard to accomplish. The mixt profile speed variation, defined in this paper, during industrial robots motion, may accomplish those two conditions. The method is linked with the location matrix computation of an industrial robot. [6]

2 The location matrix of an industrial robot

The industrial robot location, (position and orientation) is defined by the location matrix that contains axles components of position vector: $\overrightarrow{p}$ and orientation versors: $\overrightarrow{n}$; $\overrightarrow{o}$; $\overrightarrow{a}$; figure 2, [3]; a versors have the module equal to 1.

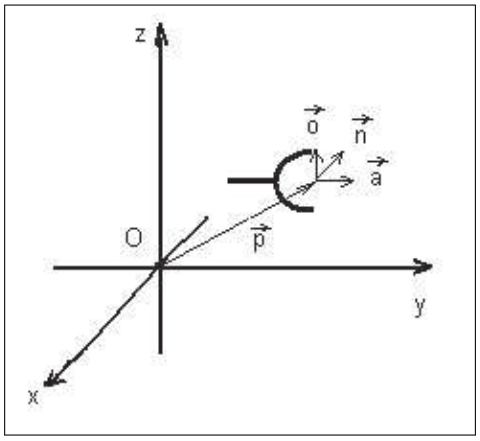

Figure 2: The position vector and orientations versors that define the location matrix of an industrial robot

A versor describes only the orientation; about an industrial robot, the orientation versors: $\overrightarrow{n}$; $\overrightarrow{o}$; $\overrightarrow{a}$; describe the orientation of tool centre point, TCP, regarding the Cartesian coordinate

system considered.

The location matrix of the industrial robot contains three components of those versors and the position vector (three components along well known three axles of a Cartesian coordinate system: OX; OY; OZ).

$$G_i = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Index i of the location matrix makes reference to index i Cartesian coordinate system.

According to Denavit-Hartenberg convention, a coordinate system, index i, is obtained by homogeneous transformations, from previous one, index $i-1$. Those homogeneous transformations are (always in this order):

1. rotation of $\Theta_i(t)+\beta$ angle, around OZ_{i-1} axle (the parameter t show that the angle varies in time, it is programmable, because that d.c.c. is an rotation one);
2. translation of d_i distance, along OZ_{i-1} axle;
3. translation of l_i distance, along OX_i axle;
4. rotation of α_i angle, around OZ_{i-1} axle. [3], [6]

The previous homogeneous transformations define the transformation matrix, ${}^{i-1}A_i$: [3]

$${}^{i-1}A_i = Rot(OZ_{i-1}, \theta_i(t)+\beta_i) \cdot Trans(OZ_{i-1}, d_i) \cdot Trans(OX_i, l_i) \cdot Rot(OX_i, \alpha_i) \tag{2}$$

For example, the forward kinematics formulas of the robotic arm from figure 1 are (a robotic arm is a particular kind of industrial robot, it is similar with the human arm):

$$\begin{array}{rcl} {}^0A_1 & = & Rot(OZ_0, \theta_1(t)+\pi/2) \cdot Trans(OZ_0, d_1) \cdot Rot(OX_1, +\pi/2) \\ {}^1A_2 & = & Rot(OZ_1, \theta_2(t)) \cdot Trans(OX_2, a_2) \\ {}^2A_3 & = & Rot(OZ_2, \theta_3(t)+\pi/2) \cdot Rot(OX_3, +\pi/2) \\ {}^3A_4 & = & Rot(OZ_3, \theta_4(t)) \cdot Trans(OZ_3, d_4) \cdot Rot(OX_4, -\pi/2) \\ {}^4A_5 & = & Rot(OZ_4, \theta_5(t)) \cdot Rot(OX_4, +\pi/2 \\ {}^5A_6 & = & Rot(OZ_5, \theta_6(t) \cdot Trans(OZ_5, d_6) \end{array} \tag{3}$$

Let us discus about transformation matrix ${}^0A_1$.

The kinematics joint, named $C1$, is a rotational one; so, relation 2 is useful and must be adapted (index i is equal with 1):

$${}^0A_1 = Rot(OZ_0, \theta_1 t+\beta_i) \cdot Trans(OZ_0, d_1) \cdot Trans(OX_1, l_1) \cdot Rot(OX_1, \alpha_1) \tag{4}$$

About relation 4, the variable parameter (programmable) is the angle: $\theta_1(t)$; the constant values are named: β_1 ; k_1 ; l_1 and α_1. In purpose to determine the constant value named β_1, the relative position of axle OX_1 to axle OX_0 must be analyzed, figure 3; it is perpendicular (it is not parallel, it define an angle); this parameter has a different from zero value [3].

In purpose to determine the α_1 constant value, it must be analyzed the axle OZ_1 relative position to axle OZ_0; figure 4; it is perpendicular, this parameter has a non-zero value.

The translations are defined by relative position of origins O_0 and O_1; so, it is necessary a translation along axle OZ_1 with constant value named d_1. [3] It result the ${}^0A_1$ transformation matrix, relation 3.

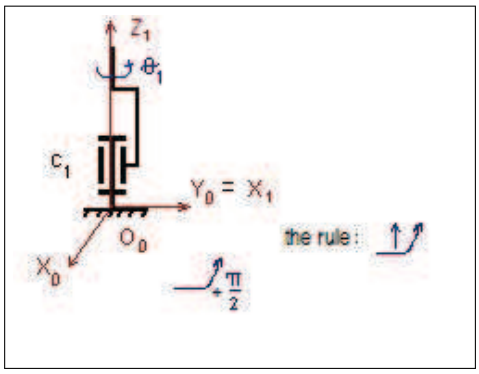

Figure 3: Analysis about β_1 value

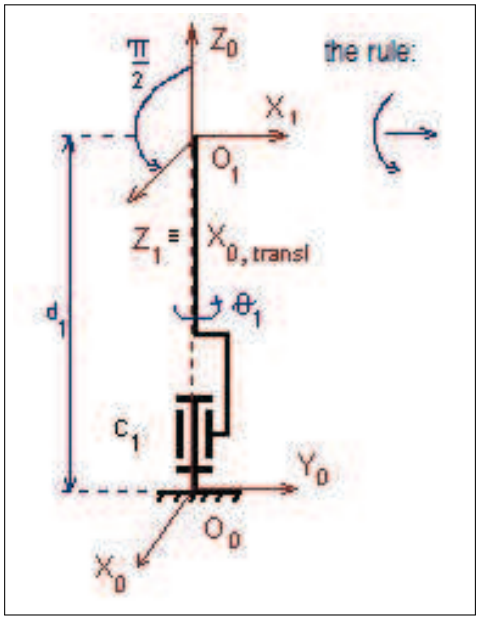

Figure 4: Analysis about α_1 value

Let us discuss other two examples [3]. From figure 5 graphical analysis, it results the ${}^1A_2$ matrix. A constant value (equal with a_2) characterizes a translation along axle OX_2; the constant value named β_2 has a zero value, because axle OX_1 is parallel with axle OX_2; the constant value named α_2 has a zero value, because axle OZ_1 is parallel with axle OZ_2.

From figure 6 graphical analysis, it results ${}^5A_6$ matrix, (relation 3); the significant constant value, d_6, is about a translation along axle OZ_6; β_6 and α_6 have zero value.

Similar analyses are performed for determination of every transformation matrix (of forward kinematics). It results all the transformation matrices (relation 4) and the location matrix of the industrial robot. More precisely, the location matrix defines the position and the orientation of the robotic arm tool center point, TCP.

Lets makes a clear distinction between parameters of a translational or a rotational kinematics joint of an industrial robot. About a rotational kinematics joint, relation 2, the variable parameter is named: $\theta_i(t)$. About a translational kinematics joint, the variable parameter is named: $d_i(t)$; the ${}^{i-1}A_i$ transformation matrix is defined as follow:

$$ {}^{i-1}A_i = Rot(OZ_{i-1}, \beta_i) \cdot Trans(OZ_{i-1}, d_i(t)) \cdot Trans(OX_i, l_i) \cdot Rot(OX_i, \alpha_i) \qquad (5) $$

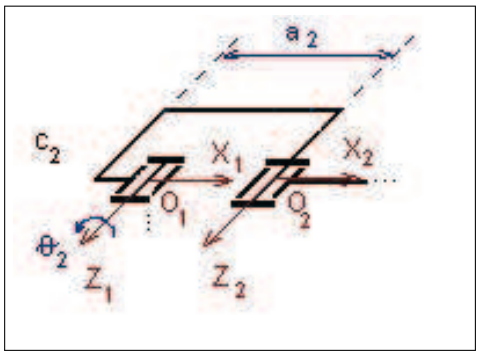

Figure 5: Analysis about ${}^{1}A_2$ matrix

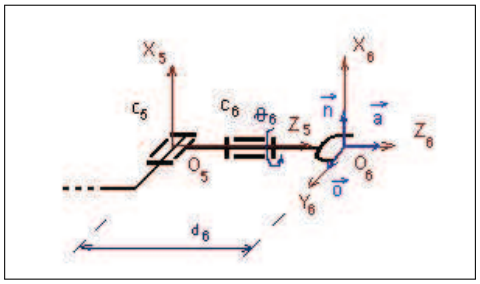

Figure 6: Analysis about ${}^{5}A_6$ matrix

In relation 2 and 5, the constant values of forward kinematics computation are named: β_i; k_i; l_i; α_i, whatever the kinematics joint is a rotational or a translational one.

Usually, about forward kinematics of industrial robots, the explanations work with a single formula (relation 6), without a clear distinction (regarding the parameter name), concerning variable and constant values:

$$ {}^{i-1}A_i = Rot(OZ_{i-1}, \theta_i) \cdot Trans(OZ_{i-1}, d_i) \cdot Trans(OX_i, l_i) \cdot Rot(OX_i, \alpha_i) \tag{6} $$

The relations 2 and 5, described in this paper, make a clear distinction between constant and variable values, involved in forward kinematics computation. According with Denavit-Hartenberg convention, the algorithm explained in this paper, determines the constant values involved in forward kinematics computations, based on observation: two axles are parallel or are not parallel.

The first step of the algorithm consists in Cartesian coordinate systems settlement, identical with Denavit-Hartenberg convention. Every kinematics joint, named C_i, is characterized by a Cartesian coordinate system, named $OXYZ_i$ properly settled; index i goes from 1 value to n, n is the number of kinematics joints. The axle OZ_i is settled along the axis of index $i+1$ kinematics joint, named C_{i+1} ; the axle OX_i is settled perpendicular of the plane formed by axles OZ_{i-1} and OZ_i. The $OXYZ_n$ Cartesian coordinate system is settled linked with TCP position and orientation; in figure 1 $n = 6$. Every Cartesian coordinate system is obtained from the previous one, by several homogenous transformations; those define the transformation matrices, ${}^{i-1}A_i$.

The next step of the algorithm consists on the characteristics identification of each C_i kinematics joint and determination of each ${}^{i-1}A_i$ matrix. Regarding a rotational kinematics joint, relationship 2 is useful for transformation matrix determination; the variable (programmable) parameter is $\theta_i(t)$; the others values are constant. Regarding a translational kinematics joint, relationship 5 is useful for transformation matrix determination; the variable (programmable) parameter is $d_i(t)$; the others values are constant.

Concerning industrial robot motion, the speed variation (as it is defined by mixt profile of speed), had influence (it changes the parameters values, in time) upon programmable (variable) parameters of kinematics joints.

The third step of the algorithm consists on transformation matrixes determination, ${}^{i-1}A_i$, (several analysis about determination of each transformation matrix, were explained on graphical analysis in figures 4, 5 and 6) it may be described as follow: the variable parameter is defined by the kinematics joint type, on step two of the algorithm; about constant values, there is a rotation, named β_i, around axle OZ_{i-1}, if axle OX_{i-1} and axle OX_i are not parallel (those axle may be perpendicular, according with the construction of the industrial robot and the angle may be $\pm\pi/2$; there are translations along axle OZ_{i-1} or along axle OX_i if the Cartesian coordinate systems origins, O_{i-1} and O_i, are not identical (very simple to be identified); there is a rotation around axle OX_i, named α_i, if axle OZ_{i-1} and axle OZ_i are not parallel, (those axle may be perpendicular and the angle may be $\pm\pi/2$).

The previous explanations develop an algorithm about forward kinematics determination, (it asks about each kinematics joint of the robotic arm: is it a translational or a rotational one; it asks about two axles: are those axles parallel or not; this algorithm may be named algorithm Or). It analysis only those two relative positions: parallel or perpendicular; the analysis described by Denavit-Hartenberg convention analysis any relative position about similar Cartesian coordinate axles involved.

The formulas of forward kinematics, to compute position vector components of an industrial robot, more precise, about the robotic arm type RRRRRR from figure 1, are [3] (notations S_i; $i = 1..6$, means sine of θ_i angle and C_i means cosine of same angle, the others notation are identical with those explained):

$$\begin{aligned}
p_x &= (S_1 \cdot C_2 \cdot S_3 + S_1 \cdot S_2 \cdot C_3) \cdot C_4 \cdot S_5 \cdot d_6 + C_1 \cdot S_4 \cdot S_5 \cdot d_6 - S_1 \cdot C_2 \cdot a_2 + \\
&\quad + (S_1 \cdot S_2 \cdot S_3 - S_1 \cdot C_2 \cdot C_3) \cdot (C_5 \cdot d_6 + d_4) \\
p_y &= (-C_1 \cdot C_2 \cdot S_3 - C_1 \cdot S_2 \cdot C_3) \cdot C_4 \cdot S_5 \cdot d_6 + S_1 \cdot S_4 \cdot S_5 \cdot d_6 + \\
&\quad + C_1 \cdot C_2 \cdot a_2 + (C_1 \cdot C_2 \cdot C_3 - C_1 \cdot C_2 \cdot S_3) \cdot (C_5 \cdot d_6 + d_4) \\
p_z &= (-S_2 \cdot S_3 + C_2 \cdot C_3) \cdot C_4 \cdot S_5 \cdot d_6 + (S_2 \cdot C_3 + C_2 \cdot S_3) \cdot (C_5 \cdot d_6 + d_4) + \\
&\quad + S_2 \cdot a_2 + d_1
\end{aligned} \tag{7}$$

The formulas to compute the orientation versors components of the same robotic arm, figure 1, are:

$$\begin{aligned}
n_x &= (S_1 \cdot C_2 \cdot S_3 + S_1 \cdot S_2 \cdot C_3) \cdot (C_4 \cdot C_5 \cdot C_6 - S_4 \cdot S_6) + \\
&\quad + C_1 \cdot (S_4 \cdot C_5 \cdot S_6 + C_4 \cdot S_6) - S_5 \cdot C_6 \cdot (S_1 \cdot S_2 \cdot S_3 - S_1 \cdot C_2 \cdot C_3) \\
n_y &= (-C_1 \cdot C_2 \cdot S_3 - C_1 \cdot S_2 \cdot C_3) \cdot (C_4 \cdot C_5 \cdot C_6 - S_4 \cdot S_6) + \\
&\quad + S_1 \cdot (S_4 \cdot C_5 \cdot C_6 + C_4 \cdot S_6) - S_5 \cdot C_6 \cdot (C_1 \cdot C_2 \cdot C_3 - C_1 \cdot S_2 \cdot S_3) \\
n_z &= (-S_2 \cdot S_3 + C_2 \cdot C_3) \cdot (C_4 \cdot C_5 \cdot S_6 - S_4 \cdot S_6) - S_5 \cdot S_6 \cdot (S_2 \cdot C_3 + C_2 \cdot S_3)
\end{aligned} \tag{8}$$

$$\begin{aligned}
o_x &= (S_1 \cdot C_2 \cdot S_3 + S_1 \cdot S_2 \cdot C_3) \cdot (-C_4 \cdot C_5 \cdot S_6 - S_4 \cdot C_6) + \\
&\quad + C_1 \cdot (-S_4 \cdot C_5 \cdot S_6 + C_4 \cdot C_6) + S_5 \cdot S_6 \cdot (C_1 \cdot C_2 \cdot C_3 - C_1 \cdot S_2 \cdot S_3) \\
o_y &= (-C_1 \cdot C_2 \cdot S_3 - C_1 \cdot S_2 \cdot C_3) \cdot (-C_4 \cdot C_5 \cdot C_6 - S_4 \cdot C_6) + \\
&\quad + S_1 \cdot (-S_4 \cdot C_5 \cdot S_6 + C_4 \cdot C_6) + S_5 \cdot C_6 \cdot (C_1 \cdot C_2 \cdot C_3 - C_1 \cdot S_2 \cdot S_3) \\
o_z &= (-S_2 \cdot S_3 + C_2 \cdot C_3) \cdot (-C_4 \cdot C_5 \cdot S_6 - S_4 \cdot C_6) + S_5 \cdot S_6 \cdot (S_2 \cdot C_3 + C_2 \cdot S_3)
\end{aligned} \tag{9}$$

$$\begin{aligned} a_x &= (S_1 \cdot C_2 \cdot S_3 + S_1 \cdot S_2 \cdot C_3) \cdot C_4 \cdot S_5 + S_1 \cdot S_4 \cdot S_5 + \\ &\quad C_5 \cdot (S_1 \cdot S_2 \cdot S_3 - S_1 \cdot C_2 \cdot C3) \\ a_y &= (-C_1 \cdot C_2 \cdot S_3 - C_1 \cdot S_2 \cdot C_3) \cdot C_4 \cdot S_5 + S_1 \cdot S_4 \cdot S_5 + \\ &\quad + C_5 \cdot (C_1 \cdot C_2 \cdot C_3 - C_1 \cdot S_2 \cdot S_3) \\ a_z &= (-S_2 \cdot S_3 + C_2 \cdot C_3) \cdot C_4 \cdot S_5 + C_5 \cdot (S_2 \cdot C_3 + C_2 \cdot S_3) \end{aligned} \tag{10}$$

The reverse kinematics (as a result of forward kinematics) computes the d.c.c. parameters starting with position matrix elements values; it is the solution of the equations system (12 equations and 12 unknown values) defined by direct kinematic. It results this conclusion: in order to command an industrial robot motion, it is necessary to compute the position matrix components, for every sampling period of time; those components describe the position and orientation of the robotic arm. The speed (velocity) of motion is defined by vector $\overrightarrow{p}$ (position vector) variation. [6]

3 Acceleration, motion on trajectory and deceleration

About motion on a trajectory, a condition could be a certain speed profile. This speed profile may be trapezoidal or parabolic, figure 7 and figure 8 (graphics consider continuous time).

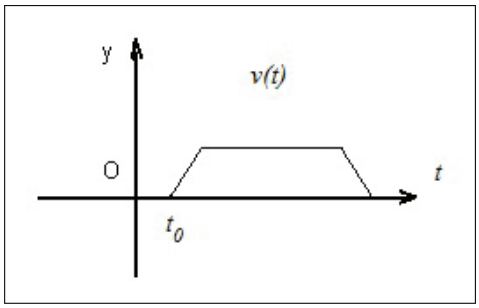

Figure 7: Trapezoidal profile (of speed)

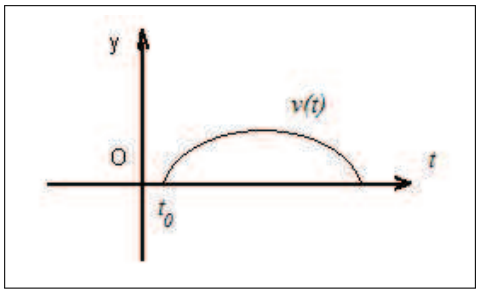

Figure 8: Parabolic profile

The motion of an industrial robot may contain three stages:

1. the acceleration from zero motion speed to the programmed motion speed;
2. the motion with programmed motion speed (constant);
3. the deceleration from programmed speed to zero. [6]

Commonly, acceleration and deceleration depend on the speed profile that was selected. This paper describes another method about deceleration; the method describes another speed profile, named mixt profile of speed, figure 9. [6]

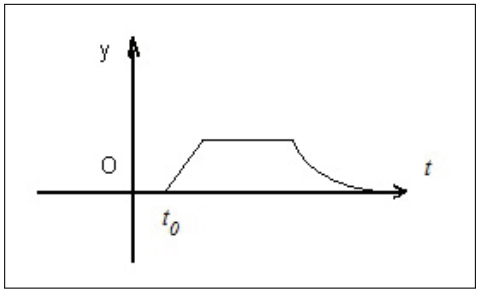

Figure 9: Mixt profile of motion speed

If the trajectory is imposed (linear or circular), it must be computed the position of the intermediary points, named waypoints, (on the trajectory), during acceleration stage, motion on trajectory stage and deceleration stage.

Intermediary positions of the robotic arm are defined by different location matrix. If the trajectory is imposed, we must compute location matrices for every waypoint. Considering reverse kinematics, it results the motion commands for kinematics joints of the robotic arm; starting with location matrix of every waypoint that composes the trajectory, the parameter of every kinematics joint may be computed.

4 Acceleration and deceleration stages for mixt profile of speed

Usually, about acceleration stage, the acceleration variation depends of the maximum acceleration possible, on a sample period of time, considering a numerical computation system with numerical processor.

About an robotic arm motion, the numerical process of command computation, is a discrete one. [1] Variation of robotic arm position, variation of motions speed, acceleration and deceleration values (and others values) depend of a discrete variable defined by relation: $k \cdot T$, where T is the sampling period of time, and k is the number of the sample periods of time considered from the commands beginning (for example, the variable had the value $11 \cdot T$ after eleven sampling periods of time from the start of motion). [6]

About computation described in this paper, the value of maximum possible acceleration in a sample period of time is named a_{max}. About this computation method described in this paper, in the acceleration stage for mixt speed profile, the variation of speed is defined by the relation: [6]

$$v(kT) = v_0 + k \cdot a_{max} \tag{11}$$

Often, the motion speed initial value is zero: $v_0 = 0$; it results: $v(kT) = k \cdot a_{max}$, figure 9, but this speed increasing computation method may be applied for any initial value.

Considering the defined speed increase, relation 11, (in the acceleration stage of mixt profile), the position varies with the values: $T \cdot v(kT) = T \cdot k \cdot a_{max}$; after each sampling period of time. The acceleration value may be considered about axle component of position vector, it results the maximum possible acceleration along every axle, named a_M (instead of a_{max}); the position vector axle components varies, during the acceleration stage: [6]

$$\begin{aligned} p_{k,x} &= p_{k-1,x} + k \cdot T \cdot a_M; k = 1..k_A \\ p_{k,y} &= p_{k-1,y} + k \cdot T \cdot a_M \\ p_{k,z} &= p_{k-1,z} + k \cdot T \cdot a_M \end{aligned} \tag{12}$$

Index k goes from 1 value to k_A value, till the end of the motion stage. The computation starts from axle components values of initial position vector, named: $p_{0,x}$; $p_{0,y}$; $p_{0,z}$.

Let consider a motion with a programmed (imposed) speed value, named v_P; the acceleration stage ends when the speed reach this programed speed value. The programed value of motion speed defines the number of sampling periods of time necessary for the acceleration stage; named k_A, it results:

$$k_A = v_p/T \cdot a_M \tag{13}$$

The variation is a discrete one, so, the value of k_A must be an integer value; the k_A value must be adapted of this condition: it will be the next bigger integer value of the computed value. Because of this aspect, the last step of speed increase value must be adapted, in purpose to reach the programmed speed value (it is obvious that the last step of speed increase value will not be bigger then: a_M).

If speed axle components, named: $v_{P,x}$; $v_{P,y}$; $v_{P,z}$; have different values, the k_A value is determined by the maximum value of speed component:

$$k_A = max(v_{P,x}; v_{P,y}; v_{P,z})/T \cdot a_M \tag{14}$$

The acceleration may be different for each axle, the maximum value of speed component, $max(v_{P,x}; v_{P,y}; v_{P,z})$, defines the axle with maximum acceleration. About other axles, the acceleration is computed, in order to have o constant value for every sampling period of time. The acceleration values are: $v_{P,x}/k_A$; $v_{P,y}/k_A$; $v_{P,z}/k_A$.

About next stage, the motion with a programmed speed, the position vector is described by the relations: [6]

$$\begin{aligned} p_{k+1,x} &= p_{k,x} + T \cdot v_{P,x} \\ p_{k+1,y} &= p_{k,y} + T \cdot v_{P,y} \\ p_{k+1,z} &= p_{k,z} + T \cdot v_{P,z} \end{aligned} \tag{15}$$

In relation 15 index k starts from k_A and goes till is necessary the deceleration stage. This relation, rel. 15 defines those significant values: $\delta_x = T \cdot v_{P,x}$; $\delta_y = T \cdot v_{P,y}$; $\delta_z = T \cdot v_{P,z}$; its mean the linear space steps (because the trajectory is linear), performed at each sampling period of time, during motion with programed speed, on every axle. Those values are named axle steps. Motion on a circular trajectory defines angle steps (about spherical coordinates).

About deceleration stage (the third stage of motion), the speed variation is not a linear one:

$$v(kT) = v_P - T \cdot a_D(kT) = v_P - b \cdot k^2; k = 1..k_D \tag{16}$$

In previous relation, the deceleration value, named a_D, is a variable value and b is a constant value. The b value defines the characteristics about robotic arm motion.

The speed decreases till the motion end; considering the condition: $0 = v_P - b \cdot k_D^2$; it results the number of sampling period of time necessary for the deceleration stage, named k_D (the axle components of speed are: $v_{P,x}$; $v_{P,y}$; $v_{P,z}$):

$$k_D = \sqrt{\frac{max(v_{P,x}; v_{P,y}; v_{P,z})}{b}} \tag{17}$$

About motion on trajectory (the second stage), the necessary distance for deceleration stage, named DD, determine its end (the motion on trajectory ends in the point situated at distance DD before the end point of motion). [6]

The resulting speed profile, named mixt profile, figure 9 (the graphic considers continuous time) ensures a better precision about stop point proximity. Typically, for precise positioning at the motion end, it can't be specified the time needed; the mixt profile of speed specifies exactly the time needed for precise positioning at the motion end. [6]

The described method, named mixt profile (of speed), was implemented at a flexible welding cellule (for manufacture of mining machinery), and the agreed motion characteristics (with the beneficiary) were ok. The maximum weight of processed pieces (with this welding cellule) was 2.5 tons. [6]

About deceleration stage, the software implementation considered 25 values about speed decrease, from speed maximum value possible, those values were written in a table, named: Deceleration table. Inertial reason imposes the number of table values (25). In purpose the determine the deceleration start, the programed speed, v_p, was compared with those values, from Deceleration table; the comparison result defines the value of k_D and every speed value, for every sampling period of time, during deceleration stage; a graphical explanation of this process is described in figure 10.

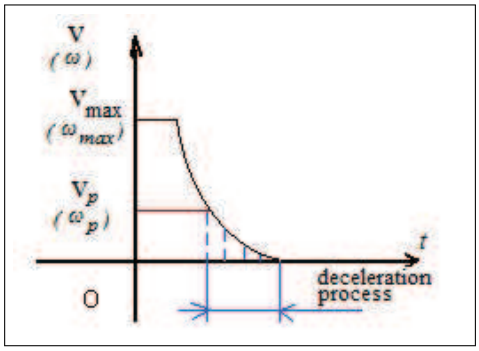

Figure 10: About software implementation of deceleration process

In figure 10, the example works with 4 steps till the end of the motion, (when the speed has zero value).

The software implementation of deceleration stage considered a different Deceleration table for OZ axle component of speed, because a vertical motion has different inertial characteristics, comparing with a horizontal motion (about axle OX and OY the Deceleration table is identical).

5 Example of computation about a linear trajectory

For example, considering a linear trajectory and constant orientation of the robotic arm (along the motion), the value of initial speed zero; the values of programed speed: $v_P = 5\sqrt{2}mm/s$; $v_{P,x} = 3mm/s$; $v_{P,y} = 4mm/s$; $v_{P,z} = 5mm/s$; $a_M = 25mm/s^2$ and $T = 10^{-2}s$; this method of computation determines: $k_A = 20$, the number of sampling periods of time necessary for acceleration stage: [6]

$$k_A = max(3; 4; 5)mm/s/(10^{-2}s \cdot 25mm/s^2) = 20 \tag{18}$$

During acceleration stage, the speed increases with those values: $\delta v_x = 3/20mm/s$; $\delta v_y = 4/20mm/s$; $\delta v_z = 5/20mm/s$; (because of inertial reasons, the acceleration have different values, for each axle components).

Considering the initial values of position vector components: $p_{0,x} = 1,1mm$; $p_{0,y} = 2,2mm$; $p_{0,z} = 3,3mm$, after first sampling period of time, the position vector has the axle components:

$$\begin{array}{rcccccccc} p_{1,x} & = & p_{0,x} + 1 \cdot T \cdot (v_{0,x} + \delta v_x) & = & 1,1 + 10^{-2} \cdot 3/20 & = & 1,1 + 0,0015 & = & 1,1015mm \\ p_{1,y} & = & p_{0,y} + 1 \cdot T \cdot (v_{0,y} + \delta v_y) & = & 2,2 + 10^{-2} \cdot 4/20 & = & 2,2 + 0,002 & = & 2,202mm \\ p_{1,z} & = & p_{0,z} + 1 \cdot T \cdot (v_{0,z} + \delta v_z) & = & 3,3 + 10^{-2} \cdot 5/20 & = & 3,3 + 0,0025 & = & 3,3025mm \end{array} \tag{19}$$

During acceleration stage, after 10 period of time the axle components of position vector differs (from the previous one, in the previous period of time) with: $\delta p_x = 10 \cdot T \cdot \delta v_x = 10 \cdot 3/2000 = 0,015mm$; $\delta p_y = 10 \cdot T \cdot \delta v_y = 10 \cdot 4/2000 = 0,02mm$; $\delta p_z = 10 \cdot T \cdot \delta v_z = 10 \cdot 5/2000 = 0,025mm$.

After 20 periods of time, begin the stage of motion on trajectory. In this moment (considering the initial values of position vector components), the position vector has the axles components: $p_{20,x} = 1,1 + 10^{-2} \cdot 3/20 \cdot (1 + 2 + ... + 20) = 1,1 + 0,315 = 1,415mm$; $p_{20,y} = 2,2 + 10^{-2} \cdot 4/20 \cdot (1 + 2 + ... + 20) = 2,2 + 0,42 = 2,62mm$; $p_{20,z} = 3,3 + 10^{-2} \cdot 5/20 \cdot (1 + 2 + ... + 20) = 3,3 + 0,525 = 3,825mm$.

The stage of motion on trajectory is described by relations relation 15, (it is similar with acceleration stage, but speed has a constant value):

$$\delta p_x = T \cdot v_{P,x}; \delta p_y = T \cdot v_{P,y}; \delta p_z = T \cdot v_{P,z}; \tag{20}$$

For each axle, the axle steps have constant values. Because axle steps have constant values, the algorithm is named: numeric difference analysis, more exactly: interpolate algorithm of numeric difference analysis. [6]

For example, after 80 periods of time, on the stage of motion with constant speed (and after 100 periods of time from the beginning of the motion) the components of position vector have the values: $p_x = 1,415 + 80 \cdot 10^{-2} \cdot 3 = 3,815mm$; $p_y = 2,62 + 80 \cdot 10^{-2} \cdot 4 = 5,82mm$; $p_z = 3,825 + 80 \cdot 10^{-2} \cdot 5 = 4,225mm$.

Let apply this computation (mixt profile of speed), to the robotic arm from figure 1. Let consider the orientation of robotic arm defined by those versors: $\overrightarrow{n} = 1 \cdot \overrightarrow{i}$; $\overrightarrow{o} = 1 \cdot \overrightarrow{j}$; $\overrightarrow{a} = 1 \cdot \overrightarrow{k}$ (where $\overrightarrow{i}$, $\overrightarrow{j}$, $\overrightarrow{k}$ are the versor defining the Cartesian coordinate axles, OX; OY and OZ). After 100 sampling period of time, the location matrix is:

$$G_0(100 \cdot T) = \begin{bmatrix} 1 & 0 & 0 & 3,815 \\ 0 & 1 & 0 & 5,82 \\ 0 & 0 & 1 & 4,225 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

During the motion, the location matrix has different values, at every sampling period of time. Knowing the location matrix, it results parameters of robotic arm kinematics joints, considering the formulas of robotic arm reverse kinematics.

The value of $\theta_1(100 \cdot T)$ parameter may be computed with this relationship (formula from reverse kinematics of the robotic arm [3]):

$$\theta_1(100 \cdot T) = \arctan \frac{-(p_{100,x} - d_6 \cdot \sin r_2 \cdot \cos r_1)}{p_{100,y} - d_6 \cdot \sin r_2 \cdot \sin r_1} \tag{22}$$

In previous relationships, the angle r_1 is the polar angle and the r_2 angle is the azimutal angle, about $\overrightarrow{a}$ versor, figure 11.

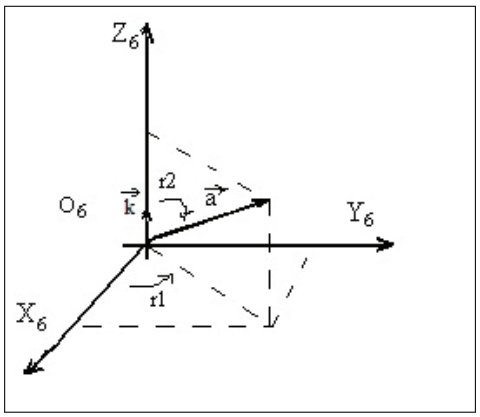

Figure 11: Polar and azimutal angle

Those angles may be computed; the value of n_1 and n_2 depends of the sign of versor components; let remember that arctan can have values in $[-\pi;\pi]$; but those angles may have values in $[0;2\pi]$:

$$r_1 = n_1 \cdot \pi + \arctan\left|\frac{a_y}{a_x}\right|; r_2 = n_2 \cdot \pi + \arctan\frac{\sqrt{a_x^2 + a_y^2}}{|a_z|} \tag{23}$$

From the considered orientation of the robotic arm: $\overrightarrow{a} = 1 \cdot \overrightarrow{k}$; it results the value of those two angles are: $r_1 = 0$; $r_2 = 0$; the parameter value is:

$$\theta_1(100 \cdot T) = \arctan\frac{-(p_{100,x})}{p_{100,y}} = \arctan(-3,815/5,82) = -33,245 \tag{24}$$

Previous relationship defines a negative angle, it means: the rotation sense of motion, about C_1 kinematics joint (the angle is the parameter of this kinematics joint) is opposite considering the positive sense, as it is designed in figure 1.

In purpose to determine the end of the second stage (motion with constant speed), it is necessary to compute the number of sampling period of time for deceleration; let considers $b = 5mm/900s$; it results: [6]

$$k_D = \sqrt{\frac{max(3;4;5)mm/s}{5mm/900s}} = 30 \tag{25}$$

During the deceleration, the computation of waypoints coordinates involves those speed values, from relation 16 it results:

$$\begin{aligned} v_x(kT) &= v_{P,x} - \tfrac{v_{P,x}}{k_D^2} \cdot k^2 = 3 - \tfrac{3}{900} \cdot k^2, k = 1..k_D \\ v_y(kT) &= v_{P,y} - \tfrac{v_{P,y}}{k_D^2} \cdot k^2 = 4 - \tfrac{4}{900} \cdot k^2 \\ v_z(kT) &= v_{P,z} - \tfrac{v_{P,z}}{k_D^2} \cdot k^2 = 5 - \tfrac{5}{900} \cdot k^2 \end{aligned} \tag{26}$$

For each sampling period of time, the position differs with values:

$$\begin{aligned} \delta p_x &= T \cdot v_x(kT), k = 1..k_D \\ \delta p_y &= T \cdot v_y(kT) \\ \delta p_z &= T \cdot v_z(kT) \end{aligned} \tag{27}$$

About deceleration stage, it must be computed the maximum value of distance axle components (necessary for deceleration stage), named DD_{max}:

$$DD_{max} = \sum_{k=1}^{k_D} T \cdot [max(v_{P,x}; v_{P,y}; v_{P,z}) - b \cdot k^2] \tag{28}$$

The DD_{max} value considers the maximum distance of the three axles, necessary for deceleration stage. The deceleration begins when it remains the distance DD_{max}, till the motion end, on respective axle.

According with the considered example, after 24 period of time on deceleration stage, the axle components of speed have the values: [6]

$$\begin{aligned} v_x(24 \cdot T) &= v_{P,x} - \tfrac{v_{P,x}}{k_D^2} \cdot 24^2 = 3 - \tfrac{3}{900} \cdot 24^2 = 1.08[mm/s] \\ v_y(24 \cdot T) &= v_{P,y} - \tfrac{v_{P,y}}{k_D^2} \cdot 24^2 = 4 - \tfrac{4}{900} \cdot 24^2 = 1.44[mm/s] \\ v_z(24 \cdot T) &= v_{P,z} - \tfrac{v_{P,z}}{k_D^2} \cdot 24^2 = 4.0752[mm/s] \end{aligned} \tag{29}$$

6 About computation for a motion on a circular trajectory, with mixt profile of speed

The previous example considered a linear trajectory. A circular trajectory imposes the computation of waypoints on spherical coordinates, named radium: R, polar angle: φ and azimuthal angle: ϕ; figure 11, and conversion on Cartesian coordinates of those values. [6] The acceleration and deceleration is similar with the method described about a linear trajectory, regarding tangential speed. The variation of tangential speed defines the variation of angular speed, named ω, figure 10. [6] Software implementation considered the maximum acceleration possible of motion speed, about rotation around each Cartesian coordinate system axle. A table about deceleration stage was defined about vertical rotations another table was defined about horizontal rotations.

An example of computation may consider $k_A = 3$; this values is defined by imposed values of motion speed. Let consider the motion a rotation of $2\pi/4 = 90°$ around axle OY. In the first period of time, the motion speed is: $v_1 = v_0 + a_M$, it result the angular speed of rotation $\omega_1 = \omega_0 + \Delta\omega$; in the second and third period of time the angular speed increase with the same value $\Delta\omega$; it results: $\omega_2 = \omega_1 + \Delta\omega = \omega_0 + 2 \cdot \Delta\omega$ and $\omega_3 = \omega_2 + \Delta\omega = \omega_0 + 3 \cdot \Delta\omega$. In the fourth period of time, (after motion beginning), the angular speed reaches the programed speed, ω_p.

Comparing the angular programmed speed value with values from Deceleration table, it may results the value $k_D = 5$ and all the values of angular speed, about deceleration stage. The distance necessary for deceleration stage on a circular trajectory, named DEC may be computed, where $\omega_{DEC}(k)$ are the smallest five values from deceleration stage (the Deceleration table simplifies the software implementation of deceleration stage):

$$DEC = \sum_{k=1}^{k_D=5} T \cdot \omega_{DEC}(k) \tag{30}$$

It results the number of sampling period of time for motion on trajectory stage (let consider $\omega_0 = 0$):

$$k_T = \frac{2\pi/4 - (\Delta\omega + 2 \cdot \Delta\omega + 3 \cdot \Delta\omega) \cdot T - DEC}{T \cdot \omega_P} \tag{31}$$

A variable orientation of robot arm, during the motion, involves a similar computation as described for a circular trajectory, but applied about computation of azimuthal and polar angle of each orientation versor; (the orientation versors are: $\vec{n}$; $\vec{o}$; $\vec{a}$).

Conclusions

About a robotic arms motion, those two conditions are very difficult to accomplish: best precision at the motion end and exact defined motion time. Those two conditions are accomplished by mixt profile (of motion speed variation), described in this paper.

The advantages of mixt profile are: the best precision to reach the end point of robotic arms motion, exact determination of motion time and minimum time of acceleration up to programed motion speed.

The method may have others diverse applications, about motion on a linear or circular trajectory; for example about turning or milling process.

Motion execution with exact speed gives processing quality; the described method of mixt profile, about speed variation, was implemented on numerical control equipment and precision was accurate (for example: numerical control equipment for workpieces of sintered metal carbides).

Bibliography

[1] Horsch, T.; Juttler, B.; Cartesian Spline Interpolation for Industrial Robots. University of Technology, Departament of mathematics, Darmstadt, Germany (http://www.ag.jku.at/pubs/csi98.pdf)

[2] Matica, L.M.; Kovendi, Z. (2011); Structure Analysis for an Industrial Robot, *Journal of Computer Science and Control Systems*, ISSN 1844-6043, 4(1): 89-92.

[3] Matica, L.M. (2008); *Conducerea robotilor industriali*, Edit. Univ. din Oradea, ISBN 978-973-759-481-5.

[4] Choset, H. and all, *Principles of Robot Motion*; https://mitpress.mit.edu/books/principles-robot-motion

[5] Laumond, J.P.; *Robot Motion Planning and Control*, ISBN: 978-3-540-76219-5 (Print) 978-3-540-40917-5 (Online), http://link.springer.com/book/10.1007%2FBFb0036069;

[6] Matica, L.M.; Oros, H (2016); Speed Computation in Movement followed by Accurate Positioning of Industrial Robots, *Computers Communications and Control (ICCCC), 2016 6th International Conference on*, IEEE Xplore, e-ISSN 978-1-5090-1735-5, DOI: 10.1109/ICCCC.2016.7496741, 75 - 79.