A combined SVM regression methodology for modelling robotic manipulators inverse dynamics

D. A. Karras*

[*]**Hellenic Aerospace Industry, Hellenic Open University and Chalkis Institute of Technology Dept. Automation, Rodu 2, Ano Iliupolis, 16342 Athens, Hellas,**

Abstract: A novel approach is presented for approximating real continuous functions of many variables using a two-stage neural network technique. In the first stage, a group of Support Vector Machines (SVMs) is involved in approximating the function based on different methods for selecting and transforming the input variables. In the second stage, an SVM is employed to combine the function approximation decisions of the previous stage's group of SVMs. Different kinds of Artificial Neural Networks are used at this stage, including SVM, Multilayer Perceptron (MLP) and Perceptron. Their performance is compared with simple averaging and other statistical methods for combining the outputs of many regressors. The proposed methodology is applied to the design of a neural-adaptive controller which combines the two-stage neural network model with a servo PD feedback controller. The different feature extraction methods involved in this case include the Fourier as well as the Wavelet transform of the input variables, which are samples belonging in the mappings of position, velocity and acceleration as functions of time.

I. ISSUES ON **CONTROL OF ROBOTIC MANIPULATORS**

- Robot manipulators have become increasingly important in the field of flexible automation. High speed and high precision trajectory tracking are indispensable capabilities for versatile applications of manipulators.

- It is very common that manipulators are subject to structured and/or unstructured uncertainties. Structured uncertainty is characterised by having a correct dynamical model but with parameter uncertainty due to imprecision of the manipulator link properties, unknown loads, inaccuracies in the torque constants of the actuators, and so on. Unstructured uncertainty is characterised by unmodelled dynamics.

- Although conventional adaptive controllers are effective in compensating for the influence of structured uncertainty, it is not clear that adaptive means alone can overcome the effects of unstructured uncertainty.

- Multiple Layer Perceptrons (MLPs) have been used for the construction of Neural - Adaptive Controllers to cope with both types of uncertainty

II. WHY TO USE NEURAL NETWORKS

- The design of current industrial robots, is based on simple joint servo-mechanisms assigned to the different joints of the arm. This results in reduced servo-response speed, thus limiting the precision and the accuracy of the end effector and producing sub-optima performance. The increasingly sophisticated tasks required of robot manipulators have called for better control techniques to enhance high-speed tracking accuracy whilst operating in uncertain enviroments.

- Many control schemes have been developed to overcome this problem. However, the performance of all these methods is highly dependent on the accurate modeling of robot dynamics.

- The advantage of a neural adaptive control technique for robot manipulators is that it avoids this *a priori* modeling problem. The control technique is generic in the sense that the controller parameters are not dependent on any parameter estimation, as opposed to many conventional adaptive control methods.

III. THE **COMPUTED TORQUE MODEL**

A very convenient starting point for the development of an ANN controller is the computed-torque algorithm. The computed torque method is a well established robot control technique which takes account of the dynamic coupling between the robot links. Its main disadvantage is the assumption of an exactly known dynamic model, which is not realisable in practice. However, the basic algorithm remains important, as it forms the basis of various adaptive and neural controllers which have been developed to overcome the requirement for an exactly known dynamic model.

In this model the robotic manipulator is consisted of n connected rigid bodies. The first end is fixed in its base while the last end is assumed to be free. These rigid links are connected through revolute or prismatic joints, a mechanical torque is applied on each joint. The dynamic ***Euler-Lagrange*** equation of the robotic manipulator is given by

$$\mathbf{M(q)\ddot{q} + V(q,\dot{q}) + F_v(\dot{q}) + F_d(\dot{q}) + G(q) + \hat{o}_d = \hat{o}} \tag{1}$$

Or equivalently

$$\mathbf{M(q)\ddot{q} + N(q,\dot{q}) + \hat{o}_d = \hat{o}} \tag{2}$$

where $q(t) \in R^n$ is the joint variable.

$\mathbf{V(q,\dot{q})}$	(nx1) representing the effects of Centipetal and Coriolis.
$\mathbf{M(q)}$	(nxn) Inertia matrix
$\mathbf{F_v(\dot{q}),F_d(\dot{q})}$	(nx1) friction vectors
$\mathbf{G(q)}$	(nx1) gravity vector
$\mathbf{q,\ \dot{q},\ \ddot{q}}$	(nx1) vectors containing the position, velocity and acceleration joint variables
$\boldsymbol{\tau(t),\ \tau_d(t)}$	(nx1) vectors of torque (input) and disturbance
$\mathbf{N(q,\ \dot{q})}$	(nx1) vector of containing all of the nonlinear terms.

Inner and Outer Loop Design

Performing feedback linearization via the use of an appropriate control input function **u** produces a linear system with respect to the joint variable tracking error and decomposes the problem in an inner and an outer loop design problems

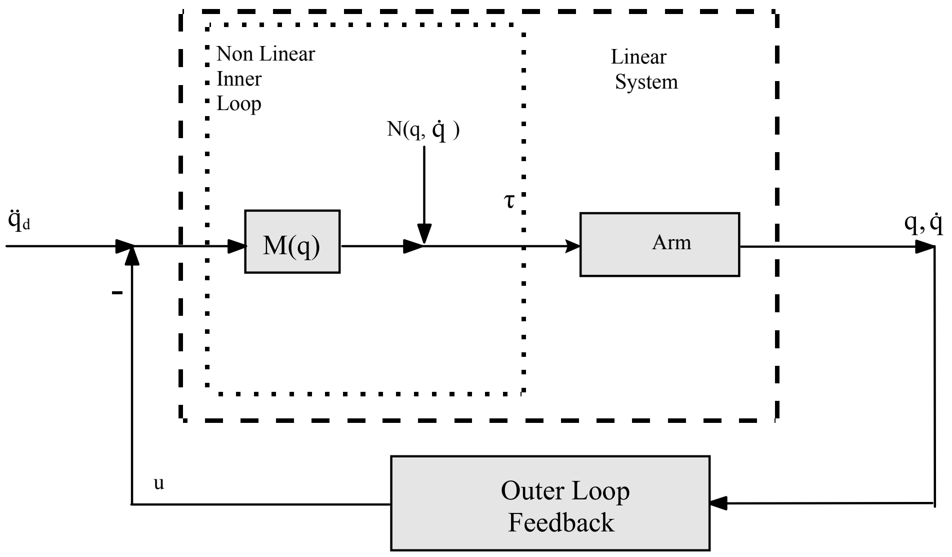


Figure 1. Computed torque control scheme, showing inner and outer loops

For example if

$$\mathbf{u} = \ddot{\mathbf{q}}_d + \mathbf{M}^{-1}(\mathbf{N} - \hat{\mathbf{o}}) \tag{3}$$

Defining

$$\mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} \tag{4}$$

then

$$\frac{d}{dt}\begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix}\mathbf{u} + \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix}\mathbf{w} \tag{5}$$

Where

$$\mathbf{w} = \mathbf{M}^{-1}\hat{\mathbf{o}}_{\mathbf{d}} \tag{6}$$

PID outer Loop Design

A very common procedure is to use a PD or PID Outer Loop design. In these cases the control function **u** produced by the outer loop takes the form

$$\mathbf{u} = -\mathbf{K}_{\mathbf{v}}\dot{\mathbf{e}} - \mathbf{K}_{\mathbf{p}}\mathbf{e} \text{ or } \mathbf{u} = \mathbf{K}_{\mathbf{v}}\dot{\mathbf{e}} + \mathbf{K}_{\mathbf{p}}\mathbf{e} + \mathbf{K}_{i}\boldsymbol{\varepsilon} \tag{7}$$

The computed torque control scheme is depicted by the following figure

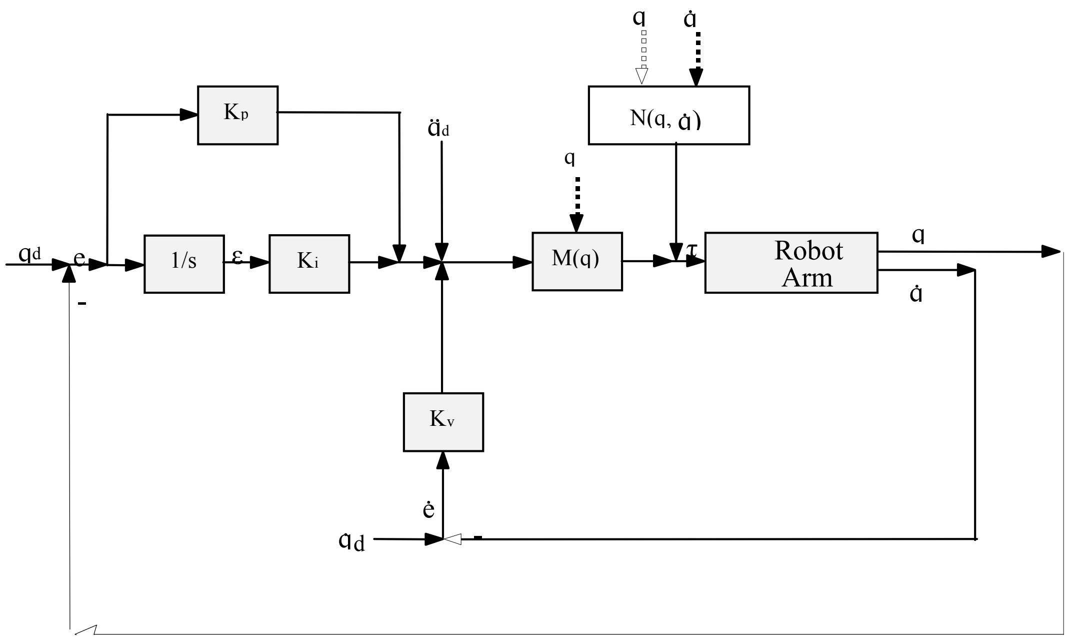


Figure 2. PID Computed torque control.

In the PD controller the overall robot arm input becomes

$$\hat{\mathbf{o}} = \mathbf{M}(\mathbf{q})(\ddot{\mathbf{q}}_{d} + \mathbf{K}_{\mathbf{v}}\dot{\mathbf{e}} + \mathbf{K}_{\mathbf{p}}\mathbf{e}) + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) \tag{8}$$

$$\ddot{\mathbf{e}} + \mathbf{K}_{\mathbf{v}}\dot{\mathbf{e}} + \mathbf{K}_{\mathbf{p}}\mathbf{e} = \mathbf{w}, \text{ where } \mathbf{w} = \mathbf{M}^{-1}\hat{\mathbf{o}}_{\mathbf{d}} \tag{9}$$

Or in a state space form

$$\frac{d}{dt}\begin{bmatrix}\mathbf{e}\\ \dot{\mathbf{e}}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ -\mathbf{K}_{p} & -\mathbf{K}_{v}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \dot{\mathbf{e}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{w} \tag{10}$$

IV. THE FEEDBACK-ERROR-LEARNING METHOD

The method was first proposed by Kawato and co-workers (1987). The basic idea is to combine an already available and tuned conventional feedback controller with an ANN acting as the feed-forward controller.

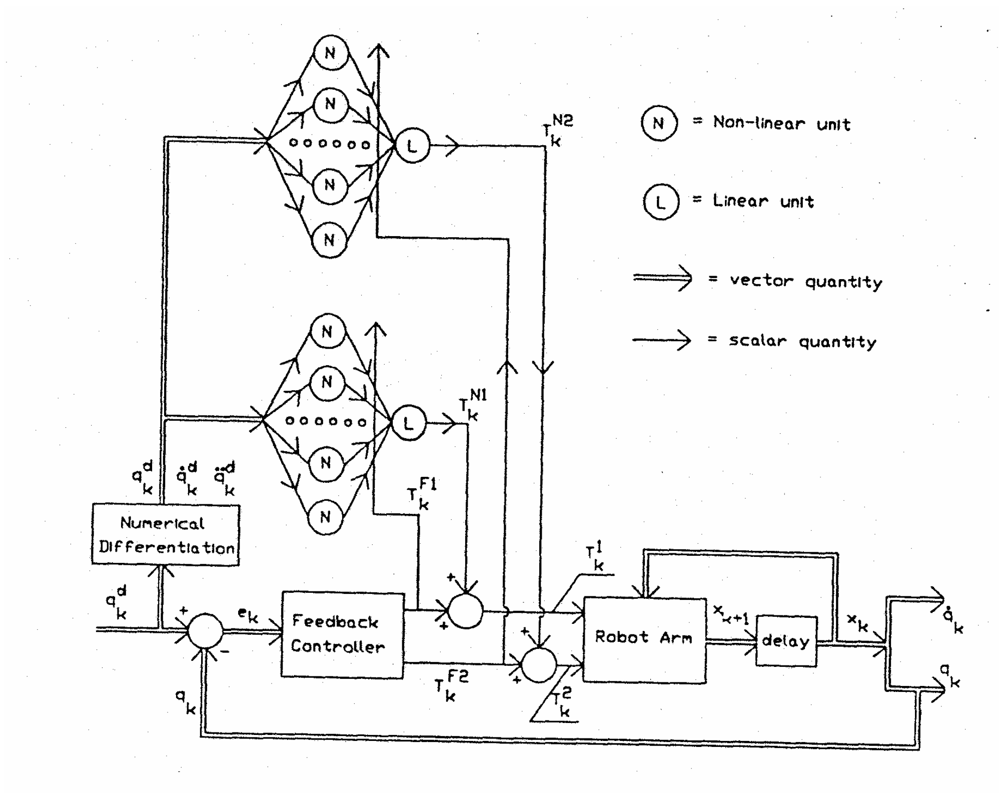
- The feedback controller should at least be good enough to stabilize the plant when used alone, but it does not need to be optimally tuned. For simplicity such a feedback controller is normally a PID controller.

- The aim is to adapt the ANN in order to minimize the tracking error $\|\mathbf{e}\|$ defined as the difference between a reference signal and the measured output (usually a subset of the state vector).

- It was proposed that the output of the feedback controller is used as the ANN output error and therefore it was called the Feedback-Error-Learning method.

- Using the feedback error signal as the ANN output error the problem of back-propagating the control error through the plant (or through the model of the pant), used in earlier ANN control techniques, is avoided.

- Before being trained, the ANN is initialized such that its output is zero for any input. Hopefully, as the ANN is being trained, it will smoothly take over control from the feedback controller and at the same time improve the overall control performance. In this way the ANN is being trained to be the inverse dynamical model of the plant.

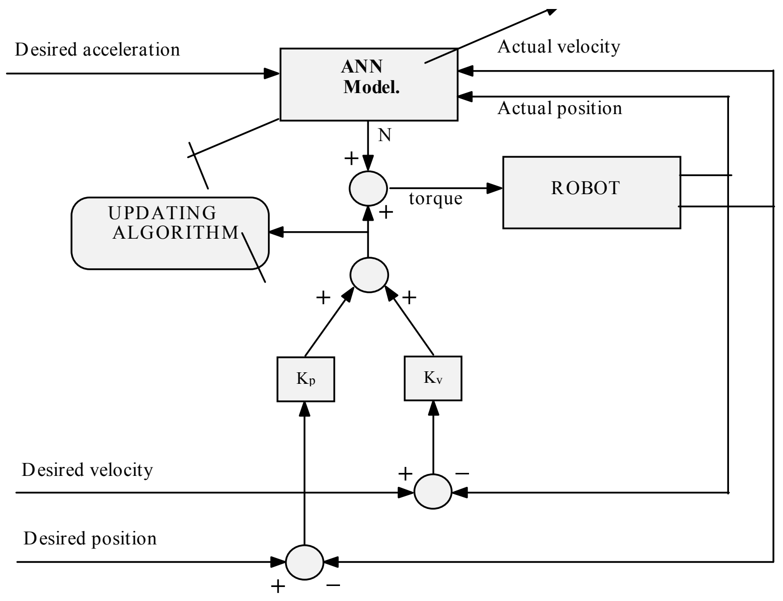


In the above scheme Kawato proposed composing the ANN input by using, at each time step, the desired angle joints and their respective desired velocities and desired accelerations.

Since in order to learn the true inverse dynamical model of the plant it is necessary to associate the vector $\left[(x_k)^T \quad (x_{k+1})^T\right]$ to T_k, the implicit assumption of the feedback-error learning method is that the feedback controller is adequate so that, when used alone to control the plant, the plant approximately follows the correct trajectory, i.e. $\left[(x_k^d)^T \quad (x_{k+1}^d)^T\right] \approx \left[(x_k)^T \quad (x_{k+1})^T\right]$. Therefore one can state that the role of the feedback controller is to provide an approximate solution for the problem. Because of this, when used within such a control structure, the ANN tends to be trained more rapidly than in other situations.

When trained this scheme is adequate for a specific trajectory. More training is demanded when a new trajectory is to be followed.

V. INVERSE DYNAMIC MODEL APPROXIMATION AND PARAMETER ADAPTATION AT THE SAME TIME



Neuro-controller based computed torque architecture

In this scheme ([Morris, Zalzala 1996], the ANN performs on-line approximation of the inverse dynamics of the manipulator and at the same time there is an adaptive element that estimates the true values of the unknown system parameters.

- It requires an on-line ANN training algorithm. Approaches taken from least squares filtering are proposed for weight adaptation.
- This requires that the output of the ANNs have to be linearized.
- The parameter adaptation mechanism (in order to be feasible) relies on its linearization around a nominal trajectory. This corresponds to a condition where the system operates in the vicinity of the desired trajectories and the ANN parameters are close to their desired values.

VI. DRAWBACKS OF CURRENT NEURO-CONTROLLER METHODOLOGIES IN THE COMPUTED-TORQUE ALGORITHM

MLPs and RBFs are universal function approximators It is practically, however, not easy to train them to approximate the arbitrarily complex continuous functions met due to the uncertainties in control tasks, since

- the lack of universally efficient learning algorithms results in poor approximation performance.

- starting from a random initial point in the weight space, the path to the global minimum is often strewn with many local minima (especially for MLPs), imposing an oscillatory convergence for the weight update algorithm and therefore, results in a very slow learning process.

VII. THE PROPOSED METHOD PRINCIPLES

- Improvement of ANN function approximation capabilities by combining different function approximation methodologies, based on different approximation features.
- A two stage methodology should be involved
  - In the first stage, a group of Support Vector Machines (SVMs) is involved in approximating the function based on different methods for selecting and transforming the input variables
  - In the second stage, an SVM is employed to combine the function approximation decisions of the previous stage's group of SVMs.
- The different feature extraction methods involved in this case include the raw input variables, their Fourier as well as their Wavelet transform. The input variables, are samples belonging in the mappings of position, velocity and acceleration as functions of time.
- We call these different feature extraction methods for function approximation as FTM [Feature Transformation Methods]

VIII. THE PROPOSED TWO-STAGE METHODOLOGY USING COMBINATION OF SVMs

- With the SVMs, the task of nonlinear regression could be defined as follows: Let f(X) be a m-D scalar valued function to be approximated. Then, a suitable regression model to be considered is: D = f(X) + n, where X is the input vector, n is the noise. Given, the training sample set {(Xi, Di)} (i=1,..,N) then, the SVM training is the optimization problem: Find the Lagrange Multipliers {λi} (i=1, ..,N) and {λ'i} (i=1, ..,N) that maximize the objective function, $Q(\lambda_i , \lambda'_i) = \Sigma_{\iota=1..N} D_i (\lambda_i - \lambda'_i) – e\, \Sigma_{\iota=1..N} (\lambda_i + \lambda'_i) – ½\, \Sigma_{\iota=1..N} \Sigma_{j=1..N} (\lambda_i - \lambda'_i) (\lambda_j - \lambda'_j) K(\mathbf{X}_i, \mathbf{X}_j)$ subject to the constraints: $\Sigma_{i=1..N} (\lambda_i - \lambda'_i) =0$ and $0<= \lambda_i <=C$, $0<= \lambda'_i<=C$ for i=1..N, where C a user defined constant. $K(\mathbf{X}_i, \mathbf{X}_j)$ are the kernel functions, in our case the radial basis kernel $K(\mathbf{X}, \mathbf{X}_j) = \exp(-1/2\sigma^2 \,||\, \mathbf{X} - \mathbf{X}_j||^2)$. => => $F(\mathbf{X}) = \Sigma_{i=1..N} (\lambda_i - \lambda'_i) K(\mathbf{X}, \mathbf{X}_i)$

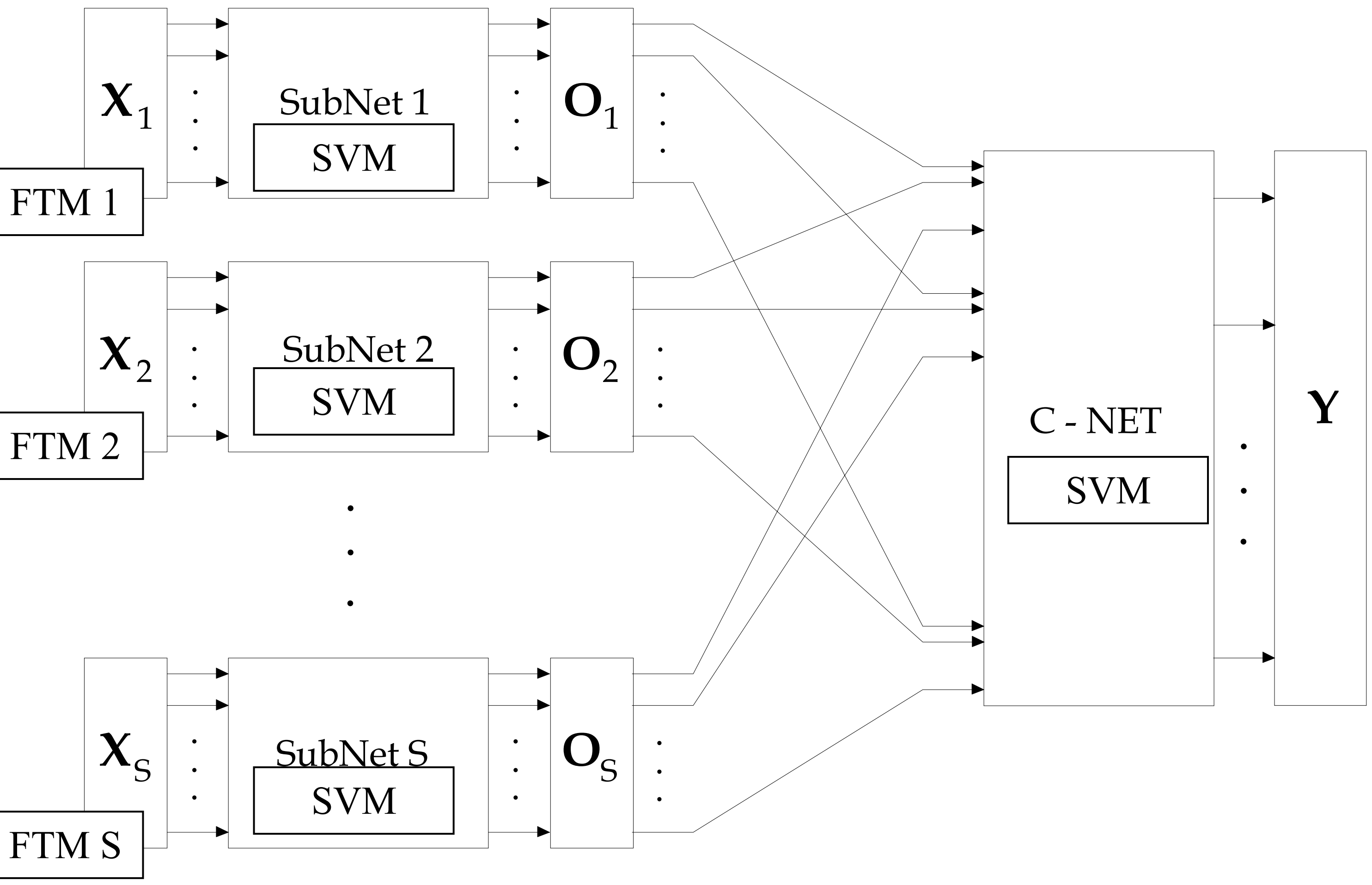


Figure 1. The schematic architecture of the two-stage function approximation system.

- The ANN controller inputs are the sampled trajectories of the actual position, velocity and desired acceleration up to time instance t. Desired output: N(t) = $T^{(true)}(t)-T^{(false\text{-}PD\ estimation)}(t)$. This comprises FTM 1.
- FTM 2 comprises the DFT transform of the sample trajectories
- FTM 3 comprises the DWT [Daubechies Basis] transform of the sample trajectories
- FTM 4 comprises the DWT [Coifflet Basis] transform of the sample trajectories

IX. SIMULATIONS

1) A two-link planar elbow arm was used. The manipulator was modeled as a two rigid links of lengths l_1 = 1m and l_2 = 1m, point masses m_1 =0.8kg and m_2 = 2.3kg corresponding to the false PD model. True masses varied within the 10% interval of these values. 12 variations were considered for each such link mass. A fourth order Runge-Kutta algorithm, step size h= 0.01, has been invoked.

2) The desired position trajectory has the components

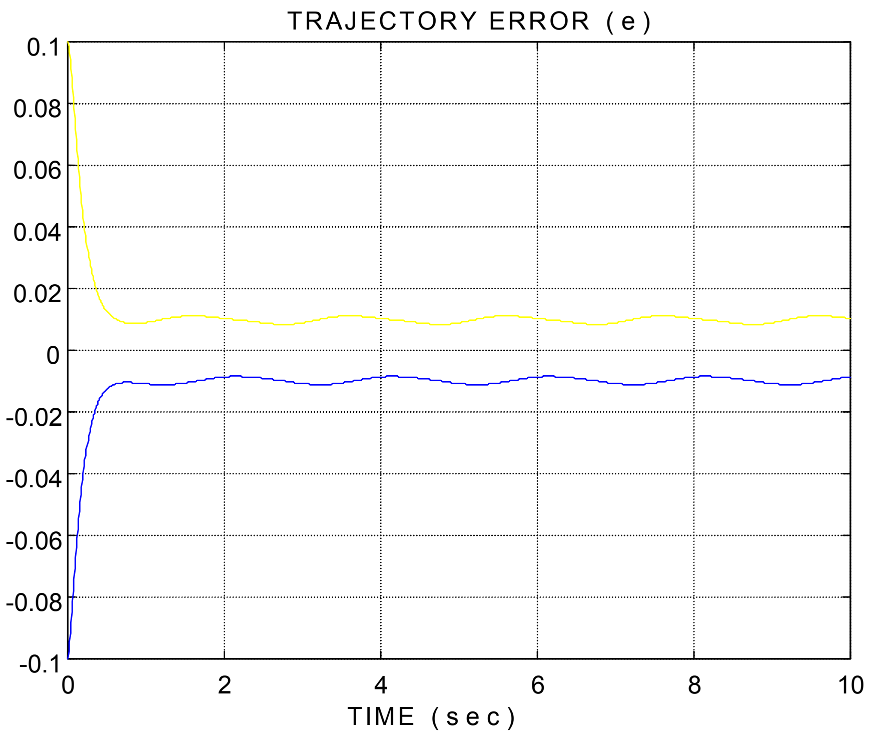
$$\theta_1 = g_1 \sin(2\pi t / T) \text{ and } \theta_2 = g_2 \sin(2\pi t / T)$$

with period T = 2s and amplitudes g_i = 0.1 rad. The time constant of the closed-loop system was selected as 0.1s.

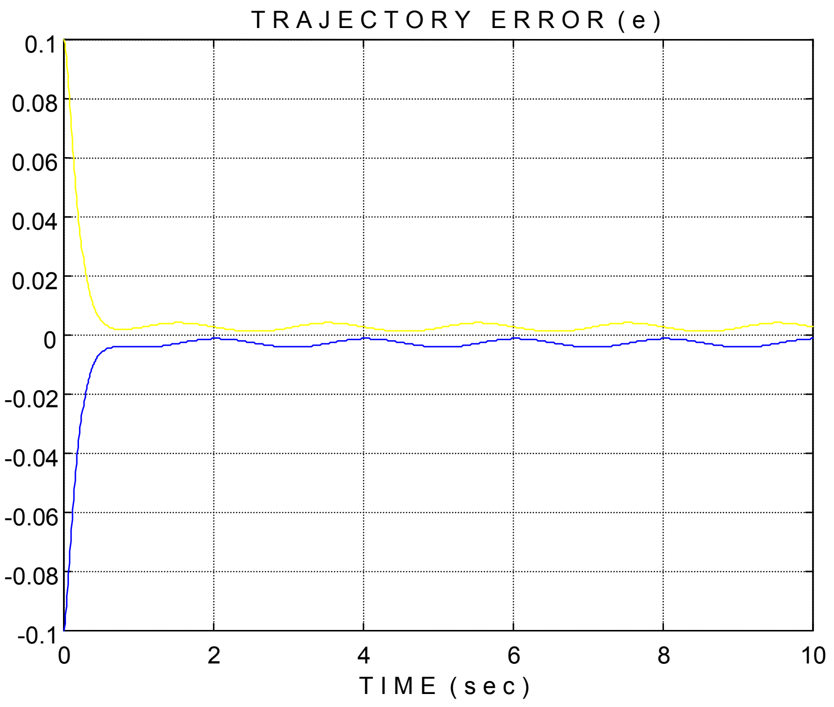
3) For every one of the 144 total pair masses variations the curves of desired acceleration, actual trajectory and actual velocity along with the associated torque N to be modelled are obtained. Each such curve has been sampled into 30 points, from which, using sliding windows of length l_w, we have formed training and test patterns for the ANN as follows.

4) Each ANN input pattern contains l_w points for the desired acceleration, the corresponding l_w points for the actual trajectory and finally, the l_w corresponding points for the actual velocity. The desired output value is the associated torque N of the l_w -th sample. In our simulations l_w = 6 and we have used 2000 patterns for ANN training and the rest 1600 patterns for testing.

5) The method was tested on a model where the false masses were assumed to be 10% lower than the true values (i.e the maximum deviation considered here). The figures show the trajectory errors for the two joints, when a correction torque, provided by the corresponding approximation approach, is added at the torque given by the PD controller.



Trajectory error when the correcting torque provided by one only approximating SVM (MLP produces worse results) (one-stage methodology) is applied



Trajectory error when the correcting torque provided by the combined SVMs two-stage methodology is applied