Real-Time Population Based Optimization for Adaptive Motion Control of Robot Manipulators

Mehmet Bodur, *Member, IEEE*

***Abstract*— A population based real-time optimization method for tuning dynamic position control parameters of robot manipulators has been proposed. Conventionally, the positional control is achieved by inverse dynamics feedforward and PID feedback controllers. The proposed method allows to tune the PID controller parameters using population based optimization methods to minimize the error while tracking a repeated desired trajectory on real-time. The stability of the system is granted by switching the inappropriate settings to a stable default using a real-time cost evaluation function.**

The proposed tuning method is tested on a two-joint planar manipulator with Cross-Entropy optimization, and on a planar inverted pendulum both with Cross Entropy, and Differential Evolutionary search methods. The test results indicated that the proposed method improves the settling time and reduces the position error over the repeated paths for both population based evolutionary optimization.

***Index Terms* — adaptive control, PID tuning, real-time CE optimization, real-time evolutionary optimization, real-time DE optimization.**

I. Introduction

Robot manipulators are commonly employed in repetitive tasks in the industry for the reduction of production costs, enhancement of precision, quality and productivity while having greater flexibility than other specialized machines as well as in hazardous environmental conditions such as in radioactive, toxic zones or where a risk of explosion exists, or spatial and submarine applications. The short-term projections show that assembly tasks will continue to be the main applications of robot manipulators [1].

Robot manipulators are mainly positioning devices with multiple degrees-of-freedom (DoF). Dynamic position control of robot manipulators is a well known problem in control engineering due to the multi-input-multi-output nonlinear behaviour of their equation of motion. There are a large number of excellent survey books and articles on the control of the manipulators [1]-[5]. In a typical position control problem, the plant is characterized by the dynamic equation of motion of the manipulator, which describes the motion of the joint displacements corresponding to the applied joint torques. The controller is designed to generate appropriate joint torques to track a desired task space trajectory, or equivalently to track the corresponding desired joint space trajectory. The conversion of trajectories from the task space to the joint space requires kinematic modeling of the robot manipulator. The equation of motion is obtained by one of the methods such as Newton-Euler, Lagrangian, and their backward and forward recursive applications.

The decentralized PD and PID control are the conventional methods for position control of the manipulator joint displacement. PD control can achieve global asymptotic stability in positioning the end-point in the absence of gravity. The integral action of the PID control further compansates any positional offset due to gravitational forces [5]. Several methods were proposed in literature to obtain suitable propotional, integral and derivative controller gain settings: K_p, K_i and K_d. In absence of friction, the gains may be selected using a Lyapunov function candidate and LaSalle's Invariance Principle [5]. Adaptive PID settings were also proposed for obtaining proper local gain settings based on plant identification and pole placement techniques [6].

Population-based metaheuristics is one of the important branches of the metaheuristic methods to search solutions of difficult optimization problems. Even though metaheuristic methods have drawbacks such as sensitivity to parameters and lack of strong proof of convergence, they are still preferred for several reasons: They do not require special conditions for the proprieties of the objective function and constraints, they are suitable for both continuous and combinatorial problems, and they are suitable as well to multiobjective optimization applications. Population-based methods exhibit further advantages compared to the neighborhood metaheuristics. They are less sensitive to misbehaving paths of certain individual solutions, and they increase the probability of attaining the global optimum by employing information on the surface of an objective function [7].

Population-based algorithms are difficult to apply to real-time optimization of robotic control problems, because of the difficulty to keep the trajectory in tolerance for all tested population members. For a reinforcement learning application Lin restricted the population only to stable parameters using a Lyapunov function [8]. This study proposes a simpler solution by real-time monitoring the tracking error. The stability problem is tackled by switching the unstable control parameters to a stable parameter set if the continuously monitored error score exceeds a tolerable threshold. The proposed method is demonstrated on two population based search algorithms: the Differential Evolution and the Cross-Entropy methods.

Dr. Mehmet Bodur is with the Computer Engineering Department at Eastern Mediterranean University, G.Magusa, TRNC, Mersin-10 Turkey. (e-mail: mehmet @ ieee . org).

Cross-Entropy (CE) Method is a statistical population based information theoretic method of inference about an unknown probability density from a prior estimate of the density and new expected values [9], [10]. CE Method was further developed as a combinatorial optimization tool to obtain the optimal solutions for rare-event problems and optimization of the scalar functions [10], [11]. It is one of the optimization methods with the proven convergence to the optimal parameters similar to Monte-Carlo and the Simulated Annealing methods.

Evolutionary Algorithms (EAs) are nature inspired population based optimization methods for both continuous variable, and combinatorial problems. The Differential Evolution is a population based search method developed from Genetic Algorithms (GAs), which provide evolution of the best solution by survival of the fittest. GAs are search algorithms inspired of natural selection and genetics to search better solutions in a solution space. A simple genetic algorithm is typically composed of three operators: reproduction, crossover and mutation [12]. The beginning of the Differential Evolution (DE) algorithm was Genetic Annealing developed by K. Price [13], [14]. Unlike to the GAs, DE employs only a simple and fast linear operator, differential mutation, which uses the differences between individuals instead of crossover and mutation operators [7], [13]. In continuous domain multimodal problems, DE finds the global best faster than GAs [7], [15]. This success resides in the manner of the trial individual creation.

This article is organized as follows: Section II presents the kinematics, dynamic modeling and control of a robot manipulator. Section III and IV introduces the Cross-Entropy and the Differential Evolution methods for optimization of the controller gains. Section V describes the proposed real-time population based optimization algorithm with its goal. Section VI and VII contains the experimental details and the results of the proposed parameter switching on a two-link planar manipulator, and on an inverted pendulum system. Section VIII concludes the article.

II. KINEMATICS DYNAMIC AND CONTROL

The forward kinematics relation maps the joint space position q to the end-effector pose

$$p = f_0(q) \tag{1}$$

using homogenous link-frame coordinate transformation matrices. Denavit-Hartenberg (D-H) convention provides the D-H transformation matrix A_k that maps a coordinate at k^{th} frame to the $(k-1)^{\text{st}}$ frame [15]. In D-H convention, the frame axis ${}^{k-1}z$ is assigned along the axis of rotation of the k^{th} link. If the joint is translational, ${}^{k-1}z$ is assigned along the movement axis of the k^{th} link. θ_k is the joint rotation from ${}^{k-1}x$ to ${}^{k}x$. The common normal from ${}^{k-1}z$ to ${}^{k}z$ is called a_k. The distance from a_{k-1} to a_k is denoted by d_k. The twist angle from ${}^{k-1}z$ to ${}^{k}z$ is called α_k. The transformation function from k^{th} to $(k-1)^{\text{st}}$ frame is obtained by a chain of rotations and translations

$${}^{k-1}A_k = \text{Rot}(z_{k-1}, \theta_k)\,\text{Trans}_{n-1}(a_k, 0, d_k)\,\text{Rot}(x_{k-1}, \alpha_k)$$

$${}^{k-1}A_k = \begin{bmatrix} \cos\theta_k & -\sin\theta_k\cos\alpha_k & \sin\theta_k\sin\alpha_k & a_k\cos\theta_k \\ \sin\theta_k & \cos\theta_k\cos\alpha_k & -\cos\theta_k\sin\alpha_k & a_k\sin\theta_k \\ 0 & \sin\alpha_k & \cos\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The *Lagrangian* provides a systematic method to obtain the equation of motion of a multi-degree-of-freedom open chain mechanism. The Lagrangian function L is defined by $L = K - P$, where K is the kinetic energy, and P is the potential energy of the analyzed system. The partial derivatives of L provide a simple means of calculation for the joint force and torques. The derivatives of the D-H transformation matrices A_i can be expressed by

$$\frac{\partial A_i}{\partial q_i} = Q_i A_i , \tag{3}$$

where $Q_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ for revolute, and $Q_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ for translational joints. This property provides a convenient form for the derivatives of the link transformation matrices ${}^0T_i$ with respect to the j^{th} generalized-joint displacement variables q_j and q_k [15].

$$U_{ij} = \frac{\partial {}^0T_i}{\partial q_j} = \frac{\partial (A_1A_2 \ldots A_j \ldots A_i)}{\partial q_j}$$
$$= A_1A_2 \ldots Q_jA_j \ldots A_i ; \qquad j \le i . \tag{4}$$

and

$$U_{ijk} = \frac{\partial {}^0T_i}{\partial q_j \partial q_k} = \frac{\partial (A_1A_2 \ldots A_j \ldots A_i)}{\partial q_j \partial q_k}$$
$$= A_1A_2 \ldots Q_jA_j \ldots Q_kA_k \ldots A_i ; \quad j \le i, k \le i . \tag{5}$$

Linear or revolute, the kinetic energy of a mass dm_i located at the position ${}^ir_{dm}$ and moving with the i-th coordinate frame is

$$d\,k_i = \tfrac{1}{2}\, d\, m_i\, {}^0v_i^{\mathrm{T}}\, {}^0v_i , \tag{6}$$

where ${}^0v_i$ denotes the velocity of mass m_i. The total kinetic energy of the link requires the integration of $d\,k_i$ over the complete mass of the i-th link.

$$k_i = \int_{m_i} dk_i = \tfrac{1}{2}\text{Trace}\left[\left(\Sigma_{p=1}^{i}\, \Sigma_{r=1}^{i} U_{ip}\, J_{pi}\, U_{ir}^{\mathrm{T}}\, \dot{q}_r\, \dot{q}_p\right)\right], \tag{7}$$

where $J_{pi} = \int_{m_i} {}^ir_{dm}\, {}^ir_{dm}^{\mathrm{T}}\, dm_i$ is the pseudo-inertia matrix of the i^{th} link. The potential energy of i^{th} link due to the gravitational field is expressed using the mass m_i, the center of mass ${}^ic_m$ and the gravitational acceleration vector g_a .

$$p_i = -\, m_i\, g_a^{\mathrm{T}}\, {}^0T_i\, {}^ic_m . \tag{8}$$

Finally, the Lagrangian of the complete system is obtained

$$L = K - P$$
$$= \tfrac{1}{2}\Sigma_{i=1}^{n}\, \Sigma_{k=1}^{i}\, \Sigma_{r=1}^{i}\, \text{Trace}\left(U_{ik}\, J_{pi}\, U_{ir}^{\mathrm{T}}\right)\, \dot{q}_r\, \dot{q}_p$$
$$- \Sigma_{i=1}^{n}\, m_i\, g_a^{\mathrm{T}}\, {}^0T_i\, {}^ic_m \tag{9}$$

and its derivatives give the generalized joint-force at the joint-j.

$$\tau_j = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} , \tag{10}$$

$$\tau_j = \Sigma_{i=j}^{n}\, \Sigma_{k=1}^{n}\, \text{Trace}\left(U_{ik}\, J_{pi}\, U_{ij}^{\mathrm{T}}\right)\, \ddot{q}_k$$

$$+ \sum_{k=1}^{n} \sum_{r=1}^{n} \text{Trace}\left(U_{ikr}\, J_{pi}\, U_{ij}^{\mathrm{T}} \right) \dot{q}_k\, \dot{q}_r$$

$$- \sum_{i=j}^{n} m_i\, g_a^{\mathrm{T}}\, U_{ij}\, {}^{i}c_m \quad . \tag{11}$$

Further, (11) can be organized to Uicker-Kahn form.

$$\tau_i = \sum_{i=j}^{n} D_{ij}\, \ddot{q}_j + \sum_{i=j}^{n} \sum_{k=1}^{n} C_{ijk}\, \dot{q}_j\, \dot{q}_k + g_i. \tag{12}$$

where

$$D_{ij} = \sum_{i=j}^{n} \text{Trace}(U_{ik}\, J_{pi}\, U_{ij}^{\mathrm{T}}) \;;$$

$$C_{ijk} = \sum_{r=1}^{n} \text{Trace}(U_{ikr}\, J_{pi}\, U_{ij}^{\mathrm{T}}),$$

and

$$g_j = - \sum_{i=j}^{n} m_i\, g_a^{\mathrm{T}}\, U_{ij}\, {}^{i}c_m \;, \tag{13}$$

are the inertial, coriolis, and gravitational terms of the equation of motion. The equation is also written in a matrix form [5]

$$\tau = D(q)\, \ddot{q} + C(q, \dot{q})\, \dot{q} + g(q) \quad . \tag{14}$$

A constant field dc-motor converts the total control action by an almost linear relation of rotor current i_{rotor} to the joint torque $\tau = k_i \cdot i_{rotor}$. Accordingly, without loss of generality, the torque coefficient k_i is assumed k_i=1 so that the joint torques are determined directly by the control action [1].

A fully actuated rigid manipulator has an independent control input for each degree-of-freedom, which simplifies its decentralized control. Robots with flexible links or joints have control problems and which may require singular perturbation and two-time-scale control techniques [5], [6].

The equation of motion is highly nonlinear to apply independent PID control directly to each joint motion. The usual method is to apply the expected joint torques for the desired trajectory by a feedforward control, and correct any deviation from the trajectory by the feedback loop of the PID control. The feedforward control action τ_c is obtained by the sum of the anticipated gravitational and inertial terms, so that all coriolis forces remains as a disturbance to the feedback loop

$$\tau_c = D_c(q_d)\, \ddot{q}_d + g_c(q_d) \;, \tag{15}$$

where D_c and g_c are the inertial coefficients and gravitational torques of the anticipated equation of motion; q_d is the desired trajectory in joint space. The discrepancy of the anticipated equation of motion from the actual dynamics of the manipulator is expected to contribute to the disturbance torques, and thus a feedback control loop is inevitable for the stability of the control action.

The difference of the desired position $q_d(t)$ and the measured position $q_f(t)$ is called the displacement error $e(t)$ of the control system. A proportional gain provides main correction of the error, an integral gain provides correction of the offset, and a derivative gain provides faster response of the controlled system.

$$\tau_f = K_P e(t) + K_I \int_{t_0}^{t} e(t)\, dt + K_D \dot{e}(t) \tag{16}$$

Assuming that the anticipated feedforward control law can be predicted by the mechanical properties of the manipulator, the PID parameters remain to be determined for the optimum tracking of the specified joint space trajectory. The following two sections contain the cross entropy and differential evolution methods, which are employed to find the optimum PID gains. In the optimization of the PID gains, The closed loop control system is considered as a stochastic system with the controller parameters.

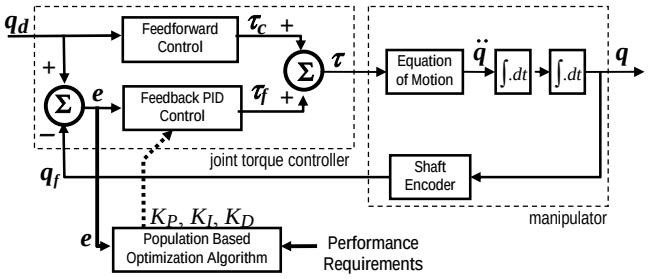

Fig. 1. Position Control System.

III. Cross Entropy Method

Cross Entropy method is a population based optimization algorithm that utilize the scores of the trial runs optimal in the information rhetoric meaning.

Let X be the set of real valued states, and let the scores S be a real function on X. CE-method targets to find the minimum of S over X, and the corresponding states $x^* \in X$ satisfying the minimum

$$\gamma^* = S(x^*) = \min_{x \in X} S(x) \,, \tag{17}$$

by employing importance sampling and minimizing the cross entropy between the samples of a family of succeeding probability mass function $f(\text{-}, \upsilon_k)$. A naive random search can find an expected value for x^* and determine γ^* with probability 1, if some of the scores $S(x)$ for the random states x can satisfy the minimum. However, methods like Crude Monte-Carlo requires considerable computational effort because it uses homogeneously distributed random states in searching x^*. CE method provides a methodology for creating a sequence of vectors x_0, x_1 ... and levels γ_0, γ_1, ... such that γ_0, γ_1, ... converges to γ^* and x_0, x_1 , ... converges to x^*.

Define a collection of functions $\{ H(\text{-} ; \gamma) \}$ on X, via

$$H(x; \gamma) = I_{\{S(x^{(i)}) > \gamma\}} = \begin{cases} 1, & \text{if } S(x) \le \gamma \\ 0, & \text{if } S(x) > \gamma \end{cases} . \tag{18}$$

for each threshold $\gamma \in R$, and $x \in X$. Let $f(\text{-}; \upsilon)$ be a family of probability mass function (pmf) on X, parameterized by a real valued vector $\upsilon \in R$. Consider the probability measure under which the state x satisfies the threshold γ

$$\ell_\upsilon(\gamma) = P_\upsilon (S(X) \ge \gamma)$$

$$= \sum_x H(x; \gamma)\, f(\text{-}; \upsilon) = E_\upsilon H(X; \gamma) \,, \tag{19}$$

where E_υ denotes the corresponding expectation operator. It converts the optimization problem to an associated stochastic rare event problem. Using the Importance Sampling (IS) simulation, the unbiased estimator of ℓ with the random sets of states $x^{(1)}$, ... taken from different independent pmf $f(x, \upsilon)$ and $g(x)$ is

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(x^{(i)}) > \gamma\}}\, W(x^{(i)}) \;, \tag{20}$$

where $W(x) = f(x, \upsilon)/g(x)$ is the likelihood ratio. Searching the optimal importance sampling density $g^*(x)$ is problematic,

since determination of $g^*(x)$ requires ℓ to be known. Instead, the optimum tilting parameter υ^* of a pmf $f(\mathrm{x}, \upsilon)$ reduces the problem to scalar case.

The tilting parameter υ can be estimated by minimizing Kullback-Leibler "distance" (also called cross entropy) between the two densities $f(x)$ and $g(x)$

$$CE(f, g)=\int f(x)\ln\frac{f(x)}{g(x)}\,dx\,. \tag{21}$$

After reducing the problem to tilt parameter υ, the cross-entropy between $f(x, \upsilon)$ and the optimal distribution $f(x, \upsilon)$ is described by

$$CE(\upsilon,\upsilon^*) := E\left(\frac{I_{\{S(x)\geq\gamma\}}}{c}\ln\frac{I_{\{S(x)\geq\gamma\}}f(x,\upsilon^*)}{c\,f(x,\upsilon)}\right), \tag{18}$$

where $c = \int I_{\{S(x)\geq\gamma\}} f(x)\,dx$. The optimal solution υ^* is obtained by solving υ^* from $\min_{\upsilon} CE_N(\upsilon)$, where

$$CE_N(\upsilon) = E_{\upsilon 1}=\frac{1}{N}\Sigma_{i=1}^{N}\ I_{\{S(x^{(i)})>\gamma\}}\ W(x^{(i)},\upsilon,\upsilon^*)\,. \tag{22}$$

For example, if all γ is equal to γ^*, in that case $\ell_{\upsilon}(\gamma)=f(x^*;\gamma)$, which typically would be a very small number. $\ell_{\upsilon}(\gamma^*)$ is estimated by Importance Sampling with $\upsilon_k=\upsilon$

$$\upsilon^* = \underset{\bar{\upsilon}}{\operatorname{argmax}}\ \Sigma_{i=1}^{N}\ E_{\upsilon}\,H(x\,;\gamma)\ln f(x\,;\,\bar{\upsilon})\,, \tag{23}$$

and using $x^{(i)}$, which are generated from pmf $f(x\,;\,\bar{\upsilon}\,)$ by

$$\underset{\bar{\upsilon}}{\operatorname{argmax}}\ \Sigma_{i=1}^{N}\ H(x^{(i)}\,;\gamma)\ln f(x^{(i)}\,;\,\bar{\upsilon})\,. \tag{24}$$

However, the estimator of υ^* is only valid when $H(x^{(i)}\,;\gamma)=1$ for enough samples.

The CE-algorithm consist of the two phases: 1) Generate random samples using $f(\text{-};\upsilon_k)$, and calculate the estimate of the objective function; 2) Update $f(\text{-};\upsilon_k)$ based on the data collected in the first phase via the CE method. The main CE Algorithm is

1) Initialize k=0, and choose initial parameter vector $\upsilon_0=\upsilon$.

2) Generate a sample of states $x^{(1)}$, ..., $x^{(M)}$ according to the pmf $f(\text{-};\upsilon_k)$.

3) Calculate the scores $S(x^{(i)})$ for all i, and order them from the biggest to the smallest: $s_1, \geq ... \geq s_N$. Define $\gamma_k = s_{[\rho N]}$, where $[\rho N]$ is the integer part of ρN, so that $\gamma_k > \gamma$, set $\gamma_k = \gamma$, this yields a reliable estimate by ensuring the target event is temporarily made less rare for the next step if the target is not reached by at least a fraction of the samples.

4) For j=1, ..., 5, let

$$\upsilon_{k+1,j}=\frac{\sum_{i=1}^{M} I_{\{S(x^{(i)})>\gamma_k\}}W(x^{(i)};\upsilon,\upsilon_k)x_j^{(i)}}{\sum_{i=1}^{M} I_{\{S(x^{(i)})>\gamma_k\}}W(x^{(i)};\upsilon,\upsilon_k)}\,. \tag{25}$$

5) Increment k and repeat steps 2, to 5, until the parameter vector has converged. Let υ^* denote the final parameter vector.

6) Generate a sample $x^{(1)}$, ..., $x^{(N)}$ according to the pmf $f(\text{-};\upsilon_k)$ and estimate ℓ via the IS estimate

$$\hat{\ell} = \frac{1}{N}\Sigma_{i=1}^{N} H(x^{(i)})\,W(x^{(i)}\,;\,\upsilon\,,\,\upsilon^*)\,. \tag{26}$$

Step (6) is not necessary in optimization of the scalar cost functions, since the goal is to minimize the cost rather than calculation of the probability of a rare event.

IV. Differential Evolutionary Optimization

For a given problem described by a parameter space X and an objective function $S(x)$ defined on this space, an initial population of randomly selected individuals $\{ x^{(i)} \mid i=1,...,M \}$ is constructed and each individual is evaluated according to the objective function. The objective function $S(x^{(i)})$ value for an individual $x^{(i)}$ is referred to its fitness and it is a relative measure of functional performance specifying how well the individual satisfies the optimization criteria. In Genetic Algorithms, this population is evolved over a number of generations, by a process based on three genetic operators: *reproduction*, *crossover*, and *mutation*. Reproduction is a process in which individuals are selected to a mating pool depending on their fitness. A higher probability of generating offspring in the next generation is assured by assigning a higher mating probability to that individual with a higher fitness value. This imitates the natural selection, in which the fittest tend to survive and reproduce, thus propagate their genetic material to future generations.

Differential evolution uses a special kind of mutation as the main search mechanism where a greedy selection approach is employed to direct the search toward the promising regions of the search space. The fundamental difference between GAs and DE is that GAs rely on crossover, while DE uses mutations of the differences of the parameter vectors as the primary search mechanism.

1. **Generate initial population**
2. **Evaluate population**
3. **Repeat**
   - **Select Parents for Reproduction**
   - **Differential Mutation**
   - **Evaluate new population**
   - **Substitute old population**

   Until (Termination Conditions are satisfied).

Fig. 3. The Differential Evolution template

DE search strategies are classified in four groups [7]. This application used the *rand3* search strategy, where the new individuals are generated from the combination of mutually distinct three random individuals within the population. Differential mutation expands the search space by adding the weighted difference of two randomly chosen vectors to a third one. That is, in k^{th} generation of an M-vector population, for a given solution $x^{(i,\,k)}$ (i =1, … , M , and k=1, … , G_{max}), three vectors $x^{(r1,\,k)}$, $x^{(r2,\,k)}$, $x^{(r3,\,k)}$ are uniformly randomly selected from the population such that the indices i, r_1, r_2 and r_3 are mutually distinct. A donor vector is formed adding the weighted difference $F \times (x^{(r2,\,k)} - x^{(r3,\,k)})$ to $x^{(r1,\,k)}$

$$v^{(i,k+1)} = x^{(r1,k)} + F\cdot(x^{(r2,k)} - x^{(r3,k)}) \tag{27}$$

where $F\in(0,2]$ is called the mutation constant. The number of donor vectors generated this way is equal to the population size [13]. Other variations of the mutation are also proposed in literature [21]. Mutation is followed by a non-uniform recombination procedure in which a trial vector $z^{(i,k+1)}$ is generated as

$$z_j^{(i,k+1)} = \begin{cases} x_j^{(i,k+1)} & \text{if rand} < CR \text{ or } j = \text{irand}(1, N) \\ x_j^{(i,k)} & \text{otherwise} \end{cases} \quad (28)$$

where CR is the crossover rate; rand generates a random number in [0,1); irand(p,q) generates an integer number in the interval,
[p, q]; and z_j ($j \in \{1, \ldots, N\}$) denotes the j^{th} entry of the trial parameter vector z. Finally, the greedy selection procedure determines the i^{th} offspring as

$$x^{(i,k+1)} = \begin{cases} z^{(i,k+1)} & \text{if } S(x^{(i,k+1)}) < S(x^{(i,k)}) \\ x^{(i,k)} & \text{otherwise} \end{cases} . \quad (29)$$

This strategy allows both generation of trial vectors $z^{(i,k+1)}$ and update of $x^{(i,\,k+1)}$ in the same pass of the loop without a loss of the major notion of Differential Evolution [7].

V. Optimization Criteria and Test Method

The optimization of motion control parameters of a manipulator using reinforcement, genetic, or other population methods comes across important problems in testing the generated parameter sets. These methods are mostly applied on simulations, because the randomly generated unrestricted population may contain unstable controller settings, which may cause failure, and even breakdown of the robot system. The problem can be solved by testing stability properties of the parameters before applying them to the dynamic control system, restricting the search space to a narrow region that guarantees the stability of the system [8].

This paper proposes a simpler, and a more flexible method for the application of population based optimization methods. The proposed method assumes that 1) a real-time tracking error measure $J(t, p, p_d(t))$ and a cost threshold J^* of the system is available, 2) a stable control parameter set x_S exists, that guarantees the stability along the trajectory for a deviation in the tolerable boundaries of the threshold J^*. The population based algorithm generates a population of controller parameter sets, $\{x^{(1,k)}, \ldots, x^{(M,k)}\}$. The evaluation module sets the control parameters of the PID controllers $K_{PID} = x^{(i,k)}$; i=1,...,M, and evaluates the real time cost of $x^{(i,k)}$ along a test period. If the cost function $J(t, p(t, x^{(i,k)}), p_d(t))$ remains less than J^* along the complete test period, a cost score $s^{(i)}$ is calculated from the cost criteria $S(x^{(i)}, p_d)$, which may include any absolute, derivative, integral and quadratic terms of tracking error. Whenever the real-time cost function $J(t)$ exceeds a prespecified critical value J^*, K_{PID} is switched to the stable control parameter set, $K_{PID} = x_S$, to reduce the tracking error, and prepare the system for the next test period. In this case, a high cost score is returned for that tested unsuccessful parameter set. The following two examples of real-time search algorithms demonstrate the proposed method on CE and DE search algorithms.

Real-time CE Algorithm to search controller parameters:

1) At initialization, specify a stable controller parameter set x_S, a real-time cost function $J(t, p(t,x), p_d(t))$, a cost threshold J^*, and a cost score function $S(x, p_d(t))$, where p_d is the periodic desired trajectory, and $p(t,x)$ is the actual trajectory obtained with the control parameter vector x. Initialize the generation count k=0, and the convergence count j=1, and choose initial $\upsilon_k = x_S$. Specify initial γ_k and elite ratio $\rho \in (0, 1)$ so that "the ratio of stable parameter vectors to all parameter vectors" is higher than ρ. This ensures reliable estimates of the target. Select the coefficients $0 < \alpha \leq 1$ and $0 < \beta \leq 1$ to update υ_{k+1} and γ_{k+1} smoothly. Pick a suitable population size M, and termination condition on number of generations G_{max}.

2) Generate a population of states $\{x^{(1)}, \ldots, x^{(M)}\}$ according to the pmf $f(\gamma_k;\upsilon_k)$. For each $x^{(i)}$, at the beginning of a trajectory period $p_d(.)$ switch the controller settings to $x^{(i)}$, and during the period, evaluate $J(t, p(t, x^{(i)}), p_d)$. Whenever $J(t)$ exceeds J^* switch the controller settings to x_S, so that system stays stable, and the trajectory deviation remains within tolerable limits. At the end of the trajectory period, calculate the cost score of the sample, $s^{(i)} = S(x^{(i)}, p_d)$.

3) Order the scores from the biggest to the smallest, i.e., $s_1, \geq \ldots \geq s_N$. Use $\gamma_k = s_{[\rho N]}$, where $[\rho N]$ is the integer part of ρN, to select the elite subset of population. Estimate υ'_{k+1}, and γ'_{k+1} from the mean and standard deviation of the parameters in the elite subset. Update

$$\upsilon_{k+1} = \alpha\, \upsilon'_{k+1} + (1-\alpha)\upsilon_k,$$
$$\beta_m = \beta - \beta\,(1-\tfrac{1}{k})\, q_S, \text{ and}$$
$$\gamma_{k+1} = \beta_m\, \gamma'_{k+1,j} + (1-\beta_m)\, \gamma_k . \quad (30)$$

If υ_{k+1} converges to υ_k, increment j=j+1; else reset j=0.

4) Repeat the steps (2) and (3) until j=5, (i.e., the last 5 iterations converge to the target x^*) or the generation count k exceeds G_{max}. At the end of each looping, update the default parameter set x_S by the lowest scoring $x^{(i,k)}$ of that generation.

Real-time DE Algorithm to search controller parameters:

1) Specify a stable controller parameter set x_S, a real-time cost function $J(t, p(t,x), p_d(t))$, a cost threshold J^*, and a cost score function $S(x, p_d(t))$. Initialize the generation count k=0, choose a population size M, maximum number of generations G_{max}, and differentiation weight F. Generate an initial population $\{x^{(1,0)}, \ldots, x^{(M,0)}\}$ with a mutation probability MP. Evaluate the scores $s^{(i,0)} = S(x^{(i,0)}, p_d)$ for the desired path p_d.

2) For each individual $x^{(i,k)}$ of the k-th generation, form a test individual $z^{(i,k+1)}$ according to (27) and (28). At the start of the period of p_d switch the controller settings to $z^{(i,k+1)}$. During this period evaluate $J(t, p(t, z^{(i,k+1)}), p_d)$. Any time $J(t)$ exceeds J^* switch the PID settings to x_S to stabilize the system. Calculate the cost score of the test sample, $s'^{(i,\,k+1)} = S(z^{(i,\,k+1)}, p_d)$. If $s'^{(i,\,k+1)} < s^{(i,\,k)}$ select $z^{(i,\,k+1)}$ for $x^{(i,\,k+1)}$ as given in (29).

3) Repeat step (2) until generation count k exceeds G_{max}. At each looping, update the default parameter set x_S by the lowest scoring $x^{(i,k)}$ of that generation.

The population based search methods provide a higher degree of flexibility in specifying the cost function compared to many other controller parameter optimization techniques which

are based on plant identification or state estimation techniques. For example, in the following simple demonstration, the cartesian absolute path deviation ($e(t)= |p(t)-p_d(.)|$) is used instead of using the conventional trajectory tracking error ($e(t)= q(t)-q_d(t)$) in joint space.

VI. Simple 2-link (2R) Demonstration

Consider the two link manipulator with two revolute joints shown in Fig. 1. The length of the links L_1 and L_2 are respectively b_1, and b_2. The masses m_1 and m_2 are homogenously distributed along the links L_1 and L_2. Coordinate frames ${}^0T$, ${}^1T$ and ${}^2T$ are assigned for the base, L_1, and L_2 by applying D-H convention as shown in Fig-1. The symbolic equation of motion for this planar manipulator is derived in symbolic toolbox of MATLAB using Lagrangian formulation

$$\tau_1 = \left(\tfrac{1}{3} m_1 b_1^2 + \tfrac{1}{3} m_2 b_1^2 + m_2 b_1 b_2 C_2 + m_2 b_1^2\right) \ddot{q}_1 + \left(\tfrac{1}{3} m_1 b_1^2 - m_1 b_1 b_2 - \tfrac{1}{2} m_1 b_1^2 C_2 + m_1 b_2^2 + m_1 b_1 b_2 C_2\right) \ddot{q}_2 - m_2 b_1 b_2 S_2 \dot{q}_1 \dot{q}_2 - \tfrac{1}{2} m_2 b_1 b_2 S_2 \dot{q}_2^2 + \tfrac{1}{2} g_a \left(b_1 m_1 C_1 + b_2 m_2 C_{12} + 2 b_1 m_2 C_1\right), \quad (24)$$

$$\tau_2 = +\left(\tfrac{1}{3} m_1 b_1^2 - m_1 b_1 b_2 - \tfrac{1}{2} m_1 b_1^2 C_2 + m_1 b_2^2 + m_1 b_1 b_2 C_2\right) \ddot{q}_1 + \tfrac{1}{3} m_2 b_2^2 \ddot{q}_2 + \tfrac{1}{2} m_2 b_1 b_2 S_2 \dot{q}_1^2 + \tfrac{1}{2} g_a m_2 b_2 C_{12}, \quad (25)$$

where C_i, and S_i, denote $\cos(q_i)$ and $\sin(q_i)$; C_{ij}, and S_{ij}, denote $\cos(q_i + q_j)$ and $\sin(q_i + q_j)$. In the simulation, the equation of motion is integrated for $m_1 = 5$ kg, $m_2 = 3$ kg, $b_1 = 0.5$ m and $b_2 = 0.4$ m, reducing the simulation model to

$$\tau_{s,1} = \left(\tfrac{199}{150} + \tfrac{3}{5} C_2\right) \ddot{q}_1 + \left(\tfrac{13}{60} + \tfrac{3}{8} C_2\right) \ddot{q}_2 - \tfrac{3}{5} S_2 \dot{q}_1 \dot{q}_2 - \tfrac{3}{10} S_2 \dot{q}_2^2 + \tfrac{1}{2} g_a \left(\tfrac{11}{2} C_1 + \tfrac{6}{5} C_{12}\right)$$

$$\tau_{s,2} = \left(\tfrac{13}{60} + \tfrac{3}{8} C_2\right) \ddot{q}_1 + \tfrac{4}{25} \ddot{q}_2 + \tfrac{3}{10} S_2 \dot{q}_2^2 + \tfrac{3}{5} g_a C_{12}. \quad (26)$$

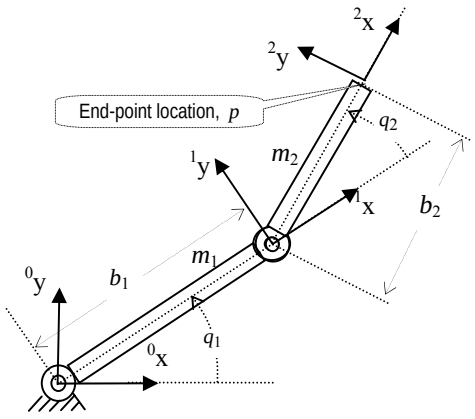

Fig. 2. Parameters of the simulated manipulator with 2-revolute joints.

All joint and actuator friction forces are assumed to be zero to prevent their stabilizing effect on the closed loop stability of the system. The feedforward control force τ_c is calculated only from the inertial and gravitational terms of a similar model ($\tau_c = D_c(q_d)\, \ddot{q}_d + g_c(q_d)$) but with a higher load mass m_2=3.5kg.

$$\tau_{c,1}(q) = \left(\tfrac{887}{600} + \tfrac{7}{10} C_2\right) \ddot{q}_1 + \left(\tfrac{13}{60} + \tfrac{3}{8} C_2\right) \ddot{q}_{2d} + \tfrac{1}{2} g_a \left(6 C_1 + \tfrac{7}{5} C_{12}\right)$$

$$\tau_{c,2}(q) = \left(\tfrac{13}{60} + \tfrac{3}{8} C_2\right) \ddot{q}_1 + \tfrac{7}{10} g_a C_{12}. \quad (27)$$

The test path is selected on a line segment in cartesian workspace starting from the point (${}^0x$, ${}^0y$) = (0.2, 0) to the point (0.7, 0.5) through the via-point (0.4, 0.2). The test trajectory is calculated for Δt = 1 ms intervals using parabolic blend with linear segments trajectory generation method with via points [11] under the constraints: maximum linear velocity= 1m/s, and linear acceleration 1 m/s^2. The points of the desired trajectory are shown in Fig. 2 for every 50 ms periods.

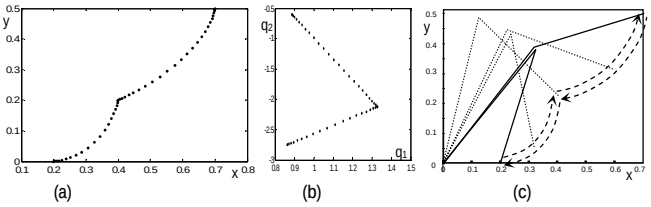

Fig. 3. Plots of the desired trajectory
(a) x vs. y (in meters), (b) q_1 vs. q_2 (in radians),
(c) the movement of the links in x-y plane.

The real-time cost function was chosen to be

$$J(t, p(t), p_d(t)) = \|p(t), p_d(t)\|$$

where $p(t)$ and $p_d(t)$ are the actual and desired end-point trajectory in the cartesian space, and $\|a, b\|$ denotes the cartesian distance between the points a and b.

The goal of the algorithm was to search smallest scoring PID parameter set $K_{PID} = (K_{1P}, K_{1I}, K_{1D}\ K_{2P}, K_{2I}, K_{2D})$ starting with a default stable parameter set

x_s = (5000, 200, 80, 5000, 200, 80),

without an intolerable tracking error cost.

The cost threshold was set to J^*= 0.002 m, and the cost score function $S(w, p_d(.))$ was evaluated by the maximum of $J(t)$ over one trajectory period. The population size was taken N= 60; elite population ratio was taken ρ=0.05; smooth update coefficients were $\alpha = 0.9$, $\beta = 0.9$, and q_s=5. In generating populations, the initial standard deviation of the gaussian pdf was set to $\gamma_0 = 0.5\, x_s$. The convergence plot of cartesian tracking displacement error for G_{max}=50 generations is shown in Fig. 4.

Fig.5 demonstrates typical tracking cases observed during the optimization of K_{PID}. The tracking error of the manipulator for with K_{PID}=x_s is seen in Fig. 5.a. The effect of switching K_{PID} from an unstable setting $x^{(12,7)}$ = (3132, 268, 50, 2121, 221, 98) to x_s=(5000, 200, 80, 5000, 200, 80) is seen in Fig. 5.b. The trajectory plots in Fig.5.c was obtained with the best scored parameter vector of 50 generations, $x^{(4,37)}$= (13032, 452, 136, 5391, 46, 161).

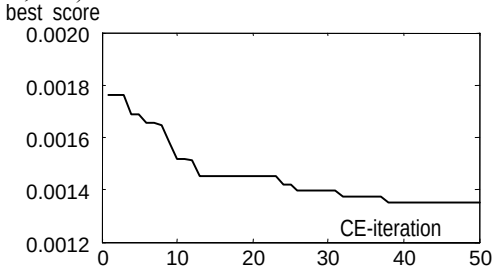

Fig. 4. Convergence plot of the best-score for two-link manipulator.

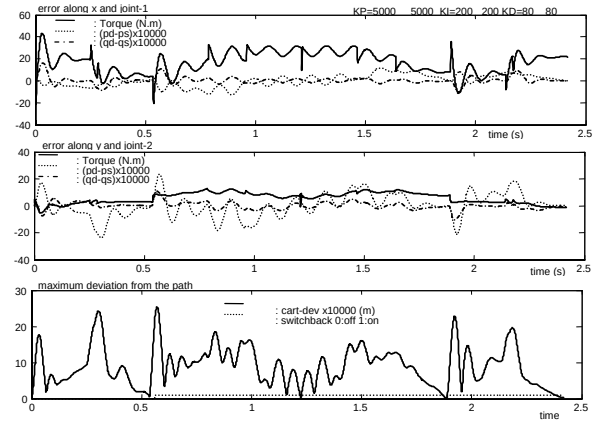

Fig. 5 a. Tracking performance of manipulator with initial stable control parameter set w_S.

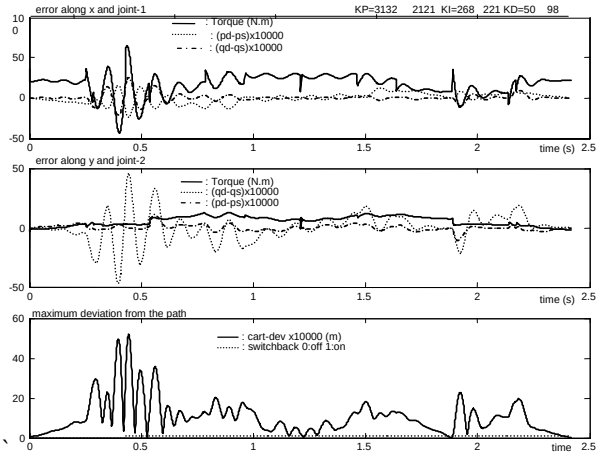

Fig. 5 b. An unstable parameter set causes excessive cartesian error at t=0.42, and control is switched back to x_S.

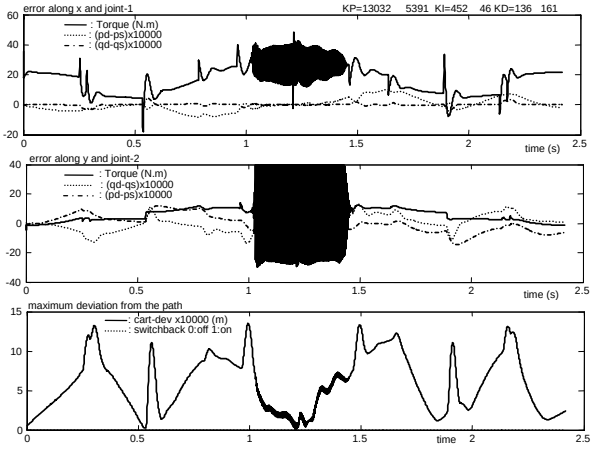

Fig. 5 c. The best parameter set reduced the cartesian error to 50%, but bounded oscillatory action of torques were observed because of the high proportional gain.

The CE-optimization provided almost 50% reduction of cartesian displacement error. At the wide open configuration of the manipulator the optimized controller settings produce an oscillatory action of actuator torques. But the oscillatory torques has a very minor effect on the tracking error. This oscillatory behavior may be prevented by assigning an additional cost on the derivative of the controller output.

VII. PID Settings of an Inverted Pendulum

The inverted pendulum is a well analyzed classical control problem [1], [18]. The objective is to track a desired cart trajectory with a small deviation e_x by applying an external force f on the cart along x-axis while keeping the pendulum stable in vertical position. The mechanical parameters of the system are introduced in Fig.6.

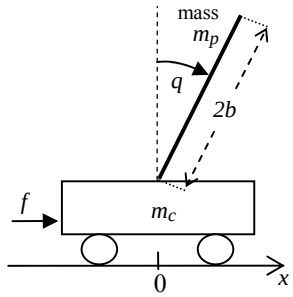

Fig. 6. Inverted Pendulum System.

The equation of motion of the cart and pole inverted pendulum system is expressed by

$$\begin{pmatrix} m_c + m_p & m_p b C_q \\ m_p b C_q & J + m_p b^2 \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{q} \end{pmatrix} + \begin{pmatrix} C_c & -m_p b \dot{q} S_q \\ 0 & C_p \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{q} \end{pmatrix} + \begin{pmatrix} 0 \\ m_p b\, g_a\, S_q \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \qquad (31)$$

where C_q and S_q are $\cos(q)$ and $\sin(q)$, g_a is the acceleration due to gravity, m_c is the mass of cart, m_p is the mass of the pole, b is the half length of the pole, C_c is the friction coefficient of cart, C_p is the friction of the pole, J is the mass moment inertial of the pole, and f is the force applied to the cart [18]. The pole mass distributed homogeneously along the pole is J= 1/3 $m_p b^2$.

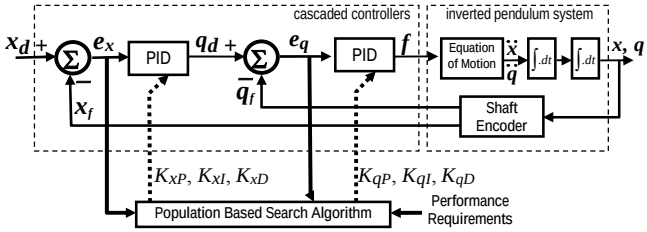

Fig. 7. Block diagram of inverted pendulum system for parameter optimization

Two PID controllers were cascaded to reduce the tracking errors $e_x = x - x_d$, and $e_q = q - q_d$, asymptotically to zero as shown in Fig 7. The goal of the exercise is to search the lowest costing PID parameters $K_{PID} = [K_{xP}, K_{xI}, K_{xD}, K_{qP}, K_{qI}, K_{qD}]$. For this cascaded PID control scheme, an oscillatory behavior was expected since no predictive or inverse dynamic feedforward action employed.

In the tests, a cart and pole system is simulated with the half-length of the pole b was taken 0.5m, the cart mass m_c=1 kg, and pole mass m_p =1 kg. The pole inertia J_p =16/3 corresponds to the homogenously distributed mass along the length of the pole. The desired cart trajectory was specified by a square function switching between –0.05m and 0.05m at every 25s, completing its period in 50s. The friction coefficients on

the cart C_c, and on the pole C_p were taken 0.01. With these coefficients the free motion amplitude of the pole diminish almost 50 percent in 50s period. The gravitational acceleration was taken g_a= 9.8 m/s^2. Equation of motion was integrated over Δt=5ms time steps using trapezoidal method. The initial parameter set

x_S = [K_{xP}, K_{xI}, K_{xD}, K_{qP}, K_{qI}, K_{qD}]
= [0.05 0.001 0.01 300 0.001 50]

managed stabilization of the system marginally with large oscillatory movements. The default parameter set was updated after every generation assigning the best scoring $x^{(i,k)}$ to the default stable parameter set x_S .

The cost function was specified using the quadratic displacement errors $e_x^2 = (x-x_d)^2$, $e_q^2=(q-q_d)^2$, and their integrals:

$$J(t, p(t, x^{(i)}), p_d) = 10\, e_x^2 + 8\, e_q^2 + \int e_x^2 dt + 0.6 \int e_q^2 dt \,. \quad (32)$$

In (32), the proportional gains on quadratic errors provide real-time cost action, while the integrals of quadratic errors accumulating the error for an end-score at the end of the cycle. At every generation k, for each member $x^{(i,\,k)}$ of the k^{th} population, the PID controller settings were set to the control parameters $x^{(i,\,k)}$ at the beginning of the desired trajectory cycle, for 50s, and corresponding score $s^{(i)}$ was evaluated using $S(x^{(i)}) = J(t_f, p(t_f, x^{(i)}), p_d)$ (32) at the end of the trajectory cycle. Control parameters were switched to the stable parameter set x_S at the threshold of $J(t, p(t, x), p_d) > J^* = 0.1$.

a) Adaptation by Real-time Cross Entropy Search Method

The initial standard deviation of the gaussian pdf of the CE search was selected $\gamma_k = 0.5\, x_S$; maximum generations and population size were selected G_{max}=50 and M = 30; elite population ratio was $\rho = 0.1$, smooth update coefficients were selected to be $\alpha = 0.9$, $\beta = 0.9$, and $q_S = 5$.

Fig.8 shows the convergence-rate of real-time CE search for three identical cases, with exactly the same initial parameter sets. The best scored parameter set of all runs reduced the tracking error score more than 50 percent.

The plot of x and q for the initial stable parameter set is shown in Fig. 9.a. A typical control parameter switching due to unstable character of the tested parameter set $x^{(i)}$ is given in Fig. 9.b, and the initial and final performance of the cart and pole system is compared in Fig. 9.c.

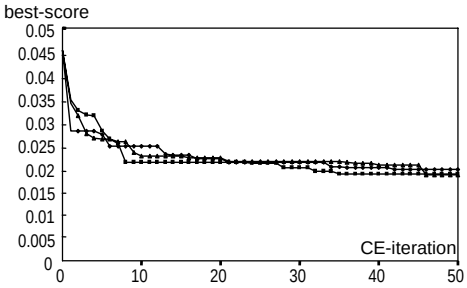

Fig. 8 Convergence-rate plots for three CE search runs with exactly the same initial parameters.

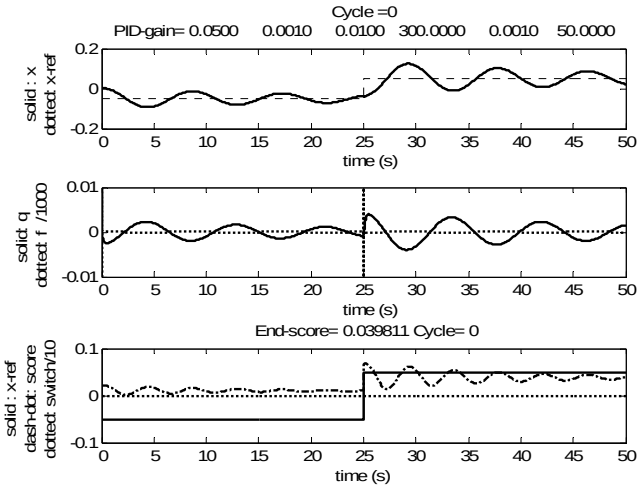

Fig. 9.a The initial stable parameter set x_S is oscillatory, but it satisfies the tracking error tolerance J^*.

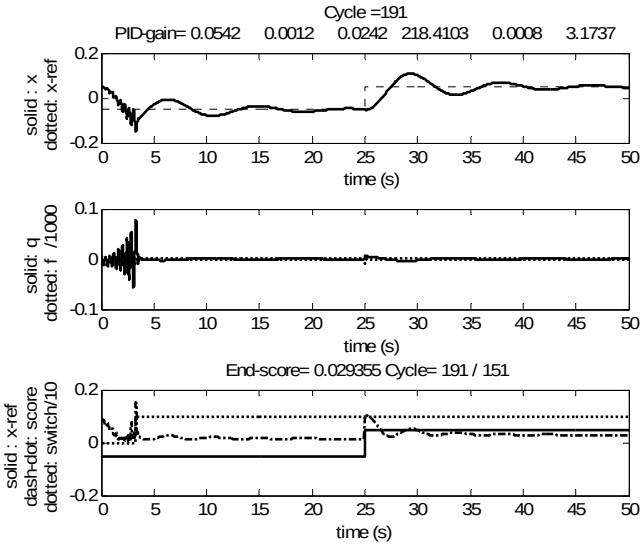

Fig. 9.b During the test of an unstable controller parameter set PID parameters were switched to xs at t =2.8s.

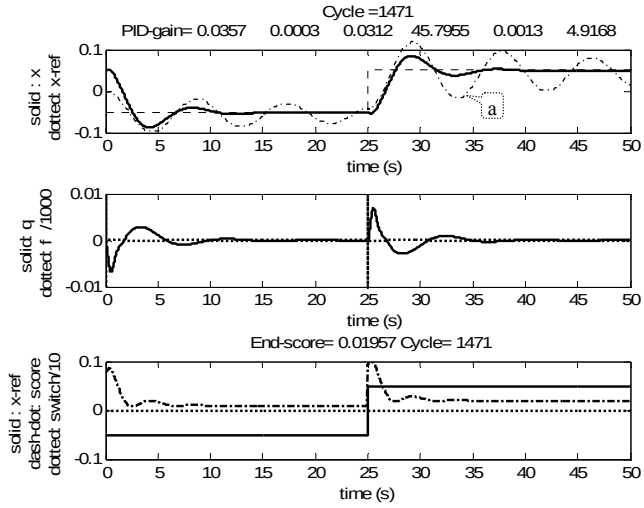

Fig 9.c Trajectory with the best scoring PID setting after 50 generations. On the first graph the trajectory (a) belongs to the initial PID settings x_S, inserted to the graph to compare initial and best-scored cases.

b) Adaptation by Real-Time Differential Evolution Method

Similar to the CE search cases, the population size and the maximum number of generations for the DE search are set to M=30 and G_{max}=50. The differentiation coefficient, the crossover rate, and the mutation probability are set to F= 0.9, CR=0.3 and p_m=0.25. The initial stable PID parameter vector x_S of real-time DE search, the real-time cost function, the score function, and the real time cost threshold value are all chosen the same as used in CE case.

Fig.10 shows the convergence-rate of real-time DE search for five identical cases, with exactly the same initial parameter sets. Each best scored parameter produced by these runs reduced the tracking error score almost 65 percent. As expected, the initial parameter set K_{PID}=x_S gives a very similar trajectory with the CE case shown in Fig. 9.a. The initial and final performance of the cart and pole system is compared in Fig. 11.

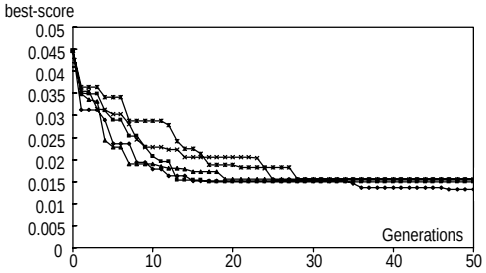

Fig. 10 Convergence-rate plots for five DE search runs with exactly the same initial parameters.

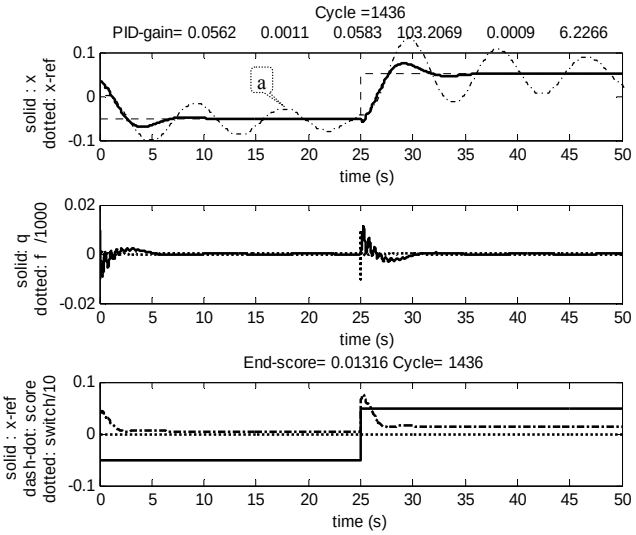

Fig. 11 Trajectory plots comparing the tracking error of (a) initial and the best scored PID parameters obtained by the real-time DE search.

VIII. Conclusion

This study proposes a real time evolutionary search method that provides stable operation within a specified cost threshold by switching the unstable control parameters with a prespecified stable parameter set whenever a tracking error based cost function exceeds a threshold. The proposed method has been tested successfully on an open chain manipulator dynamic control, and on the control of an inverted pendulum, where without a feedforward or predictive control the system is stable only in a very narrow band of PID settings.

During the tests none of these two systems has been fallen into non controllable states. In CE method, even though some of the oscillatory controller settings collected better scores than similar but less oscillatory settings, the mean of the elites always remained preferably stable. In both 2R and inverted pendulum cases, over 50% reduction of the specified cost function is achieved by population-based real-time search method during the progress of its repetitive operation.

Acknowledgment

Special thanks go to my colleague Dr. Adnan Acan for discussing several issues on implementation of the proposed method. Thanks to my graduate robotic course students: Maneli Badakshan, Vassilya Abdulova, Dilek Beyaz, Duygu Çelik for their efforts in reproduction of the trajectory planning algorithms; M. Reza Najiminaini, and A.A. M.Abed for their efforst in deriving the symbolic equation of motion.

References

[1] Kelly, R. Control of robot manipulators in joint space. (Advanced textbooks in control and signal processing)Control of Robot Manipulators in Joint Space), Springer-Verlag London Limited 2005

[2] Paul, R.C., "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm" Stanford A.I. Lab, A.I. Memo 177, Stanford, CA, Nov.1972.

[3] Paul, R.P., Robot Manipulators: Mathematics Programming, and Control, MIT Press. Cambridge 1982.

[4] Fu, K.S., Gonzales, R.C., Lee C.S.G., Robotics Control, Sensing, Vision, and Intelligence, Mc.Graw-Hill International Ed. Singapore, 1988.

[5] Mark W. Spong", "Motion Control of Robot Manipulators", online article: url:citeseer.ist.psu.edu/165889.html.

[6] Bodur M., Sezer M. E., Adaptive control of flexible multilink manipulators, Int.J.Control, vol.58, no.3, pp 519-536, 1993.

[7] V. Feoktistov, Differential Evolution In Search of Solutions *Optimization and Its Applications* Vol 5, Springer, USA, 2006

[8] Chuan-Kai Lin, Reinforcement learning adaptive fuzzy controller for robots, Fuzzy Sets and Systems, Elsevier Vol.137, pp.339-352, 2002.

[9] Kullback S. Information theory and statistics. NY: Wiley, 1959.

[10] J. Shore and R. Johnson. Properties of cross-entropy minimization. IEEE Trans. Information Theory, 27(4):472-482, 1981

[11] P.T. de Boer, D.P. Kroese, S. Mannor, and R.Y. Rubinstein. A tutorial on the cross-entropy method. Annals of Operations Research, 2004

[12] D.E. Goldberg, *Genetic Algorithms in search, Optimization, and Machine Learning,* Addison Wesley, Massachusetts, USA, 1989.

[13] R. Storn and K. Price, Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces, *Journal of Global Optimization Vol. 11*, pp. 341-359, 1997.

[14] Kenneth V. Price. Genetic annealing. Dr. Dobb's Journal, pages 127–132, October 1994.

[15] Bodur, M.; Acan, A.; Akyol, T. Fuzzy System Modeling with the Genetic and Differential Evolutionary Optimization, International Conference on Computational Intelligence for Modelling, Control and Automation, Volume 1, pp. 432 - 438, 28-30 Nov. 2005

[16] Uicker, J. J., Denavit, J. , Hartenberg, R. S., An iterative method for the displacement analysis of spatial mechanisms. Journal of Applied Mechanics. 26. 309-314, 1964.

[17] S. B. Niku, Introduction to Robotics, Analysis, Systems, Applications. Prentice Hall Inc. pp 153-165, 2001.

[18] Shozo Mori, et al., Control of unstable mechanical system, control of pendulum, Internat. J. Control 23 (5) pp. 673–692, 1976

[19] Rubinstein, R. Y. and Kroese, D.P. (2004). The Cross-Entropy Method:A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning. Springer-Verlag, New York.

[20] D.P. Kroese and K.-P. Hui: Applications of the Cross-Entropy Method in Reliability, Computational Intelligence in Reliability Engineering (SCI) 40, 37–82 Springer-Verlag Berlin Heidelberg (2007).

[21] H.Y. Fan, J. Lampien, A trigonometric mutation operation to differential evolution, *J. of Global Opt.*, Vol. 27, pp. 105-129, 2003.