

# Real-Time Evolutionary Algorithms for Constrained Predictive Control

Mario Luca Fravolini, Antonio Ficola and Michele La Cava  
*Dipartimento di Ingegneria Elettronica e dell'Informazione, Università di Perugia  
Italy*

## 1. Introduction

In the last years, thanks to the great advancements in computing technology, Evolutionary Algorithms (EA) have been proposed with remarkable results as robust optimisation tools for the solution of complex real-time optimisation problems. In this chapter we review the most important results of our research studies concerning the design of EA-based schemes suitable for real-time optimisation problems for Nonlinear Model Based Predictive Control (MBPC). In the first part of the chapter it will be discussed some modifications of a standard EA in order to address some real-time implementation issues.

The proposed extension concerns the adoption of a new realtime adaptive mutation range to generate smooth commands, and the adoption of an intermittent feedback to face the computational delay problem. It will be shown that the main advantage of the improved technique is that it allows an effective real-time implementation of the Evolutionary-MBPC with a limited computing power. The real-time feasibility of the proposed improved Evolutionary-MBPC will be demonstrated by showing the experimental results of the proposed method applied to the control of a laboratory flexible mechanical system characterized by fast dynamics and a very small structural damping.

Later, the application of real-time EAs is extended to real-time motion planning problems for robotic systems with constraints either on input and state variables. Basing on a finite horizon prediction of the future evolution of the robot dynamics, the proposed EA-based device online preshapes the reference trajectory, minimizing a multi-objective cost function. The shaped reference is updated at discrete time intervals taking into account the full nonlinear robot dynamics, input and state constraints. A specialized Evolutionary Algorithm is employed as search tool for the online computation of a sub-optimal reference trajectory in the discretized space of the control alternatives. The effectiveness of the proposed method and the online computational burden are analyzed numerically in two significant robotic control problems.

## 2. Improved evolutionary predictive controllers for real-time application

*Part of the following article has been previously published in: M.L. Fravolini, A. Ficola, M. La Cava, "Improved Evolutionary Predictive Controllers for real time application", Control and Intelligent Systems, vol. 31, no.1, pp.16-29,2003, Acta Press, Calgary Canada, ISSN: 1480-1752(201) .*

The employment of a model of a dynamical system to plan in real-time optimal control policies is a very attractive approach for the generation of sophisticated and effective control laws. The Model Based Predictive Control (MBPC) is based on this idea. In the last years, much progress has been made toward the development of the stability theory of nonlinear MBPC (Maine & Michalaska, 1993; Scokaert et al., 1999; Gyurkivics, 1998), but the resulting algorithms are quite difficult to be implemented in real-time. The difficulty in the implementation of nonlinear MBPC lies in the fact that a constrained non-convex nonlinear optimization problem has to be solved on-line. Indeed, in the case of nonlinear dynamics, the optimization task is highly computationally demanding and, for this reason, the online optimization problem is an important issue in the implementation of MBPC (Camacho & Bordons, 1995). The undesirable effects caused by the numerical optimization are mainly two. The first is due to the time required by the optimization procedure to compute a solution; this time is often much longer than the sampling time, thus introducing a significant delay in the control loop. The second effect is related to the fact that, in general, a numerical algorithm for nonlinear constrained MBPC cannot guarantee the optimality of the solution.

The complexity of the online optimization problem is strictly related to the structure of the nonlinear dynamics and also to the number of the decision variables and to the constraints involved. Many approaches have been proposed to face the online optimization task in nonlinear MBPC; a widely applied approach consists of the successive linearization of the dynamics at each time step and of the use of standard linear predictive control tools to derive the control policies, as proposed in (Mutha et al., 1997; Ronco et al. 1999). Although these methods greatly reduce the computational burden, the effect of the linearization could generate poor results when applied to the real nonlinear system. Other methods utilize iterative optimization procedures, as Gradient based algorithms or Sequential Quadratic Programming techniques (Song & Koivo 1999). These methods, although can fully take into account the nonlinear dynamics of the system, suffer from the well known problem related to local minima. A partial remedy is to use these methods in conjunction with a grid search as proposed in (Fischer et al., 1998). Another approach is represented by discrete search techniques as Dynamic Programming (Luus, 1990) or Branch and Bound (Roubos et al. 1999) methods. In this approach the space of the control inputs is discretized and a smart search algorithm is used to find a sub optimum in this space. The main advantages of search algorithms are that they can be applied regardless of the structure of the model; furthermore, the objective function can be freely specified and it is not restricted to be quadratic. A limitation of these methods is the fast increase of the algorithm complexity with the number of decision variables and grid points, known as "the curse of the dimensionality".

In the last years, another class of search techniques, called Evolutionary Algorithms (EAs), showed to be quite effective in the solution of difficult optimal control problems (Foegel, 1994; Goldberg 1989). Although at the beginning the largest part of the EA applications were developed for offline optimization problems, recently, thanks to the great advancements in computing technology, many authors have applied with remarkable results EAs for online performance optimization in the area of nonlinear and optimal control. Porter and Passino showed clearly that Genetic Algorithms (GA) can be applied effectively either in adaptive and supervisory control (Porter & Passino, 1998; Lennon & Passino 1999) or as nonlinear observers (Porter & Passino 1995). Liaw (Liaw & Huang 1998) proposed a GA for online

learning for friction compensation in robotic systems. In references (Fravolini et Al. 1999 (a); Fravolini et Al. (b)) the authors proposed to apply EAs for the online trajectory shaping for the reduction of the vibrations in fast flexible mechanical systems.

Recently, some authors have pointed out the possibility of applying EAs as performance optimizer in MBPC. Onnen in (Onnen et Al., 1999) proposed a specialized GA for the optimization of the control sequence in the field of process control. He showed the superiority of the evolutionary search method in comparison to a branch-and-bound discrete search algorithm. Similar algorithms have been also proposed in (Martinez et Al. 1998; Shin & Park, 1998).

Although very interesting from a theoretical point of view, all these studies were carried out only by means of simulations and scarce attention was paid to computational and real-time implementation issues; furthermore, to the best of our knowledge, no practical application of this technique has been reported in literature. In the first part of this paper, therefore, we discuss in details some practical real-time implementation issues of the evolutionary MBPC that have not been discussed in (Onnen et Al., 1999; Martinez et Al. 1998; Shin & Park, 1998). In fact, we experimented that the application the algorithm [19] could generate practical problems. As noticed also in (Roubos et Al. 1999), the suboptimal command signal often exhibits excessive chattering that cause unnecessary solicitations of the actuation system. To cope with this problem we propose to improve the basic algorithm (Linkens, 1995) by applying an online adaptation of the search space by defining a new adaptive mutation operator. We will show that smoother solutions can be obtained without significant degradation of the performance. A second aspect that should be faced is the computational delay that originates when computational intensive algorithms, as EAs, are applied for real-time optimization. This aspect is particularly important in the case of a limited computational power and fast sampling rates. To overcome this problem we modified the basic algorithm by inserting an intermittent feedback strategy. The advantage of this strategy is the possibility of decoupling the computing time by the system sampling time; this makes the EA based optimization procedure more flexible than the basic one for real-time implementation.

Another aspect that has not been discussed in previous works is the analysis of the repeatability of the control action of the EA based MBPC; this aspect is strictly related to the reliability of the proposed strategy. This issue has been addressed by means of a stochastic analysis. The first part of the paper terminates with a comparison of the performance provided by the improved EAs and an iterative gradient-based optimization procedure. In the second part of the paper we report the experimental results obtained by implementing the improved algorithm for the real-time MBPC of a nonlinear flexible fast-dynamic mechanical system.

### 3. Nonlinear Predictive Control

Model Based Predictive Control includes a wide class of control strategies in which the control law is derived on the basis of the estimation of the future evolution of the system predicted by a model. A nonlinear MBPC is based on the following three main concepts (Garcia et Al. 1989; Levine, 1996):

- A) the use of a model to predict the future evolution of the real system;
- B) the online computation of the optimal control sequence, by minimizing a defined index of performance that quantifies the desired behavior;

C) the receding horizon: only the first value of the optimal sequence is applied; then the horizon is shifted one sample in the future and a new sequence is recalculated.

One of the main advantages of the MBPC is the possibility to handle general constraints either on the inputs or on state variables. A block diagram of a MBPC is shown in Fig. 1.

- *The Model and the System:* In MBPC the model of the system is expressly used to compute the control law; for this reason the model should be as accurate as possible. There is no restriction on the structure of the model. If complete physical insight on the system is available, the model could be analytically expressed in state space form; on the other hand, if only a scarce knowledge is available, a black box model can be employed. In the last case, the system is often implemented by a neural network or a fuzzy system.

*Remark 1:* In most of the applications, the system block in Fig. 1 represents the open loop dynamics of the nonlinear systems; however in some applications it can comprise either a preexistent inner control loop or an expressly designed feedback controller. In the first case as in (Shiller & Chang, 1995) the MBPC is employed to improve the performance of the existing feedback controller. In the second case as in (Fravolini et Al., 2000) an inner feedback controller is designed in order to directly compensate the effects that are difficult to be included in system prediction model, as friction and stiction.

- *The Optimal Control Law:* In MBPC the optimal command sequence  $u^*(k)$  is determined as the result of an optimization problem at each sampling instant. All the performance specifications are quantified by means of a cost index. An index, which is often employed, is the following one:

$$J = \sum_{j=N_1}^{N_2} p_k \|\hat{y}(k+j|k) - y_d(k+j)\|^2 + \sum_{j=1}^{N_u} q_k \|\Delta u(k+j-1)\|^2 + J_1 \quad (1)$$

The first term of J evaluates the square error between the desired future output signal  $y_d(k+j)$  and the predicted future output  $\hat{y}(k+j|k)$ , estimated on the basis of the prediction model and the feedback information available at instant  $k$ . This error is evaluated over a defined *prediction horizon*, which is a window of  $N_2 - N_1$  samples. The second term of J is the control contribution over the *control horizon* window of  $N_u$  samples, where ( $\Delta u(k) = u(k) - u(k-1)$ ). Coefficients  $p_k$  and  $q_k$  weights the different terms of J. The term  $J_1$  is a further cost function that can be used to take into account other specifications or constraints. MBPC is mainly used in conjunction with constraints on the amplitude and the rate of variation of the input and output variables, namely:

$$U^- < u(k) < U^+ \quad \Delta U^- < \Delta u(k) < \Delta U^+ \quad (2)$$

$$Y^- < y(k) < Y^+ \quad \Delta Y^- < \Delta y(k) < \Delta Y^+ \quad (3)$$

The fulfillment of the constraints (2) are required to take into account the practical limitations of the actuation system, while constraints (3) can prevent the system to work in undesirable regions. The minimization of index J is performed with respect to the

decision variables, which is the sequence of the control increments:  $[\Delta u(k), \Delta u(k+1), \dots, \Delta u(k+N_u-1)]$ .

- *The receding horizon and the feedback filter:* In the receding horizon strategy only the first sample of the optimal sequence  $u^*(k+j)$  is applied to the system; subsequently, the horizon is shifted one step in the future and a new optimization is repeated on the basis of the measured feedback information. In the basic form, the horizons are shifted at each sampling instant. Although this strategy introduces feedback information in the control loop at the sampling rate, this can require a large computational power in the online implementation; on the other hand (see section 6), it is possible to decrease the computational burden by applying an intermittent feedback (Ronco et Al., 1999). The structure and complexity of the feedback filter depends on the strategy employed to insert the feedback information in the optimization process. In the simplest case the filter is low-pass and is used only to reduce the measurement noise. This approach is often used in conjunction to input-output models where the measured output is mainly used to recover steady state errors. In case of state space models, the feedback filter could be more complex: it can be either a bank of filters used to estimate derivatives or integrals, or a model based nonlinear observer. In the last case the possibility to recover the system states allows periodic system/model realignment. The realignment prevents excessive drift of the prediction error when a long-range prediction is required.

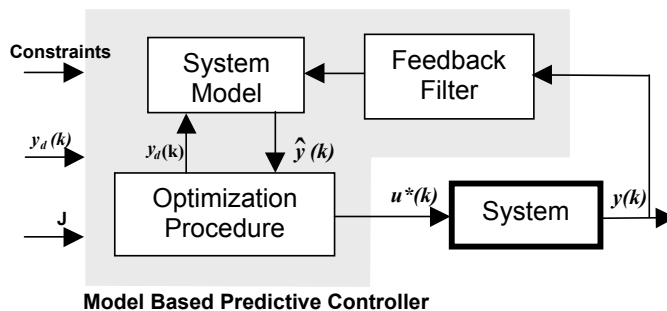


Figure 1. Nonlinear Model Predictive control

#### 4. The benchmark laboratory nonlinear system

The performance of the improved Evolutionary MBPC has been experimentally tested on a nonlinear laboratory flexible mechanical system. The system is composed of a flexible beam clamped at one end; a controlled pendulum is hinged on the free tip and is used to damp out the beam oscillations; the motion occurs in the horizontal plane. A DC motor is employed to regulate the angular position of the pendulum in order to damp out the vibrations on the flexible beam. Fig. 2 shows the experimental set-up. The system is characterized by a very small structural damping and its stabilization in a short time represents a significant test because of the fast dynamics and the presence of oscillatory modes that require either an accurate model or a short sampling time or a long prediction horizon. This situation is known to be critical in the implementation of MBPC algorithms.

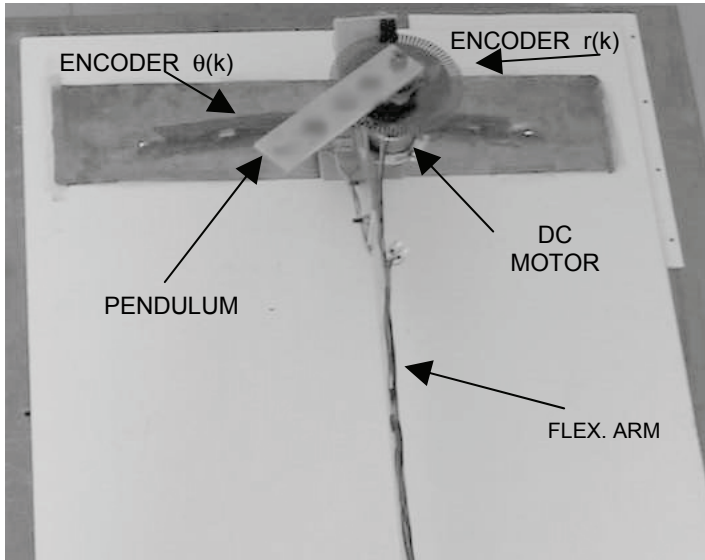


Figure 2. The laboratory benchmark flexible systems

Fig. 3a shows the mechanical model of the system, the main parameters of which are reported in table 1. The system is equipped with optical encoders that allow the measurement of the tip deflection  $r$  and the angle  $\phi$  of the pendulum (since the beam deflection is small we exploited the relation  $r \approx L \cdot \theta$ ). Because of the most of the vibration energy is associated to the first vibration mode of the beam, its dynamics can be described with a sufficient accuracy by a mass-spring-damper mechanical system (Fig. 3b). The equations of motion of a flexible mechanical system moving in a horizontal plane are of the form:

$$B(q)\ddot{q} + C(q, \dot{q}) = Q \tag{4}$$

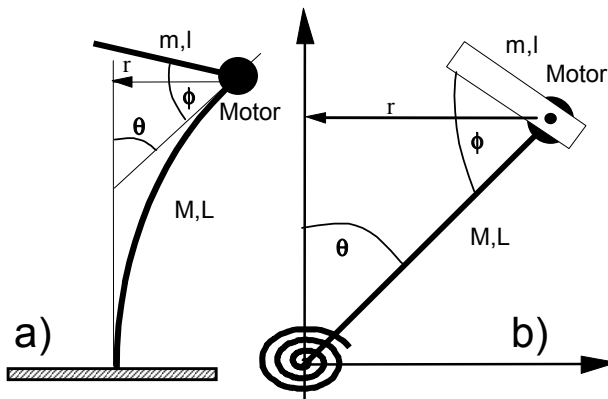


Figure 3. The structure (a), and its model (b)

$M$	0.689 kg	Mass of the beam
$m$	0.070 kg	Mass of the pendulum
$L$	1.000 m	Length of the beam
$l$	0.086 m	Length of the pendulum
$K_1$	25.3 Nm/rad	Elastic coeff. of the beam
$C_1$	0.008 Nm/s rad	Damping coeff. of the beam
$C_2$	0.050 Nm/s rad	Damping coeff. of the pendulum

Table 1. Parameters of the Model

where  $\mathbf{B}$  is the inertia matrix,  $\mathbf{C}$  comprises Coriolis, centrifugal, stiffness and damping terms,  $\mathbf{q}=[\theta \ \phi]^T$  is the vector of Lagrangian coordinates and  $\mathbf{Q}$  is the generalized vector of the input forces. By applying the standard Lagrangian approach the following model matrices were obtained:

$$\mathbf{B} = \begin{bmatrix} (M+m)L^2 + ml^2 - 2mLl \cos \phi & ml^2 - mL \cos \phi \\ ml^2 - mL \cos \phi & ml^2 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 2mL\dot{\phi}\dot{\theta} \sin \phi + mL\dot{\phi}^2 \sin \phi + K_1\theta + C_1\dot{\theta} \\ -mL\dot{\theta}^2 \sin \phi + C_2\dot{\phi} + C_{cou} \operatorname{sgn}(\dot{\phi}) \end{bmatrix} \quad (5)$$

$$\mathbf{Q} = [0 \ \tau]^T$$

where  $\tau$  is the control torque and the term  $C_{cou}$  represents the Coulomb friction coefficient acting on the motor; this term cannot be modeled accurately. The inaccuracy produced by this uncertainty can generate a significant discrepancy between the simulated and the real pendulum position. A method of compensating for modeling errors that cause inaccuracies in the estimation of the pendulum angle is to redefine the model such that the input is not the motor *torque*  $\tau$ , but rather the motor *position*  $\phi(t)$  and its derivatives (Geniele et Al., 1997)]. In this case, using an inner position feedback controller (for instance, a PD controller), it is possible to track accurately a reference trajectory  $\phi_d(t)$ . If the desired trajectory is smooth, the tracking error can be very small. In this case we can assume  $\phi(t) \cong \phi_d(t)$  and the variables  $\phi_d(t), \dot{\phi}_d(t), \ddot{\phi}_d(t)$  are the new inputs to the model (5). By applying this strategy it is possible to recast the model equations (4-5) obtaining a reduced order model relating the motor position  $\phi_d(t)$  to the beam deflection  $\theta(t)$ . The reduced dynamics satisfy the equation:

$$b_{11}(\phi_d)\ddot{\theta} + b_{12}(\phi_d)\ddot{\phi}_d + c_1(\theta, \phi_d, \dot{\theta}, \dot{\phi}_d) = 0 \quad (6)$$

where  $b_{11}, b_{12}, c_1$  are the entries of matrix  $\mathbf{B}$  and vector  $\mathbf{C}$ . Equation (6) describes the nominal model used for the prediction by the MBPC.

Since a MBPC is discrete time device, it updates the outputs at a defined sampling rate  $T_s$ , the control inputs  $\phi_{des}(t)$  and its derivatives are kept constant during a sampling interval  $[kT_s, (k+1)T_s]$  by a zero-order hold filter. The decision variables for the optimization problem (1) are the control increments:

$$\begin{aligned} & [\Delta u(k+1), \Delta u(k+2), \dots, \Delta u(k+N_2)] = \\ & [\Delta \ddot{\phi}_{des}(k+1), \Delta \ddot{\phi}_{des}(k+2), \dots, \Delta \ddot{\phi}_{des}(k+N_2)] \end{aligned} \quad (7)$$

The desired position  $\phi_d(t)$  and velocity  $\dot{\phi}_d(t)$  are computed by numerical integration using the 4<sup>th</sup> order Runge Kutta method. The EA based MBPC procedures was written in C code and run in real-time on a dSPACE 1003 DSP board, based on a TMS320C40 DSP.

## 5. Evolutionary Algorithms

Evolutionary Algorithms are multi point search procedures that reflect the principle of evolution, natural selection, and genetics of biological systems (Goldberg, 1989; Porter & Passino 1998). An EA explores the search space by employing stochastic rules; these are designed to quickly direct the search toward the most promising regions. The EAs work with a population of solutions rather than a single point; each point is called chromosome. A chromosome represents a potential solution to the problem and comprises a string of numerical and logical variables, representing the decision variables of the optimization problem. An EA maintains a population of chromosomes and uses the genetic operators of "selection" (it represents the biological survival of the fittest ones and is quantified by a fitness measure that represents the objective function), "crossover" (which represents mating), and "mutation" (which represents the random introduction of new genetic material), to generate successive generations of populations. After some generations, due to the evolutionary driving force, the EA produces a population of high quality solutions to the optimization problem.

The implementation of an EA can be summarized by the following sequence of standard operations:

1. (Randomly) Initialize a population of  $N$  solutions
2. Calculate the fitness function for each solution
3. Select the best solutions for reproduction
4. Apply crossover and mutation to selected solutions
5. Create the new population
6. Loop to step 2) until a defined stop criterion is met

The implementation of the evolutionary operators of selection, crossover and mutation is not unique. The first and most known EA is the simple GA (Goldberg 1989), but a large number of modifications have been proposed and applied by many authors. In this paper, the attention will be focused mainly on the evolutionary operators expressly designed for MBPC. These operators were introduced in previous works (Fravolini et Al. 1999 (a); Fravolini et Al. (b)) and tested by mean of extensive simulation experiments. Some of these operators are similar to the corresponding ones reported by (Onnen, 1997; Martinez, 1998) and constitute a solid, tested and accepted base for Evolutionary MBPC.

### 5.1 The Standard Evolutionary MBPC operators

- *Fitness function:* The natural choice for the cost function to be minimized online is represented by the index  $J$  in (1). Since in the EA literature it is more common to refer to fitness function optimization than to cost function minimization, the following fitness function is defined:

$$f = 1/J \quad (8)$$

- *Decision variables*: The decision variables are the sequence of the future  $N_u$  input increments  $\Delta u(k+j)$  in (7).
- *Chromosome structure and coding*: A chromosome is generated by the juxtaposition of the coded sequence of input increments (7). Concerning the codification of the decision variables, some alternatives are possible. Binary or decimal codifications, although used in many simulative studies as in (Onnen, 1997; Martinez, 1998), are not particularly suited in the real-time application due to the time consuming coding and decoding routines. Real and integer coded variables do not require any codification; but the evaluation of evolutionary operators for real numbers is slower than the corresponding routines working with integers; therefore, in this work a decimal codification was employed. A coded decision variable  $x$  can assume an integer value in the range  $\Omega: 0, \dots, L_0, \dots, L_x$  where  $L_x$  represents the discretization accuracy; this set is uniformly mapped in the bounded interval  $\Delta U^- \leq \Delta u \leq \Delta U^+$ .  $L_0$  is the integer corresponding to  $\Delta u=0$ ; The  $i$ -th chromosome in the population at  $t$ -th generation is a vector of integer numbers  $X_i^t = [x_{i,1}^t, x_{i,2}^t, \dots, x_{i,N_u}^t]$ , the actual values of the corresponding decision variables  $\Delta u(k)$  can be obtained by the following decoding scheme:

$$\Delta u(k+j) = \Delta U^- + \Delta_U \cdot x_{i,j} \quad j = 1, \dots, N_u \quad (9)$$

where  $\Delta_U = |\Delta U^+ - \Delta U^-|/L_x$  is the control increment resolution. The sequence of the applied controls is:

$$u(k+j) = u(k-1+j) + \Delta u(k+j) \quad j = 1, \dots, N_u \quad (10)$$

- *Selection and Reproduction* :  
Selection and reproduction mechanisms act at two different levels during the on-line optimization:
  - *Selection and Reproduction within time interval  $k$* . Since a limited computational time is available within the time interval  $k$ , it is essential to not loose the best solutions during the evolution and to use them as "hot starters" for the next generation. For this reason a *steady state* reproduction mechanism is used, namely the best  $S$  chromosomes in the current generation pass unchanged in the next one. The remaining part of the population is originated on the basis of a *rank selection* mechanism: given a population of size  $N$ , the parent solutions are selected within the mating pool of the best ranked  $D$  individuals. This approach is similar to the algorithm described in (Yao & Sethares, 1994).
  - *Heredity (between time interval  $k$  and  $k+1$ )*. At the beginning of the  $k+1$ -th time interval, since the horizon is shifted one step in the future, the genes of the best  $S'$  chromosomes are shifted back of one location within the chromosome; in this way the first values are lost and replaced by the second ones and so on. The values on the last positions are simply filled by keeping the current value. The shifted  $S'$  chromosomes represent the "hot starters" for the optimization in the  $k+1$ -th time interval; the remaining  $N-S'$  chromosomes are randomly generated.

- *Crossover*: Uniform crossover has been implemented. Given two selected chromosomes  $X_i^t$  and  $X_j^t$ , two corresponding variables are exchanged with a probability  $p_c$ , namely:

$$x_{i,k}^{t+1} = x_{j,k}^t \text{ and } x_{j,k}^{t+1} = x_{i,k}^t. \quad (11)$$

- *Mutation*: Mutation is applied with probability  $p_m$  to each variable of a selected chromosome  $X_i$ , according to:

$$x_{i,k}^{t+1} = x_{i,k}^t + \text{rand}(\bar{\Delta}) \quad (12)$$

where  $\text{rand}(\bar{\Delta})$  is a random integer in the range  $[-\Delta_x, +\Delta_x]$  and  $\Delta_x$  is the maximum mutation amplitude.

- *Constraints*: During the optimization it is possible that a decision variable  $x_{i,j}^t$  (due to mutation) violates the maximum or minimum allowed value in  $\Omega$ ; this can be easily avoided by applying the following threshold operators:

$$\begin{cases} \text{if } x_{i,j}^t > L_x \Rightarrow x_{i,j}^t = L_x \\ \text{if } x_{i,j}^t < 0 \Rightarrow x_{i,j}^t = 0 \end{cases} \quad (13)$$

The application of (13) automatically guarantees that  $\Delta U^- \leq \Delta u \leq \Delta U^+$ . Similarly, the fulfillment of constraint  $U^- \leq u \leq U^+$  is guaranteed by applying the thresholds:

$$\begin{cases} \text{if } u(k) + \Delta u(k) > U^+ \Rightarrow x_{i,k}^t = L_0 + \text{int}\left(\frac{U^+ - u(k)}{\Delta_U}\right) \\ \text{if } u(k) + \Delta u(k) < U^- \Rightarrow x_{i,k}^t = L_0 + \text{int}\left(\frac{U^- - u(k)}{\Delta_U}\right) \end{cases} \quad (14)$$

It is also possible to take into account other constraints as (3) by employing the penalty function strategy.

- *Termination criterion*: In a real-time application the termination criterion must take into account the hard bound given by the real time constraint; this implies that the maximum number of generations is limited by the fulfillment of this constraint. When the real-time deadline is met, the optimization is stopped, and the first decoded value of the best chromosome  $\Delta u^*(k)$  is applied to the system.

## 6. Improved Operators for Real-Time Implementation

The following sections describe the improvements of the standard algorithm reported in section 4 that we propose to apply in the real-time implementation. Some of these operators are not new in the field of MBPC; however, the novelty consists of the adaptation of these concepts within an Evolutionary real-time optimization procedure.

### 6.1 Adaptation of the search space

The employment of a discrete search technique to minimize the index (1) introduces a tradeoff between the number of discrete control alternatives and the computational load required for the exploration of the discrete space. The performance of an EA, although it does not perform an exhaustive exploration of the search space, is anyway influenced by the dimensionality of the grid; this aspect is particularly relevant in real-time applications where the computing time is limited. A coarse discretization could generate unsatisfactory results as undesired oscillations of the output variable, while a too fine grid could require a prohibitive time to locate a satisfactory solution. The problems related to the excessive chattering in the control signal or in controller gains have been previously mentioned by other authors as in (Roubos et Al., 1999; Lennon & Passino, 1999). By applying a scaling factor to the size of the region in which the control alternatives are searched can significantly decrease this problem. As proposed in (Sousa & Setnes, 1999), a way to achieve this goal is to adapt the dimension of the search space depending on the predicted deviation of the output from the desired reference signal. When the prediction error keeps small within the prediction horizon, it is probably not convenient to explore the whole allowed region of the control alternatives; rather, it is opportune to concentrate the search around the origin. On the other hand, when a large error is predicted, the range of the control alternatives should be large to enable a fast recover.

In the context of evolutionary MBPC, the exploration of new regions is mainly carried out by the application of the mutation operator (12) that causes a random incremental variation in the range  $\pm\Delta_x$  around the current value. For this reason, we propose to improve the basic algorithm by conveniently scaling during the real-time operation the mutation range  $\bar{\Delta}$  by means of an adaptive gain  $\alpha(k) \in [0,1]$ . The adaptive mutation range is therefore defined as:

$$\bar{\Delta}(k) = \alpha(k) \cdot \Delta_x \quad (15)$$

Different criteria can be used to adapt the parameter  $\alpha(k)$ . In (Sousa, 1999) the adaptation of the parameter  $\alpha(k)$  is carried out by means of a fuzzy filter on the basis of the current estimation error and the current change of error. In this work we propose to adapt the parameter in function of the predicted mean absolute output error  $\bar{E}^*$  for the best solution:

$$\bar{E}^*(k) = \frac{1}{(N_2 - N_1)} \sum_{j=N_1}^{N_2} |y_d(k+j) - \hat{y}(k+j)| \quad (16)$$

$\bar{E}^*$  is a meaningful index because it takes into account not only the current value of the error but also the predicted behavior of the estimation error. When  $\bar{E}^*(k)$  is small,  $\alpha(k)$  should be small to allow a fine regulation near the steady state; otherwise, when  $\bar{E}^*(k)$  is large,  $\alpha(k)$  should be large to allow big corrections. The following law defines a possible scaling:

$$\alpha(k) = 1 - \exp(-\bar{E}^{*2} / \sigma) \quad (17)$$

where  $\sigma > 0$  is a parameter that regulates the adaptation speed. The proposed law has only one design parameter ( $\sigma$ ) that is not really critical and, thus, allows for the use of some heuristics to cope with the uncertainty in its definition. By employing the adaptive law (15),

the choice of the maximum mutation amplitude  $\Delta_x$  is no more critical, because the adaptive gain  $\alpha(k)$  scales its range accordingly.

### 6.1.1 A simulation example

To show the effect of the adaptation of the search space we report the results of a simulation study performed on the model of the mechanical system (6). The decision variables is the sequence (7). To find suitable values for the setting of the MBPC parameters some trials were performed. The length of prediction horizon was chosen in order to cover a complete period of oscillation of the first mode of vibration of the beam; shorter horizons gave unsatisfactory responses, that is a very long stabilization transient. In the simulation we chose equal values for the prediction and control horizons; the final values were:  $N_1=0, N_2=N_u=32$ ; the sampling time was  $T_s=0.04s$  (this implies a horizons length of  $N_u \cdot T_s = 1.28s$ ). The other parameters of the MBPC are reported in table 2.

Since the scope of the control system is to damp out the oscillations of the tip position  $r(k)$  of the beam ( $r(k) \cong L \cdot \theta(k)$ ) (see Fig. 3), we assumed as the desired target position the trajectory  $y_d(k) = \theta_d(k) = 0$ ; the resulting objective function is:

$$J = \sum_{j=0}^{N_u} |\theta(k+j) - y_d(k+j)| = \sum_{j=0}^{N_u} |\theta(k+j) - 0| \quad (18)$$

$N = 20$	$D = 6$	$S = 2$	$S = 4$
$p_m = 0.1$	$p_c = 0.8$	$K = 5$	$T_s = 0.04$
$L_x = 4000$	$\Delta_x = 1000$	$\sigma = 0.001$	$N_u = 32$

Table 2. Parameters of the MBPC

It is worth noting that, although cost function (18) is different from index (1), it does not influence the functionality of the EA. In index (18), no cost is added to weight the contribution of the control effort, because it was observed in the real implementation that the most important constraint on the control signal is imposed by its rate of variation, due to the limited bandwidth of the actuator. The constraint  $-8 < \Delta u(k) < 8$  allows the generation of reference signal that can be accurately tracked by actuator. Fig. 4a shows the response of the controlled system in the case of no adaptation of the mutation range; in the same figure it is also reported the uncontrolled response (dotted line). Fig. 4b shows the same response in the case that the adaptive law of the mutation range is active. While in the two cases a not significant difference in the performance variable is observed, on the other hand, a big discrepancy is present in the sequence of the control input increments (Fig. 5a-b). In the case of fixed mutation range the control increments exhibit an excessive chattering also when the system is near the steady state. The presence of persistent spikes in the control signal is due to the difficulty of the basic algorithm to select a smooth sequence of small increments in a large space of control alternatives in a limited number of generations. In the case of adaptation of the mutation range, the control increments are more reasonable. Fig. 6 a-b shows the control signals applied to the system in the two cases. Although an unavoidable chattering is present during the transitory, it completely disappears when the system is near the steady state. This example shows that the adaptive reduction of the dimensionality of

the search alternatives helps the algorithm to select a sufficiently smooth signal able to drive the mechanical system to the steady state; furthermore, the adaptive mutation range does not degrade the performance of the controlled system. For this reason the adaptive mutation ranges represent a valid improvement of the basic algorithm for real-time Evolutionary MBPC of the system under inspection. Fig. 7 shows the adaptation of the gain  $\alpha(k)$  during the transitory.

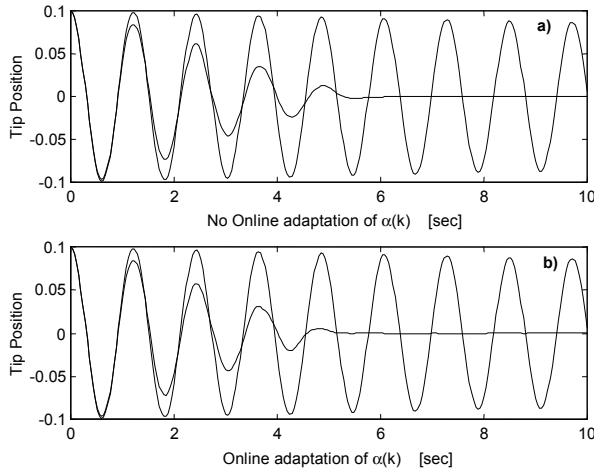


Figure 4. Tip position (controlled and free) without the adaptation of the search space (a), and with the adaptation of the search space (b)

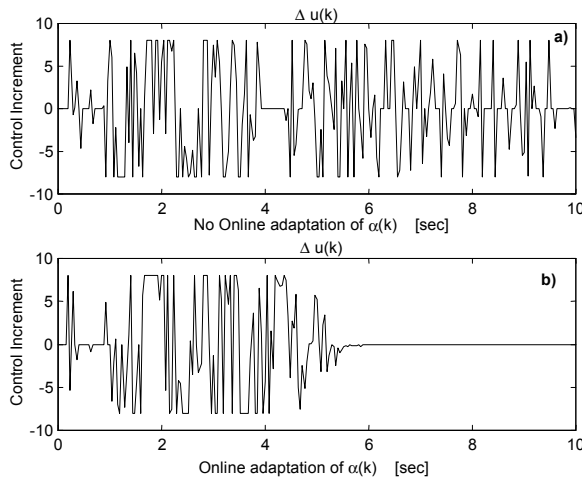


Figure 5. Control input increments without the adaptation of the search space (a), and control input increments with the adaptation of the search space (b)

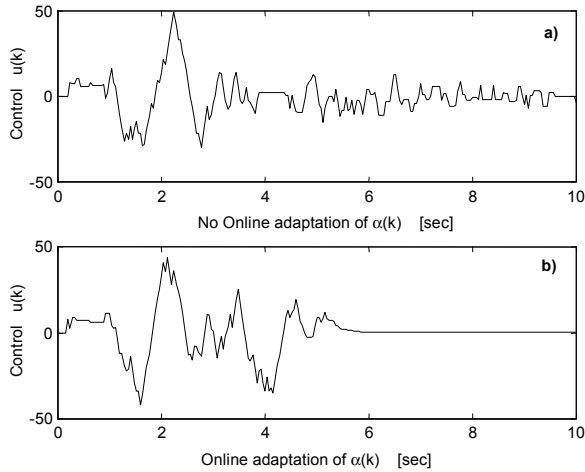


Figure 6. Control signal without the adaptation of the search space (a), control signal with the adaptation of the search space (b)

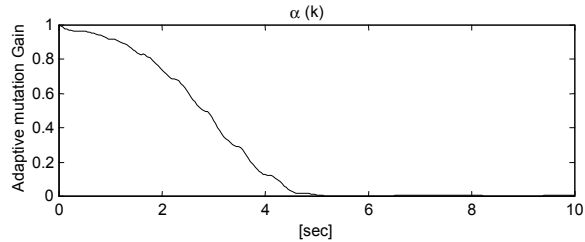


Figure 7. Online adaptation of gain  $\alpha(k)$

## 6.2 The computational delay problem

The implementation of a MBPC procedure implies the online optimization of the cost index  $J$  at every sampling period  $T_s$ . In most of theoretical and simulation studies concerning the MBPC, the problems related to the computational delay, that is the CPU time  $T_c$  required for the numerical optimization of index (1), are seldom taken into account. In the ideal situation ( $T_c=0$ ), the optimal control signal applied in the  $m$ -th sampling interval depends directly on the current state,  $x(kT_s)$ , at the same instant. Under this hypothesis the optimal control law is defined by the following function  $f_s(\cdot)$ :

$$u_k^*(t) = f_s(x(kT_s), u(kT_s), t) \quad t \in [mT_s, (m+1)T_s] \quad (19)$$

Although this hypothesis can be reasonable in some particular cases, the computational delay is a major issue when nonlinear systems are considered, because the solution of a nonlinear dynamic optimization problem with constraints is often computationally intensive. In fact, in many cases the computation time  $T_c$  required by the optimization

procedure could be much longer than the sampling interval  $T_s$ , making this control strategy not implementable in real-time. Without loss of generality we assume that the computing time is a multiple of the sampling time:

$$T_c = H \cdot T_s \quad (20)$$

where  $H$  is an integer. The repetition of the optimization process in each sampling instant is related to the desire of inserting robustness in the MBPC by updating the feedback information at the beginning of each sampling interval ( $T_s$ ) before the new optimization is started. Generally, the mismatches between system and model cause the prediction error to increase with the prediction length and for this reason the feedback information should be exploited to realign the model toward the system to keep the prediction error bounded. The simplest strategy to take into account the system/model mismatches is to use a parallel model and to add the current output model error  $e(k) = y(k) - \hat{y}(k)$  to the current output estimation. In this way the improved prediction to be used in index (1) is:

$$\hat{y}(k+j) \leftarrow \hat{y}(k+j) + e(k) \quad j = N_1, \dots, N_2 \quad (21)$$

This approach works satisfactory to recover steady state errors and for this reason it is widely used in process control and in presence of slow and not oscillatory dynamics (Linkens & Nyongesa, 1995). In case a white box model of the system is available, a more effective approach is to employ the inputs and the measured outputs to reconstruct the unmeasured states of the system by means of a nonlinear observer (Chen, 2000); this allows a periodic realignment of the model toward the system.

### 6.2.1 Intermittent feedback

In the case the prediction model generates an accurate prediction within a defined horizon, it is not really necessary to perform the system/model realignment at the sampling rate  $T_s$ . As proposed in (Chen et Al., 2000) an intermittent realignment is sufficient to guarantee an adequate robustness to system/model mismatches. Following this approach, the effects of the computational delay are overcome by applying to the system, during the current computation interval  $[kT_c, (k+1)T_c]$ , not only the first value of the optimal control sequence yielded in the previous computation interval  $[(k-1)T_c, kT_c]$ , but also the successive  $H-1$  values ( $T_c = H \cdot T_s$ ). When the current optimization process is finished, the optimal control sequence is updated and the feedback signals are sampled and exploited to perform a new realignment; then a new optimization is started. By applying this strategy the realignment period is therefore equal to the computation time  $T_c$ . According to this approach the system is open loop controlled during two successive computational intervals and the optimal control profile during this period is defined by the following function  $f_c(\cdot)$ :

$$\begin{aligned} u_{k|k-1}^*(t) &= f_c \left( x(kT_c), u([kT_c, (k+1)T_c], t) \Big|_{k-1} \right); \\ t &\in [kT_c, (k+1)T_c] \end{aligned} \quad (22)$$

where  $u_{k|k-1}^*(t)$  is the predicted sequence to be applied in the  $k$ -th computational interval that has been computed in the previous interval. The main advantage of the intermittent feedback strategy is that it allows the decoupling between the system sampling time  $T_s$  and

the computing time  $T_c$ ; this implies a significant decrease in the computational burden required for the real-time optimization. On the other hand, a drawback of the intermittent feedback is that it inserts a delay in the control action, because the feedback information has an effect only after  $T_c$  seconds. The evolutionary MBPC algorithms described in (Onnen et Al., 1999; Martinez et Al. 1998) do not take into account the computational delay problem; therefore, their practical real-time implementation strictly depends on the computing power of the available processor that should guarantee the execution of an adequate number of generations within a sampling interval  $[mT_s, (m+1)T_s]$ . The employment of an intermittent feedback strategy allows the enlargement of the computation time available for the convergence of the algorithm and makes the algorithm implementable in real-time also with no excessively powerful processors.

The choice of the computing time  $T_c$  (realignment period) represents an important design issue. This period should be chosen as a compromise between the two concurrent facts:

1. The enlargement of the computing time  $T_c$  allows to refine the degree of optimality of the solution by increasing the number of generations within an optimization period.
2. Long realignment periods cause the prediction error to increase, as a consequence of system/model mismatches.

### 6.2.2. Effects of the intermittent feedback

To evaluate the effects of the intermittent feedback we considered two exemplificative simulations.

Reference is made to the model (4-5) of the flexible mechanical system. We investigated the following situations:

*Case A) No system/model mismatches (ideal case)*

*Case B) Significant modeling error (realistic case)*

Since the Evolutionary MBPC optimization procedure is based on a pseudo randomized search, it is unavoidable that the repetition of the same control task generates slightly different sub optimal control sequences. For this reason the investigation of the performance should be carried out by means of a stochastic analysis by simulating a significant number of realizations (in our analysis 20 experiments of 10 seconds each). We choose as performance measure the mean and the standard deviation of the mean absolute tracking error  $\bar{e}$  for the tip position of the beam starting from the deflected position  $\theta = 0.1 \text{ rad}$ ,  $\dot{\theta} = 0$ ,  $\phi = 0$ ,  $\dot{\phi} = 0$ . This variable is defined as:

$$\bar{e}(\xi) = \frac{1}{N} \sum_{m=1}^{T_{end}} |\theta(m, \xi) - 0| \quad (23)$$

where the (stochastic) variable  $\xi$  is used to put into evidence the stochastic nature of the variable  $\bar{e}$ .

- *Case A, no model uncertainty:* The Evolutionary MBPC is set according to the parameters of table 2. The scope of the analysis is to show the effect on the performance caused by the enlargement of the computing time. As the computing time  $T_c$  is expressed as a multiple of the sampling time  $T_s$ , its enlargement is obtained by increasing the value of the integer  $H$  in (20). The analysis is performed by varying the number of the generations  $K$  that are computed during  $T_c$ . The computational power ( $P$ ) required to

make the MBPC controller implementable in real-time is proportional to the number of generations  $K$  that can be evaluated in a computing interval  $T_c$ , namely:

$$P \propto K / H \quad (24)$$

Table 3 shows the value of the mean value of variable  $\bar{e}(\xi)$  for different values of  $K$  and  $H$ . Some general design considerations can be drawn. The computing power  $P$  required to implement in real-time the algorithm keeps almost constant along the same diagonal of table 3. It is not surprising that, in this ideal case, controllers with the same value of  $P$  give comparable performance. Actually, the increase of the prediction error with the increase of the realignment period has no effect; therefore in case of not significant modeling error, given a certain computing power, the choice of the realignment period is not critical.

mean value on 20 exp.		$H$ ( $T_c=H \cdot T_s$ )				
		1	2	4	8	16
K Gen	1	<b>0.0191</b>	0.0196	<b>0.0201</b>	0.0229	<b>0.0349</b>
	2	0.0187	<b>0.0189</b>	0.0195	<b>0.0203</b>	0.0279
	4	<b>0.0167</b>	0.0180	<b>0.0189</b>	0.0196	<b>0.0223</b>
	8	0.0159	<b>0.0176</b>	0.0185	<b>0.0189</b>	0.0212
	16	<b>0.0148</b>	0.0171	<b>0.0178</b>	0.0180	<b>0.0204</b>
	32	0.0147	<b>0.0166</b>	0.0176	<b>0.0177</b>	0.0193

Table 3. Mean absolute tracking error for  $\bar{e}(\xi)$ , (Case A)

- Case B, significant model uncertainty:* To examine the effect of modeling uncertainty, we assumed an inaccuracy in the value of the mass of the pendulum in the prediction model; its value was increased of 30% with respect to the nominal one. Fig. 8 shows the comparison between the system output (solid line) and the output predicted by the model (dashed line) for the Evolutionary MBPC controller characterized by parameters  $K=32$  and  $H=16$ . The two systems are driven by the same optimal input signal calculated online on the basis of the inaccurate prediction model. At the realignment instants the modeling error is zeroed, subsequently it increases due to the system/model mismatch. The corresponding performance of the evolutionary MBPC are reported in table 4. Some general consideration can be drawn. As expected, due to the system/model mismatches, a decrease in the controller performance with respect to the case (A) is observed. Given a certain computing power  $P$ , the performance degrades significantly by increasing the realignment period. This is due to the fact that the prediction error increases by enlarging the realignment period. For these reasons, in case of significant modeling error, given a defined value of  $P$ , it is preferable to choose the controller characterized by the minimum value of  $T_c$ . It is worth of mention that by using the basic Evolutionary MBPC (Onnen, 1997) we are limited by the constraint  $T_c=T_s$ , namely it is possible to implement only the controllers of the first column ( $H=1$ ) of tables 3 and 4. Clearly, the adoption the proposed

intermittent feedback strategy allows more flexibility in the choice of the parameters of the algorithm to achieve the best performance. In particular, it is not required to compute at least one EA generation in one sampling interval, but this can be computed in two or more sampling intervals thus decreasing the computational load; in fact, as shown by the simulation study, there are many controllers with a  $K/H$  ratio less than one that give satisfactory performance.

mean value on 20 exp.		$H$ ( $T_c=H \cdot T_s$ )				
		1	2	4	8	16
$K$ Gen	1	<b>0.0274</b>	0.0311	<b>0.0332</b>	0.0361	<b>0.0498</b>
	2	<b>0.0267</b>	0.0294	<b>0.0318</b>	0.0329	<b>0.0441</b>
	4	<b>0.0249</b>	0.0268	<b>0.0297</b>	0.0332	<b>0.0394</b>
	8	<b>0.0237</b>	0.0258	<b>0.0278</b>	0.0302	<b>0.0366</b>
	16	<b>0.0231</b>	0.0255	<b>0.0273</b>	0.0297	<b>0.0368</b>
	32	<b>0.0220</b>	0.0253	<b>0.0270</b>	0.0293	<b>0.0340</b>

Table 4. Mean absolute tracking error for  $\bar{e}(\xi)$ , (Case B)

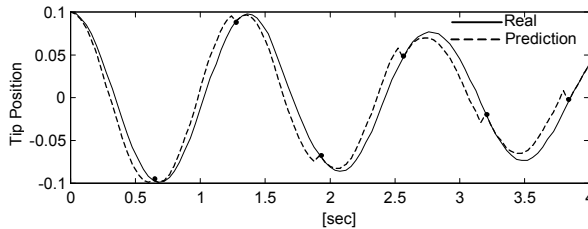


Figure 8. Effect of the system/model realignment in presence of significant modeling error (case  $K=32$ ,  $L=16$ )

### 6.3 Repeatability of the control action

The degree of repeatability of the control action of the controllers described in tables 3 and 4 is investigated in this section. The standard deviation (STD) of the variable  $\bar{e}(\xi)$  can capture this information; in fact a big STD means that the control action is not reliable (repeatable) and the corresponding controller should not be selected. Table 5 reports the STD of  $\bar{e}(\xi)$  in the case of no model uncertainty (case A). In general, given a defined realignment period  $T_c$ , the increase of the computational power  $P$  reduces the variability in the control action. Acceptable performance can be obtained by employing controllers characterized by the ratio  $K/H > 0.5$ . Similar considerations are valid also in the case in which the STD is evaluated for a significant model error (case B) and for this reason are not reported.

	$L \quad (T_c=H \cdot T_s)$				
	1	2	4	8	15
1	5.857e-4	9.319e-4	0.0021	0.0039	0.0044
2	4.395e-4	7.736e-4	0.0018	0.0033	0.0042
K 4	4.217e-4	5.759e-4	0.0023	0.0023	0.0015
8	3.802e-4	5.682e-4	9.340e-4	0.0011	0.0017
16	3.568e-4	3.258e-4	4.326e-4	0.0010	0.0020
32	2.165e-4	2.267e-4	2.896e-4	4288e-4	0.0015

Table 5. Standard Deviation of  $\bar{e}(\xi)$  (Case A)

**6.4 Comparison with conventional optimization methods**

The comparison of the Evolutionary MBPC with respect to conventional methods was first carried in (Onnen, 1997), where it was showed the superiority of the EA on the branch-and-bound discrete search algorithm. In this work the intention is to compare the performance provided by the population-based global search provided by an EA with a local gradient-based iterative algorithm. We implemented a basic gradient steepest descent algorithm and used the standard gradient projection method to fulfill the amplitude and rate constraints for the control signal (Kirk,1970); the partial derivatives of the index J with respect to the decision variables were evaluated numerically. Table 6 reports the result of the comparison of the performance provided by the two methods regarding the simulation time, the mean absolute tracking error  $\bar{e}$  and the number of simulations required by the algorithm, by varying the number of algorithm cycles K in the case  $T_c=T_s$ . The Evolutionary MBPC gave remarkably better performance than the gradient-based MBPC regarding the performance  $\bar{e}$ ; furthermore, the Evolutionary MBPC requires a minor number of simulations that imply also a minor simulation time. This comparison clearly shows that, in this case, the gradient-based optimization get tapped in local minima, while the EA provides an effective way to prevent the problem.

K	GA (Ls=32,N=20)			GRAD (Ls=32)		
	sim Time	$\bar{e}$	N° sim	sim Time	$\bar{e}$	N° sim
1	3.79	1.55	20	4.16	1.51	32
2	5.99	0.33	40	6.81	0.66	64
4	9.31	0.33	80	12.08	0.78	128
8	15.99	0.34	160	30.25	0.59	256
16	41.13	0.32	320	50.32	0.54	512
32	65.87	0.35	640	92.81	0.52	1024

Table 6. Comparison between GA and Gradient Optimization

### 7. Experimental Results

Basing on the results of the previous analysis, we were able to derive the guidelines to implement the improved Evolutionary MBPC for the real-time control of the experimental laboratory system of Fig. 2. The EA was implemented by means of a C procedure and the 4<sup>th</sup> order Runge-Kutta method was used to perform the time domain integration of the prediction model (6) of the flexible system. As in section 5, the scope of the control system is to damp out the oscillations of the tip of the flexible beam, that starts in the deflected position  $\theta = 0.1 \text{ rad}$ ,  $\dot{\theta} = 0$ ,  $\phi = 0$ ,  $\dot{\phi} = 0$ . The desired target position is  $y_d(k) = \theta_d(k) = 0$ . Fig. 9 shows the experimental free response of the tip position that put into evidence the very small damping of the uncontrolled structure. In the same figure it is reported the response obtained by employing a co-located dissipative PD control law of the form

$$\tau(k) = -0.1\phi(k) - 0.5\hat{\phi}(k) \tag{25}$$

where velocity  $\hat{\phi}(k)$  has been estimated by means of the following discrete time filter

$$\hat{\phi}(k) = 0.78\hat{\phi}(k-1) + 10[\phi(k) - \phi(k-1)] \tag{26}$$

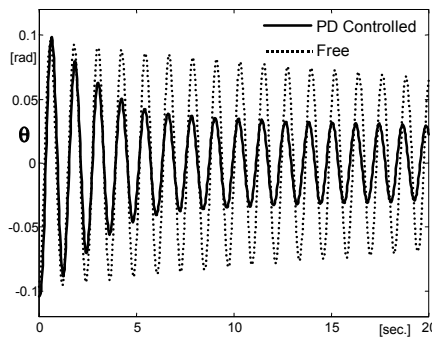


Figure 9. Experimental response

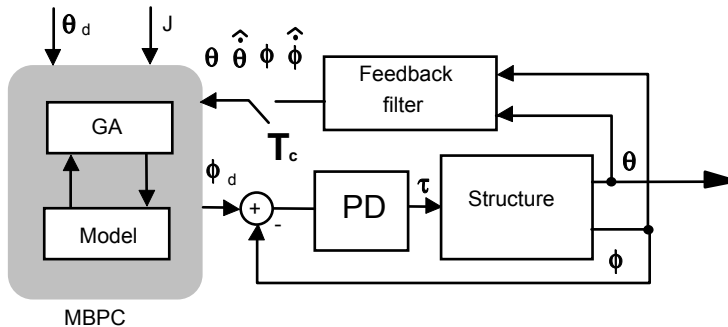


Figure 10. The MBPC scheme for the experimental validation

The conventional PD controller is not able to add a satisfactory damping to the nonlinear system. It is expected that a proper MBPC shaping of the controlled angular position  $\phi_d(k)$  of the pendulum could improve the damping capacity of the control system. Fig. 10 shows the block diagram of the implemented MBPC control system.

In order to perform system/model realignment, it is necessary to realign intermittently all the states of the prediction model (6) basing on the measured variables. Because of only positions  $\theta(k)$  and  $\phi(k)$  can be directly measured by the optical encoders, it was necessary to estimate both the beam and pendulum velocities. The velocity were estimated with a sufficient accuracy by applying single pole approximate derivative filters to the corresponding positions; the discrete time filter for  $\hat{\phi}(k)$  is eq (26);  $\hat{\theta}(k)$  is estimated by:

$$\hat{\theta}(k) = 0.78\hat{\theta}(k-1) + 10[\theta(k) - \theta(k-1)] \quad (27)$$

Every  $T_c$  seconds the vector  $[\theta \ \hat{\theta} \ \phi \ \hat{\phi}]$  is passed to the MBPC procedure to perform the realignment. For the reasons explained in section 3, the redefined inputs of the system is the pendulum position  $\phi(k)$ ; therefore a PD controller has been designed to guarantee an accurate tracking of the desired optimal pendulum shaped position  $\phi_d(k)$ ; the PD regulator is:

$$\tau(k) = -5.0(\phi(k) - \phi_{des}(k)) - 0.6(\hat{\phi}(k) - \dot{\phi}_{des}(k)) \quad (28)$$

The accurate tracking of the desired trajectory  $\phi_d(k)$  is essential for the validity of the predictions carried out by exploiting the model (6). Fig. 11 shows the comparison of shaped reference  $\phi_d(k)$  and the measured one  $\phi(k)$  in a typical experiment. The tracking is satisfactory and the maximum error  $|\phi(k) - \phi_d(k)|$  during the transient is 0.11 rad. This error is acceptable for the current experimentation. Note that this PD regulator has a different task respect to regulator (25), employed in the test of Fig. 9; in fact regulator (28) is characterized by higher gains to achieve trajectory tracking, which cause almost a clamping of the pendulum with the beam.

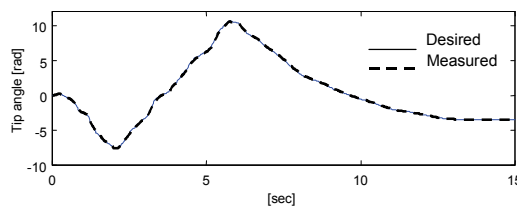


Figure 11. On-line shaped  $\phi_d(t)$  and  $\phi(t)$  ( $T_c = 2T_s$ )

### 7.1 Settings of the experimental evolutionary MBPC

The settings of the MBPC are the same used for the simulations of section 5 and reported in table 2. The decision variables are the sequence of the control input increments  $[\Delta u(k+1) \Delta u(k+2) \dots \Delta u(k+N_2)]$ . The corresponding input signals  $\phi_d(k), \dot{\phi}_d(k), \ddot{\phi}_d(k)$  are obtained by integration of the nominal model equations (6) driven by the sub-optimal control input sequence determined in real-time by the MBPC. The choice of the realignment period  $T_c$  is strongly influenced by the available computational power. In this experiment at least two sampling periods  $T_s$  are required to compute one generation of the EA when the settings of table 2 are employed, therefore it cannot be implemented with a standard Evolutionary MBPC. On the other hand, the improved algorithm can be easily implemented in real-time by choosing a computing power ratio  $K/H \leq 0.5$ . An idea of the performance achievable can be deduced by inspecting tables 2, 3 and 4.

### 7.2 Results

In the experimental phase, it has been evaluated the performance of the MBPC for 4 values of the realignment period  $T_c$  ( $T_c = HT_s$   $H = [2, 4, 8, 16]$ ) in the case of a computing power  $K/H = 0.5$ . Figs. 12 a-d show the measured tip position and the respective value of index  $\bar{e}$  for different values of  $T_c$ . In all the laboratory experiments a significant improvement of the performance with respect to the co-located PD controller (25) is achieved. In fact, after about 6 seconds the main part of the oscillation energy is almost entirely damped out. In all the experiments the performance does not undergo a significant degradation with the increase of the realignment period, showing that in this case an accurate model of the system has been worked out. The values of the index  $\bar{e}$  are in good agreement with the corresponding predicted in table 3 in the case of small modeling error. Anyway, in the case  $T_c = 2 T_s$  (Fig. 12a) a superior performance was achieved near the steady state; in this case, the prediction error is minimum and the residual oscillations can be entirely compensated. On the other hand, in the case  $T_c = 16 \cdot T_s$  (Fig. 12d) some residual oscillations remain, because the prediction error becomes large due to the long realignment period. To underline the effects of the realignment, in Fig. 13 the error  $\theta_d(k) - \theta(k)$  in the case  $T_c = 16 \cdot T_s$  is reported. Every 0.64 seconds, thanks to the realignment, the prediction error is zeroed and a fast damping of the oscillations is achieved; near the steady state, the occurrence of high frequency small amplitude oscillations, cannot be recovered effectively. Fig. 14 reports the sequence of the sub optimal control increments applied to the system for the experiment of Fig. 12a. As expected, the adoption of the adaptive mutation range drives to zero the sequence of control increments near the steady state, allowing a very accurate tracking of the desired trajectory. As for the repeatability of the control action no significant difference was observed on the performance in comparable experiments. Repeating 10 times the experiment of Fig. 12a gave a mean of 0.0205 for the  $\bar{e}(\xi)$  index and a standard deviation of  $1.112e-3$ ; these are in good accordance with the predicted results of table 5.

The results of the experiments clearly demonstrate that the proposed improved Evolutionary MBPC is able to guarantee an easy real-time implementation of the algorithm giving either excellent performance and a high degree of repeatability of the control action.

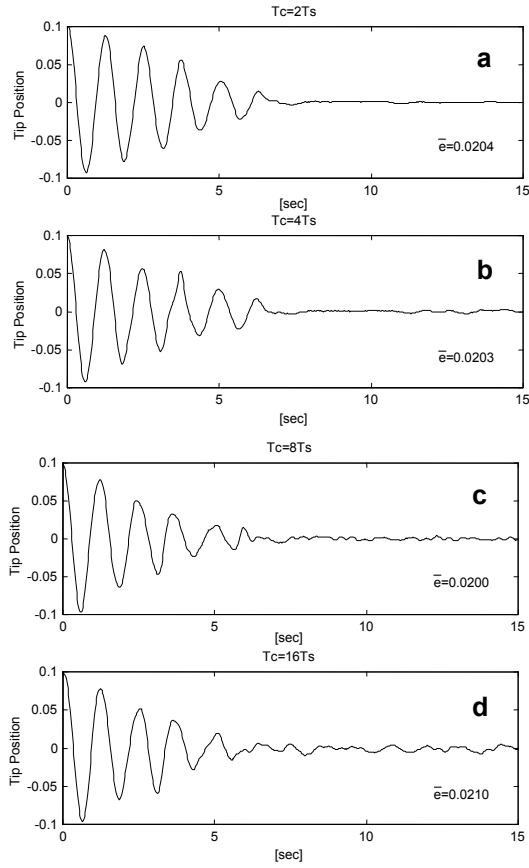


Figure 12. Measured tip position  $\theta(t)$  for different values of the system/model realignment time  $T_c$

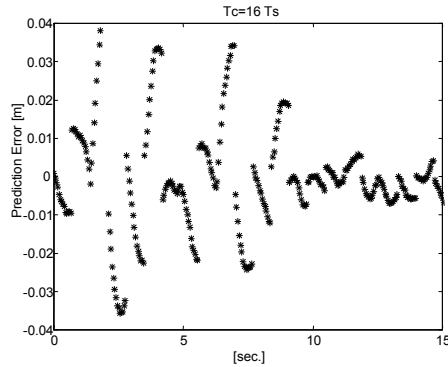


Figure 13. The effect of the system/model realignment on  $\theta_d(t) - \theta(t)$

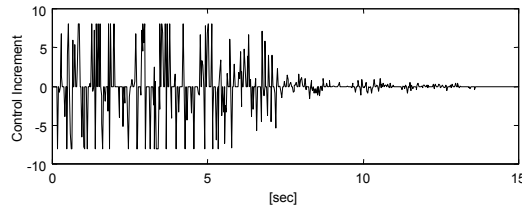


Figure 14. The sequence of input increments in the case  $T_c = 2T_s$ .

## 8. Conclusion

This work introduces an improved Evolutionary Algorithm for the real-time Model Based Predictive Control of nonlinear dynamical systems. The main issues involved in the practical real-time implementation of this control scheme have been pointed out and addressed by adapting and extending some known concepts of the conventional Evolutionary MBPC. The advantage of the online adaptation of the dimension of the search space has been pointed out and a new adaptive mutation range operator in function of the actual prediction error has been proposed. The problem related to the computational delay has been faced by inserting an intermittent feedback strategy in the basic Evolutionary MBPC. This extension allows the computation of one generation of the EA in more than one sampling interval, thus decreasing the required computational power for the real-time implementation. The application of the improved EA allows the real-time MBPC of fast dynamics systems by employing a CPU with a limited computing power. The improved algorithm has been experimented with remarkable results for the stabilization of oscillations of a laboratory nonlinear flexible mechanical system. A stochastic analysis showed that improved Evolutionary Algorithm is reliable in the sense that a good repeatability of the control action can be achieved; furthermore, the EA outperforms a conventional iterative gradient-based optimization procedure. Although the potentiality of the improved Evolutionary MBPC have been shown only for a single laboratory experiments, the analysis and design guidelines are general and for this reason can be easily applied to the design of real-time Evolutionary MBPC for a general nonlinear constrained dynamical systems.

## 9. Predictive Reference Shaping for Constrained Robotic Systems Using Evolutionary Algorithms

*Part of the following article has been previously published in: M.L. Fravolini, A. Ficola, M. La Cava. "Predictive Reference Shaping for Constrained Robotic Systems Using Evolutionary Algorithms", Applied Soft Computing, Elsevier Science, in stampa, vol. 3, no. 4, pp.325-341, 2003, ISSN:1568-4946.*

The manufacturing of products with a complex geometry demands for efficient industrial robots able to follow complex trajectories with a high precision. For these reasons, tracking a given path in presence of task and physical constraints is a relevant problem that often occurs in industrial robotic motion planning. Because of the nonlinear nature of the robot dynamics, the robotic optimal motion-planning problem cannot be usually solved in closed form; therefore, approximated solutions are computed by means of numerical algorithms.

Many approaches have been proposed concerning the constrained robot motion-planning problem along a pre specified path, taking into account the full nonlinear dynamics of the manipulator. In the well-known technique described in (Bobrow et Al. 1985; Shin, & McKay, 1985), the robot dynamics equations are reduced into a set of second order equations in a path parameter. The original problem is then transformed into finding the curve in the plane of the path parameter and its first time derivative, while the constraints on actuators torque are reduced into bounds on the second time derivative of the path parameter. Although this approach can be easily extended to closed loop robot dynamics, it essentially remains an *offline*-planning algorithm; indeed, in presence of unmodeled dynamics and measurement noise this approach reduces its effectiveness when applied to a real system. Later, to overcome these robustness problems, some *online feedback* path planning schemes have been formulated. Dahl in (Dahl & Nielsen, 1990) proposed an online path following algorithm, in which the time scale of the desired trajectory is modified in real time according to the torque limits. A similar approach has been also proposed by Kumagai (Kumagai et Al., 1996).

Another approach to implement an online constrained robotic motion planning is to employ Model Predictive Control (MPC) strategies (Camacho & Bordons, 1996; Garcia et Al., 1989). A MPC, on the basis of a nominal model of the system, online evaluates a sequence of future input commands minimizing a defined index of performance (tracking error) and taking into account either input or state constraints. The last aspect is particularly important because MPC allows the generation of sophisticated optimal control laws satisfying general multiobjective constrained performance criteria.

Since only for linear systems (minimizing a quadratic cost function) it is possible to derive a closed form solution for MPCs, an important aspect is related to the design of an efficient MPC optimization procedure for the online minimization of an arbitrary cost function, taking into account system nonlinearities and constraints. Indeed, in a general formulation, a constrained non-convex nonlinear optimization problem has to be solved on line, and in case of nonlinear dynamics, the task could be highly computationally demanding; therefore, the online optimization problem is recognized as a main issue in the implementation of MPC (Camacho & Bordons, 1996). Many approaches have been proposed to face the online optimization task in nonlinear MPC. A possible strategy, as proposed in (Mutha et Al., 1997; Ronco et Al., 1999) consists of the linearization of the dynamics at each time step and of the use of standard linear predictive control tools to derive the control policies. Other methods utilize iterative optimization procedures, as gradient based algorithms (Song & Koivo, 1999) or discrete search techniques as Dynamic Programming (Luus, 1990) and Branch and Bound (Roubos et Al., 1999) methods. The main advantages of a search algorithm is that it can be applied to general nonlinear dynamics and that the structure of the objective function is not restricted to be quadratic as in most of the other approaches. A limitation of these methods is the fast increase of the algorithm complexity with the number of decision variables and grid points. Recently, another class of search techniques, called Evolutionary Algorithms (EAs) (Foegel, 1999; Goldberg, 1989) showed to be very effective in *offline* robot path planning problems (Rana, 1996); in the last years, thanks to the great advancements in computing technology, some authors have also proposed the application of EAs for *on-line* performance optimization problems (Lennon & Passino, 1999; Liaw & Huang, 1998; Linkens & Nyongesa, 1995; Martinez et Al., 1998; Onnen et Al., 1997 ; Porter & Passino, 1998).

Recently the authors, in (Fravolini et Al., 2000) have applied an EA based optimization procedure for the online reference shaping of flexible mechanical systems. The practical *real-time* applicability of the proposed approach was successively tested with the experimental study reported in (Fravolini et Al., 1999). In this work the approach is extended to the case of robotic motion with constraints either on input and state variables. Two significant simulation examples are reported to show the usefulness of the online reference shaping method; some considerations concerning the online computational load are also discussed. The paper is organized as follows. Section 10 introduces the constrained predictive control problem and its formalization. Section 11 introduces the EA paradigm, while section 12 describes in details either the online EA based optimization procedure and the specialized EA operators required for MPC. In section 13 the proposed method is applied to two benchmark systems; in section 14 a comparison with a gradient-based algorithm is discussed. Finally, the conclusions are reported in section 15.

## 10. The Robotic Constrained Predictive Control Method

In the general formulation it is assumed that an inner loop feedback controller has already been designed to ensure stability and tracking performance to the robotic system, as shown in figure 15. In case of fast reference signals  $r(k)$ , the violation of input and state constraints could occur; to overcome this problem the authors, in (Fravolini et Al., 2000) proposed to add to the existing feedback control loop an online predictive reference shaper. The predictive reference shaper is a nonlinear device, that modifies in real-time the desired reference signal  $r(k)$  on the basis of a prediction model and of the current feedback measures  $y(k)$ . The scope is that the shaped reference signal  $r_s(k)$  allows a more accurate track of the reference signal without constraints violation. Usually, these requirements are quantified by an index of cost  $J$  and a numerical procedure is employed to online minimize this function with regard to a set of decision variables. Typically, the decision variables are obtained by a piecewise constant parameterization of the shaped reference  $r_s(k)$ .

As for predictive controllers strategies, the reference shaper evolves according to a receding horizon strategy: the planned sequence is applied until new feedback measures are available; then a new sequence is calculated that replaces the previous one. The receding horizon approach provides the desired robustness against both model and measurement disturbance.

*Note 1:* The trajectory shaper of figure 15 could also be employed without the inner feedback control loop. In this case, the shaper acts as the only feedback controller and it directly generates the control signals; in this case  $u(k) \equiv r_s(k)$ .

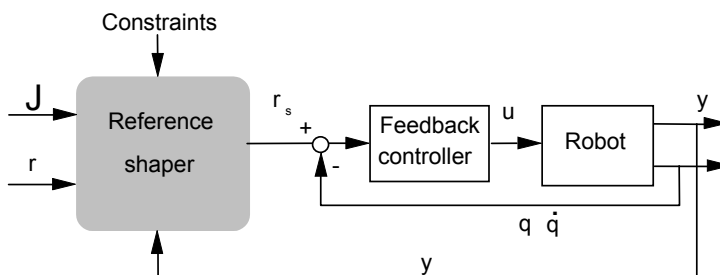


Figure 15. Online reference shaper

### 10.1 Problem Formulation

The equation of motion of a constrained robot dynamics can be formally expressed as follows:

$$\begin{cases} M(q)\ddot{q} + C(q, \dot{q}) + G(q) = Qu \\ u = u(q, \dot{q}, \xi, r_s) \quad y = F \cdot [q^T \quad \dot{q}^T]^T \\ h = h(q, \dot{q}, \xi, r_s) \\ q(0) = q_o \quad \dot{q}(0) = \dot{q}_o \quad \xi(0) = \xi_o \end{cases} \quad (1)$$

where  $q \in R^m$  is the vector of robot positions,  $\dot{q}$  is the vector of robot velocities,  $\xi$  collects the states either of the controller or of the actuation system,  $M$  is the inertia matrix,  $C$  is the Coriolis vector,  $G$  the gravity vector,  $Q$  is the input selection matrix,  $u \in R^m$  is the control vector,  $y \in R^n$  is the output performance vector,  $F$  is the output selection matrix,  $r_s \in R^n$  is the shaped reference to be tracked by  $q$ . The vector  $[q_o, \dot{q}_o, \xi_o]$  represents the initial condition of the feedback-controlled system;  $h \in R^p$  is the vector of the  $p$  constraints that must satisfy the relation:

$$h(q, \dot{q}, \xi, r_s, t) \in H \quad t > 0 \quad (2)$$

where  $H$  is the set where all the constraints (2) are fulfilled.

The aim of the reference shaper is to online compute the shaped reference  $r_s(k)$  in order to fulfill all the constraints (2) while minimizing a defined performance measure  $J$  that is function of tracking error of the performance variables  $y$ . Since the online optimization procedure requires a finite computing time  $T_c$  (optimization time) to compute a solution, the proposed device is discrete time with sampling instants  $k_c \cdot T_c$ . The inner feedback controller is allowed to work at a faster rate  $kT_s$  (where  $T_c = a \cdot T_s$  and  $a > 1$  is an integer, therefore  $k_c = a \cdot k$ ). The optimal sequence  $r_s^*(k_c)$  is determined as the result of an optimization problem during each computing interval  $T_c$ . The cost index  $J$ , which was employed, is:

$$J(k_c) = \sum_{i=1}^n \alpha_i \sum_{j=1}^{N_{yi}} |\hat{y}_i(k_c + j | k_c) - r_i(k_c + j)| + \sum_{i=1}^m \beta_i \sum_{j=1}^{N_{ui}} |\Delta u_i(k_c + j - 1)| + J_1(k_c) \quad (3)$$

The first term of (3) quantifies the absolute predicted tracking error between the desired output signal  $r_i(k_c + j)$  and the predicted future output  $\hat{y}_i(k_c + j | k_c)$ , estimated on the basis of the robot model and the feedback measures available at instant  $k_c$ ; this error is evaluated over a defined prediction horizon of  $N_{yi}$  samples. The second term of (4) is used to weight the control effort that is quantified by the sequence of the input increments  $\Delta u_i(k) = u_i(k) - u_i(k-1)$  (evaluated over the control horizon windows of  $N_{ui}$  samples). The coefficients  $\alpha_i$  and  $\beta_i$  are free weighting parameters. The term  $J_1(k_c)$  in (5) is a further cost function, which can be used to take into account either task or physical constraints. In this work the following constraints on control and state variables have been considered:

$$U_i^- < u_i(k) < U_i^+ \quad \Delta U_i^- < \Delta u_i(k) < \Delta U_i^+ \quad i = 1, \dots, n \quad (6)$$

$$X_i^- < x_i(k) < X_i^+ \quad \Delta X_i^- < \Delta x_i(k) < \Delta X_i^+ \quad i = 1, \dots, n \quad (7)$$

Constraints (6) take into account possible saturations in the amplitude and rate of the actuation system, while constraints (7) prevent the robot to work in undesirable regions of the state space. The optimization problem to be solved during each sampling interval  $T_c$  is formalized as follows:

$$\underset{R_{si}(k_c)}{\text{mim}} \quad J(k_c) \quad (8)$$

taking into account the fulfillment constraints (6) and (7). The optimization variables of the problem are the elements of the sequences  $R_{si}(k_c)$  evaluated within the control horizon:

$$R_{si}(k_c) = [\Delta r_{si}(k_c), \Delta r_{si}(k_c + 1), \dots, \Delta r_{si}(k_c + N_{Ui} - 1)] \quad i = 1, \dots, n \quad (9)$$

## 11. Evolutionary Algorithms

The Evolutionary Algorithms are multi point search procedures that reflect the principle of evolution, natural selection and genetics of biological systems (Foegel; 1994; Goldberg, 1989). An EA explores the search space by employing stochastic rules; these are designed to quickly direct the search toward the most promising regions. It has been shown that the EAs provide a powerful tool that can be used in optimization and classification applications. The EAs work with a population of points called chromosomes; a chromosome represents a potential solution to the problem and comprises a string of numerical and logical variables, representing the decision variables of the optimization problem. The EA paradigm does not require auxiliary information, such as the gradient of the cost function, and can easily handle constraints; for these reasons EAs have been applied to a wide class of problems, especially those difficult for hill-climbing methods.

An EA maintains a population of chromosomes and uses the genetic operators of "selection" (it represents the biological survival of the fittest ones and is quantified by a fitness measure that represents the objective function), "crossover" (which represents mating), and "mutation" (which represents the random introduction of new genetic material), with the aim of emulating the evolution of the species. After some generations, due to the evolutionary driving force, the EA produces a population of high quality solutions for the optimization problem.

The implementation of a basic EA can be summarized by the following sequence of standard operations:

- a. (Random) Initialization of a population of  $N$  solutions
- b. Calculation of the fitness function for each solution
- c. Selection of the best solutions for reproduction
- d. Application of crossover and mutation to the selected solutions
- e. Creation of the new population
- f. Loop to point b) until a defined stop criterion is met

## 12. The Online Optimization Procedure for MPC

In order to implement the predictive shaper described in the previous section, it is necessary to define a suitable online optimization procedure. In this section it is described the MPC

algorithms based on an EA; the resulting control scheme is reported in figure 16. The flow diagram of the online optimization procedure implemented by the Evolutionary reference shaper is reported in figure 17.

*Note 2:* In this work, to keep notation simple, the prediction ( $N_{Yi}$ ) and control ( $N_{Ui}$ ) horizons are constrained to have the same length for each output and each input variable ( $N_Y = N_U$ ).

*Note 3:* Because the feedback controller sampling interval  $T_s$  is often different from the optimization time  $T_c$  ( $T_c = a \cdot T_s$ , often  $T_c \gg T_s$ ), during a period  $T_c$  it is required to perform predictions with a prediction horizon  $N_Y$  longer at least  $2T_c$  ( $2a$  samples,  $T_c = 2a \cdot T_s$ ). More precisely, during the current computation interval  $[k_c, k_c+1]$  the first  $a$  values of the optimal sequences  $R_{si}^*(k_c)$  that will be applied to the real plant are fixed and coincide with the optimal values computed in the previous computational interval  $[k_c-1, k_c]$ ; the successive  $N_U - a$  values are the actual optimization variables that will be applied in the successive computation interval  $[k_c+1, k_c+2]$ .

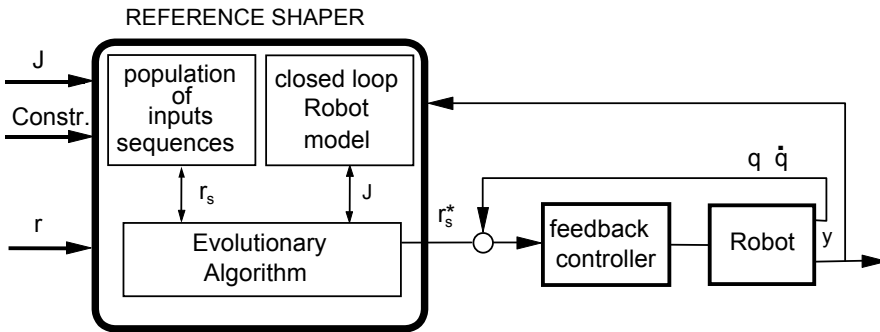


Figure 16. The proposed evolutionary reference shaper

### 12.1 Specialized Evolutionary Operators for MPC

The application of EA within a MPC requires the definition of evolutionary operators expressly designed for real time control. These operators were introduced in previous works (Fravolini et Al, 1999; Fravolini et Al. 2000) and tested by mean of extensive simulation experiments. Some of these operators are similar to those reported in (Goggos; 1996; Grosman & Lewin, 2002; Martinez et Al., 1998; Onnen et Al., 1997) and constitute a solid, tested, and accepted base for evolutionary MBPC.

In this paragraph the main EA operators expressly specialized for online MPC are defined.

- *Fitness function:* The objective function to be online minimized (with rate  $T_c$ ) is the index  $J(k_c)$  in (3). The fitness function is defined as:

$$f = 1/J \quad (10)$$

- *Decision variables:* The decision variables are the elements of the sequences  $R_{si}(k_c)$  (9).
- *Chromosome structure and coding:* A chromosome is generated by the juxtaposition of the coded sequence of the increments of the shaped reference  $R_{si}(k_c)$ . With regard to the codification of the decision variables, some alternatives are possible. Binary or decimal codification are not particularly suited in online applications, since they require time consuming coding and decoding routines. Real coded variables, although do not require any codification, have the drawback that the implementation of evolutionary

operators for real numbers is significantly slower than in the case of integers variables. Therefore, the best choice is an integer codification of the decision variables, which can guarantee a good accuracy of the discretization while operating on integer numbers. A coded decision variable  $x_{ij}$  can assume an integer value in the range  $0, \dots, L_0, \dots, L_i$ , where  $L_i$  represents the quantization accuracy. This set is uniformly mapped in the bounded interval  $\Delta R_i^- \leq \Delta r_{si} \leq \Delta R_i^+$ .  $L_0$  represents the integer corresponding to  $\Delta r_{si} = 0$ . The  $l$ -th chromosome in the population at  $t$ -th generation during the computing interval  $[k_c, k_c+1]$  is a string of integer numbers:

$$X_l^t = [x_{11}, x_{12}, \dots, x_{1, N_{U1}} \mid x_{21}, x_{22}, \dots, x_{2, N_{U2}} \mid, \dots, x_{m1}, x_{m2}, \dots, x_{m, N_{Um}}] \quad (11)$$

the actual value of the decision variables  $\Delta r_{si}(k_c+j)$  are obtained by applying the following decoding rule:

$$\Delta r_{si}(k+j) = \Delta R_i^- + \Delta_{Ri} \cdot x_{i,j} \quad j = 1, \dots, N_{Ui} \quad i = 1, \dots, n \quad (12)$$

where  $\Delta_{Ri} = |\Delta R_i^+ - \Delta R_i^-| / L_i$  is the control increment resolution for the  $i$ -th input. The sequences of shaped references applied in the prediction window result:

$$r_{si}(k_c+j) = r_{si}(k_c-1+j) + \Delta r_{si}(k_c+j) \quad j = 1, \dots, N_{Ui} \quad i = 1, \dots, n \quad (13)$$

- *Selection and Reproduction mechanisms (during a computing interval  $T_c$ )* Selection and reproduction mechanisms act at two different levels during the on-line optimization. The lower level action concerns the optimization of the fitness function  $f$  during  $T_c$ . Because a limited computation time is available, it is essential that the good solutions founded in the previous generations are not lost, but are used as "hot starters" for the next generation. For this reason a *steady state* reproduction mechanism is employed, namely the best  $S$  chromosomes in the current generation are passed unchanged in the next one; this ensures a not decreasing fitness function for the best population individual. The remaining part of the population is originated on the basis of a *rank selection* mechanism; given a population of size  $N$ , the best ranked  $D$  individuals constitute a mating pool. Within the mating pool two random parents are selected and two child solutions are created applying crossover and mutation operators; this operation lasts until the new population is entirely replaced. This approach is similar to the algorithm described in (Yao & Sethares, 1994).
- *Receding Horizon Heredity (between two successive computational intervals)*: The second level of selection mechanism implements the receding horizon strategy. The best chromosomes computed during the current  $T_c$  are used as starting solutions for the next optimization. At the beginning of the next computational interval, because the prediction and control horizons are shifted in the future, the values of the best  $S'$  chromosomes are shifted back of  $a$  locations ( $T_c = a \cdot T_s$ ). In this way the first  $a$  values are lost and their positions are replaced by the successive  $a$ . The values in the last positions (from  $a+1$  to  $N_{Ui}$ ) are simply filled by keeping constant the value of the  $a$ -th variable. The shifted  $S'$  chromosomes represent the "hot starters" for the optimization in the next computational interval; the remaining  $N-S'$  chromosomes are randomly generated. The

application of hereditary information is of great relevance, because a significant improvement in the convergence speed of the algorithm has been observed.

- *Crossover*: during an optimization interval  $T_c$  uniform crossover has been implemented. Given two selected chromosomes  $X_a^t$  and  $X_b^t$  in the current generation  $t$ , two corresponding variables are exchanged with a probability  $p_c$ , namely the crossed elements in the successive generation result:

$$x_{a,(i,j)}^{t+1} = x_{b,(i,j)}^t \quad \text{and} \quad x_{b,(i,j)}^{t+1} = x_{a,(i,j)}^t \quad (14)$$

- *Mutation*: random mutation are applied with probability  $p_m$  to each variable of a selected chromosome  $X_a^t$ , in the current generation according to the formula:

$$x_{a,(i,j)}^{t+1} = x_{a,(i,j)}^t + \text{rand}(\bar{\Delta}) \quad (15)$$

where  $\text{rand}(\bar{\Delta})$  is a random integer in the range:  $[-\Delta_{xi}, \dots, 0, \dots, \Delta_{xi}]$  and  $\Delta_{xi}$  is the maximum mutation amplitude.

- *Constraints*: One of the main advantages of MPC is the possibility of taking into account of constraints during the online optimization. Different kinds of constraints have been considered:

i) *Constraints on the shaped reference*: In order to generate references that could be accurately tracked by means of the available actuation system, it can be required to constrain either the maximum/minimum value of the shaped signals or and their rate of variation. For this reason, if, during the optimization a decision variable  $x_{i,j}$  violates its maximum or minimum allowed value in the range  $0, \dots, L_i$ , the following threshold is applied:

$$\begin{cases} \text{if } x_{i,j}^t > L_i \text{ then } x_{i,j}^t = L_i \\ \text{if } x_{i,j}^t < 0 \text{ then } x_{i,j}^t = 0 \end{cases} \quad i = 1, \dots, n \quad (16)$$

The application of (16) automatically guarantees that  $\Delta R_i^- \leq \Delta r_{si}(k) \leq \Delta R_i^+$ . In a similar fashion it is possible to take into account of the constraint on the amplitudes  $R_i^- \leq r_{si}(k) < R_i^+$  by applying the thresholds:

$$\begin{cases} \text{if } r_{si}(k) + \Delta r_{si}(k) > R_i^+ \text{ then } x_{i,k}^t = L_0 + \text{int} \left( \frac{R_i^+ - r_{si}(k)}{\Delta_{Ri}} \right) \\ \text{if } r_{si}(k) + \Delta r_{si}(k) < R_i^- \text{ then } x_{i,k}^t = L_0 + \text{int} \left( \frac{R_i^- - r_{si}(k)}{\Delta_{Ri}} \right) \end{cases} \quad i = 1, \dots, n \quad (17)$$

Thresholds (16) and (17) ensure the desired behavior of  $r_{si}(k)$ .

ii) *Constraints on control signals*: In the case the inner control loop is not present, then  $u(k) \equiv r_s(k)$  (see Note1) therefore the constraints are automatically guaranteed by thresholds

(16)-(17). In the case the inner loop is present, constraints (18) are implicitly taken into account in the optimization procedure by inserting in the prediction model an amplitude and a rate saturation on the command signals generated by the inner controller. These thresholds are implemented by:

$$\begin{cases} \text{if } u_i(k) > U_i^+ & \text{then } u_i(k) = U_i^+ \\ \text{if } u_i(k) < U_i^- & \text{then } u_i(k) = U_i^- \end{cases} \quad (19)$$

and:

$$\begin{cases} \text{if } \Delta u_i(k) > \Delta U_i^+ & \text{then } \Delta u_i(k) = \Delta U_i^+ \\ \text{if } \Delta u_i(k) < \Delta U_i^- & \text{then } \Delta u_i(k) = \Delta U_i^- \end{cases} \quad (20)$$

Generally, in the design phase of the inner feedback controllers, it is difficult to take into account the effects of the possible amplitude and rate limit on the inputs; on the other hand constraints (19) and (20) are easily introduced in the MPC approach. The resulting shaped references are thus determined by taking explicitly into account the physical limitations of the actuation systems.

iii) *Constraints on state variables*: These constraints are taken into account by exploiting the penalty function strategy; namely a positive penalty term  $J_1$ , proportional to the constraint violation, is added to the cost function  $J(k_c)$  in (21). Let the set  $H$  defining the  $p$  constraints in (2) be defined as:

$$H = \{h_i \in R^p : h_i(q, \dot{q}, x, r_s) \leq 0, i = 1, \dots, p\} \quad (22)$$

then, the penalty function taking into account the violation of constraints along the whole prediction horizon has been defined as:

$$J_1(k) = \sum_{i=1}^p \gamma_i \sum_{j=1}^{N_y} \max(h_i(k+j), 0) \geq 0 \quad (23)$$

When all the constraints are fulfilled,  $J_1$  is equal to zero, otherwise it is proportional to the integral of the violations. By increasing the values of the weights  $\gamma_i$ , it is possible to enforce the algorithm toward solutions that fulfill all the constraints.

- *Choice of Computing time  $T_c$* : It should be chosen as a compromise between the two concurrent factors:

1) The enlargement of the computing time  $T_c$  allows to refine the degree of optimality of the best solutions by increasing the EA generation number,  $gen$ , that can be evaluated within an optimization period.

2) A large  $T_c$  causes the increase of the delay in the system. Excessive delays cannot be acceptable for fast dynamics systems.

Obviously, the computational time is also influenced by the computing power of the processor employed.

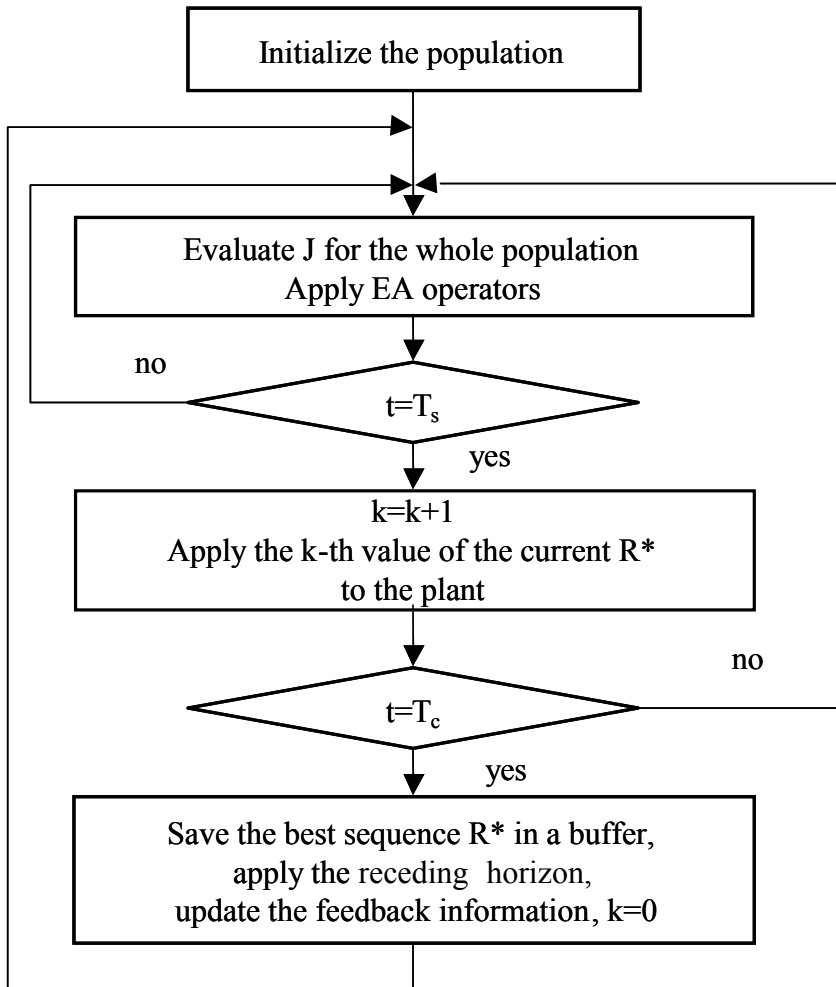


Figure 17. Flow diagram of the evolutionary reference shaper

### 13 Applications

In this section the proposed evolutionary shaper is applied to two significant robotic control problems.

#### 13.1 Coupled Flexible Beam/Pendulum (Ex. 1)

In this example the performance of the trajectory shaper has been tested on a nonlinear flexible mechanical system. The system is composed of a flexible beam clamped at one side, with a controlled pendulum hinged on the other; the motion occurs in the horizontal plane (figure 18a). The stabilization of this system represents an excellent benchmark because of the fast dynamics and the presence of oscillatory and scarcely damped modes. The scope of

the control law is to add damping to the system by properly driving the pendulum, when the beam starts in a deflected position. The beam is modeled as a first order mass-spring-damper element (figure 18b). The matrices of the system according to model (1) are:

$$M = \begin{bmatrix} (M+m)L^2 + ml^2 - 2mlL \cos \phi & ml^2 - mL \cos \phi \\ ml^2 - mL \cos \phi & ml^2 \end{bmatrix}$$

$$C = \begin{bmatrix} 2mL\dot{\phi}\dot{\theta} \sin \phi + mL\dot{\phi}^2 \sin \phi + K_1\theta + C_1\dot{\theta} \\ -mL\dot{\theta}^2 \sin \phi + C_2\dot{\phi} \end{bmatrix} \quad (24)$$

$$Q = [0 \ u]^T \quad q = [\theta \ \phi]^T$$

The parameters of the model (25) are reported in table 7. The inner loop is controlled by a PD law; this controller was expressly tuned to increase the damping effect induced by the controlled pendulum on the beam oscillations. In figure 19a it is reported the deflection of the tip position of the beam in the case of free and of PD-controlled response, while in figure 19b it is reported the corresponding PD control signal  $u(k)$ . Although a significant reduction of the oscillations is observed, the PD controller is not able to add a large damping to the system.

To improve the performance of the PD controlled system a reference MPC shaper is added to the inner loop according to general scheme of figure 15. The decision variables are the elements of the sequence  $[\Delta r_s(k_c+1), \Delta r_s(k_c+2), \dots, \Delta r_s(k_c+N_U-1)]$  of the shaped  $r_s(k)$  reference for the feedback PD controller. The horizon length was set to  $N_U=N_V=32$ ; this implies a control horizon of  $N_U T_s = 1.28$  s ( $T_s=0.04$ s); the employment of shorter horizons has generated unsatisfactory responses. The other parameters of the evolutionary shaper are reported in table 8. Since the scope is to damp out the oscillation of the tip of the beam, the objective function (25) was particularized as follows:

$$J(k_c) = \sum_{j=0}^{N_U} |\theta(k_c + j) - y_d(k_c + j)| = \sum_{j=0}^{N_U} |\theta(k_c + j) - 0| \quad (26)$$

where the angle  $\theta(k)$  is chosen as performance variable ( $\theta(k)=y(k)$ ) (it was assumed that for small deflections the tip position is  $L\ddot{\theta}(k)$ ). Furthermore, a saturation block ( $-20 < u(k) < 20$ ) was added at the output of the PD controller either in the "real" system and in the prediction model to take into account the saturation of the actuation system. Some simulations were performed with the aim of comparing the performance of the reference shaper by varying the computational power required for real-time computation. This aspect has been emulated by fixing the number of generations ( $gen=10$ ) evaluated during the interval  $T_c$  and by varying its duration; indeed, the longer is  $T_c$ , the less is the computing power required. The computing times  $T_c=[1, 2, 4, 8, 16] \cdot T_s$  have been evaluated; the test results are summarized by the performance indexes reported in table 9. The reported indexes were evaluated over a period of 15 s. In column 2 the percent performance increase (evaluated by means of the mean absolute tracking error) with regard to the PD controller case is reported; the shaper produced a significant reduction of about 30% in the case of

$T_c=[1, 2, 4] \cdot T_s$ . Conversely, for  $T_c=16 \cdot T_s$  the shaper has an evident negative effect (-56%), because in this case, the number of generations evaluated in a sampling time  $T_s$  ( $gen/T_s=0.625$ , see column 7) is too small to reach a satisfactory solution. In the successive columns of table 9 the mean, maximum and minimum value of the control signal are reported. In column 6 it is reported the ratio ( $t_s/t_r$ ) between the simulation time  $t_s$  employed to run the simulation and the real-time  $t_r$ . All the described simulations were carried out by employing a PentiumII 200 MHz processor; the code was written in C, and the dynamics were simulated with a fourth order Runge-Kutta algorithm. In more details, for  $T_c \geq 2 \cdot T_s$ , the normalized simulation time is less than one; this implies that the proposed procedure could be employed on the physical system with the current processor as real-time controller. In figure 20a it is shown the tip position when the reference shaper is applied when  $T_c=2 \cdot T_s$ . A significant increase in the oscillation damping is observed with respect to the PD law. In figure 20b the corresponding control signal is shown; it should be noted that, although the active constraint on the command amplitude, the performance are not degraded. In figure 21a the online shaped reference  $r_s(k)$  is shown, while in figure 21b it is reported the motor position  $\phi(k)$ . Overall, the results clearly show that the evolutionary shaper was able to give a substantial improvement to the performance of the existing PD feedback controller.

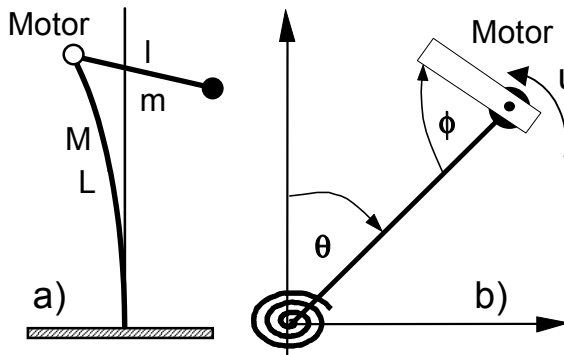


Figure 18. The structure (a), and the model (b)

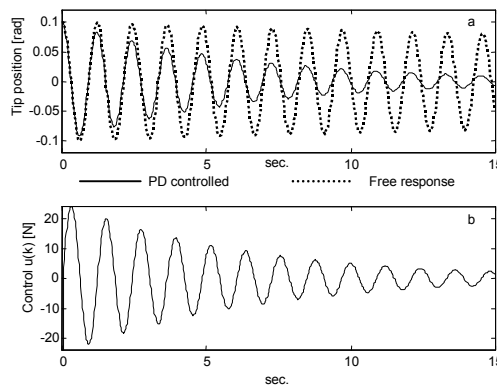


Figure 19. Free and PD controlled response (a), PD control effort (b)

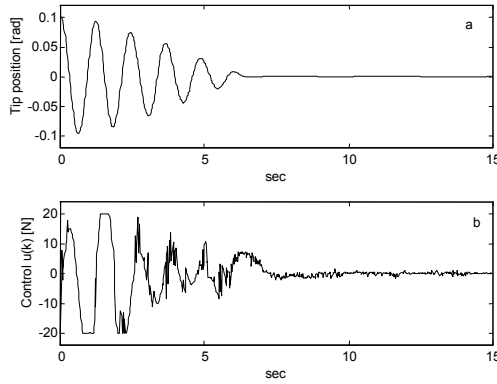


Figure 20. The response with the reference shaper PD+MPC ( $T_c = 2 \cdot T_s$ )

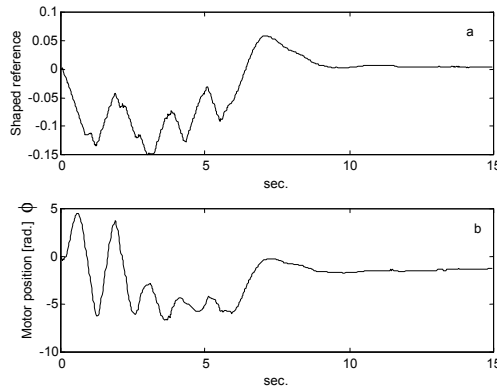


Figure 21. Shaped reference  $r_s$  (a) and motor position (b)

M	0.689 kg	Mass of the beam
m	0.070 kg	Mass of the pendulum
L	1.000 m	Length of the beam
l	0.086 m	Length of the pendulum
$K_1$	25.3 Nm/rad	Elastic coeff. of the beam
$C_1$	0.008 Nm/s rad	Damping coeff. of the beam
$C_2$	0.050 Nm/s rad	Damping coeff. of the pendulum

Table 7. Parameters of the model (Example 1)

$N = 20$	$D = 6$	$S = 2$	$S' = 4$
$p_m = 0.1$	$p_c = 0.8$	$T_s = 0.04$	$L = 4000$
$N_U = 32$	$N_Y = 32$	$\Delta_X = 1000$	$R^+ = 0.4$
$R^- = -0.4$	$\Delta R^+ = 0.01$	$\Delta R^- = -0.01$	$gen = 10$

Table 8. Parameters of the evolutionary shaper (example 1)

	$T_c$	$ e(k) _{mean}$	$ u(k) _{mean}$	$U_{max}$	$U_{min}$	$t_s/t_r$	$Gen/T_s$
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Free	-	0.0569	-	-	-	-	-
PD	$T_s$	0.0243 +0%	5.87	21.8	-20.1	-	-
PD+MPC	$T_s$	0.0164 +32%	4.05	20	-20	1.19	10
PD+MPC	$2 T_s$	0.0167 +31%	4.11	20	-20	0.73	5
PD+MPC	$4 T_s$	0.0193 +20%	5.37	20	-20	0.33	2.5
PD+MPC	$8 T_s$	0.0231 +5%	6.02	20	-20	0.18	1.25
PD+MPC	$16 T_s$	0.0380 -56%	9.27	20	-20	0.11	0.625

Table 9. Performance of PD and PD+MPC (example 1). Col.1 Computing time, Col.2 Mean tracking error, Col.3 mean control effort, Col.4 Maximum control, Col.5 Minimum control, Col.6 Normalized simulation time, Col.7 Generation evaluated in a sampling interval

**13.2 Two Links Planar Robot (Ex. 2)**

In this study a well-known benchmark system in robotic optimal control was considered (Bobrow et Al., 1985): the two-links planar robot shown in figure 22. The matrices of the model according to (1) are:

$$\begin{aligned}
 M &= \begin{bmatrix} \beta_5 + 2\beta_6 c \phi + \beta_7 & \beta_6 c \phi + \beta_7 \\ \beta_6 c \phi + \beta_7 & \beta_7 \end{bmatrix} \\
 C &= \begin{bmatrix} -\beta_6 \dot{\psi}(2\dot{\theta} + \dot{\phi}) \text{sen}(\phi) \\ -\beta_6 \dot{\phi} \dot{\theta}^2 \text{sen}(\phi) \end{bmatrix} \\
 Q &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}
 \end{aligned} \tag{27}$$

where:  $\beta_4 = I_2 + m_3 l_2^2$ ,  $\beta_5 = I_1 + l_2^2(m_2 + m_3)$ ,  $\beta_6 = l_1(a_2 m_2 + l_2 m_3)$ ,  $\beta_7 = I_3 + \beta_4$

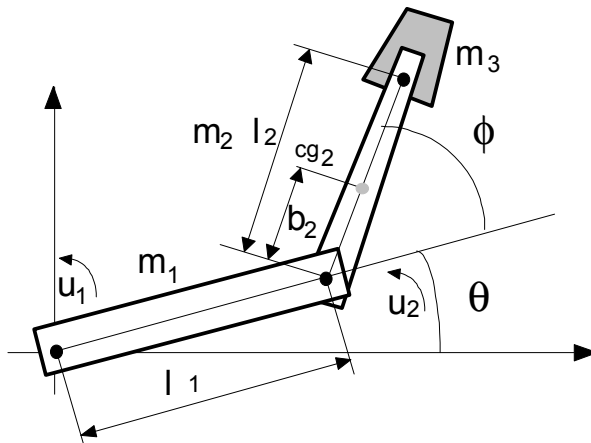


Figure 22. The model of the two-links planar robot

The model parameters are reported in table 10. In this example it has been assumed that the reference shaper directly generates the control torques and for this reason no inner control loop was employed. The desired trajectory to be tracked by the end effector (tip position) is the rectangular trajectory defined (in Cartesian coordinates) by:

$$\begin{cases} x_d = +0.21 \cdot (2-t) & y_d = 0 & 0 \leq t < 4 \\ x_d = -0.42 & y_d = -10.5 \cdot (t-4) & 4 \leq t < 8 \\ x_d = -0.21 \cdot (10-t) & y_d = 0.42 & 8 \leq t < 12 \\ x_d = +0.42 & y_d = 10.5 \cdot (t-16) & 12 \leq t < 16 \end{cases} \quad (28)$$

Length of the first link ( $l_1$ )	0.4 m
Length of the second link ( $l_2$ )	0.25 m
Length $b_2$	0.125 m
Inertia of the 1th. link about its C.G. ( $I_1$ )	1.6 m <sup>2</sup> ·kg
Inertia of the 2th. link about its C.G. ( $I_2$ )	0.43 m <sup>2</sup> ·kg
Inertia of the hand w.r. the hand ( $I_3$ )	0.01 m <sup>2</sup> ·kg
Mass of the second link ( $m_2$ )	15 kg
Mass of the load ( $m_1$ )	6 kg

Table 10. Parameters of the model (Example 2)

In the simulation study, the first joint was fixed at the origin of the reference frame. In figure 23 the desired trajectory (28) and the corresponding velocities are shown. It has to be noted that, due to the discontinuity of the velocities, it is not possible to track the reference trajectory accurately near the discontinuity; anyway, thanks to the predictive action this effect can be reduced also in presence of constraints. The purpose of the MPC shaper is to achieve a small tracking error while fulfilling the following constraints on inputs:

$$-10 \leq u_1(k) \leq 10 \quad -3 \leq \Delta u_1(k) \leq 3 \quad (29)$$

$$-2 \leq u_2(k) \leq 2 \quad -0.3 \leq \Delta u_2(k) \leq 0.3 \quad (30)$$

and the constraints on joint velocities:

$$-0.9 \leq \dot{\theta}(k) \leq 0.9 \quad -0.9 \leq \dot{\phi}(k) \leq 0.9 \quad (31)$$

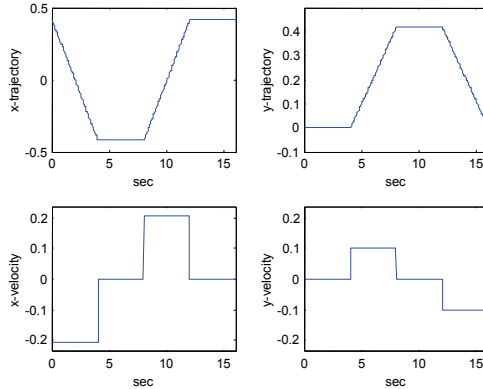


Figure 23. The cartesian trajectory and velocity

All the values are expressed in MKS. The decision variables of the trajectory shaper are the elements of the sequences of the torques that have to be applied to joints, namely:

$$R_s(k_c) = \begin{bmatrix} \Delta u_1(k_c + 1), \Delta u_1(k_c + 2), \dots, \Delta u_1(k_c + N_{u1}) \\ \Delta u_2(k_c + 1), \Delta u_2(k_c + 2), \dots, \Delta u_2(k_c + N_{u2}) \end{bmatrix} \quad (32)$$

The following objective function have been employed:

$$J(k_c) = \sum_{j=1}^{N_U} |x(k_c + j) - x_d(k_c + j)| + |y(k_c + j) - y_d(k_c + j)| + J_1(k_c) \quad (33)$$

where the first term represents the absolute tip position tracking error, while the penalty term  $J_1$  is employed to take into account the constraints on joint velocities and it is defined, according to the penalty approach (34), as:

$$J_1(k) = 5 \left[ \sum_{j=1}^{N_U} \max(|\dot{\theta}(k + j)| - 0.9, 0) + \sum_{j=1}^{N_U} \max(|\dot{\phi}(k + j)| - 0.9, 0) \right] \quad (35)$$

Note that in this example, since the trajectory shaper directly generates the command signals ( $r_s(k) \equiv u(k)$ ), it is not required to explicitly insert the thresholds (36) and (37) in the prediction model, because thresholds (38) and (39) directly act on the control signals  $[u_1, u_2]$ . Some simulations were performed, in the case  $T_c = 2T_s$ , to determine the appropriate values for the shaper parameters, that are reported in table 11. The performance of the shaper has been detailing tested in three different contexts.

$N = 20$	$D = 6$	$S = 2$	$S = 4$
$p_m = 0.1$	$p_c = 0.8$	<b>gen</b> = 10	$T_s = 0.02$
$L = 4000$	$\Delta_x = 1000$	$N_{u1} = 12$	$N_{u2} = 12$
$\Delta U_1^- = -3$	$\Delta U_1^+ = 3$	$U_1^- = -10$	$U_1^+ = 10$
$\Delta U_1^- = -0.3$	$\Delta U_2^+ = 0.3$	$U_2^- = -2$	$U_1^+ = 2$

Table 11. Parameters of the evolutionary shaper (example 2)

*Experiment 1:* In this experiment no constraints were imposed either on inputs or on states; the scope was just to obtain a very accurate tracking and therefore the constraints (40)(41) were disabled. In figure 24a it is reported the trajectory of the tip position, while in figure 24b the corresponding absolute tracking error is shown. Although a very good tracking have been achieved, the constraints on inputs, joint velocities and the corresponding rates are violated. In figure 25 the corresponding shaped input torques and the joint velocities are shown. A resume of the performance is reported in table 12. In this and in the following experiments the computational load was not demanding ( $t_s/t_r=0.86$ ) and could be implemented in real time with the available computing power.

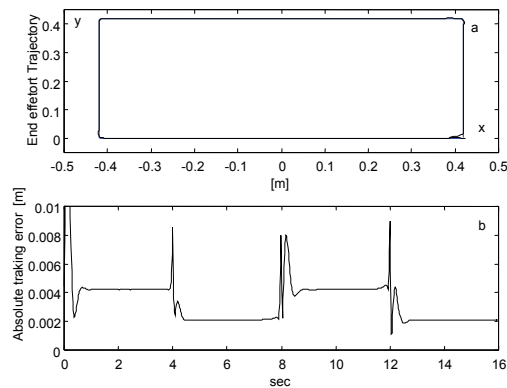


Figure 24. Tracking performance for example 2, experiment 1. Constraints on inputs and angular rates are disabled

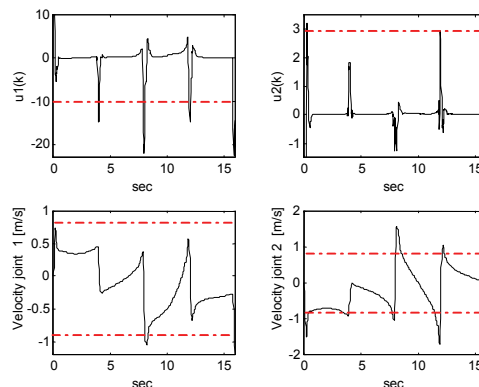


Figure 25. Shaped torques and joints velocities example 2, experiment 1

Case	$ e(k) _{mean}$ (1)	$ e(k) _{max}$ (2)	$ u_1(k) _{max}$ (3)	$ u_2(k) _{max}$ (4)	$ \Delta u_1(k) _{max}$ (5)	$ \Delta u_2(k) _{max}$ (6)	$ \dot{\theta}(k) _{max}$ (7)	$ \dot{\psi}(k) _{max}$ (8)	$t_s/t_r$ (9)	$gen/t_s$ (10)
Exp.1	0.0033	0.0093	22,01	2,93	6	2	1.05	1.76	086	5
Exp.2	0.0056	0.041	10	2	3	0.3	1.383	1.66	086	5
Exp.3	0.0087	0.0463	9,84	1.77	3	0.3	0.881	0.900	086	5

Table 12. Performance of MPC for  $T_c=2T_s$ : example 2. Col.1-2 Computing time, Col.3-4 Maximum control effort, Col.5-6 Maximum control increment, Col.7-8 Maximum angular rate, Col.9 Normalized simulation time, Col.10 Generation evaluated in a sampling interval

*Experiment 2:* In this experiment the constraints (29) and (30) on inputs were activated while the constraint (31) on the velocities remain still inactive ( $J_I=0$ ). Due to torque saturations, an increase in the tracking error is observed particularly in the corners of the trajectory; anyway a good tracking is still achieved. It should be noted that also in this case the constraints on joint velocities are violated. In figure 26a and 26b the trajectory of the tip position and the corresponding absolute tracking error are reported respectively. In figure 27 the shaped input torques and the velocities of joints are shown. A resume of the performance is reported in table 12.

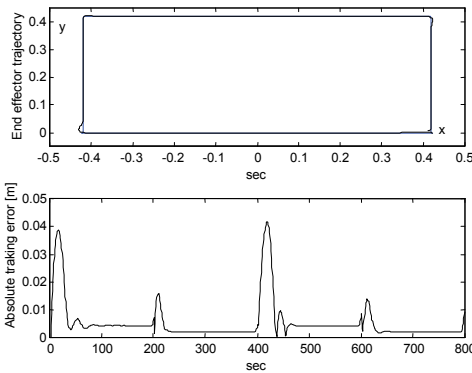


Figure 26. Tracking performance for example 2, experiment 2. Only the constraints on inputs are active

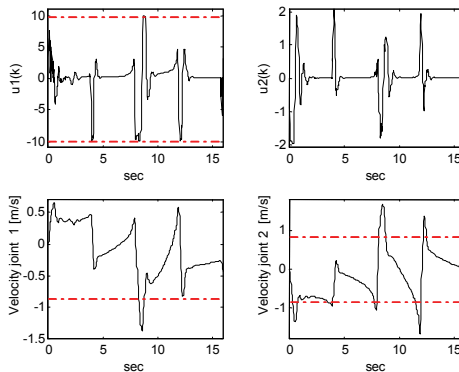


Figure 27. Shaped torques and joints velocities example2, experiment 2

*Experiment 3:* In the last experiment also the constraints (31) on the velocities have been activated ( $J_1 \neq 0$ ). Due to the presence of these additional constraints, it was not possible to maintain a good tracking in some parts along the trajectory. In figure 28a and 28b the trajectory of the tip position and the corresponding absolute tracking errors are reported. In figure 29 the shaped input torques and the joints velocities are shown. It should be observed that in this case all the constraints were fulfilled; in particular the most decisive constraint was that on the second joint velocity. Indeed, in the periods when this signal is saturated the biggest tracking error was observed. The control torques never reached the saturation. In all the three experiments the trajectory shaper has given an appropriate solution to the constrained robot motion planning problem.

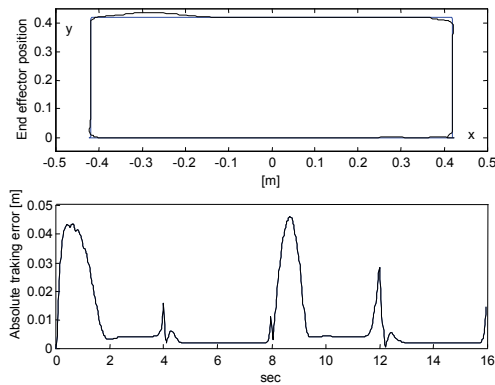


Figure 28. Tracking performance for example 2, experiment 3. Constraints on inputs and angular rates are active

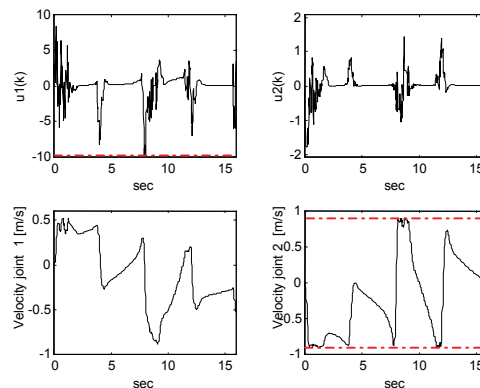


Figure 29. Shaped torques and joints velocities example 2, experiment 3

#### 14. Comparison with conventional optimization methods

The comparison of the Evolutionary optimization for nonlinear MPC with respect to conventional methods was first carried out in (Onnen et Al., 1997); in this study it was showed the superiority of the EA on the branch-and-bound discrete search algorithm. In this study, our intention was to compare the performance provided by the proposed EA with a local gradient-based iterative algorithm. It was implemented a basic gradient steepest descent algorithm and used the standard gradient projection method to fulfill the amplitude and rate constraints for the control commands (Kirk, 1970); the partial derivatives of the index  $J$  with respect to the decision variables were evaluated numerically. The performance comparison was performed on the more challenging two-links planar robot system in the case of the experiment 2. Table 13 reports the performance provided by the two methods in terms of the normalized simulation time, the mean absolute tracking error and the number of prediction required by increasing the number of algorithm cycles per sampling interval  $T_s$ . The Evolutionary optimization gave remarkably better performance in terms of the mean tracking error. This fact clearly puts into evidence that, in this case, the gradient-based optimization gets trapped in local minima, while the EA provides an effective way to prevent the problem. As for the computational power ( $N_{sim}/T_s$ ) required by the two methods to converge to a sub optimal solution within 5% of the stationary value, the EA required 160 simulations while the gradient-based 196; the corresponding normalized simulation times were comparable. The possibility of achieving remarkable performance improvement with a comparable computational cost clearly demonstrates that the EA-based predictive shaper offers a valid alternative to iterative local optimization methods especially in the case of multi modal objective functions.

		EA ( $N_U=12, N=20$ )		GRAD ( $N_U=12$ )		
$n^\circ\text{-cycles}/T_s$	$t_s/t_r$	$ e(k) _{mean}$	$N_{sim}/T_s$	$t_s/t_r$	$ e(k) _{mean}$	$N_{sim}/T_s$
1	0.28	0.0228	20	0.17	0.0311	12
2	0.45	0.0069	40	0.25	0.0135	24
4	0.71	0.0057	80	0.40	0.0122	48
8	1.21	0.0055	160	0.70	0.0115	96
16	3.10	0.0054	320	1.26	0.0104	192
32	4.97	0.0054	640	2.39	0.0103	384

Table 13. MPC performance comparison of EA and gradient-based optimization

#### 15. Conclusions

In this work an online predictive reference shaper has been proposed as a method to improve the tracking performance of robotic systems subjected to constraints on input and state variables. The method allows the choice of an arbitrary multi-objective cost function

that quantifies all the objectives of the desired constrained motion. The global nonlinear optimization is performed in discrete time employing an online specialized Evolutionary Algorithm. The trajectory shaper has been applied in two examples of constrained robotic motion. Both the simulation tests have clearly pointed out the benefits on the performance given by the proposed Predictive Evolutionary Trajectory Shaper; furthermore, the EA algorithm outperformed a conventional iterative gradient-based optimization procedure. It has also been shown that real time implementation can be easily achieved using a not excessively powerful CPU. Finally, it is worth mentioning that the proposed method is sufficiently general and for this reason could be easily extended to other kinds of nonlinear dynamic systems with arbitrary references and constraints.

## 16. References

- J.E Bobrow, S. Dubowsky, J.S.Gobson, (1985), Time Optimal Control of Robotic Manipulator Along Specified path, *Int. J. Robotics Research*, 4(3), 3-17.
- E.F. Camacho, C. Bordons, (1995) *Model Predictive control in the process industry*, (Berlin: Germany, Springer Verlag.
- W.H. Chen, D.J. Balance, J. O'Reilly, (2000), Model predictive control of nonlinear systems: computational burden and stability, *IEE Proceedings Control Theory and Applications*, 147 (4), , 387 -394.
- O. Dahl, L. Nielsen, (1990) Torque limited path following by online trajectory time scaling, *IEEE Trans. Robotics Automation*, 6(5) 554-561
- B. Foegel, (1994) Applying Evolutionary Programming to Selected Control Problems, *Computers Math. Applic.* 27(11) 89-104
- M. Fischer, O. Nelles, R. Isermann, (1998) Adaptive predictive control of a heat exchanger based on a fuzzy model, *Contr. Eng. Practice* 6(2), , 259-269.
- M.L. Fravolini, A. Ficola, M. La Cava (1999) Improving trajectory tracking by feed forward evolutionary multivariable constrained optimization, *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, 985-990.
- M.L. Fravolini, A. Ficola, M. La Cava, (1999) Vibration Reduction by Predictive Evolutionary Trajectory Shaping, *IEEE 38th Conference on Decision and Control*, 5289-5291.
- M.L. Fravolini, A. Ficola, M. La Cava (2000) Intermittent Feedback Strategies for the Predictive Control of a Flexible link: Experimental Results, *Proc. 6<sup>th</sup> IFAC Symposium on Robot Control*, 2000, 391-396.
- C.E. Garcia, D.M. Petit, M. Morari (1989) Model Predictive control: Theory and Practice a survey, *Automatica*, 25(3), 1989, 335-348.
- H. Geniele, R.V. Patel, K. Khorasani (1997), End-point control of a flexible-link manipulator: theory and experiment, *IEEE Trans. Contr. Syst. Technology*, 5(6), 556-570.
- V. Gogos, R.E. King (1996), Evolutionary Predictive Control (EPC), *Computers Chem. Eng.*, 20 suppl. 817-822
- D. Goldberg (1989), *Genetic Algorithms, Search Optimization and Machine Learning*, (Addison Wesley Publishing Company, INC.
- B. Grosman, D.R. Lewin (2002), Automated nonlinear model predictive control using genetic programming, *Computers and Chemical Engineering*, 26, 631-640
- E. Gyurkvcis (1998), Receding horizon control via bolza-type optimization, *Sys. & Contr. Letters*, 35, 195-200.
- D.E. Kirk (1970), *Optimal Control Theory*, Prentice-Hall.

- K. Kumagai, D. Kohli, R. Perez, (1996) Near minim time feedback controller for manipulators using online time scaling of trajectories, *ASME J. Dynamic Sys. Meas. Contr.*, 118, 300-308
- W.K. Lennon, K.M. Passino, (1999) Intelligent Control for brake systems, *IEEE Trans. Contr. Sys. Tech.* 7(2), 188-202.
- W.S. Levine, (1996) *The control handbook*, CRC Press.
- D.C. Liaw, J.T. Huang, (1998) Contact Friction Compensation for Robots Using Genetic Learning, *J. of Intelligent and Robotic Systems*, 23(2/4), 331-349.
- D.A. Linkens, H.O. Nyongesa, (1995) Genetic Algorithms for Fuzzy Control part 2: Online system development and application, *IEE Proc. Control Theory Appl.*, 142(3), 177 - 185.
- R. Luus, (1990) Optimal control by dynamic programming using a systematic reduction of the grid size, *Int. J. Control*, 51, 995-1013.
- D.Q. Maine, H. Michalaska, (1993) Robust receding horizon control of nonlinear systems, *IEEE Trans. Automatic Control*, 38, 1623-1633.
- M. Martinez, J. Senent, X. Blasco, (1998) Generalized predictive control using genetic algorithms, *Eng. Application of Artificial Intelligence*, 11(3), 1998, 355-367.
- R.K Mutha, W.R Cluett, A. Penlidis (1997), Nonlinear model-based predictive control of non affine systems, *Automatica*, 33(5), 907-913.
- C. Onnen , R. Babuska, U. Kaymak, J.M. Sousa, H.B. Verbruggen, R. Iserman, (1997), Genetic algorithms for optimization in predictive control, *Contr. Eng. Practice*, 5(10), 1997, 1363-1372.
- L.L. Porter, K.M. Passino (1995), Genetic adaptive observers, *Eng. Application. of Artificial Intelligence*, 8(3), 261-269.
- L.L. Porter, K.M. Passino, (1998) Genetic Adaptive and Supervisory control, *Int. J. Intelligent Control Sys*, 2(1), 1-41.
- A.S. Rana, A.M.S. Zalzal, (1996) Near time optimal collision used free motion planning of robotic manipulators using evolutionary Algorithm, *Robotica*, 14 621-632
- E. Ronco, T. Arsan, P.J. Gawthrop, D.J. Hill, (1999) A globally valid continuous-time GPC through successive system linearisation, *Proc. 38th IEEE Conf. Decision and Control*, 726-731.
- E. Ronco, T. Arsan, P.J. Gawthrop, (1999) Open loop intermittent feedback control: practical Continuous-time GPC, *IEE Proceedings* 146(5), 426-434.
- J.A. Roubos, S. Mollow, R. Babuska, H.B. Verbruggen (1999), Fuzzy model based predictive control using Takagi-Sugeno models, *Int. J. Approximate Reasoning* , 22(1-2), 1-30.
- Z. Shiller, W.S. Chang (1995), Trajectory preshaping for high speed articulated systems, *J. of Dynamic Sys. Meas. and Control*, 117, 304-310.
- P.O.M. Sokaert, D.Q. Maine, J.B. Rawings (1999), Suboptimal model predictive control (feasibility implies stability), *IEEE Trans. Automatic Control*, 44(3), 648-658.
- Z. Shin, N.D. Mc Kay, (1985) Minimum time control of robotic manipulators with geometric path constraints, *IEEE Trans. Robot. Automation*, 30 531-541
- S.C. Shin, S.B. Park, (1998), GA based predictive control for nonlinear processes, *Electronic Letters*, 34(20), , 1980-1981.
- B.J Song, A.J. Koivo, (1999), Nonlinear predictive control with application to manipulator with flexible forearm, *IEEE Trans. Ind. Electr.*, 46(5), 923 -932.

- 
- J.M. Sousa, M.S. Setnes, (1999), Fuzzy predictive filters in model predictive control, *IEEE Trans. Ind. Electr.* 46(6), 1225 -1232.
- S. Sun, A.S. Morris, A.M.S. Zalzal, (1996) Trajectory planning of multiple coordinating robots using genetic algorithms, *Robotica*, 14 227-234
- L. Yao, W. Sethares, (1994) Nonlinear parameter estimation via the genetic algorithm, *IEEE Trans. on Sign. Proc.*, 42(4), 927 -935.