18th Mediterranean Conference on Control & Automation
Congress Palace Hotel, Marrakech, Morocco
June 23-25, 2010

Redundant Robot Manipulator Control with Obstacle Avoidance Using Extended Jacobian Method

M. BENZAOUI, H. CHEKIREB and M. TADJINE

Abstract— **In this paper, we develop a control of a redundant robot manipulator. That has to carry out a trajectory tracking in operational space while avoiding an obstacle. For this purpose, extended Jacobian method is used. The Self-motion vector is introduced at the level of the inverse kinematic solution in order to produce the obstacle avoidance (the secondary task). Thus, robot avoids obstacles without influencing the main task (trajectory tracking). The self-motion is computed from the optimization of scalar function depending on an anti collision constraint. Finally, this control method has leaded to a trajectory tracking in Cartesian space while avoiding the obstacle.**

I. INTRODUCTION

Robots manipulators are used to achieve repetitive tasks, dull works and used even in hostile environment. Their ability on repetitivity is necessary for industrial applications, but it is even more significant, if this quality is preserved even in presence of constraints related to the execution of task that may prevent the robot from achieving desired task such as obstacles.

Obstacle's avoidance problem is often met in industry and it forms a geometrical constraint, when several robots are placed in close places in order to save space. This means that it is necessary to develop control algorithms which consider the workspace restriction problems.

Such difficulties led to the development of new control methods where the main task (desired task) and the secondary task (obstacle avoidance) are considered at the same time. One suitable solution is the use of Kinematic redundancy [1]-[9].

Beside, robots manipulators kinematically redundant are those with a degree of freedom (DOF) *n* more than the necessary degree m to describe position and orientation. Redundancy can be classified according to *r*, the order of redundancy, the difference between n and m. It can include the intrinsic redundancy (due to the robot structure) or the functional redundancy (due to the exerted task). Independently of the redundancy causes, its use is the same one: the *r* additional DOF are used in the resolution of inverse kinematics to create an internal motion of the joints (self-motion) in order to avoid difficulties which obstruct the realization of desired task such as obstacles.

Avoiding obstacles problems are tackled in optimization terms of a scalar function *h(.)* reflecting the secondary task which must be carried out in addition to the main task (trajectory tracking in Cartesian space). That means for obstacle avoidance, a possibility would be to define a function *h(.)* depending on the Euclidean minimal distances between the obstacle and the robot arms[3].

This paper is organized as fellows. In section II, the position of problem is introduced and a redundant planar robot is presented resulting from 560 PUMA robot. The latter must carry out a trajectory tracking in Cartesian space while avoiding an obstacle. For this purpose, in section III, the self-motion is introduced to compute the secondary task (obstacle avoidance) without influencing the main task (trajectory tracking). This self-motion is deduced from an optimization of a scalar function *h(.)*. The section IV introduces the method which allows extending the Jacobian matrix by the self-motion vector in order to make it square. The choice of the scalar function *h(.)* depending on the anti collision constraint and it is given in section V. Finally, in section VI, the developed method control is tested in the case of this 3 DOF planar robot which realizes a trajectory tracking in Cartesian space in presence of an obstacle.

II. POSITION OF THE PROBLEM

The robot used derives from PUMA 560 by limiting its workspace to the vertical plan. So, the θ_1, θ_4 and θ_6 joints are locked so only the three other joints are free (*n=3*) [9]. As the task, to be realized, is a simple positioning of the end-effector in a vertical plan, it requires only two degrees of freedom (*m=2*) and the redundancy degree r of this robot is one: *r=n-m=1*. In this case, the state vector is reduced as following:

$$q = [q_1 \quad q_2 \quad q_3]^T = [\theta_2 \quad \theta_3 \quad \theta_5]^T \tag{1}$$

The direct geometrical model (DGM), gives the position (*x1, x2*) of the end-effector in this vertical plan according to the joint position (q):

M. Benzaoui is with the DGEE, FSI, University M'hamed Bougara, Boumerdès, Independence avenue 35000 Boumerdès Algeria (e-mail: benzaouimess@mail.com).

H. Chekireb, is with Process Control Laboratory,Department of Electrical Engineering, Polytechnic National School, BP 182, 10 avenue Hassan Badi, El-Harrach, Algeria (e-mail: chekireb@yahoo.fr).

M. Tadjine is with Process Control Laboratory,Department of Electrical Engineering, Polytechnic National School, BP 182, 10 avenue Hassan Badi, El-Harrach, Algeria (e-mail: tadjine@yahoo.fr)

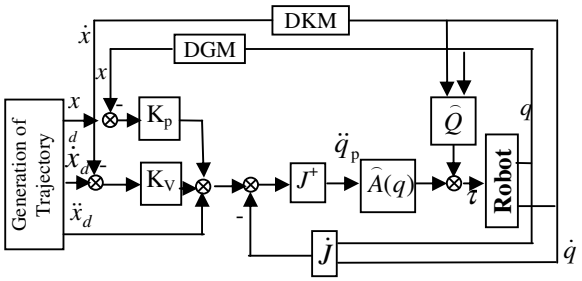


Fig.1. Control Scheme without obstacle avoidance

$$x_1 = a_2.\cos(q_1) + a_3.\cos(q_1+q_2) + d_4.\sin(q_1+q_2) + L.\sin(q_1+q_2+q_3) \quad (2.a)$$

$$x_2 = -a_2.\sin(q_1) - a_3.\sin(q_1+q_2) + d_4.\cos(q_1+q_2) + L.\cos(q_1+q_2+q_3) \quad (2.b)$$

Where a2, a3, d4 and L are geometrical parameters of the robot.

The Direct kinematic model (DKM) gives Joint velocity according to Cartesian velocity such as:

$$\dot{x} = J.\dot{q} \quad (3)$$

Where elements of J are given by:

$$\begin{aligned}
J_{11} &= -a_2.\sin(q_1) - a_3.\sin(q_1+q_2) + d_4.\cos(q_1+q_2) + L.\cos(q_1+q_2+q_3)\\
J_{12} &= -a_3.\sin(q_1+q_2) + d_4.\cos(q_1+q_2) + L.\cos(q_1+q_2+q_3)\\
J_{13} &= L.cos(q_1+q_2+q_3)\\
J_{21} &= -a_2.\cos(q_1) - a_3.\cos(q_1+q_2) - d_4.\sin(q_1+q_2) - L.\sin(q_1+q_2+q_3)\\
J_{22} &= -a_3.\cos(q_1+q_2) - d_4.\sin(q_1+q_2) - L.\sin(q_1+q_2+q_3)\\
J_{23} &= -L.\sin(q_1+q_2+q_3)
\end{aligned} \quad (4)$$

The actuator torques vector τ is related to the robot dynamic by [9]:

$$\tau = A(q)\ddot{q} + Q(q,\dot{q}) \quad (5)$$

With

$$Q(q,\dot{q}) = B(q,\dot{q}) + C(q,\dot{q}) + G(q)$$

Where:

- A: the inertia matrix of the robot
- B: Vectors of the Coriolis forces
- C: Vectors of the centrifugal forces
- G: Vector of the gravitational forces.

The end-effector has to follow a trajectory in Cartesian space, which is defined by the half circle of radius R and whose centre has the co-ordinates (x_{10}, x_{20}).

The trajectory can be determined by using an auxiliary variable $\alpha(t)$ varying according to the bang-bang law. Thus, the trajectory is determined as follow:

$$\begin{aligned}
x_{1d} &= R\cos(\alpha) + x_0;\ \dot{x}_{1d} = -\dot{\alpha}R\sin(\alpha);\\
x_{2d} &= R\sin(\alpha) + x_0;\ \dot{x}_{2d} = \dot{\alpha}R\cos(\alpha);\\
\ddot{x}_{1d} &= -\ddot{\alpha}R\sin(\alpha) - \dot{\alpha}R\cos(\alpha);\\
\ddot{x}_{2d} &= \ddot{\alpha}R\cos(\alpha) - \dot{\alpha}R\sin(\alpha);
\end{aligned} \quad (6)$$

Using, computed torque algorithm to obtain the trajectory tracking, the control law is calculated as follows [9]:

$$\tau = A.J^{-1}(w - \dot{J}.\dot{q}) + Q(q,\dot{q}) \quad (7)$$

$$w(t) = \ddot{x}_d + K_v.(\dot{x}_d - \dot{x}) + K_p.(x_d - x)$$

That can be implemented as schematized on figure1.

Since the Jacobian J is not square, its inverse J^{-1}appearing in control law (7) and represented by the J^+ term in the control scheme (Fig.1), is computed by the pseudo-inverse method. So, joint velocities are given by:

$$\dot{q}_p = J^+\dot{x} \text{ and } J^+ = J^T.(J.J^T)^{-1} \quad (8)$$

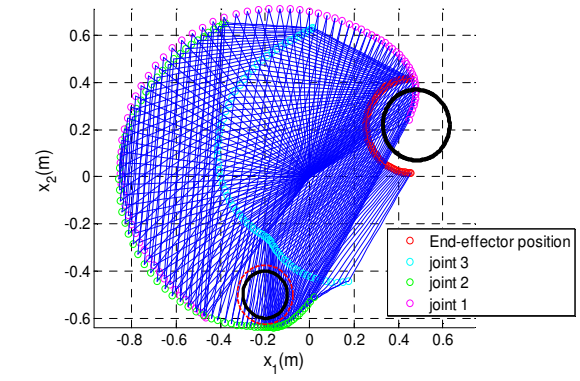
The control law (7), with $J^{-1}=J^+$, is applied in order to make the end-effector tracking the trajectory represented by the half-circle in a vertical plan (fig.2a). Moreover, it is assumed that there exists an object materialized by a circle. At first, initial time, the end-effector is assumed at the initial position of the trajectory (the top of the half-circle) and it must go along the trajectory in direct (the anti-clockwise) direction. Results are carried out in the case where the control coefficients are set such that:

$$K_v = diag\left[\sqrt{12500}\quad \sqrt{12500}\quad \sqrt{12500}\right];$$

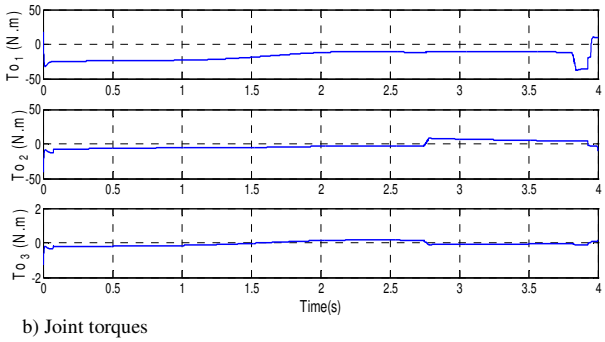
$$K_p = diag\left[12500\quad 12500\quad 12500\right]$$

The obtained figures (fig.2b and 2c) show that the trajectory tracking is carried out with tolerable tracking errors and realizable actuator torques. However, according to the figure 2a, arms 1 and 2 come into the area normally occupied by the object and then, product a collision with the object. Moreover, the situation is worsened by the fact that some arms come to cross the trajectory itself. This means that the control law (7) is unreached in a realistic context.

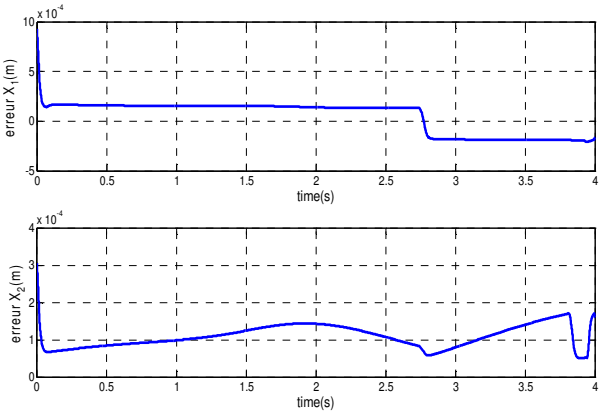
To overcome those difficulties, one can use a method for obstacle avoidance exploiting the robot redundancy. Such as developed in the following sections.



a)Cartesian trajectory



b) Joint torques



c) Tracking errors

Fig. 2. Control without obstacle avoidance

I. Introduction of the Self-Motion

It is known that one of methods to improve the dexterity of a robot manipulator is to increase its joints number. That means, the same pose of the end-effector is realized with infinity of joint position possibilities. The desired trajectory $x_d(t)$, for a redundant robot manipulator, is represented by a vector of dimension m required for the main task. The r DOF's robot has in more, can be exploited to achieve a constraint $h(.)$ due to the secondary task. In this way, the robot arms will acquire the capacity of reconfiguration (internal motion) without affecting the end-effector's pose. This is obtained by modifying the $\dot{q}_p$ solution as follows [1-7, 11, 12]:

$$\dot{q} = \dot{q}_{\mathrm{p}} + \dot{q}_n = J^{+}\dot{x} + \alpha(I - J^{+}J)z \tag{9}$$

z is the matrix (vector for the case $r=1$) of an arbitrary projection in this null space. It can be the gradient projection of the scalar function $h(q)$:

$$z = \nabla h \tag{10}$$

II. The Extended Jacobian Method

The extended Jacobian method is due to Baillieul [8] and it was exploited initially to avoid problems of instability encountered in generalized inverse method. The Jacobian of the robot is enlarged by the self-motion vectors as follows:

$$\left.\begin{array}{l} \dot{x} = J(q).\dot{q} \\ \hline\hline \dot{x}_a = J_a(q).\dot{q} \end{array}\right\} \Rightarrow \dot{x}_e = \left(\begin{array}{c} \dot{x} \\ \hline\hline \dot{x}_a \end{array}\right) = \left(\begin{array}{c} J \\ \hline\hline J_a \end{array}\right).\dot{q} = J_e.\dot{q} \tag{11}$$

Where $J(q)$ is a $m\times n$ Jacobian matrix and $Jn(q)$ comprises the $r\times n$ self-motion matrix. So, the $Je(q)$ extended Jacobian becomes $n\times n$ square matrix.

However, the additional term $Jn(q)$ cannot be arbitrary as for any mathematical function. Indeed, the obtained $n\times n$ matrix must be necessary invertible and allows to include a secondary task $h(.)$.

By the use of self-motion vector, the r lines can be added in the condition as [7]:

$$V_i^T \nabla h(q) = 0 \quad \text{and} \quad i=1,\ldots,r \tag{12}$$

Where $V = \left[V_1 \quad \cdots \quad V_r\right]$ is the set of column vectors that generate the kernel matrix of J. So that:

$$J_a(q) = \nabla^T (V^T \nabla h) = \frac{\partial (V(q)^T . \nabla h(q))}{\partial q} \tag{13}$$

In this case the control low (7) becomes:

$$\tau = A.J_e^{-1}(w_e - \dot{J}_e.\dot{q}) + Q(q,\dot{q}) \tag{14}$$

With

$$w_e(t) = \ddot{x}_{de} + K_v.(\dot{x}_{de} - \dot{x}_e) + K_p.(x_{de} - x_e)$$

And

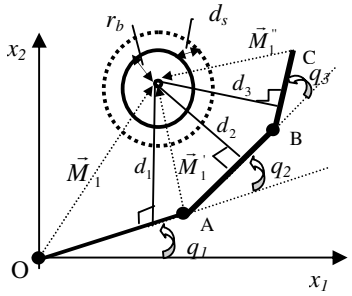


Fig. 3. Outdistance obstacle planar robot

$$x_e = \begin{pmatrix} x \\ 0_r \end{pmatrix};\ \dot{x}_e = \begin{pmatrix} x \\ 0_r \end{pmatrix};\ \dot{x}_{de} = \begin{pmatrix} \dot{x}_d \\ \dot{x}_{ad} \end{pmatrix};\ \ddot{x}_{de} = \begin{pmatrix} \ddot{x}_d \\ 0_r \end{pmatrix};$$

with $\dot{x}_{ad} = V^T \nabla h$

For a redundancy *r=1*, $V = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}^T$ is a simple vector and its elements vi can be computed such as in [10]:

$$v_i = (-1)^{i+1} \det(J_i(q)) \text{ and } i=(1,n) \quad (15)$$

Where, $J_i(q)$ represents the Jacobian J without its ith column.

As r=1, the computation of Ja is simplified such as in [7]:

$$J_a = V^T(q).Hu(h(q)) - \mu \quad (16)$$

Where μ is a vector ant its elements μ_i are determined by:

$$\mu_i = (\nabla^T h).J^T.(J.J^T)^{-1}.\frac{\partial J}{\partial q_i}V \text{ and } i=(1,n) \quad (17)$$

and Hu(h(q)) is the hessian matrix given by:

$$Hu(h(q)) = \frac{\partial}{\partial q}\left(\frac{\partial h(q)}{\partial q}\right)$$

Thus, it steel one problem, how one specify the scalar function h(.) ?

III. Obstacles Avoidance Method

The function *h(q)* includes necessarily the effect of the constraint due to the obstacle. In the case of obstacle avoidance, this returns mainly to determine the distance obstacle-robot.

A. Euclidean distance criteria

The method is based on Euclidean distances between the obstacle center and each of robot arms. One solution is the use of projection within the scalar product related to the vectors $\vec{M}_1$ drawn from the beginning of this moving arm to the center of obstacle $\vec{M}_2$ the arms itself [3]. This allows to determine the Euclidean distances *di (i=1,n)* between the obstacle and the different robot arms. But, it is rather the anti-collision constraint Di which is actually used. The computation with the anti-collision variables is significant, but to supplement the problem formulation it is necessary to define a scalar function dependent on these variables Di, anti-collision functions. Those latter are often taken in the following form:

$$h(q) = h_0(q) + \sum_{i=1}^{n} \rho_i .P(D_i(q)) \quad (18)$$

Where h_0 is used in order to minimize an energy criterion and it is often taken as $\frac{1}{2}q^T q$, the ρ_i terms are constant and the function *P(.)* is known as penalty function.

Generally, this function is represented by:

$$P(D_i) = \frac{-1}{D_i(q)} \quad (19)$$

This leads to the following scalar function h(q):

$$h(q) = \frac{1}{2}q^T.q + \sum_{i=1}^{n} \frac{-\rho_i}{D_i(q)} \quad (20)$$

The gradient of h(q) is given by:

$$\nabla h = \begin{bmatrix} \partial h/\partial q_1 & \dots & \partial h/\partial q_n \end{bmatrix}^T \quad (21)$$

So,

$$\frac{\partial h}{\partial q_i} = q_i - \sum_{k=1}^{n} \frac{(\partial D_k/\partial q_i)}{D_k^2}\rho_k \text{ and } i=(1,n) \quad (22)$$

B. Case of Planar Robot

When it is a plan workspace, the obstacle can be seen as circular such as defined on figure 3. Their co-ordinates are such as:

$$\vec{M}_1 = \begin{bmatrix} x_{1b} & x_{2b} \end{bmatrix}^T$$

The coordinates related to the tip of the first arm are given by:

$$\vec{M}_2 = \vec{OA} = \begin{bmatrix} l_1.\cos(q_1) & l_1.\sin(q_1) \end{bmatrix}^T \quad (23)$$

The following relations are fulfilled:

$$\cos(\vec{M}_1, \vec{M}_2) = \frac{\vec{M}_1\vec{M}_2}{\|\vec{M}_1\|.\|\vec{M}_2\|} \quad (24)$$

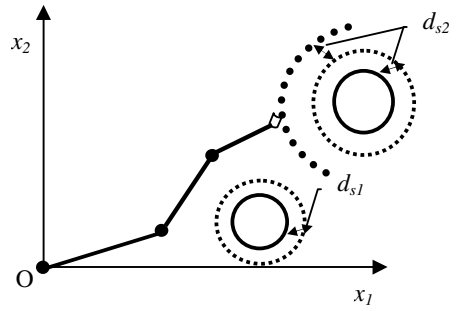


Fig. 4. Workspace of application

$$\sin(\vec{M}_1, \vec{M}_2) = \frac{d_1}{M_1} \tag{25}$$

Therefore,

$$\frac{d_1^2}{M_1^2} + \left(\frac{\vec{M}_1 \vec{M}_2}{\|\vec{M}_1\| . \|\vec{M}_2\|} \right)^2 = 1 \tag{26}$$

By using (24) and (25) in (26), the Euclidean distance d_1 is given by:

$$d_1^2 = x_{1b}^2 + x_{2b}^2 - (x_{1b}.\cos q_1 + x_{2b}.\sin q_1)^2 \tag{27}$$

And thus the anti-collision constraint is then:

$$\begin{aligned} D_1 &= -d_1^2 + (r_b + d_s)^2 \\ &= -x_{1b}^2 - x_{2b}^2 + (x_{1b}.\cos q_1 + x_{2b}.\sin q_1)^2 \\ &\quad + (r_b + d_s)^2 \end{aligned} \tag{28}$$

Where, *ds* is a safety distance.

The same procedure is used for *D2*, with:

$$\vec{M}_1' = \begin{bmatrix} x_{1b} - l_1.\cos q_1 & x_{2b} - l_1.\sin q_1 \end{bmatrix}^T$$

$$\vec{M}_2' = \begin{bmatrix} l_2.\cos(q_1 + q_2) & l_2.\sin(q_1 + q_2) \end{bmatrix}^T , \tag{29}$$

This gives:

$$\begin{aligned} D_2 &= -(x_{1b} - l_1.\cos q_1)^2 + (x_{2b} - l_1.\sin q_1)^2 + \\ &\quad [(x_{1b} - l_1.\cos q_1).\cos(q_1 + q_2) + (x_{2b} - \\ &\quad l_1.\sin q_1).\sin(q_1 + q_2))]^2 + (r_b + d_s)^2 \end{aligned} \tag{30}$$

For *D3*, the vectors $\vec{M}_1$ and $\vec{M}_2$ are replaced by

$$\begin{aligned} \vec{M}_1'' &= [(x_{1b} - l_1.\cos q_1 - l_2.\cos(q_1 + q_2)\, x_{2b} \\ &\quad - l_1.\sin q_1 - l_2.\sin(q_1 + q_2)]^T \\ \vec{M}_2'' &= \begin{bmatrix} l_3.\cos(q_1 + q_2 + q_3) & l_3.\sin(q_1 + q_2 + q_3) \end{bmatrix}^T \end{aligned} \tag{31}$$

So that:

$$\begin{aligned} D_3 &= -(x_{1b} - l_1.\cos q_1 - l_2.\cos(q_1 + q_2))^2 - (x_{2b} - l_1.\sin q_1 \\ &\quad - l_2.\sin(q_1 + q_2))^2 + [(x_{1b} - l_1.\cos q_1 - l_2.\cos(q_1 + q_2)). \\ &\quad \cos(q_1 + q_2 + q_3) - (x_{2b} - l_1.\sin q_1 - l_2.\sin(q_1 + q_2)) \\ &\quad \times \sin(q_1 + q_2 + q_3)]^2 + (r_b + d_s)^2 \end{aligned} \tag{32}$$

Thereafter, the expressions (28), (30) and (32) are introduced into the relation (20) of the anti-collision function h(q) and its gradient is determined by the relation (22).

IV. CONTROL OF THE ROBOT WITH OBSTACLE AVOIDANCE

The previous method is used to make the robot avoid the obstacle (circle in the bottom of figure 4). While ensuring the realization of the tracking trajectory, the half-circle representing the desired trajectory shares the plan in 2 zones: the concave zone and the convex zone. To prevent the arms from coming to cross the trajectory it is enough that they evolve in the convex zone (on the left of the trajectory). Such difficulty can also be solved as avoidance obstacle. Thus, it is assumed that there is a virtual obstacle materialized by a circle with the same center than the desired trajectory and of the radius R2 = R-d_{s2} (d_{s2} is a safety distance).

The tracking trajectory represented by a half-circle (fig 4), is involved by the control law (14). The implementation can be realized by the same scheme of the figure 1 in which:

-- x, $\dot{x}$, x_d, $\dot{x}_d$ and $\ddot{x}_d$ are replaced respectively by x_e, $\dot{x}_e$, x_{de}, $\dot{x}_{de}$ and $\ddot{x}_{de}$;

-- Jacobian J+ and $\dot{J}$ are replaced respectively by J_e^{-1} and $\dot{J}_e$.

The extended Jacobian *Je* is calculated according to (11), (15) , (16) and (17).

The scalar function *h(q)* used in the determination of *Ja* is given by relation (18). Since, this function must incorporates both: constraints due to the real obstacle, and the assumed obstacle (virtual). So, it is of the form:

$$h(q) = h_0(q) + \sum_{i=1}^{3} \rho 1_i . P(D1_i(q)) + \sum_{i=1}^{3} \rho 2_i . P(D2_i(q))$$

where (ds1, $\rho 1_i$, $D1_i$) and (ds2, $\rho 2_i$, $D2_i$) are related respectively to the real obstacle and the assumed one.

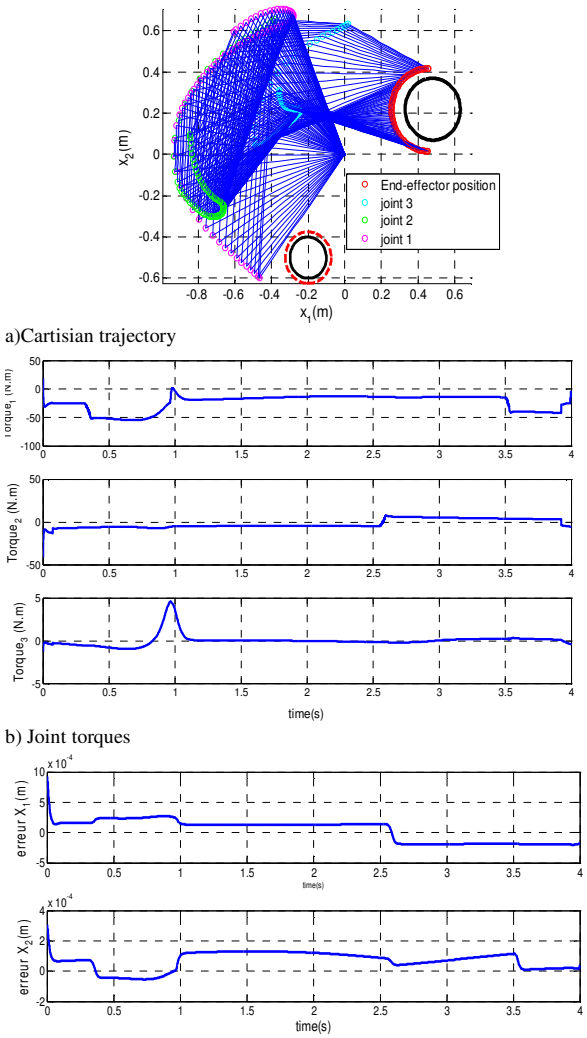
Tests are carried out in the same conditions related to the trajectory realization given in section II and moreover the control coefficients are not changed.

The parameters of the obstacle avoidance are set as:

$$\rho 1 = diag\begin{bmatrix} 0.02 & 0.07 & 0.001 \end{bmatrix}$$

$$\rho 2 = diag\begin{bmatrix} 0.2 & 0.1 & 0.01 \end{bmatrix}$$

The obtained results appear on figure 5. This one shows that with this method the obstacle avoidance is realized without disturbing the execution of the position tracking task.



a)Cartisian trajectory

b) Joint torques

c) Tracking errors

Fig.5. Robot Control with obstacle avoidance

Indeed, no robot arm comes into the protected zone around the obstacle or comes to cross the desired trajectory (arms evolve in the left side of the trajectory).

Moreover, the position tracking errors are around 0.1 mm and the joint torques remain within a suitable range.

V. CONCLUSION

In this paper, redundancy is applied to solve the problem of obstacles avoidance in robot control. Indeed, the r DOF in more are exploited to create self-motion which acts to optimize a scalar function. This function is defined in order to incorporate the anti collision constraints. The self motion vectors can be introduced in control by using the extended Jacobean method.

This method is applied in the case of 3DOF planar robot resulting from the PUMA 560 robot. The control incorporating this procedure is applied in the case where the end-effector tracks trajectory in a plane close to an obstacle. The obtained results show that this control method allows avoiding obstacle while assuring the desired performances of trajectory tracking.

REFERENCES

[1] Yunong Zhang and Jun Wang , "Obstacle Avoidance for Kinematic Redundant Manipulators Using a Dual Neural Network," , *IEEE Transactions on systems, vol. 34, N°1, Feb. 2004, pp. 752-759.*

[2] Fan-Tien Cheng, Jeng-Shi Chen and Fan-chu Kung, "Study and Resolution OF Singularities for 7-DOF Redundant Manipulator", *IEEE Transactions on industrial electronic, Vol. 45, N°. 3,June 1998, pp. 469-480.*

[3] Benallegue B.Daachi. and A Ramdane Cherif, "Commande neuronale adaptative de robot Manipulateur Redondant avec évitement d'obstacles fixes", International Francophone Conférence of Automation, Nantes, 08-10 July 2002, pp. 33-38.

[4] Kang Teresa, Solving inverse Kinematics Constraint problem for Highly Articulated Models. Master of Mathemtics in Computer Science, GE.University of Walerloo, Canada 2000.

[5] V. Perdereau, C.Passi and M. Drouin, " Real-time control of redundant robotic manipulators for mobile obstacle avoidance", Robotics and Autonomous Systems, 2002, pp. 41-59.

[6] Jin-Liang Chen,Jing-Sin Liu, Wan-Chi Lee and Tzu-Chen Liang, "On-line multi-criteria based collision-free posture generation of redundant manipulator in constrained workspace", Robotica(2002),Vol. 20,pp.625-636.

[7] Charles A.Klein, Caroline Chu-Jenq, and Shamir Ahmed , "A new formulation of the Extended Jacobien Method and its Use in Mapping Algorithmic Singularities for Kinematically Redundant Manipulators", *IEEE transaction on robotics and automation, vol.11N°.1, february 1995, pp. 50-55.*

[8] Krzysztof Tchon , "Quadratic Normal Forms of Redundant Robot Kinematics with application to singularity avoidance", *Robotics and Automation, IEEE Transactions,Vol.14, Issue 5, Oct. 1998, pp. 834 – 837.*

[9] B. Armstrong, O. Khatib and J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm", *IEEE International Conference on Robotics and Automation, 1986, pp. 510-518.*

[10] T.Shamir, "The Singularities of redundant Robot Arms". The International Journal of Robotics Research, Vol.9, No.1, February 1990 Page(s):113 – 121.

[11] Degao Li, Andrew A. Groldenberg, and Jean W. Zu "A New Method of Peak Torque Reduction With Redundant Manipulators". *Robotics and Automation, IEEE Transactions on Vol. 13, Issue 6, Dec. 1997 Page(s):845 - 853.*

[12] John M.Hollerbach,Member,IEEE, And Ki suh "Redundancy Resolution of Manipulators Through Torque Optimization" *Robotics and Automation, IEEE Journal of [legacy, pre - 1988] Vol. 3, Issue 4, Aug 1987 Page(s):308 – 316.*