SVM-Based Control System for a Robot Manipulator

Foudil Abdessemed

1 Department of Electronics, College of Technology, University of Batna, Algeria
* Corresponding author E-mail: foudil.abdessemed@univ-batna.dz

Received 30 Apr 2012; Accepted 30 Jun 2012

© 2012 Abdessemed; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract Real systems are usually non-linear, ill-defined, have variable parameters and are subject to external disturbances. Modelling these systems is often an approximation of the physical phenomena involved. However, it is from this approximate system of representation that we propose - in this paper - to build a robust control, in the sense that it must ensure low sensitivity towards parameters, uncertainties, variations and external disturbances. The computed torque method is a well-established robot control technique which takes account of the dynamic coupling between the robot links. However, its main disadvantage lies on the assumption of an exactly known dynamic model which is not realizable in practice. To overcome this issue, we propose the estimation of the dynamics model of the nonlinear system with a machine learning regression method. The output of this regressor is used in conjunction with a PD controller to achieve the tracking trajectory task of a robot manipulator. In cases where some of the parameters of the plant undergo a change in their values, poor performance may result. To cope with this drawback, a fuzzy precompensator is inserted to reinforce the SVM computed torque-based controller and avoid any deterioration. The theory is developed and the simulation results are carried out on a two-degree of freedom robot manipulator to demonstrate the validity of the proposed approach.

Keywords Two-degree of Freedom Robot Manipulator, Inverse Dynamics Control, Support Vector Machine (SVM), Fuzzy Precompensator

1. Introduction

For many years, the problem of controlling highly non-linear systems has received much attention within the control community, especially fast manipulation control. As a result of the strong need to find easy and robust methods to control robots, many solutions have been proposed. When the controlled part of the process is slightly disturbed, classical control laws - such as PID - may be sufficient if the accuracy requirements and system performance are not too severe. Otherwise, control laws must be developed that ensure the robustness of the processes' behaviour in the face of parameter uncertainties and variations.

One of the most popular and famous techniques is inverse dynamics control - also called the computed
torque control. Borrowed from the rigid robot manipulator [1], it can be easily applied to many other nonlinear systems [2]. This technique offers a wide variety of advantages. However, it requires precise knowledge of the analytical model of the system in order to compute the feed-forward torques required for the execution of trajectory tracking. Nevertheless, the dynamics of robot manipulators are coupled and highly nonlinear. They contain uncertainties such as friction and un-modelled nonlinearities. This is why a precise model cannot be obtained and why many efforts have been deployed to solve the problem of dynamic uncertainties. Many robot adaptive controllers have been proposed to solve the problem of parameter uncertainties and disturbances. However, few have been proposed to deal with both dynamic and kinematics uncertainties [3, 4]. To deal with these issues, researchers have demonstrated the positive use of neural networks in learning system uncertainties [5] and in helping in the generation of control laws [6, 7, 8]. In fact, instead of generating the feed-forward signal mathematically, it can be learnt from the feedback signal using neural networks. However, neural networks often suffer from the curse of dimensionality and the over-fitting problem. The theory and application of fuzzy logic is another tool used to control such highly nonlinear systems [9, 10]. It uses fuzzy linguistic variables and fuzzy logic to allow for smooth interpolation. Fuzzy logic is an alternative way to model human expertise. In [11], the authors combined the advantages of fuzzy and neural network intelligence to improve the overall learning ability and achieve robust control. In our previous work [12], we combined a PID controller with a fuzzy precompensator to control a robot manipulator tracking a pre-specified trajectory. We used fuzzy logic to design a precompensator on the basis of decision-making rules. It is tuned to minimize the output error when the PID controller exhibits significant steady-state error. However, fuzzy control methodology requires the tuning of membership functions and may not scale well to complex systems. It deals with imprecision and vagueness, but not with uncertainty.

Recently, Support Vector Machines (SVMs) have been introduced by Vapnik as an intriguing alternative machine learning method. They were originally designed for pattern recognition and classification tasks [13], and have been successfully used in a wide variety of classification problems, such as 3D object recognition [14], biomedical imaging [15], image compression [16] and remote sensing [17]. Furthermore, they exhibit very interesting behaviours when extended to solve regression and function approximation problems. SVMs do not suffer from the over-fitting problem and have a good generalization property. SVMs are based on the idea of structural risk minimization, which shows that the generalization error is bounded by the sum of the training error and a term depending on the Vapnik–Chervonenkis dimension. By minimizing this bound, high generalization capabilities can be achieved. In [18], the authors survey recent developments in the research and application of SVMs in intelligent robot information acquisition and processing. A comparison between two non-parametric regression methods for model approximation is proposed [19]. The outcomes yield positive results to the detriment of high computational cost.

Based on the work of Cheah et al. [20], in which they propose an adaptive Jacobian controller for the trajectory tracking of robot with uncertain kinematics and dynamics, we propose to exploit support vector regression for designing an efficient and robust control law to track robot trajectories. As an extension to our previous results [21], this paper studies this subject thoroughly. Uncertainties in robot parameters and nonlinearities can be dealt with more efficiently than model-based techniques. To avoid using online learning, we propose offline learning for the model’s system. This model is not overly sensitive to disturbances. This is explained by the fact that the complexity of the model’s representation by support vectors is independent of the dimensionality of the input space and depends only upon the number of support vectors. To further improve the performance of the above approach and ultimately lessen error bounds in case the robot undergoes changes in its kinematic and/or dynamic parameters, a fuzzy precompensator is inserted as reinforcement to the proposed scheme [12]. Simulations are carried out on a two-degree robot manipulator. The effectiveness of the proposed controller is demonstrated for different operating conditions and the results are very satisfactory. This paper is organized as follows: Section 2 formulates the robot dynamic equations; Section 3 presents the inverse dynamic control concept; Section 4 presents the SVM regression technique and model estimation with non-parametric regression techniques; Section 5 presents the development of the fuzzy precompensator; Section 6 describes the way the fuzzy rule base is obtained using evolutionary programming; Section 7 reports the simulations’ results and how the proposed approach provides positive outcomes, thereby justifying their applicability; finally, Section 8 concludes the paper.

2. Problem formulation

The dynamic equations of the robot manipulator are found through the use of the Lagrangian formulation, and the dynamic equation of an n-degree of freedom manipulator can be written as:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + F(q)$$  \hspace{1cm} (1)
where \( q \in \mathbb{R}^n \) is the generalized joint coordinate; \( M(q) \in \mathbb{R}^{n \times n} \) is the symmetric positive definite inertia matrix, bounded as \( m \leq M(q) \leq \overline{m}I \) for some constant \( m \) and \( \overline{m} \). Vector \( C(q, \dot{q}) \dot{q} \in \mathbb{R}^n \) presents the centrifugal and Coriolis torques, bounded as \( \|C(q, \dot{q})\| \leq a_1 \|\dot{q}\| \), the terms \( G(q) \in \mathbb{R}^n \) and \( F(q) \in \mathbb{R}^n \) are the vectors of gravitational torques and friction respectively, bounded as \( \|G(q) + F(q)\| \leq a_2 + a_3 \|\dot{q}\| \), and \( \tau \in \mathbb{R}^n \) is the vector of applied joint torques. The design of the controllers for the fine motion of the manipulator may be considered in two stages: the primary controller, which is nothing but the inverse dynamics of the manipulator inserted in the feedforward path to compensate for the nonlinear effects and which attempts to cancel the nonlinear terms in the model. However, the mathematical model used is usually not exact, and since the system is subject to disturbances, undesirable errors can be corrected by means of an additional controller, called a secondary controller (cf. Fig. 1).

![Figure 1. Primary and secondary robot manipulator control](image)

### 3. Inverse Dynamics Control

Let us consider the robot dynamics given by equation (1). The objective in using the inverse dynamic control is to eliminate the nonlinearities in the dynamics, as shown in Fig. 1. The inverse dynamic control technique is based on a controller command apprehending a priori knowledge about the system written in a dynamic model. This approach is often referred to as the computed torque method. Since the control objective is trajectory tracking in a joint space, the desired trajectories are joint angles, velocities and accelerations. The tracking errors are defined as:

\[
\begin{align*}
\epsilon &= q^d - q \\
\dot{\epsilon} &= \dot{q}^d - \dot{q}
\end{align*}
\]

where the measurements of \( q \) and \( \dot{q} \) are required in the subsequent design. To conform to the inverse nonlinear control given in Fig. 1, we rewrite equation (1) as:

\[
\tau = M(q)\ddot{q} + V(q, \dot{q})
\]

where:

\[
V(q, \dot{q}) = C(q, \dot{q}) \dot{q} + G(q) + F(q)
\]

\( V(q, \dot{q}) \) is a term that contains all the forces acting on the system. The functions \( C, G \) and \( F \) are defined as above. The problem is to determine the input vector \( \tau(t) \) so that \( q(t) \) will track the desired trajectory \( q^d(t) \) with tolerable errors. The input will be generated as:

\[
\tau(t) = u^d(t) + u^s(t)
\]

where:

\[
u^d(t) = M(q^d) \ddot{q}^d + V(q^d, \dot{q}^d)
\]

and:

\[
u^s(t) = M(q^d) (K_p \epsilon + K_v \dot{\epsilon})
\]

\( q^d \), \( \dot{q}^d \) and \( \ddot{q}^d \) denote the desired position, velocity and acceleration, respectively. Under ideal conditions and perfect knowledge of the parameter values with no disturbances, controller (6) causes the exact cancellation of the nonlinear expression in equation (3); that is, if \( V(q^d, \dot{q}^d) = V(q, \dot{q}) \), \( M(q^d) = M(q) \) and \( q(0) = q^d(0) \), then \( q(t) = q^d(t) \) for all \( t \geq 0 \). The implementation and the proper operation of the primary controller in equation (6) require that the dynamical model and the numerical values of the parameters are accurately known. If this is not the case, the cancellation of the nonlinear terms may not be perfect. Therefore, the secondary controller given by equation (7) is needed to compensate for the effects of internal modelling errors and disturbances. It will regulate small deviations of the system response about a nominal trajectory. Hence, it will be designed on the basis of a model obtained by the linearizing equation (3) about the pre-specified trajectory. For the analysis of the control system specified by equations (3) and (5), the following equalities are assumed:

\[
V(q^d, \dot{q}^d) = V(q, \dot{q}) \quad \text{and} \quad M(q^d) = M(q)
\]

After substituting equation (3) into equation (5) and recognizing the relations (8) in the resulting equation, we obtain the following error equation:

\[
K_v \dot{\epsilon} + K_p \epsilon = 0
\]

The stability and the dynamics of the error are controlled by the feedback gains \( K_p \) and \( K_v \). They can be chosen so
that the error equation is stable [1]. They can, for example, be tuned so that the eigenvalues of system in equation (8) assume the values that result in the acceptable responses of the system states. Unfortunately, in practice, robot dynamics cannot be modelled exactly; they are usually subject to torque disturbances and/or modelling errors. In this case, the closed-loop system will be different to that of equation (9). By equating equations (3) and (5) and by taking the estimates of $M(q^d)$ and $V(q^d,q^d)$ as $\hat{M}(q)$ and $\hat{V}(q,q)$ respectively, we obtain:

\[
\ddot{\mathbf{e}} + K_v \dot{\mathbf{e}} + K_p \mathbf{e} = M^{-1}(q)[\hat{M}(q)\dot{q} + \hat{V}(q,q) + \mathbf{v}]
\]

(10)

where $\hat{M}(q) = M(q) - \hat{M}(q)$ and $\hat{V}(q,q) = V(q,q) - \hat{V}(q,q)$ represent errors in the dynamic model. $\mathbf{v}$ is a vector function due to all the disturbances being completely unknown, but it is known to be upper bounded. The error equation (10) may be written in the form:

\[
\ddot{\mathbf{e}} + K_v \dot{\mathbf{e}} + K_p \mathbf{e} = M^{-1}(q)[Y(q,q,q)\tilde{\mathbf{e}} + \mathbf{v}]\]

(11)

where $\tilde{\mathbf{e}}$ is the vector of the parameter errors and $Y(q,q,q)$ is a matrix of functions.

In this work, and instead of using the analytical dynamics model and mathematically computing the required feed-forward compensation, we propose to use a kernel regressor to learn the feed-forward torques and a servo feedback control scheme to improve robustness. This may have the advantages of not only approximating and compensating for unknown system functions but also for unknown system properties, such as friction. In this case, and in absence of unknown disturbances, the second term of equation (10) is zero and the stability and the dynamics of the system are governed by equation (9). In the sequel, the learning feed-forward control is developed using the support vector machine (SVM).

If we approximate the model using a regression technique, equation (5) is changed to:

\[
\mathbf{u} = T(q^d,\dot{q}^d,\dot{q}^d)\hat{M}(q)\dot{\mathbf{e}} + K_v \dot{\mathbf{e}} + K_p \mathbf{e}
\]

(12)

such that $T(q^d,\dot{q}^d,\dot{q}^d) = M(q^d)\dot{q}^d + \hat{V}(q^d,\dot{q}^d)$, $M(q^d)$ and $\hat{V}(q^d,\dot{q}^d)$ are the approximate of $M(q^d)$ and $V(q^d,\dot{q}^d)$ respectively, determined by an appropriate kernel regressor. Actually, the inverse dynamics control based on SVM models will have the structure shown in Fig. 2.

4. Support Vector Regression and Model Estimation

4.1 Regression with the SVM

For a better understanding of the support vector machine, let us first consider a supervised binary classification problem. Let us assume that the training set consists of $N$ vectors $x_i \in \mathbb{R}^d \ (i = 1, 2, \ldots, N)$ from the $d$-dimensional feature space $X$. To each vector $x_i$, we associate a target $y_i \in \{-1, +1\}$. The linear SVM classification approach consists of looking for a separation between the two classes in $X$ by means of an optimal hyperplane that maximizes the separating margin. In the nonlinear case, which is the most commonly used since the data is often linearly non-separable, they are first mapped with a kernel method in a higher dimensional feature space, i.e. $\Phi(x) \in \mathbb{R}^{d'} \ (d' > d)$. The membership decision rule is based on the function $\text{sgn}(f(x))$, where $f(x)$ represents the discriminant function associated with the hyperplane in the transformed space and is defined as:

\[
f(x) = w^* \cdot \Phi(x) + b^*
\]

(13)

The optimal hyperplane defined by the weight vector $w^* \in \mathbb{R}^{d'}$ and the bias $b^* \in \mathbb{R}$ is that which minimizes a cost function that expresses a combination of two criteria, namely: 1) margin maximization and 2) error minimization. It is expressed as:

\[
\Psi(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i
\]

(14)

This cost function minimization is subject to the constraints:

\[
y_i \cdot (w \cdot \Phi(x_i) + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, N
\]

(15)

and:

\[
\xi_i \geq 0 \quad i = 1, 2, \ldots, N
\]

(16)
where $\xi_i$ is the so-called slack variable introduced to account for non-separable data. The constant $C$ represents a regularization parameter that allows the control of the shape of the discriminant function and, consequently, the decision boundary when data is non-separable. The above optimization problem can be reformulated through a Lagrange function for which the Lagrange multipliers can be found by means of a dual optimization leading to a quadratic programming solution [22], i.e.:

$$\max_a \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Subject to the following constraints:

$$\begin{align*}
\alpha_i & \geq 0, \quad i=1,2, \ldots, N \\
\sum_{i=1}^{N} \alpha_i y_i & = 0
\end{align*}$$

where $a = [\alpha_i, \alpha_j, \ldots, \alpha_N]$ is the vector of the Lagrange multipliers. The final result is a discriminant function conveniently expressed as a function of the data in the original dimensional feature space $X$.

Now, rather than dealing with outputs $y \in \{\pm 1\}$, regression estimation is concerned with estimating real valued functions. The SVM regression technique stems from the idea of deducing an estimate $\hat{g}(x)$ of the true and unknown relationship $y = g(x)$ between the vector of observations $x$ and the desired biophysical parameter $y$ from a given set of $N_T$ training samples. In our case, $N_T$ will include labelled as well as semi-labelled samples, such that $\hat{g}(x)$, first, has a maximum deviation from the desired targets $y_i$, $(i=1, \ldots, N_T)$ and, second, is as smooth as possible [22]. This is usually performed by mapping the data from the original $d$-dimensional feature space to a higher dimensional transformed feature space - i.e., $\Phi(x) \in \Re^{d'}$ ($d' > d$) - in order to increase the flatness of the function and to approximate it in a linear way as follows:

$$\hat{g}(x) = w^* \cdot \Phi(x) + b^*$$

where $w^*$ is a vector of weights and $b^*$ is the offset. ‘Flatness’ means that one seeks a small $w$, and one way to ensure this is by minimizing the norm - i.e. $||w||^2$. The optimal linear function in the higher dimensional transformed feature space is that which minimizes the following cost function:

$$\Psi(w, \xi) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{N_T} (\xi_i + \xi_i^*)$$

and:

$$\begin{align*}
y_i - (w \cdot \Phi(x_i) + b) & \leq \varepsilon + \xi_i \\
(w \cdot \Phi(x_i) + b) - y_i & \leq \varepsilon + \xi_i^*; i=1,2, \ldots, N_T \\
\xi_i, \xi_i^* & \geq 0
\end{align*}$$

where $\varepsilon > 0$ is a predefined constant which controls the noise tolerance. The $\xi_i$ and $\xi_i^*$ are the so-called slack variables introduced to account for those samples that do not lie in the $\varepsilon$-deviation tube, the constant $C$ represents a regularization parameter that allows the tuning of the tradeoff between the flatness of the function $\hat{g}(x)$ and the amount up to which deviations larger than $\varepsilon$ are tolerated. It is well known that $\varepsilon$ is proportional to the input noise level and should also depend on the number of training samples [23, 24]. The above optimization problem reformulated through a Lagrange function into a dual optimization problem leads to a solution that is a function of the data conveniently expressed in the original dimensional feature space as:

$$\hat{g}(x) = \sum_{i \in W} (\alpha_i - \alpha_i^*) K(x_i, x) + b^*$$

where $K(.,.)$ is a kernel function and $W$ is the subset of indices $(i=1, 2, \ldots, N_T)$ corresponding to the non-zero Lagrange multipliers $\alpha_i$ or $\alpha_i^*$’s. The training samples associated to non-zero weights are called support vectors. The kernel $K(.,.)$ should be chosen so that it satisfies the condition imposed by Mercer’s theorem [25, 26]. Examples of common nonlinear kernels that fulfill Mercer’s condition are the polynomial and the Gaussian kernel functions.

4.2 Robot Model Estimation with Support Vector Regression

In this second part of the section, we will generate the approximate model of the robot manipulator with nonparametric regression techniques. We consider a two-link robot manipulator whose dynamics equations are derived using the Euler-Lagrange formulation. Based on relation (6), the equations of the motion of the two-link manipulator are expressed for each joint as:

$$\begin{align*}
u_1^d & = \tau_1 + \tau_2 \\
u_2^d & = \tau_3 + \tau_4
\end{align*}$$

such that

$$\begin{align*}
\tau_1 & = M_1^T(q) \ddot{q}^d \\
\tau_2 & = V_1(q^d, \dot{q}) \\
\tau_3 & = M_2^T(q) \ddot{q}^d \\
\tau_4 & = V_2(q^d, \dot{q})
\end{align*}$$

where $M_1^T$ and $M_2^T$ are the row vectors of the inertia matrix and $V_1$ and $V_2$ are vectors regrouping the terms of the Coriolis and centrifugal torques as well as the gravity and friction torques. A model identification procedure is carried out to identify these terms using
SVM regression. Each term is identified separately to get - at the end - four approximate models, namely $M_1$, $M_2$, $V_1$, and $V_2$ which are used to generate the following approximate robot manipulator torques:

$$u_1 = M_1^2 q^d + V_1(q^d, q^d) + M_1^2 (k_p e + k_v e)$$
$$u_2 = M_2^2 q^d + V_2(q^d, q^d) + M_2^2 (k_p e + k_v e)$$  \hspace{1cm} (23)

The purpose of the PD-controller is to servo the system so that the actual displacement of the joint will track a desired angular displacement specified by a pre-planned trajectory. The starting point is the use of the desired joint angle, velocity and acceleration trajectories to generate the corresponding torques for every sampling time. The obtained torque vectors are stored with the position, velocity and acceleration vectors in lookup tables to be referred to afterwards by the SVM regressor for training.

The entries of the \textit{svmtrain} function are, consequently, the torques $\tau_1$ and $\tau_3$, as well as the position and acceleration vectors needed to implement $M_1$ and $M_2$. The torques $\tau_2$ and $\tau_4$, as well as the position and velocity vectors, are needed to implement $V_1$ and $V_2$. In this work we have adopted the nonlinear SVM regressor based on the Gaussian kernel [27] as a result of the good performances generally achieved by this type of kernel. It is worth mentioning that the performance of SVMs depends on the choice of kernels and there is no theory which states a way to choose good kernels but rather experiment with specific problems. A typical example of such a kernel is represented by the Gaussian function:

$$K(x_i, x) = \exp(-\gamma \|x_i - x\|^2)$$  \hspace{1cm} (24)

where the parameter $\gamma$ is inversely proportional to the width of the Gaussian kernel.

During the training phase, the SVM regressor was selected according to a k-fold cross-validation (CV) procedure by first randomly splitting the training data into k mutually exclusive subsets (folds) of equal size [28]. Then, we trained, k times, an SVM classifier modelled with the predefined values of $C$ and $\gamma$. Each time, we left out one of the subsets from training and used it (the omitted subset) only to obtain an estimate of the classification accuracy. From k instances of training and mean square error computation, the average accuracy yielded a prediction of the regression accuracy of the considered SVM regressor. The best SVM regression parameter values were chosen to maximize such a prediction. It is noteworthy that when k is equal to the number of samples, the k-fold CV becomes a leave-one-out CV procedure.

4.3 Discussion

To study the stability of the system, consider equation (9) which describes the evolution of errors relative to the desired trajectory. If the model is perfect and if there is no noise and no initial error, There will be an exact tracking trajectory. In fact, the values of the gains $K_p$ and $K_v$ can be adjusted so that the eigenvalues of the system in equation (8) assume values that result in acceptable responses to the system states. However, and in case of any inaccuracy in the manipulator model, the analysis of the resulting closed-loop system becomes difficult and the stability of the system may be lost. To overcome this problem, we proposed a support vector machine in learning feed-forward control. The advantage of this scheme is its ability to learn unknown system properties - such as friction - that can be compensated in this way. Unfortunately, some external disturbances or noise cannot be compensated and instability may result. This leads us to propose - in the next section - a fuzzy precompensator, which when added to the proposed scheme helps in obtaining good transient responses and stability in face of disturbances and noise, and discards any analysis of any eventual solution of the differential equation governing the system.

5. The Fuzzy Precompensator

5.1 Analysis and Design

In order to compensate for losses, a fuzzy precompensator is integrated into the existing system. This is simply achieved by placing it in front of the SVM-inverse dynamics controller (Fig. 3). The fuzzy precompensator compensates for dynamic changes in the system and unpredictable external disturbances. It helps the inverse dynamics controller to provide the necessary actions that allow the end effector track the desired trajectory with minimum errors. The block diagram of the precompensator is shown in Fig. 4.

The two inputs to the PD controller are $e_i(k)$ and $\dot{e}_i(k)$, where:

$$e_i(k) = q_i^d(k) - q_i(k), \quad i=1,...,n$$  \hspace{1cm} (25)

is the trajectory tracking error vector and its first-order derivative is given by (26), where $i$ refers to the $i$th joint and $n$ to the number of joints.

$$\dot{e}_i(k) = q_i^d(k) - \dot{q}_i(k), \quad i=1,...,n.$$  \hspace{1cm} (26)

The output of the precompensator is considered now as the new reference input to the system; one can write:

$$\bar{q}_i^d(k) = q_i^d(k) + \delta_i(k), \quad i=1,...,n$$  \hspace{1cm} (27)
5.2 Stability Analysis

The presence of the fuzzy precompensator introduces a new joint position reference \( q^c \), such that:

\[
q^c = q^d + \delta
\]

In this case, (11) is rewritten as:

\[
\ddot{e} + K_e \dot{e} + K_p e = \Psi \hat{\theta} + v - u_c
\]

where \( \Psi = \dot{M}^{-1} Y \), \( v = \dot{M}^{-1} v' \) and \( u_c = K_p \delta \). To study the stability of the tracking errors, we start by writing the system (31) in the state space:

\[
\dot{x} = Ax + B\Psi \hat{\theta} + Bv - Bu_c
\]

where \( Bv \) is a 2nx1 with an upper bound.

\[
\|Bv(t)\| < v_{\text{max}}
\]

A is the Hurwitz matrix:

\[
A = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}, \quad \text{and} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix}
\]

In [27], it is seen that the adaptive control system designed for structured uncertainties will exhibit robustness in relation to bounded external disturbances. If we can compensate for the disturbing terms in equation (32), we may obtain disturbance rejection over the entire workspace of the robot manipulator. This can be done by synthesizing fuzzy control rules from human experts or using evolutionary algorithms. Our task is to guarantee the stability of the closed system in the sense that the state \( x \) remains uniformly bounded in face of parameter uncertainties and unknown disturbances. Looking at Fig. 4, the 2nx1 vector function approximator \( \delta = F(e, \Delta e) \) is used to compensate any disturbances by providing the right amount of signal. Let us define the Lyapunov function candidate:

\[
V = \frac{1}{2} x^T P x
\]

where \( P \) is a symmetric positive definite matrix satisfying the Lyapunov equation:

\[
A^T P + PA = -Q
\]

for a given symmetric, positive definite matrix \( Q \). Because \( A \) is stable, such \( P \) always exists. Using (32) and (35), the time derivative of \( V \) is obtained as:

\[
\dot{V} = -\frac{1}{2} x^T Q x + x^T PB (\psi \dot{\theta} + v - u_c) \leq x^T PB (|\psi \dot{\theta} + v - u_c|)
\]

Our goal is to design \( u_c \) such that \( \dot{V} \leq 0 \) - i.e., the controller \( u_c \) guarantees that \( |x| \) is decreasing. Therefore we should always have:

\[
u_c \geq |\psi \dot{\theta} + v|
\]

To implement this command online, the fuzzy precompensator should provide the quantity that should be added in such a way that constraint (37) is always satisfied. Therefore, the idea is to analyse the output response.

An increase in the command signal is needed whenever the error is around zero and the change-in-error is negative. Correspondingly, an increase in the command is necessary when we observe an increase in the error and the change-in-error. The other rules are obtained in a similar manner. For example, suppose that the error is zero and that the change-in-error is negative. This means that the output is increasing, heading in the direction of an overshoot. To compensate for this, we have to decrease the command signal. This corresponds to applying a correction term that is negative. The fuzzy precompensator is based on the Takagi-Sugeno type fuzzy model [30]. The realization of the function
F[e(k), Δe(k)] obeys the procedures of fuzzification, fuzzy rule inference and defuzzification. Each membership function is a map from the universe of discourse to the interval [-1 1]. The Takagi-Sugeno’s model is formulated following the form:

\[ R_{ij}: \text{if } e \in A_i^e \text{ and } \Delta e \in A_i^{\Delta e} \text{ then } \delta_{ij} = m_{ij} ; \quad i=1..7; j=1..7. \]

Where \( R_{ij} \) denotes the \( ij \)th implication and \( m_{ij} \) is the consequent parameter, \( A_i^e \) and \( A_i^{\Delta e} \) are fuzzy sets of the input membership functions. The output of the fuzzy precompensator is performed by defuzzification method, given the input pair \((e(k), \Delta e(k))\); the final output is:

\[ \delta(k) = C_r \frac{\sum_{i=1}^{49} \mu(\xi_i)\zeta_i}{\sum_{i=1}^{49} \mu(\xi_i)} \]

(38)

where \( \zeta_i \) is the support value at which the membership function reaches the maximum value and \( \delta(k) \) is the consequent of the \( i \)th implication; the weight \( \mu(\xi_i) \) implies the overall truth of the premise of the \( i \)th implication, calculated as:

\[ \mu(\xi_i) = A_i^e \bullet A_i^{\Delta e} \]

(39)

\( C_r \) is a scaling factor, defined for the three universes of discourse and whose values are chosen as: \( C_r=3.15; C_{\Delta e}=5; C_r=10.0 \). The main difficulty in designing a fuzzy logic controller is the efficient formulation of the fuzzy If-Then rules. It is well-known that if it is easy to produce the antecedent parts of a fuzzy control rule, it will be difficult to produce the consequent parts of the rules without expert knowledge. The derivation of such rules is often based on the experience of skilled operators, the use of heuristic thinking [31, 32] or by modelling an expert’s action [33].

6. Evolutionary Programming

In recent years, due to the availability of powerful computer platforms, the theory of evolutionary algorithms gained popularity to the problem of parameter optimization, although its origins can be traced back to the 1960s. They represent a well known family of optimization methods that have proven very attractive both for their stability and their effectiveness. These techniques have been exploited successfully in a broad range of applications including, for instance, biology, chemistry, control systems and image processing. This method is regarded as a parallel global search technique evaluating many points in the parameter space and it is more likely to converge towards a global solution. These algorithms perform a search by evolving a population of candidate individuals modelled with “chromosomes”. From one generation to the next, the population (set of candidate individuals) is improved by mechanisms inspired by genetics - i.e., through the use of both deterministic and nondeterministic genetic operators. Evolutionary programming (EP) is used to obtain the optimal consequent parts of the fuzzy precompensator rules.

6.1 Representation

Each individual chromosome represents a complete rule-base solution. The components \((a_1 = m_1, a_2 = m_2, ..., a_L = m_L)\) determine the consequent parts of the fuzzy rules, and the remaining components \((a_{L+1} = \sigma_1, a_{L+2} = \sigma_2, ..., a_{2L} = \sigma_L)\) contain standard deviations which control the mutation process. A complete string of chromosomes could be written in the following way, with \( a_1a_2...a_La_{L+1}a_{L+2}...a_{2L} \), representing one individual. The set of all the individuals represents a population. If we denote by \( P(t) \) a population at time \( t \), then we can write:

\[ P(t) = \{I^1(t), I^2(t), ..., I^p(t)\} \]

(40)

where:

\[ I^1 = \{a_j; i = 1, ..., L; j = 1, ..., 2L\} \]

(41)

\( I^1 \) designates the \( i \)th individual in which \( a_j \) describes the \( j \)th consequent part and the standard deviations, as was defined above. The algorithm seeks many local optima and increases the likelihood of finding the global optimum representing the problem goal.

6.2 Initialisation

The algorithm is initiated by designating an initial population formed by 20 identical individuals. The number 20 is only an experiment number. Each solution is taken as a pair of real-valued vectors \((\vec{m}_i, \vec{\sigma}_i)\), such that \( \dim(\vec{m}_i)=\dim(\vec{\sigma}_i)=\)number of rules. \( \vec{m}_i \) is the consequent part vector of the fuzzy If-Then rules and \( \vec{\sigma}_i \) is an adaptable standard deviation vector. Initially, chromosomes in the current population are selected for reproduction according to the roulette wheel spinning method [34]. The adaptable standard deviations \( \sigma \) have been assigned an ad hoc choice of \( \sigma_i = 0.05; i=1,...,49 \).

6.3 Evaluation

The fitness function is evaluated for each set of rules and based on the function chosen to be:

\[ \Phi_F = \frac{10^9}{\text{PE TE}} \]

(42)
where:
\[
PE = \sum_{t=1}^{2} \sum_{i=1}^{2} (\theta_i(t) - \theta_i(t))^2,
\]
\[
TE = \sum_{t=1}^{2} \sum_{i=1}^{2} \tau_i(t) - \tau_i(t).
\]

### 6.4 Generation

Generation in EP differs from that in GAs. The variation operators such as crossover and mutation perform differently. In fact, one child from one parent is generated by adding a normally distributed random value with an expectation zero and a unity standard deviation [35] as:

\[
m_{new}^j = m_{old}^j + \bar{\sigma}_{old}^j.N_j(0,1) \quad (43)
\]

N\(_j\)(0,1) indicates that the random variable is sampled anew for each value of the counter \(j\). The step size \(\sigma\) is controlled by equation (44). The parameters \(\tau\) and \(\tau'\) are the learning rates and adopt the conventional values of \((\sqrt{2}n)^{-1}\) and \((\sqrt{2}n)^{-1}\) respectively, where \(n\) is the total number of rules to be evolved (in our case, \(n=49\)).

\[
\sigma_{new}^j = \sigma_{old}^j.\exp[\tau.N(0,1) + \tau.N(0,1)] \quad (44)
\]

To help the system operating in a convergence and a stable region, certain constraints are utilized that are derived when the following assumptions are made:

- **Assumption 1**: The action values within the upper triangle of Table.1, denoted by \(m^U\), are constrained to take negative values. Here: \(m^U(j,k) \in [-2, 0]\);
- **Assumption 2**: The action values within the lower triangle of Table.1, denoted by \(m^L\), are constrained to take positive values. Here: \(m^L(j,k) \in [0, 2]\);
- **Assumption 3**: The action values within the off-diagonal of the lower triangle of Table.1, denoted by \(m^{UL}\), are free to take either positive values or negative values, with the one in the middle of the table constrained to take a zero value. i.e.:

\[
\begin{align*}
(m^U / (j,k) & \in [-2, 2]; \ j \neq k \\
(m^U / (j,k) & = 0.0; \ j = k
\end{align*}
\]

The values -2 and 2 are chosen as lower and upper limits, successively and arbitrarily. To explain the natural meaning of the above assumptions, it is necessary to establish the link that gives input and output values based either on experience, a system step response or phase plane trajectory, so as to form the rule base. For example, if the reference signal is kept constant while the error is positive and the error change is negative, this means that the output is increasing towards an overshoot. Therefore, the input to the system should correspond to a negative value. After creating the new generation, a fitness score is evaluated for each new member \(m_i\).

### 6.5 Selection

The next generation is selected based solely on the fitness values of the individuals. The selection is conducted by taking a random uniform sample of individuals of a size equal to half the population size among that of all the parents and offspring. Each solution from offspring and parent individuals is evaluated against the “" win"” randomly chosen individuals. For each comparison, a “" win"” is assigned if an individual score is better than or equal to that of its opponent, and the 20 new individuals with the greatest number of wins are retained as parents for the next generation. Their associated adaptable standard deviations are included.

### 7. Computer Simulation and Comparison

For simulation and comparison and without lost of generalities, a two-link robot manipulator was used to analyse and test the performance of the proposed approach. The rigid manipulator was modelled as two rigid links of lengths \( l_s=0.432m \) and \( l_s=0.432m \) with corresponding masses \( m_1=15.91Kg \) and \( m_2=11.36 Kg \).

The simulations were carried out using the fourth-order Runge-Kutta algorithm, with the step \( h=0.01 \). The objective is to track the desired joint trajectories chosen to be:

\[
\begin{align*}
q_1 & = 1 + 2(sint + sin2t) \\
q_2 & = 2 + cos4t + cos6t
\end{align*}
\]

Two Matlab SVM libraries are used. The trainer SVM classifier, *svmtrain*, has the following syntax:

\[
\text{model} = \text{svmtrain}(\text{training_label_vector}, \text{training_instance_matrix}, 'libsvm_options')
\]

and the prediction class of the new input data according to a pre-trained model, *svmpredict*, having the following syntax:

\[
\text{predicted_label} = \text{svmpredict}(\text{testing_label_vector}, \text{testing_instance_matrix}, \text{model}, 'libsvm_options')
\]

During the training phase, we used a 3-fold CV, whereas the values of \( C \) and \( \gamma \) are found to be (8192, 0.25), (2048, 0.25), (5064, 0.25) and (1991, 0.25) for the four models \( M_1, M_2, V_1 \) and \( V_2 \) respectively. It is worth mentioning that the quality of SVM models depends on the proper settings of the SVMs’ meta-parameters. However, there is no consensus on a given method for the choice of these parameters. In our case, the parameters are determined by a simple loop using a prepared Matlab script, summarized by the following algorithm:
Start

**Step 1)** Set initial values for best_rate, best_C, and best_γ.

**Step 2)** Set the initial value and the final value of cost C.

**Step 3)** Set the initial value and the final value of γ in the kernel function.

**Step 4)** Train an SVM classifier on the feature Space and assign the result to rate equ.(46)

**Step 5)** If the rate is greater than or equal to best_rate, set best_rate as equal to rate, best_C as equal to C and best_γ as equal to γ.

**Step 6)** Increment γ by some amount.

**Step 7)** If γ is not equal to γ_{\text{final}}, go to step 4.

**Step 8)** Increment C by some amount.

**Step 9)** If C is not equal to C_{\text{final}}, go to step 4.

End

The ε value of the intensive tube was fixed to 10⁻¹. On the other hand, the rule base of the fuzzy precompensator is found by running the evolutionary algorithm with a population of 20 for about 500 generations. The resultant rule-base is depicted in Table 1.

First, the classical computed torque controller is implemented. As stated before, the values of the gains $K_p$ and $K_\phi$ can be adjusted so that the eigenvalues of the system in equation (9) assume the values that result in the acceptable responses of the system states.

To have an idea of the precision of the system, we propose to evaluate the proposed scheme in terms of the rising time and the settling time. We tested the position control in terms of RMS values, given by equation (48), when the first joint is made to move from 0 to (π/4) and the second joint from 0 to (-π/4).

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2} \quad (48)$$

The simulation is carried out for three cases for a lap of time equal to 1.12 sec. We report in Table 2 the results of the simulations, from where it is clear that the SVM precompensated system based on computed torque provides better results compared to the computed torque and the computed torque-based SVM approaches. Moreover, we show in Figs. 5, 6 and 7 the input step responses when the two links are moving in opposite directions. One can see the better performances observed on the computed torque-based SVM reinforced by the fuzzy precompensator (Figs. 7). The robustness of the last controller is obvious compared with the other two controllers when unknown payloads are added. In fact, this controller succeeded in rejecting the perturbation and maintaining its desired step response. On the other hand, the other two controllers show mediocre performances when the plant is perturbed, and the outputs move away from their desired step responses (Figs. 5 and Figs. 6).

<table>
<thead>
<tr>
<th>Δe</th>
<th>NL</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.614</td>
<td>-0.70</td>
<td>-8.94</td>
<td>-1.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMD</td>
<td>-0.54</td>
<td>-0.14</td>
<td>-1.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSD</td>
<td>0.812</td>
<td>-0.46</td>
<td>0.847</td>
<td>0.349</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZED</td>
<td>-0.55</td>
<td>0.308</td>
<td>0.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSd</td>
<td>-0.60</td>
<td>0.605</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMd</td>
<td>0.91</td>
<td>0.828</td>
<td>0.487</td>
<td>0.555</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLd</td>
<td>0.717</td>
<td>0.61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1.** Fuzzy logic rules for the precompensator.

<table>
<thead>
<tr>
<th>RMSE</th>
<th>Joint (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Computed torque</td>
<td>0.1941</td>
</tr>
<tr>
<td>Computed torque based SVM</td>
<td>0.2031</td>
</tr>
<tr>
<td>Computed torque based SVM precompensated</td>
<td>0.0557</td>
</tr>
</tbody>
</table>

**Table 2.** Values of the position tracking error as RMSE.

**Figure 5(a).** Position tracking error of the 1st joint (computed torque).

**Figure 5(b).** Position tracking error of the 2nd joint (computed torque).
On the other, and for better illustration, we have plotted the desired and measured trajectories to show the tracking performances of the three techniques without perturbation (Figs. 8, 9 and 10). One can see that the tracking performances are excellent, since nonlinearities are cancelled.

The trajectory tracking obtained using the SVM regressor-based inverse dynamic control which shows excellent tracking and a surprisingly better curve response in the transient part of the first joint - as can be seen in Fig. 9 - as determined by the accuracy of the regression model. These results illustrate the good performance of the SVM regression algorithm in approximating nonlinear functions and does not care about errors so long as they are less than \( \varepsilon \), but will not accept any deviation larger than this.

The transient responses for the two joints are further improved (Figs. 10) when inserting the fuzzy precompensator in the previous configuration. In fact, the fuzzy precompensator is designed such that the inputs act like input reinforcement having the amount that must be added to the process input to compensate for current poor performances.

To see the precision performance when perturbations are added, simulations were carried out by suddenly doubling the mass of the second link at time 2.5 sec and changing the values of both the viscous and Coulomb friction coefficients at the learning stage. In this case, the precision is relatively degraded for the computed torque method because the original design is perturbed and the parameters of the PD controller need to be updated to new values (cf. Figs. 11).

However, for the SVM regressor-based inverse dynamic control, the error tolerance seems to be much better for the two joints (Figs. 12). We explain this fact by the concept that states that the complexity of the model’s representation by support vectors is independent of the dimensionality of the input space, depending instead only upon the number of support vectors.

Now, to further improve the performance of the above approach and achieve global asymptotical tracking with fast transient responses and disturbance rejection, we activate the precompensator in the controlled system. In this case, the system can track the reference input much better despite any disturbances, as shown by Fig. 13. The evolution of joint position errors are depicted in Figs. 14, 15 and 16. They illustrate the tracking performances experienced by each method. One can note that in the absence of perturbations, the performances are almost the same. However, in presence of perturbation, the tracking performances are degraded for the computed torque and the SVM-based inverse dynamic, but remain adequate for
the SVM-based inverse dynamics reinforced with the fuzzy precompensator, as is depicted in Figs. 17, 18 and 19 respectively. In fact, with the third scheme the maximum position errors are effectively decreased. This could be explained by the ability of the fuzzy precompensator to work around the goal positions, even though the PD controller parameters need not to be well tuned.

On the other hand, the quality of tracking can be measured in terms of the normalized mean square error, defined as the Mean squared error/variance of the target, given by expression (49) as:

\[ nMSE = \frac{\sum_{i=1}^{N} (f(x_i) - \bar{f}(x_i))^2}{\sum_{i=1}^{N} (f(x_i) - \bar{f}(x_i))^2} \]

(49)

where:

\[ \bar{f}(x_i) = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

and N is the number of samples.

Table 3 gives the normalized squared error (nMSE) of the three studied methods when no perturbation is considered. When analyzing the results obtained, we come to the conclusion that the proposed novel approach attained excellent performances and that it is not contingent upon any assumptions related to the actual system. Comparing control quality using the three methods, the nMSE of the modelled system is mostly the same as the real system, due to better approximation.

Using the required stored torques, we can solve for the time-consuming torque calculation, for which the values have to be provided as fast as the servo-loop. The role of the fuzzy precompensator is obvious through the results obtained in Table 4, which shows the performances achieved by this controller when the system is perturbed.

It is also important to have an idea of the effort developed on the joints. If we assume that each joint is driven by a DC motor via a transmission gear system, the control inputs are obtained in terms of voltage by multiplying the calculated torques by \( N^1 K_m^{-1} \), where N represents the gear ratio and \( K_m \) is the torque constant, such that:

- \( N_1 = 120 \) for the first DC motor
- \( N_2 = 50 \) for the second DC motor
- \( K_{mi} = K_{mc} = 0.18 \text{Nm/Volt} \) for the torque constant

We report on Figs. 20, 21 and 22 the curves of the control inputs when the robot manipulator evolves over the entire workspace without perturbations. On the other hand, we depict in Figs. 23, 24 and 25 the curves of the control inputs when the robot manipulator is perturbed by doubling the mass of the second link. The comparison between the Figures shows that the controller has compensated for the mass disturbance by an increase demand in energy. There is no doubt that the results obtained by the SVM-based computed torque controller enhanced with the fuzzy precompensator are highly

<table>
<thead>
<tr>
<th>nMSE</th>
<th>Joint (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Computated torque</td>
<td>0.0027</td>
</tr>
<tr>
<td>Computated torque-based SVM</td>
<td>0.0021</td>
</tr>
<tr>
<td>Computated torque-based SVM precompensated</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

**Table 3. Tracking error as the nMSE for each joint using reference trajectories without perturbation**

<table>
<thead>
<tr>
<th>nMSE</th>
<th>Joint (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Computated torque</td>
<td>0.0028</td>
</tr>
<tr>
<td>Computated torque-based SVM</td>
<td>0.0019</td>
</tr>
<tr>
<td>Computated torque-based SVM precompensated</td>
<td>0.0009</td>
</tr>
</tbody>
</table>

**Table 4. Tracking error as the nMSE for each joint using reference trajectories with perturbation.**

satisfactory, as is clear from Figs. 22 and 25. The demand in energy to conduct the experiment with this controller is much less than that of its predecessors. Moreover, one can reduce any initial chattering by carefully tuning the scaling factor \( C_r \) in (37) or by choosing small initial amplitude responses. Comparing the three approaches, we come to the conclusion that the proposed scheme has the significant advantage in improving performance relative to the computed torque or the fixed gain controller. The simulation results showed that the computed torque based on SVM learning is a very interesting approach when a precise model of the robot manipulator is unknown or when the system is confronted with disturbances and uncertainties. SVM-based computed torque can achieve a relative low tracking error. In cases where high performances are needed, we can always support SVM-based computed torque with a fuzzy precompensator, which provides the amount needed by the plant when variations in the load or changes in the parameters of the plant are observed.
Figure 8. Trajectory tracking without perturbation, computed torque.

Figure 9. Trajectory tracking without perturbation, computed torque-based SVM.

Figure 10. Trajectory tracking without perturbation, computed torque-based SVM precompensated.

Figure 11. Trajectory tracking with perturbation, computed torque.
Figure 12. Trajectory tracking with perturbation, computed torque-based SVM.

Figure 13. Trajectory tracking with perturbation, computed torque-based SVM precompensated.

Figure 14. Performance of the computed torque control without perturbation.

Figure 15. Performance of the SVM-based inverse dynamic control without perturbation.
Figure 16. Performance of the SVM-based inverse dynamic reinforced with precompensator without perturbation

Figure 17. Performance of the computed torque control with perturbation

Figure 18. Performance of the SVM-based inverse dynamic control with perturbation

Figure 19. Performance of the SVM-based inverse dynamic reinforced with precompensator with perturbation
Figure 20. Control effort for both joints without perturbation (computed torque)

Figure 21. Control effort for both joints without perturbation (SVM-based inverse dynamic)

Figure 22. Control effort for both joints without perturbation (SVM-based inverse dynamic with precompensator)

Figure 23. Control effort for both joints with perturbation (computed torque)
8. Conclusion

In this paper, we propose a novel approach to nonlinear system control based on model dynamic compensation using one type of regressor brought from machine learning concepts, reinforced by a fuzzy precompensator. The idea is based on the computed torque method, which has been one of the successful tools used to control robot manipulators, since it eliminates the nonlinear parts of the system allowing the mechanical system to be controlled solely by a servo feedback PD controller. Unfortunately, the computed torque method gives rise to some drawbacks due to uncertainties in parameters and unforeseen perturbations. In this work, we propose a kernel regressor which is used to predict the feedforward torques when the desired trajectories are given. The learning is done offline and the results are most likely comparable to the computed torque method. The introduction of the fuzzy precompensator within the controlled system demonstrated improved performances due to fast transient responses and disturbance rejection. For the purpose of comparison, we included in the simulation results obtained by the computed torque control method. The simulation results show a better tracking trajectory, achieved by the SVM-based inverse dynamics control whenever any variation in the parameters occurred. Disturbance rejection and sensitivity were further improved when we inserted the fuzzy precompensator as part of the controller in order to reinforce the previous actions. All the simulation results were presented to demonstrate and prove the simplicity and the applicability of the SVM regression approach reinforced by a fuzzy precompensator to stabilize the robot manipulator with reduced steady-state error and less demand on effort.

9. Acknowledgments

This work was supported by the NPST programme of King Saud University, Project No: 08-ELE-300-02.

10. References


