# An evolutionary approach for the motion planning of redundant and hyper-redundant manipulators

Maria da Graça Marcos, J. A. Tenreiro Machado, T.-P. Azevedo-Perdicoúlis

# An Evolutionary Approach for the Motion Planning of Redundant and Hyper-redundant Manipulators

Maria da Graça Marcos [*], J. A. Tenreiro Machado [1], T-P Azevedo-Perdicoúlis[2]

[1] *Dept. of Electrotechical Eng, Institute of Engineering, Polytechnic Institute of Porto, Porto-Portugal*
*Email: jtm@isep.ipp.pt*
[2] *Dept. of Mathematics, University of Trás-os-Montes and Alto Douro, Vila Real-Portugal*
*Email: tazevedo@utad.pt*

**Abstract**

The trajectory planning of redundant robots is an important area of research and efficient optimization algorithms are needed. The pseudoinverse control is not repeatable, causing drift in joint space which is undesirable for physical control. This paper presents a new technique that combines the closed-loop pseudoinverse method with genetic algorithms, leading to an optimization criterion for repeatable control of redundant manipulators, and avoiding the joint angle drift problem. Computer simulations performed based on redundant and hyper-redundant planar manipulators show that, when the end-effector traces a closed path in the workspace, the robot returns to its initial configuration. The solution is repeatable for a workspace with and without obstacles in the sense that, after executing several cycles, the initial and final states of the manipulator are very close.

*Keywords:* Redundant Manipulators, Hyper-redundant Manipulators, Robots, Kinematics, Genetic Algorithms, Trajectory Planning.

## 1. Introduction

Kinematic redundancy occurs when a manipulator possesses more degrees of freedom than the required to execute a given task. In this case the inverse kinematics admits an infinite number of solutions, and a criterion to select one of them is required. Most of the research on redundancy deals with the use of these extra degrees of freedom and is referred to in the literature as the resolution of redundancy [1].

Many techniques for solving the kinematics of redundant manipulators that have been suggested control the end-effector indirectly, through the rates at which the joints are driven, using the pseudoinverse of the Jacobian (see, for instance, [2]). The pseudoinverse of the Jacobian matrix guarantees an optimal reconstruction of the desired end-effector velocity – in the least-squares sense – with the minimum-norm joint velocity. However, even though the joint velocities are instantaneously minimized, there is no guarantee that the kinematic singularities are avoided [3]. Moreover, this method has the generally undesirable property that repetitive end-effector motions do not necessarily yield repetitive joint motions. Klein and Huang [4] were the first to observe this phenomenon for the case of

[*]Corresponding author.
*Dept. of Mathematics, Institute of Engineering, Polytechnic Institute of Porto, Porto-Portugal*
*E-mail address:* mgm@isep.ipp.pt

the pseudoinverse control of a planar three-link manipulator. A large volume of research has been produced in the last few years in this topic [5-8]. For example, Zhang *et al.* [9] solve the joint angle drift problem by means of a dual-neural-network based quadratic-programming approach.

Baillieul [10] proposed a modified Jacobian matrix called the extended jacobian matrix. The extended jacobian is a square matrix that contains the additional information necessary to optimize a certain function. The inverse kinematic solutions are obtained through the inverse of the extended jacobian. The algorithms, based on the computation of the extended jacobian matrix, have a major advantage over the pseudoinverse techniques, because they are locally cyclic [11]. The disadvantage of this approach is that, while mechanical singularities may be avoided, typical algorithmic singularities [12] arise from the way the constraint restricts the motion of the mechanism [13].

One optimization method that is gaining popularity for solving complex problems in robotics is the Genetic Algorithm (GA). GAs are population-based stochastic and global search methods. Their performance is superior to that revealed by classical optimization techniques [14] and has been used successfully in robot path planning.

Parker *et al*. [15] used GAs to position the end-effector of a robot at a target location, while minimizing the largest joint displacement. This method has some shortcomings, such as the lack of precision and is affected by the values of the weights. Arakawa *et al*. [16] proposed a virus-evolutionary genetic algorithm, composed of a host population and a virus population with subpopulations, for the trajectory generation of redundant manipulators without collision, that optimize the total energy. The operators of crossover, mutation, virus infection and selection are executed in each subpopulation independently. Kubota *et al*. [17] studied a hierarchical trajectory planning method for a redundant manipulator using a virus-evolutionary GA. This method runs, simultaneously, two processes. One process calculates some manipulator collision-free positions and the other generates a collision-free trajectory by combining these intermediate positions. De la Cueva and Ramos [18] proposed a GA for planning paths without collisions for two robots, both redundant and non-redundant, sharing the same workspace. The GA works directly over the task space adopting the direct kinematics. Each robot is associated to one population and each string of a population represents a complete robot path. Nishimura *et al*. [19] proposed a motion planning method using an artificial potential field and a GA for a hyper-redundant manipulator whose workspace includes several obstacles. The motion planning is divided into two sub problems. The first is the "Path planning" that generates a trajectory leading the tip of manipulator to the goal without collisions, using the artificial potential field concept. The second consists in the "Collision-free sequence generation" that generates a sequence of movements by which distinct parts of the manipulator can avoid collisions with the obstacles. McAvoy and Sangolola [20] proposed an approach with GAs for optimal point-to-point motion planning of kinematically redundant manipulators. Their approach combines B-spline curves, for the generation of smooth trajectories, with Gas, for obtaining the optimal solution. Peng and Wei [21] presented the ASAGA trajectory planning method of redundant manipulators by combining a stochastic search algorithm (simulated annealing algorithm) and a GA. In the ASAGA the selection, crossover and mutation operators are adjusted by using an adaptive mechanism based on the fitness value. Zhang *et al.* [22] proposed an algorithm to solve the inverse kinematics of a flexible macro-micro manipulator system which combines a GA and a neural network. Pires *et al.* [23] proposed a multi-objective genetic algorithm, when considering up to five simultaneous objectives, to generate manipulator trajectories and for obstacle avoidance.

Having these ideas in mind, the paper is organized as follows. Section 2 introduces the fundamentals of the kinematics of redundant manipulators. Based on these concepts, section 3 presents the new closed-loop inverse kinematics algorithm with genetic algorithms (CLGA) and the open-loop genetic algorithm (OLGA). Section 4 presents the simulation results in a workspace without and with obstacles. For

comparison purposes some results for the closed-loop pseudoinverse (CLP) method are also presented. Finally, section 5 draws the main conclusions.

## 2. Kinematics and dynamics of redundant manipulators

We consider a manipulator with $n$ degrees of freedom whose joint variables are denoted by $\mathbf{q} = \left[q_1,\ q_2,\ \ldots,\ q_n\right]^T$. We assume that the class of tasks we are interested in can be described by $m$ variables, $\mathbf{x} = \left[x_1,\ x_2,\ \ldots,\ x_m\right]^T$, $m < n$, and that the relation between $\mathbf{q}$ and $\mathbf{x}$ is given by the direct kinematics:

$$\mathbf{x} = f\left(\mathbf{q}\right) \tag{1}$$

Differential kinematics of robot manipulators was introduced by Whitney [24] that proposed the use of differential relationships to solve for the joint motion from the Cartesian trajectory of the end-effector. Differentiating (1) with respect to time yields:

$$\dot{\mathbf{x}} = \mathbf{J}\left(\mathbf{q}\right)\dot{\mathbf{q}} \tag{2}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^m$, $\dot{\mathbf{q}} \in \mathbb{R}^n$ and $\mathbf{J}\left(\mathbf{q}\right) = \partial f\left(\mathbf{q}\right)/\partial \mathbf{q} \in \mathbb{R}^{m \times n}$. Hence, it is possible to calculate a path $\mathbf{q}\left(t\right)$ in terms of a prescribed trajectory $\mathbf{x}\left(t\right)$ in the operational space.

Equation (2) can be inverted to provide a solution in terms of the joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\left(\mathbf{q}\right)\dot{\mathbf{x}} \tag{3}$$

where $\mathbf{J}^{\#}$ is the Moore-Penrose generalized inverse of the Jacobian $\mathbf{J}$ [2, 25].

## 3. Robot trajectory control

The Jacobian of a $n$-link planar manipulator (i.e., $m = 2$) has a simple recursive nature according with the expressions:

$$\mathbf{J} = \begin{bmatrix} -l_1 S_1 - \ldots - l_n S_{1\ldots n} & \cdots & -l_n S_{1\ldots n} \\ l_1 C_1 + \ldots + l_n C_{1\ldots n} & \cdots & l_n C_{1\ldots n} \end{bmatrix} \tag{4}$$

where $l_i$ is the length of link $i$, $q_{i\ldots k} = q_i + \ldots + q_k$, $S_{i\ldots k} = Sin\left(q_{i\ldots k}\right)$ and $C_{i\ldots k} = Cos\left(q_{i\ldots k}\right)$, $i, k \in \mathbb{N}$.

In the closed-loop pseudoinverse (CLP) method the joint positions can be computed through the time integration of the expression:

$$\Delta\mathbf{q} = \mathbf{J}^{\#}\left(\mathbf{q}\right)\Delta\mathbf{x} \tag{5}$$

where $\Delta\mathbf{x} = \mathbf{x}_{ref} - \mathbf{x}$ and $\mathbf{x}_{ref}$ is the vector of reference position in the operational space. Nevertheless, in a previous study, addressing the CLP method [26], it was concluded that this method leads to unpredictable, not repeatable, arm configurations and reveals properties resembling those that occur in chaotic systems.

Genetic algorithms (GAs) are a method for solving both constrained and unconstrained optimization problems, based on the mechanics of natural genetics and selection, that was first introduced by Holland [27]. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the fitness or the cost function. The GA modifies repeatedly the population of individuals (possible solutions). At each step, the genetic algorithm selects individuals at random, from the current population, to be parents, and uses them to produce the offspring for the next generation. Over successive generations, the population evolves towards an optimal solution. The GAs can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, not differentiable, stochastic, or highly nonlinear.

Bearing these facts in mind, in this paper we propose a new method that combines the CLP with a GA, namely the closed-loop inverse kinematics algorithm with genetic algorithms (CLGA).

In order to find an initial joint configuration of the manipulator is used another GA, adopting the direct kinematics, which is denoted as OLGA.

In both cases the optimal configuration is the one that minimizes the fitness function according to some specified criteria.

### 3.1 The CLGA formulation

The CLGA adopts the closed-loop structure without requiring the calculation of the pseudoinverse. The CLGA uses an extended Jacobian matrix, $\mathbf{J}^* \in \mathbb{R}^{n\times n}$, and an extended vector, $\Delta\mathbf{x}^* \in \mathbb{R}^n$, as a way to limit the joint configurations for a given end-effector position.

The definition of $\mathbf{J}^*$ and $\Delta\mathbf{x}^*$ take the form:

$$\mathbf{J}^* = \begin{bmatrix} -l_1 S_1 - \ldots - l_n S_{1\ldots n} & \cdots & -l_n S_{1\ldots n} \\ l_1 C_1 + \ldots + l_n C_{1\ldots n} & \cdots & l_n C_{1\ldots n} \\ j_{(m+1)1} & \cdots & j_{(m+1)n} \\ \cdots & \cdots & \cdots \\ j_{n1} & \cdots & j_{nn} \end{bmatrix} \qquad \Delta\mathbf{x}^* = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_{m+1} \\ \vdots \\ \Delta x_n \end{bmatrix} \tag{6}$$

where the matrix elements $j_{ik}$ and $\Delta x_i$, $i = m+1,\ldots,n$ and $k = 1,\cdots,n$, are values generated by the GA, satisfying the additional imposed constraints.

#### 3.1.1 Representation and operators in the CLGA

An initial population of strings, with dimension $n_P = N$, is constructed at random and the search is then carried out among this population. Each chromosome (string) is implemented by a matrix of

$n_V = (n-m)\times(n+1)$ values (genes), with $j_{dk}$ and $\Delta x_d$, $d = 1,\ldots,n-m$, $k = 1,\ldots,n$, consisting in floating-point numbers initialized in the range $[var_{min}, var_{max}]$. For the generation $T$, the i[th] chromosome of the population is represented as:

$$\left[\mathbf{J}^{(T,i)} \vdots \Delta\mathbf{x}^{(T,i)}\right] \tag{7}$$

where $\mathbf{J}^{(T,i)} = \left[ j_{dk}^{(T,i)} \right] \in \mathbb{R}^{(n-m)\times n}$ and $\Delta\mathbf{x}^{(T,i)} = \left[ \Delta x_d^{(T,i)} \right] \in \mathbb{R}^{n-m}$.

The three different operators used in the genetic algorithm are reproduction, crossover and mutation. In what respecting the reproduction operator, the successive generations of new strings are generated on the basis of their fitness function. In this case, it is used a rank weighting to select the strings from the old to the new population. For the crossover operator, the strings are randomly grouped together into pairs and a crossover point is randomly selected for each one of the $n-m$ lines of the parent. Then crossover is performed among pairs. Finally, for the mutation operator, one variable value from the $n_V$ values of the chromosome is replaced with a new random one.

The CLGA procedure is shown in figure 1, where $\mathbf{x}_{ref}$ is the vector of reference position in the operational space and $\mathbf{x}_{ini}$ is a vector representing the position of the end-effecter in the operational space.

**Begin**
  $T = 0$
  *calculate* $\Delta\mathbf{x} = \mathbf{x}_{ref} - \mathbf{x}_{ini}$, $\mathbf{J}$
  *initialize random population* $P(T) = \left[\left[\mathbf{J}^{(T,1)} \vdots \Delta\mathbf{x}^{(T,1)}\right], \ldots, \left[\mathbf{J}^{(T,N)} \vdots \Delta\mathbf{x}^{(T,N)}\right]\right]$
  *get* $\Delta\mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q})\Delta\mathbf{x}^*$ *and* $\mathbf{q} = \int\Delta\mathbf{q}$
  *evaluate* $P(T)$
  ***repeat***
    *selection parents from* $P(T)$
    *crossover* $P(T)$
    *mutation* $P(T)$
    *form new generation* $P(T)$
    *get* $\Delta\mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q})\Delta\mathbf{x}^*$ *and* $\mathbf{q} = \int\Delta\mathbf{q}$
    *evaluate* $P(T)$
    $T = T + 1$
  ***until*** *termination condition is TRUE*
  *get new* $\mathbf{q}$
***End***

Fig. 1. Procedure for the CLGA.

3.1.2 Optimization criteria

The fitness function minimizes the joint displacement between the current joint position and the initial joint position, through the following function:

$$f_1 = A\dot{\mathbf{q}}^T\dot{\mathbf{q}} + B\left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right)^T\left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right) \tag{8}$$

where $A, B \in \mathbb{R}^+$ denotes weighting factors, $\mathbf{q}_0$ and $\mathbf{q}$ represents the initial and current joint configurations, respectively, and $\Delta t$ is a step time increment.

3.2 The OLGA formulation

The OLGA trajectory planning adopts a simple open-loop structure, as we can see in figure 2. An initial population of strings, with dimension $n_P = N$, is constructed at random and the search is then carried out among this population. Each chromosome is defined through an array of $n_V = n$ values, $q_i$, $i = 1,\ldots,n$, represented as floating-point numbers initialized in the range $[q_{\min}, q_{\max}]$. For the generation $T$, the i[th] chromosome of the population is represented as:

$$\mathbf{q}^{(T,i)} = \left(q_1^{(T,i)}, \ \ldots, q_n^{(T,i)}\right) \tag{9}$$

The end-effector position, $\mathbf{x}^{(T,i)}$, for each configuration, $\mathbf{q}^{(T,i)}$, is easily calculated using the direct kinematics.

3.2.1 Optimization criteria

The evaluation function is defined based on the positional error of the end-effector:

$$P_{error} = \sqrt{\left(x_c - x_f\right)^2 + \left(y_c - y_f\right)^2} \tag{10}$$

where $\mathbf{x}_c = \left(x_c, y_c\right)$ and $\mathbf{x}_f = \left(x_f, y_f\right)$ are vectors representing the end-effector current position and the desired final position, respectively. Therefore, the algorithm minimizes the function $f_2 = P_{error}$.

**1** ***Begin***
**2** $T = 0$
**3** *initialize random population* $P(T) = \left[ \mathbf{q}^{(T,1)},\ \mathbf{q}^{(T,2)},\ \ldots,\ \mathbf{q}^{(T,N)} \right]$
**4** *get* $\mathbf{X} = \left[ \mathbf{x}^{(T,1)},\ \ldots, \mathbf{x}^{(T,N)} \right]$ *using direct kinematics*
**5** *evaluate* $P(T)$
**6** ***Repeat***
**7** *selection parents from* $P(T)$
**8** *crossover* $P(T)$
**9** *mutation* $P(T)$
**10** *form new generation* $P(T)$
**11** *get* $\mathbf{X}$ *using direct kinematics*
**12** *evaluate* $P(T)$
**13** $T = T + 1$
**14** ***until*** *termination condition is TRUE*
**15** *get new* $\mathbf{q}$
**16** ***End***

Fig. 2. Procedure for the OLGA.

## 4. Simulation results

In this section we start by analyzing the performance of the CLGA for a free workspace and then we study the effect of including several types of obstacles in the working environment.

Without lacking of generality, in the following experiments are adopted arms having identical link lengths, $l_1 = l_2 = \ldots = l_n$.

The experiments consist in the analysis of the kinematic performance of a planar manipulator with $n = \{3,\ 4,\ 5,\ 6,\ 7\}$ rotational joints, denoted as *nR*-robot, that is required to repeat a circular motion in the operational space with frequency $\omega_0 = 7.0 \text{ rad sec}^{-1}$, center at $r = \left( x_1^{\,2} + x_2^{\,2} \right)^{1/2}$, radius $\rho = 0.5$ and a step time increment of $\Delta t = 10^{-3}$ sec. The goal here is to position the end-effector of the *nR*-robot at a target location while minimizing the joint angle drift using the fitness function $f_1$ with $A = B = 1$. The initial joint configuration is obtained using the OLGA with the fitness function $f_2$.

The average of the positional error for $n_C$ cycles is given by the expression:

$$\overline{P}_{error} = \frac{\sum_{i=1}^{k} P_{error}}{k} \tag{11}$$

where $k$ is the number of sampling points and is defined as:

$$k = \frac{2\pi}{\omega_0 \Delta t} n_C \tag{12}$$

The average of the total joint displacement for the *nR*-robot is given by the expression:

$$\overline{\Delta \mathbf{q}} = \frac{\|\Delta \mathbf{q}\|}{n} \tag{13}$$

where $\Delta \mathbf{q} = \mathbf{q}_f - \mathbf{q}_0$, is the vector of the joint displacement between the final, $\mathbf{q}_f$, and the initial, $\mathbf{q}_0$, joint configurations, and $\| \cdot \|$ represents the Euclidean norm.

The CLGA algorithm adopts crossover probabilities of $p_c = 0.5$ for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$ and mutation probabilities of $p_m = 0.5$ and $p_m = 0.3$ for $n = \{3,\ 4\}$ and $n = \{5,\ 6,\ 7\}$, respectively. The string population is $n_P = \{200,\ 400,\ 800,\ 1200,\ 1600\}$ for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$, respectively, and the results are obtained for $n_G = 200$ consecutive generations. Each variable value is initialized in the range $[-1,\ 1]$.

The OLGA algorithm adopts crossover and mutation probabilities of $p_c = 0.5$ and $p_m = 0.5$, respectively, a string population of $n_P = 1600$ and the results are obtained for $n_G = 200$ consecutive generations. Each variable value is initialized in the range $[-2\pi,\ 2\pi]$.

4.2.1 The CLGA performance in a workspace without obstacles

The average of the positional error, $\overline{P}_{error}$, for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$ rotational joints, $n_C = 50$ cycles and radial distance $r = \{0.7,\ 1.0,\ 2.0\}$, is depicted in figure 3.

We observe that:

i) the CLGA gives good precision in the task of positioning the end-effector at the target position;
ii) in general, we get better results for the radial distance $r = 2.0$ and worse results for the radial distance $r = 0.7$;
iii) in general, the positional error gets worse when the number of joints increases.

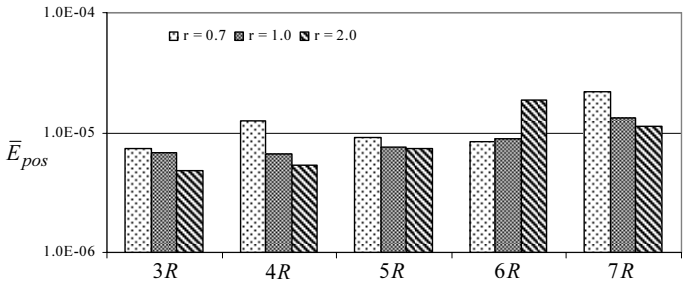


Fig. 3. $\overline{P}_{error}$ of the *nR*-robot during $n_C = 50$ cycles for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$ and $r = \{0.7,\ 1.0,\ 2.0\}$.

Figures 4-5 show successive robot configurations, for $n = \{3,\ 7\}$ rotational joints and $r = 2.0$, during the 1st and 50th cycles, respectively. As we can see, the joint configurations are very similar, for both cycles, revealing that the joint positions are repetitive. For comparison purposes, figures 6-7 show successive robot configurations, for $n = \{3,\ 7\}$ rotational joints and $r = 2.0$, during the 1st and 50th cycles, respectively, for the CLP method. As we can see, the joint configurations are very different revealing that the joint positions are not repetitive.

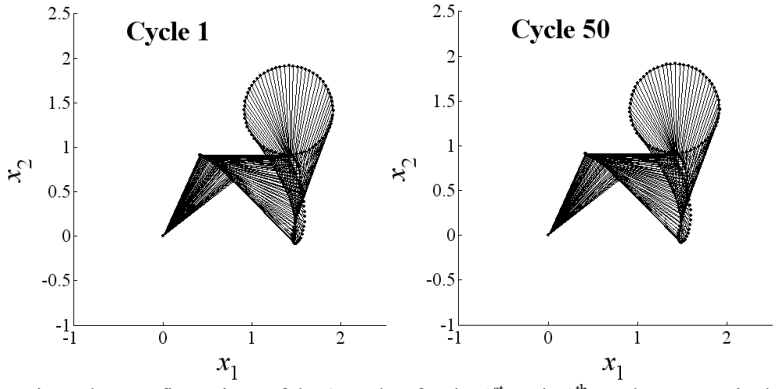


Fig. 4. CLGA successive robot configurations of the 3*R*-robot for the 1st and 50th cycles, respectively, and $r = 2.0$.

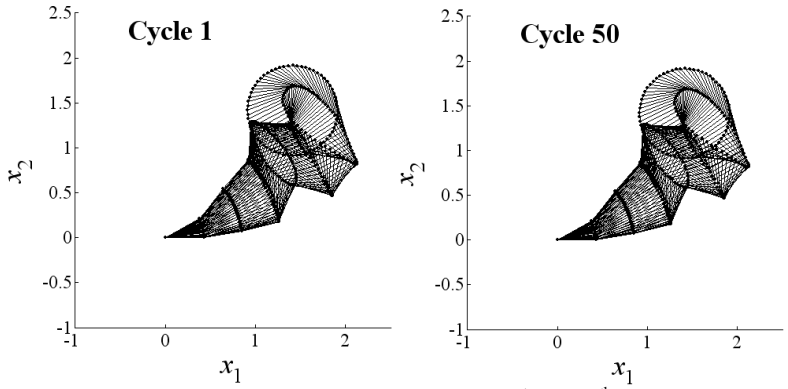


Fig. 5.CLGA successive robot configurations of the 7*R*-robot for the 1st and 50th cycles, respectively, and $r = 2.0$.

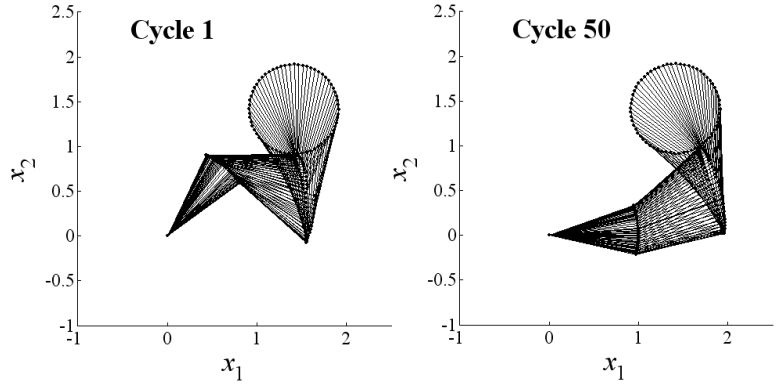


Fig. 6. CLP successive robot configurations of the 3*R*-robot for the 1st and 50th cycles, respectively, and $r = 2.0$.

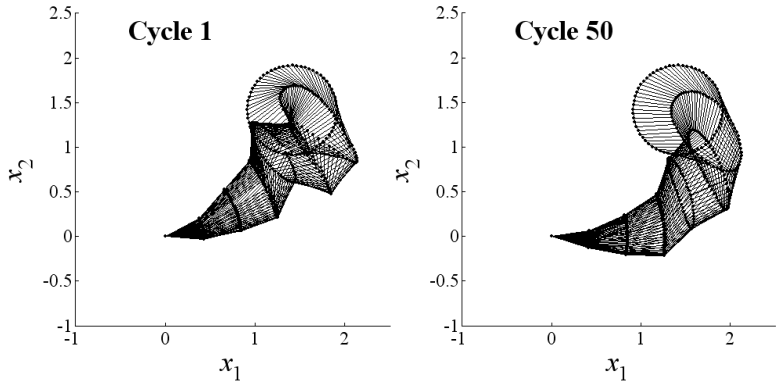


Fig. 7. CLP successive robot configurations of the 7*R*-robot for the 1st and 50th cycles, respectively, and $r = 2.0$.

In order to evaluate the joint angle drift after executing $n_C = 50$ cycles, table 1 shows the average of the total joint displacement, $\overline{\Delta \mathbf{q}}$, when $n = \{3,\ 4,\ 5,\ 6,\ 7\}$ and $r = \{0.7,\ 1.0,\ 2.0\}$, for the CLGA and the CLP methods. We verify that for the CLGA we get some drift in joint positions, but the values are very similar for all the manipulators and for all of the radial distances. For the CLP we get an high drift in the joint positions revealing that this method leads to unpredictable arm configurations.

| **CLGA** | $r = 0.7$ | $r = 1.0$ | $r = 2.0$ |
|---|---|---|---|
| 3*R* | 9.96E-04 | 8.84E-04 | 1.08E-03 |
| 4*R* | 7.12E-04 | 7.38E-04 | 5.70E-04 |
| 5*R* | 6.73E-04 | 5.42E-04 | 6.15E-04 |
| 6*R* | 5.98E-04 | 4.81E-04 | 8.57E-04 |
| 7*R* | 1.26E-03 | 5.44E-04 | 5.39E-04 |

| **CLP** | $r = 0.7$ | $r = 1.0$ | $r = 2.0$ |
|---|---|---|---|
| 3*R* | 1.35E+01 | 6.41E+00 | 5.80E-01 |
| 4*R* | 8.2E+00 | 4.4E+00 | 5.8E-01 |
| 5*R* | 7.2E+00 | 2.2E+00 | 4.4E-01 |
| 6*R* | 5.4E+00 | 4.9E+00 | 3.0E-01 |
| 7*R* | 4.2E+00 | 2.4E+00 | 2.0E-01 |

Table 1. CLGA and CLP average of the total joint displacement, $\overline{\Delta \mathbf{q}}$, of the *nR*-robot, $r = \{0.7,\ 1.0,\ 2.0\}$ and after $n_C = 50$ cycles.

Figure 8-9 shows the joint positions and the Fourier transform of the robot joint velocities for the 1st joint and $n_C = 50$, $n = \{3,\ 7\}$ and $r = \{0.7,\ 2.0\}$, with the CLGA and the CLP methods. For the other joints the results are similar to those of the 1st joint.

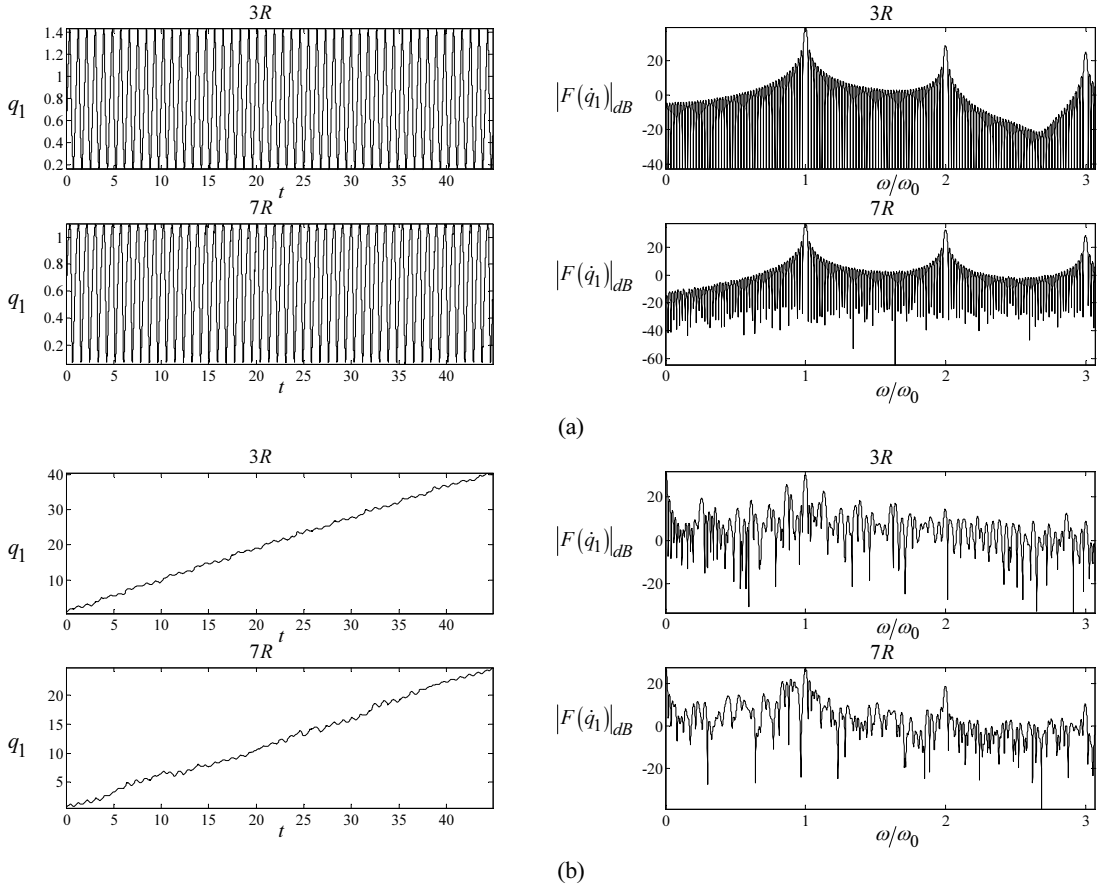


Fig. 8. The $\{3,\ 7\}$ *R*-robot 1st joint positions *versus* time and $\left|F\left\{\dot{q}_1(t)\right\}\right|$ *vs* $\omega/\omega_0$ during $n_C = 50$ cycles for $r = 0.7$ and (a) CLGA (b) CLP .

For the CLGA, we conclude that:

i) repetitive trajectories in the operational space lead to periodic trajectories in the joint space;
ii) the initial and final joint positions, for each of the cycles, are very close;
iii) the signal energy is concentrated essentially in the fundamental and multiple higher harmonics.

For the CLP, we conclude that the results depend on the circle being executed. Besides the position drifts, occur unpredictable motions with severe variations that lead to high joint transients. Moreover, for $r = 0.7$ we verify that a large part of the energy is distributed along several sub-harmonics, revealing properties similar to those that occur in chaotic systems.

We verify that the CLGA has a good performance in repetitive motion tasks because we get not only a good positioning but also a repetitive behavior.

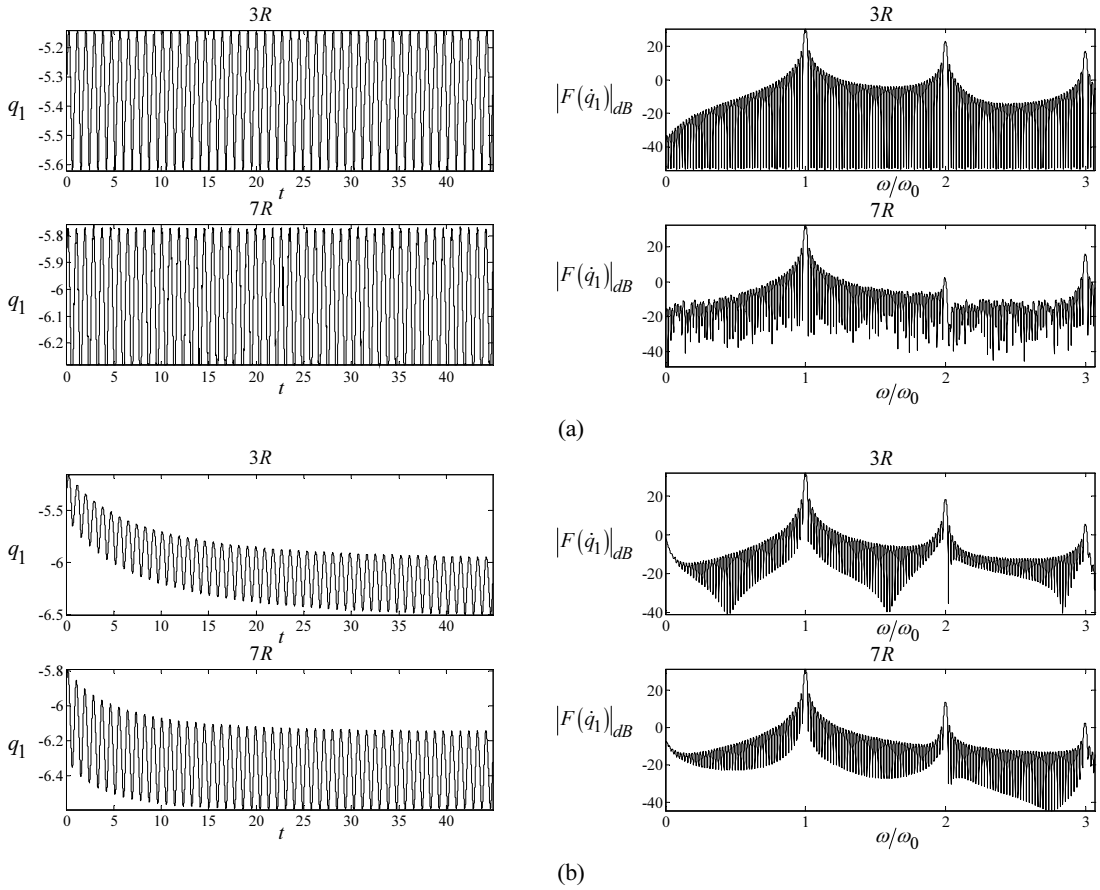


Fig. 9. The $\{3,\ 7\}$ *R*-robot 1st joint positions *versus* time and $\left|F\left\{\dot{q}_1(t)\right\}\right|$ *vs* $\omega/\omega_0$ during $n_C = 50$ cycles for $r = 2.0$ and (a) CLGA (b) CLP.

4.2.2 The CLGA performance in a workspace with obstacles

This section presents the results of several simulations, when considering two obstacles in the workspace. At the end of the section, are presented some examples in a workspace with three obstacles. When, for a given joint configuration, some part of the manipulator is inside an obstacle, the CLGA simply rejects the configuration and generates a new population element.

4.2.2.1 The CLGA performance in a workspace with two obstacles

In the experiments, the position of the obstacles in the workspace depends on the radial distance. In figures 10-12 are represented the positions of the obstacles in the workspace for each radial distance, $O(r;\ i)$, for $r = \{0.7,\ 1.0,\ 2.0\}$ and $i = 1, 2, 3$.

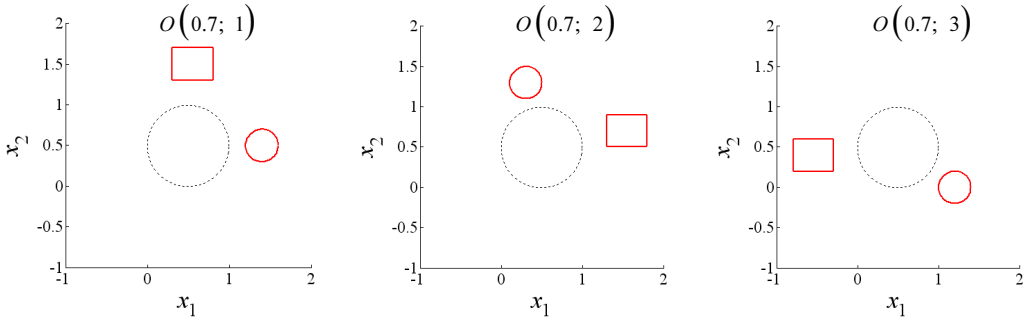


Fig. 10. Workspaces for $O(0.7;\ i)$, $i = 1, 2, 3$.

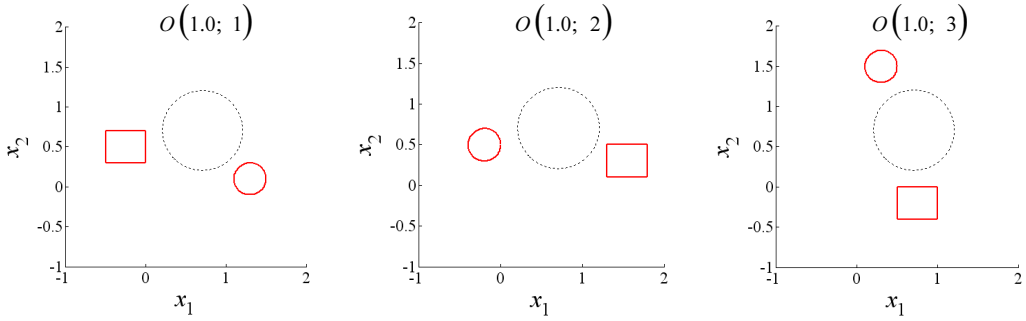


Fig. 11. Workspaces for $O(1.0;\ i)$, $i = 1, 2, 3$.

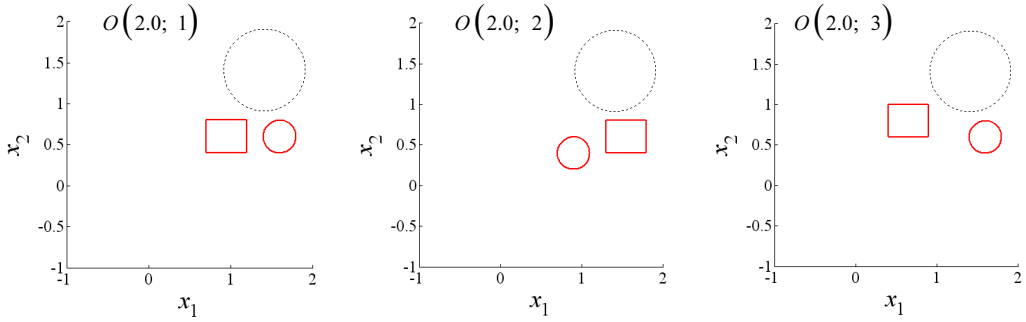


Fig. 12. Workspaces for $O(2.0;\ i)$, $i = 1, 2, 3$.

The average of the positional error, $\overline{P}_{error}$, for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$, $n_C = 50$ cycles and $O(r;\ i)$ for $r = \{0.7,\ 1.0,\ 2.0\}$, $i = 1, 2, 3$, is presented in figures 13-15, respectively.

We observe that:

i) the CLGA gives good precision in the task of positioning the end-effector at the target position, while avoiding the obstacles in the workspace;

ii) in general, we get better results for the radial distance $r = 2.0$ and worse results for $r = 0.7$;

iii) in general, the positional error, $\overline{P}_{error}$, gets worse when the number of joints increases.

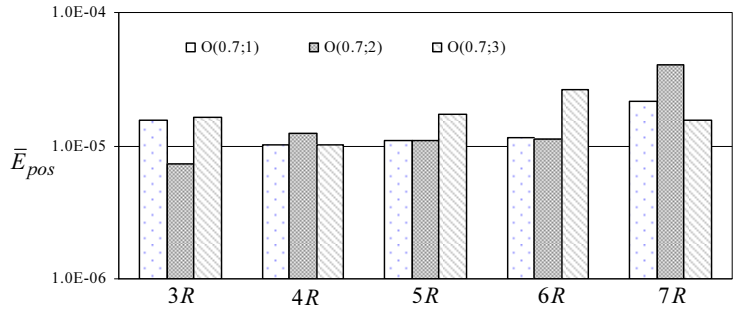


Fig. 13. $\overline{P}_{error}$ of the *nR*-robot during $n_C = 50$ cycles for $O\left(0.7;\ i\right)$, $i = 1, 2, 3$.

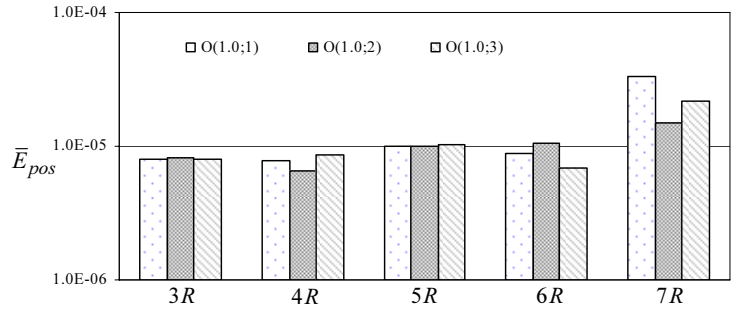


Fig. 14. $\overline{P}_{error}$ of the *nR*-robot during $n_C = 50$ cycles for $O\left(1.0;\ i\right)$, $i = 1, 2, 3$.

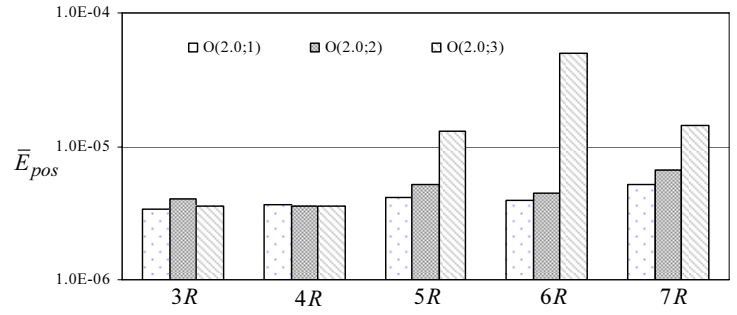


Fig. 15. $\overline{P}_{error}$ of the *nR*-robot during $n_C = 50$ cycles for $O\left(2.0;\ i\right)$, $i = 1, 2, 3$.

Figures 16-17 show successive robot configurations, for $n = \{3,\ 7\}$ and $O\left(2.0;\ i\right)$, $i = 1, 2, 3$, during the 1st and 50th cycles, respectively, revealing that the motion is repetitive.

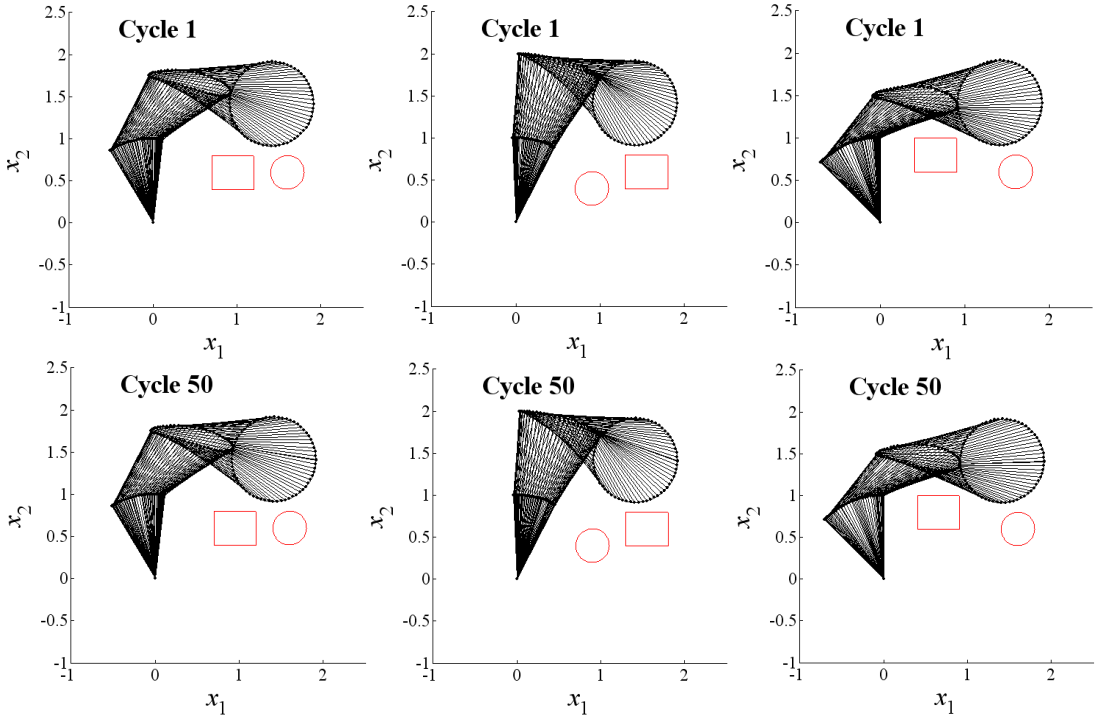


Fig. 16. Successive robot configurations of the 3*R*-robot for the 1st and 50th cycles, respectively, and $O\left(2.0;\ i\right)$, $i = 1, 2, 3$.

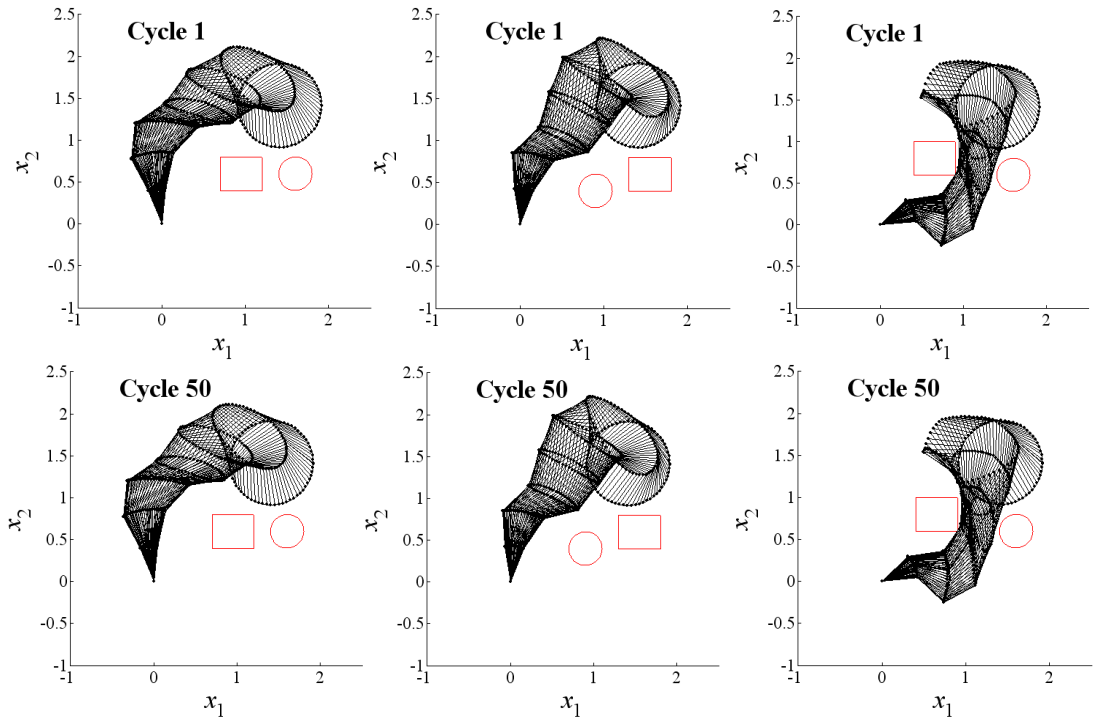


Fig. 17. Successive robot configurations of the 7*R*-robot for the 1st and 50th cycles, respectively, and $O\left(2.0;\ i\right)$, $i = 1, 2, 3$.

Table 2 shows the average of the joint displacement, $\overline{\Delta \mathbf{q}}$, for $n = \{3,\ 4,\ 5,\ 6,\ 7\}$, and $O(r;\ i)$, $r = \{0.7,\ 1.0,\ 2.0\}$, $i = 1, 2, 3$. We conclude that the results are consistent with those we obtained in a workspace without obstacles.

| | $O(0.7;\ 1)$ | $O(0.7;\ 2)$ | $O(0.7;\ 3)$ | $O(1.0;\ 1)$ | $O(1.0;\ 2)$ | $O(1.0;\ 3)$ | $O(2.0;\ 1)$ | $O(2.0;\ 2)$ | $O(2.0;\ 3)$ |
|---|---|---|---|---|---|---|---|---|---|
| $3R$ | 8.63E-04 | 1.11E-03 | 1.02E-03 | 7.57E-04 | 7.63E-04 | 8.02E-04 | 6.15E-04 | 6.64E-04 | 6.51E-04 |
| $4R$ | 6.16E-04 | 6.86E-04 | 5.98E-04 | 6.07E-04 | 7.39E-04 | 6.85E-04 | 4.29E-04 | 4.19E-04 | 4.22E-04 |
| $5R$ | 7.31E-04 | 7.04E-04 | 5.40E-04 | 5.63E-04 | 4.63E-04 | 4.74E-04 | 3.56E-04 | 4.16E-04 | 7.14E-04 |
| $6R$ | 4.66E-04 | 6.28E-04 | 5.92E-04 | 4.67E-04 | 3.45E-04 | 6.23E-04 | 2.38E-04 | 2.96E-04 | 7.33E-04 |
| $7R$ | 5.16E-04 | 3.75E-04 | 3.95E-04 | 4.63E-04 | 5.39E-04 | 2.95E-04 | 7.07E-04 | 3.94E-04 | 1.96E-03 |

Table 2. The average of the joint displacement of the *nR*-robot, $\overline{\Delta \mathbf{q}}$, after $n_C = 50$ cycles for $O(r;\ \mathrm{i})$, $r = \{0.7,\ 1.0,\ 2.0\}$, $i = 1, 2, 3$.

Figures 18-20 show the joint positions and the Fourier transform of the robot joint velocities for the 1st joint and $n_C = 50$ cycles, $n = \{3,\ 7\}$, $O(2.0;\ i)$, $i = 1, 2, 3$. For the others joints the results are similar to the verified ones for the 1st joint.

We conclude that:

i) repetitive trajectories in the operational space lead to periodic trajectories in the joint space;
ii) the initial and final joint positions for each one of the cycles are very close;
iii) the signal energy is concentrated essentially in the fundamental and multiple higher harmonics.

We observe also that the results are consistent with those of the previous section and that the presence of obstacles does not present an additional complexity for the CLGA to reach a repetitive solution.

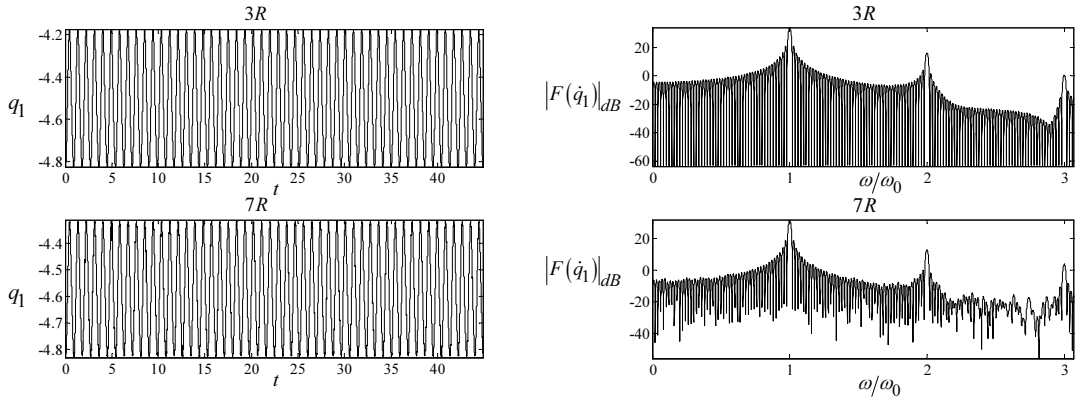


Fig. 18. The $\{3,\ 7\}$ *R*-robot 1st joint positions *versus* time and $\left|F\{\dot{q}_1(t)\}\right|$ *vs* $\omega/\omega_0$ during $n_C = 50$ cycles for $O(2.0;\ 1)$.

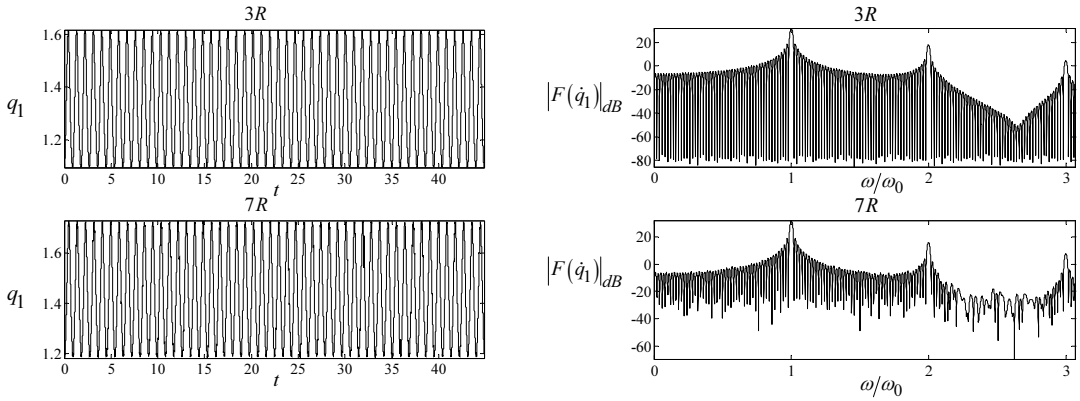


Fig. 19. The $\{3,\ 7\}$ *R*-robot 1st joint positions *versus* time and $\left|F\left\{\dot{q}_1(t)\right\}\right|$ *vs* $\omega/\omega_0$ during $n_C = 50$ cycles for $O(2.0;\ 2)$.

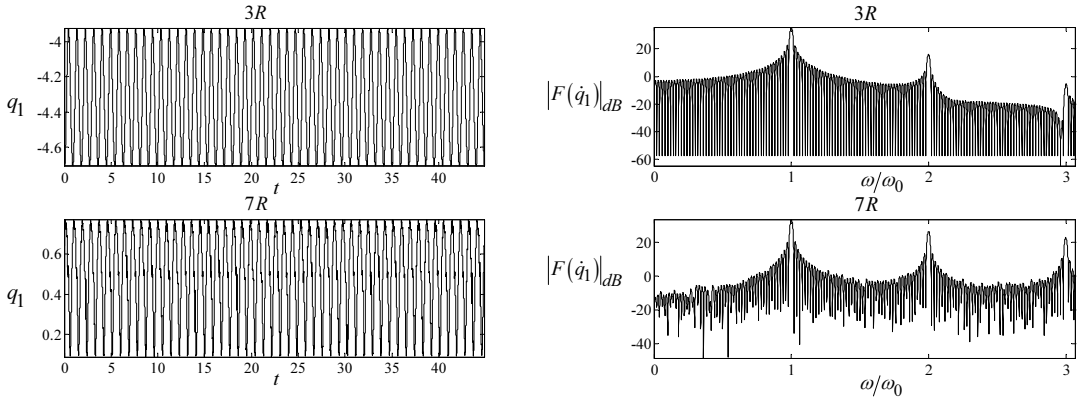


Fig. 20. The $\{3,\ 7\}$ *R*-robot 1st joint positions *versus* time and $\left|F\left\{\dot{q}_1(t)\right\}\right|$ *vs* $\omega/\omega_0$ during $n_C = 50$ cycles for $O(2.0;\ 3)$.

4.2.2.2 The CLGA performance in a workspace with three obstacles

One of the problems that occurs when we use the repeatability criterion is that, if the initial joint configuration of the manipulator is not well adjusted for the task, then will occur 'jumps' of the robot structure to prevent the collision with the obstacles. In the experiments the trajectory is repetitive and, therefore, these jumps may occur during several cycles.

In figures 21-22 we use a workspace with three obstacles. For $n = 3$, after two different executions of the algorithm, we obtained two different initial joint configurations with distinct consequences, in terms of the positional error and performance in the execution of the task. However, if we use a manipulator with $n = 7$ rotational joints there is no problem to find a good solution for the initial positions. Anyway, the manipulator motion is always repeatable and the drift in the joint positions is similar to the one we obtained in the previous experiments.

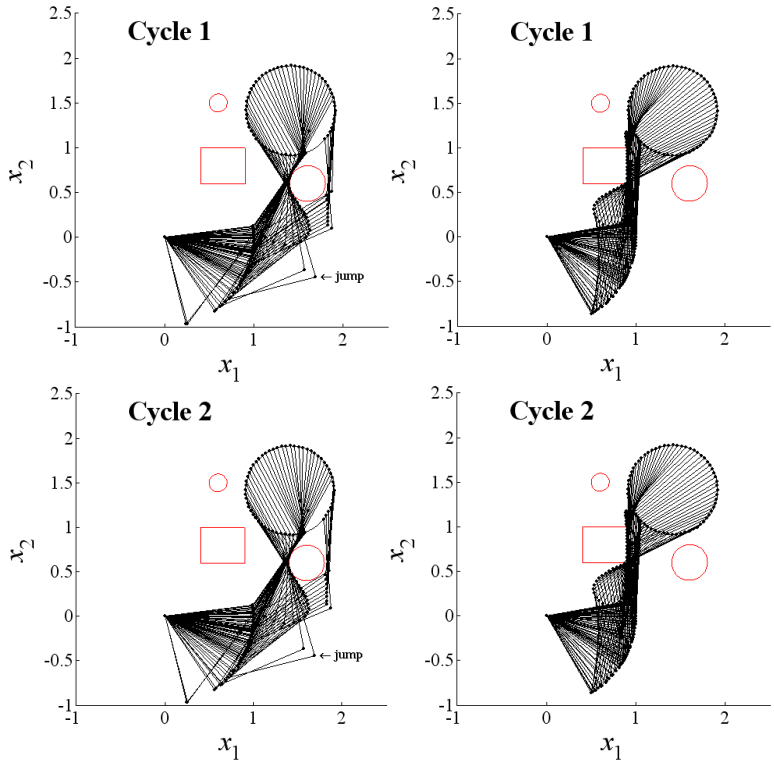


Fig. 21. Successive robot configurations of the 3*R*-robot for the 1st and 2nd cycles, respectively, for a workspace with three obstacles and $r = 2.0$, with two different initial joint configurations.

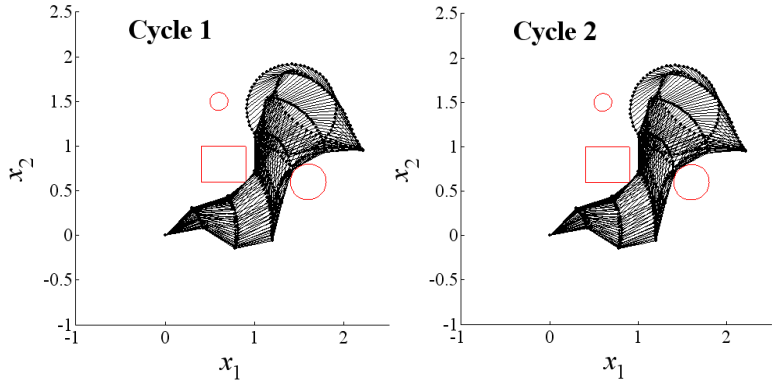


Fig. 22. Successive robot configurations of the 7*R*-robot for the 1st and 2nd cycles, respectively, for a workspace with three obstacles and $r = 2.0$.

## 5. Conclusions

A CLGA algorithm that combines the CLP with a GA scheme was presented. Several experiments were developed to study the performance of the CLGA when the manipulator is required to repeat a

circular motion in the operational space, while satisfying an optimization criterion, in a workspace without and with obstacles.

The results show that the CLGA gives good results in the perspective of the positional error and the repeatability of the joint positions. In what concerns the drift in the joint angles, there is no zero-drift but the results seem to be acceptable for all manipulators under analysis.

It is shown that the presence of obstacles does not present an additional complexity for the CLGA. However, the initial configuration of the manipulator plays an important role in the performance of the manipulator in a workspace with obstacles. In fact, if the initial configuration is not adjusted adequately for the required task, then jumps may occur in the joint positions to prevent the collision with the obstacles.

## References


[1] Dragomir N. Nenchev and Yuichi Tsumaki. Motion analysis of a kinematically redundant seven-DOF manipulator under the singularity-consistent method. Proc. of the 2003 IEEE Int. Conf. on Robotics and Automation 2003; 2760 – 2765.

[2] Keith L. Doty, C. Melchiorri and C. Bonivento. A Theory of Generalized Inverses Applied to Robotics. International Journal of Robotics Research 1993; 12:1-19.

[3] J. Baillieul, J. Hollerbach and R. Brockett. Programming and control of kinematically redundant manipulators. Proc. of the 23rd IEEE Conf. on Decision and Control 1984; 768 – 774.

[4] C. A. Klein, and C.-H. Huang. Review of Pseudoinverse Control for Use With Kinematically Redundant Manipulators. IEEE Trans. Syst. Man, Cyber 1983; Vol SMC-13, nº3, 245-250.

[5] R. G. Roberts and A. Maciejewski. Repeatable Generalized Inverse Control Strategies for Kinematically Redundant Manipulators. IEEE Transactions on Automatic Control 1993; 38:5:689-699.

[6] C. A. Klein and S. Ahmed. Repeatable Pseudoinverse Control for Planar Kinematically Redundant Manipulators. IEEE Transactions on Systems, Man and Cybernetics, 1995; 25:12:1657-1662.

[7] Y. Zhang, H. Zhu, X. Lv and K. Li. Joint Angle Drift Problem of PUMA560 Robot Arm Solved by a Simplified LVI-Based Primal-Dual Neural Network. Proc. of the IEEE Int. Conf. on Industrial Technology 2008; 1–26.

[8] Y. Zhang, X. Lv, Z. Li, Z. Yang and H. Zhu. Effective neural remedy for drift phenomenon of planar three-link robot arm using quadratic performance index. Electronics letters, 2008; 44:6:436-437. DOI 10.1049/el:20080455.

[9] Y. Zhang, Z. Tan, Z. Yang and X. Lv. A Dual Neural Network Applied to Drift-Free Resolution of Five-Link Planar Robot Arm. Proceedings of the 2008 IEEE International Conference on Information and Automation 2008; 1274-1279.

[10] J. Baillieul. Kinematic programming alternatives for redundant manipulators. Proc. of the IEEE Int. Conf. on Robotics and Automation 1985; 722 – 728.

[11] D. R. Baker and C. W. Wampler II. On the Inverse Kinematics of Redundant Manipulators. The Int. Journal of Robotics Research 1988; 7: 2: 11: 3 – 21.

[12] Ki-Cheol Park, Pyung-Hun Chang and Sukhan Lee. A new kind of singularity in redundant manipulation: semi algorithmic singularity. Proc. of the IEEE Int. Conf. on Robotics and Automation 2002; 2:1979-1984.

[13] J. Park, W.-K. Chung, and Y. Youm. Characteristics of optimal solutions in kinematics resolutions of redundancy. IEEE Transactions on Robotics and Automation 1996; 12:3:471-478.

[14] D. E. Goldenberg. Genetic algorithms in search optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.

[15] J. K. Parker, A. R. Khoogar and D. E. Goldberg. Inverse kinematics of redundant robots using genetic algorithms. Proc. of the 1989 IEEE Int. Conf. on Robotics and Automation 1989; 271 – 276.

[16] T. Arakawa, N. Kubota and T. Fukuda. Virus-evolutionary genetic algorithm with subpopulations: application to trajectory generation of redundant manipulator through energy optimization. Proc. of the 1996 IEEE Int. Conf. on Systems, Man, and Cybernetics 1996; 14 – 17.

[17] N. Kubota, T. Arakawa, T. Fukuda and K. Shimojima. Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm. Proc. of the IEEE Int. Conf. on Robotics and Automation 1997; 205 – 210.

[18] V. de la Cueva and F. Ramos. Cooperative genetic algorithms: a new approach to solve the path planning problem for cooperative robotic manipulators sharing the same work space. Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems 1998; 267 – 272.
[19] T. Nishimura, K. Sugawara, I. Yoshihara and K. Abe. A motion planning method for a hyper multi-joint manipulator using genetic algorithm. Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics 1999; 645 – 650.
[20] B. McAvoy and B. Sangolola. Optimal trajectory generation for redundant planar manipulators. Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics 2000; 3241 – 3246.
[21] Y. Peng and W. Wei. A New Trajectory Planning Method of Redundant Manipulator Based on Adaptive Simulated Annealing Genetic Algorithm (ASAGA). Proc. of the IEEE Int. Conf. on Computational Intelligence and Security 2006; 262 – 265.
[22] Y. Zhang, Z. Sun and T. Yang. Optimal Motion Generation of a Flexible Macro-micro Manipulator System Using Genetic Algorithm and Neural Network. Proc. of the 2006 IEEE Conf. on Robotics, Automation and Mechatronics 2006; 1 – 6.
[23] E. J. S. Pires, J. A. T. Machado and P. B. M. OliveiraY. Manipulator Trajectory Planning using a MOEA. Applied Soft Computing Journal 2007; 7:3:659-667.
[24] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on Man-Machine Systems 1969; MMS-10: 2: 47 – 53.
[25] Bruno Siciliano. Kinematic Control of Redundant Robot Manipulators: A Tutorial. Journal of Intelligent and Robotic Systems 1990; 3:201-212.
[26] M. G. Marcos, F. B. Duarte and J. A. T. Machado. Complex Dynamics in the Trajectory Control of Redundant Manipulators. Transactions of Nonlinear Science and Complexity (World Scientific) 2007; 1:134 – 143.
[27] J. H. Holland. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor. 2nd ed. Mit Press, 1992.