Fuzzy-Neuro Optimal Time-Energy Control of a Three Degrees of Freedom Planar Manipulator

Amar KHOUKHI, Luc BARON, Marek BALAZINSKI
Department of Mechanical Engineering,
École Polytechnique de Montréal,
C. P. 6079, Succ. CV Montreal PQ, Canada H3C 3A7
(amar.khoukhi, luc.baron, marek.balazinski)@polymtl.ca

Kudret DEMIRLI
Department of Mechanical and Industrial Engineering
Concordia University,
1455 de Maisonneuve Blvd. W., EV4.173, Montreal, PQ,
Canada H3G 1M8
demirli@alcor.concordia.ca

Abstract - In this paper, the optimal time-energy trajectory planning problem for a three degrees-of-freedom planar manipulator is considered using a multi-level fuzzy-neural system. First, a neural network is achieved to solve the inverse kinematics problem (IKP). After pre-processing steps characterizing the minimum time trajectory and the corresponding torques, a fuzzy-neuro controller is developed to deal with the minimum energy trajectory planning. The basic function of this controller is to select the minimum energy link actuator torques allowing the robot end-effector to move from a starting point $S$ to a target point $T$. This optimization is performed under actuators and workspace constraints. It starts with a fuzzy clustering of input-output data, uses Tsukamoto fuzzy inference and back-propagation algorithm to train the resulting adaptive neural network. The premises parameters (antecedent membership functions parameters) as well as rule-consequence parameters are then learned and optimized, giving the optimal energy torques. Simulation results are provided to prove the efficiency of the proposal.

Index Terms: Robot Manipulators, Neural Inverse Kinematic, Minimum Time-Energy Trajectory Planning, Fuzzy-Neuro Control.

I. INTRODUCTION

The primary weakness of conventional robot’s trajectory planning methods is the massive amount of computer time needed to obtain a solution. As robot tasks are demanding in precision and diversity, these tasks execution might be difficult to achieve because of variations of the dynamic parameters, such as Coriolis and centrifugal wrenches, inertia parameters, and friction forces [1, 2]. The dynamic coupling between robot’s axes makes even more complicated the design of a trajectory planning or a control system for the robot [3, 4]. To overcome these difficulties, several researchers have been used soft computing techniques. Neural networks and fuzzy control constitute the widely applied artificial intelligence techniques in control systems of industrial manipulators, mobile robots and many related engineering fields [5-8]. However, the whole neuro-fuzzy systems reported in the literature to solve the trajectory design problem of robotic manipulators are basically geometric and kinematic based and did not consider dynamic parameters and actuator limitations, neither time or energy minimization [1, 2]. In this article, the use a multi-level fuzzy-neuro time-energy trajectory planner for a three degrees-of-freedom planar manipulator is proposed. First, a neural network is used to solve the inverse kinematics problem (IKP). The IKP consists in transforming Cartesian coordinates (within the task execution space) into the corresponding joint coordinates (in the actuation space) of the manipulator. Then the Fuzzy-Neuro Optimal Time-Energy Controller (FuNOTEC) is introduced to generate an optimal time-energy trajectory allowing the robot end-effector to move from a starting point $S$ to a target point $T$. This optimization is performed under actuator dynamics constraints and workspace limitations. The idea in the proposed fuzzy-neuro controller is that the rule base of the fuzzy logic system is replaced with a feed-forward neural network [9-12]. The proposed system starts with a fuzzy sub-clustering of input-output data, uses Tsukamoto fuzzy inference and back-propagation algorithm to train the resulting adaptive neural network. The premise parameters (antecedent membership functions parameters) as well as rule-consequences parameters are then learned and optimized, giving the optimal energy torques.

In the next section, the trajectory definition and the minimum time trajectory planning are considered. Section 3 develops the inverse kinematic problem where the neural inverse kinematic, namely NetIKP, is proposed. Section 4 considers dynamic trajectory generation. In section 5, the fuzzy-neuro control system is introduced. In section 6, preliminary simulation results are provided to show the effectiveness of the proposed approach, and section 7 concludes this paper.

* This work is supported by Natural Science and Engineering Council of Canada (NSERC) grants ES D3-317622, RGPIN-203618, RGPIN-105518 and STPGP-269579.
II. OPTIMAL TIME TRAJECTORY DEFINITION AND COMPUTING

A three degrees-of-freedom planar manipulator operating in a two dimensional space is considered (Fig. 1). The robot task is to execute a pick and place operation in a limited workspace and within a normalized time interval \([0, 1]\). Furthermore, each robot joint motion is to produce zero rates and accelerations at the ends of the interval. For that reason a cycloidal function is chosen to parameterize the joints motion. In normal form, this function is given by:

\[
s(t) = t - \frac{1}{2\pi} \sin(2\pi t)
\]

Its first and second derivatives being easily derived as:

\[
s'(t) = 1 - \cos(2\pi t)
\]

\[
s''(t) = 2\pi \sin(2\pi t)
\]

The cycloidal motion and its first two time-derivatives are normalized to fall within the range \((-1, 1)\). While this motion, indeed, has zero velocity and acceleration at the ends of the interval \([0, 1]\), its jerk is nonzero at these points implying jump discontinuities at the ends of that interval. This issue will be considered in the remainder of this paper when developing the fuzzy-neuro optimal energy controller for the mechanical system.

Moreover, by defining the initial and target joint positions respectively as \(q^i_j, q^f_j\), \(j = 1, 2, 3\), the reader can readily verify, under the assumption that \(\dot{q}^i_j > \dot{q}^f_j\) this motion attains its maximum velocity at the center of the interval, i.e., at \(t = 0.5\), the maximum being \(s_{\text{max}} = s'(0.5) = 2\), and hence,

\[
(q^j_j)_{\text{max}} = \frac{2}{T}(q^f_j - q^i_j)
\]

Likewise, the \(j\)th joint acceleration attains its maximum and minimum values at \(t = 0.25\), and \(t = 0.75\), respectively, i.e.,

\[
s_{\text{max}}' = s'(0.25) = s'(0.75) = 2\pi
\]

Hence,

\[
(q^j_j)_{\text{max}} = \frac{2\pi}{T}(q^f_j - q^i_j), \quad (q^j_j)_{\text{min}} = -\frac{2\pi}{T}(q^f_j - q^i_j)
\]

Thus, if the motion is constrained by the maximum joint rates and accelerations delivered by the motors, the minimum time \(T_j\) for the \(j\)th joint to produce the given task can be readily determined by considering boundary conditions for both joint rates and accelerations such that:

\[
|\dot{q}^j_j| \leq |\dot{q}^j_{\text{max}}|, \quad \text{and} \quad |\ddot{q}^j_j| \leq |\ddot{q}^j_{\text{max}}|
\]

with

\[
|\dot{q}^j_{\text{max}}| = \frac{2}{T}(q^f_j - q^i_j)
\]

and

\[
|\ddot{q}^j_{\text{max}}| = \frac{2\pi}{T}(q^f_j - q^i_j)
\]

It follows that the minimum time trajectory for the \(j\)th joint considering velocity and acceleration boundary conditions simultaneously is:

\[
T_j = \min \left\{ \frac{2(q^f_j - q^i_j)}{|\dot{q}^j_{\text{max}}|}, \frac{2\pi}{|\ddot{q}^j_{\text{max}}|} \right\}
\]

And the overall minimum time (for all of the three joints simultaneously) is therefore given by:

\[
T_{\text{min}} = \max \left\{ T_j \right\}
\]

The minimum time trajectory is henceforth characterized in the joint space by joints, joint rates and accelerations variations would be:

\[
\begin{align*}
q^j_j(t) &= q^f_j + (q^f_j - q^i_j)(1 - \frac{t}{T_{\text{min}}}) - \frac{1}{2\pi} \sin(2\pi \frac{t}{T_{\text{min}}}) \\
q^j_j(t) &= \frac{q^f_j - q^i_j}{T_{\text{min}}} (1 - \cos(2\pi \frac{t}{T_{\text{min}}})) \\
q^j_j(t) &= \frac{q^f_j - q^i_j}{T_{\text{min}}} 2\pi \sin(2\pi \frac{t}{T_{\text{min}}})
\end{align*}
\]

On the other hand, the robot forward kinematics equations are (Fig. 1):

\[
\begin{align*}
x &= L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) + L_3 \cos(q_1 + q_2 + q_3) \\
y &= L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) + L_3 \sin(q_1 + q_2 + q_3)
\end{align*}
\]

where \(L_1 , L_2 , \text{and} L_3\) represent respectively the lengths of the first, second and third link of the robot.

III. NEURAL INVERSE KINEMATICS PROBLEM

The function NetIKP gets the Cartesian coordinates \((x, y)\) of the robot end-effector defined within the task space as inputs to give the corresponding joint variables \((q_1, q_2, q_3)\) as outputs. The forward kinematics (FK) of equations (15) act as a
supervisor, generating thereby sequences of \((x, y)\) and their corresponding joint variables \((q_1, q_2, q_3)\), (a set of \((q_1, q_2, q_3)\)) is used to construct the true \((x, y)\), and thus to get an error on which to apply the back-propagation algorithm). NetIKP is a multi-layer perceptron. The first layer is a two input entry layer. The last layer is a three output-layer. Furthermore, it involves two hidden layers. The first one contains 60 units, with log-sigmoidal transfer function, while the second contains 55 units with tan-sigmoidal transfer function. The transfer function from the second hidden layer to the output layer is the log-sigmoidal. Figure 2 shows NetIKP function. The training rule is the Levenberg-Marquard version of the gradient back-propagation algorithm [13]. This choice allows speeding up the learning process substantially with less iteration as compared to standard (gradient descent) back-propagation. The training data are obtained by sampling the unit circle into 20 equal samples, applying forward kinematic equations then removing singular configurations that zero the robot jacobian to finally construct NetIKP. The function \texttt{trainlm} and command \texttt{sim} from Neural Networks toolbox of Matlab [14] are used to train and simulate NetIKP. The training error is shown in Figure 3.

where the vector \(\tau=(\tau_1, \tau_2, \tau_3)^T\) corresponds to the torques produced by the joint actuators, \(q, \dot{q}, \ddot{q}\) are \(3\times1\) columns vector describing respectively joint, joint rates and accelerations. This model takes into account of the \(3\times3\) inertia matrix \(D(q)\), \(3\times1\) Coriolis and Centrifugal wrenches \(C(q)\), \(3\times1\) gravitational wrench \(G(q)\), and the \(3\times1\) friction force \(F_{fric}(q)\) (including viscous and Coulomb frictions).

The robot inverse dynamic model is detailed in Ref. [13]. From the generated kinematic-based minimum time trajectory according to section II, the inverse dynamic solution (16) allows us to generate dataset sequences associating each \((q_i, q_{\dot{i}}, q_{\ddot{i}})\) \(i\in\{1, 2, 3\}\) to its corresponding actuator torques \((\tau_i)\) \(i\in\{1, 2, 3\}\). The obtained torques is then projected into an admissible domain of torque limits (which might be provided by the manufacturer). In order to keep the constraint \(\tau_{\text{Max}} \leq \tau \leq \tau_{\text{Min}}\) satisfied, the following projections are performed:

\[
\tau_i = \text{Max}(\text{Min}(\tau_i, \tau_{\text{Max}}), \tau_{\text{Min}}), \ i \in \{1, 2, 3\}
\]  

(17)

In order to have the necessary minimum energy torques to execute the trajectory, a fuzzy-neuro controller is required.

V. FUZZY-NEURO OPTIMAL TIME-ENERGY TRAJECTORY PLANNING (FuNOTEC)

A. Introduction

The definition of the fuzzy membership functions used for the fuzzification and defuzzification of the input and output variables plays a significant role in the ability of a fuzzy-neuro controller to learn and generalize. An adaptive neuro-fuzzy network based on Tsukamoto fuzzy inference is considered to learn the premise parameters as well as the crisp outputs of the fuzzy rules. The proposed system, called Fuzzy Neural Optimal Time Energy Controller (FuNOTEC) starts with a fuzzy sub-clustering of input-output data, then applies Tsukamoto fuzzy inference with back-propagation gradient for forward passes and minimum least square for consequent part parameters to train the resulting adaptive neural network and giving thereby the optimal energy actuator torques.

From the previous steps, one gets datasets of t-uples \((q, \dot{q}, \ddot{q})\) and torques \(\tau_1, \tau_2, \tau_3\) characterizing the minimum time trajectory for a given starting and target end-effector positions as:

\[
X = (q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3) = (x_1, x_2, x_3, x_4, x_5, x_6).
\]

Let us consider the following three datasets: \((X_1, \tau_1, \ldots, X_K, \tau_K)\), \((X_1, \tau_1, \ldots, X_K, \tau_K)\) and \((X_1, \tau_1, \ldots, X_K, \tau_K)\). In FuNOTEC, fuzzy logic rules with
constant consequence part are adopted in the following format:

\[ R^i: \text{If } (x_1 \in A^i_1) \text{ and } (x_2 \in A^i_2) \text{ and } \ldots \text{ and } (x_n \in A^i_n) \]

Then (\( \tau_i = Z^i_1 \)) and (\( \tau_2 = Z^i_2 \)) and (\( \tau_3 = Z^i_3 \))

also \ldots 

\[ R^j: \text{If } (x_1 \in A^j_1) \text{ and } (x_2 \in A^j_2) \text{ and } \ldots \text{ and } (x_n \in A^j_n) \]

Then (\( \tau_j = Z^j_1 \)) and (\( \tau_2 = Z^j_2 \)) and (\( \tau_3 = Z^j_3 \))

\ldots 

\[ R^n: \text{If } (x_1 \in A^n_1) \text{ and } (x_2 \in A^n_2) \text{ and } \ldots \text{ and } (x_n \in A^n_n) \]

Then (\( \tau_1 = Z^n_1 \)) and (\( \tau_2 = Z^n_2 \)) and (\( \tau_3 = Z^n_3 \))

Layer 4: This is the normalization layer. Every node in this layer is a circle labelled \( N \). The \( j \)-th node calculates the ratio of the \( j \)-th rule’s firing strength to the sum of all rules’ firing strengths:

\[ \beta_j = \frac{\alpha_j}{\sum_{j=1}^{N} \alpha_j} \]

The outputs \( \beta_j \) of this layer are called the normalized firing strengths.

Layer 5: This is the consequent layer. Every node in this layer is a square and its outcome is given by: \( o_j = \beta_j Z^j_1 \), where \( \beta_j \) are the normalized firing strengths from layer 4 associated with the \( j \)-th rule. \( Z^j_1 \) are the consequent parameters, optimized by a last square estimate [10].

Layer 6: This layer acts as a defuzzifier. It has a single circle node labelled \( \Sigma \). At this node, the overall system output is obtained as the summation of all incoming signals i.e.

\[ o_k = \sum_{j=1}^{N} \beta_j Z^j_1 \]

**C. FuNOTEC Learning Procedure**

FuNOTEC learning proceeds with two phases: the initializer phase and the optimizer phase.

**Initializer Phase**

The initializer phase is for structure identification. It finds proper input space fuzzy partitions and fuzzy logic rules with two objectives: (1) - minimize the number of rules generated and (2) - minimize the number of fuzzy sets on the universe of discourse for each input variable. To initialize the premise parameters \( a^i_j \) and \( b^i_j \) of the membership functions in the antecedent part and the conditional parameters \( Z^i_1 \) in the consequent part of the fuzzy rule \( R^j \), a clustering method is applied. A basic advantage of using a clustering method as a pre-processor for determining the initial rules is that the resultant rules are more tailored to the input data than they are in a fuzzy inference system generated without clustering. The second advantage is that it reduces the problem of combinatorial explosion of rules when the input data has a high dimension (as in the case at hand). This phenomenon is commonly known as the curse of dimensionality problem. We assume that we don’t have a clear idea on how many clusters there should be for our input-output dataset generated from the pre-processing steps. To find the number of clusters and the cluster centers in the dataset, we apply the subtractive clustering method developed in Ref. [11]. This approach provides a fast; one-pass algorithm to take input-output training data to generate our Tsukamoto-type fuzzy inference system that models the robot dynamic behaviour. In this method, a cluster radius indicates the range of its influence. Specifying a small cluster radius yields many small clusters in
the data, (resulting in many rules). And vice versa, specifying a large cluster radius yields a few large clusters in the data, (resulting in fewer rules). In the performed simulations runs, after several trials, the cluster radius 0.4 had been found as more convenient for our dataset. The Matlab function subelust is used to implement this method and initialize our fuzzy identification model.

**Optimizer Phase**
The optimizer phase is a supervised learning iterative algorithm. This provides adaptive rules by adjusting the parameters of the network based on a given input-output dataset pairs. The link weights in the consequent part and the parameters of membership functions are adjusted to minimize the necessary electric energy function to have the robot end-effector moving from a starting point $X_s$ to a target point $X_t$. FuNOTEC works with a hybrid learning algorithm. First, in the forward pass, the antecedent premises parameters of the network based on a given input-output dataset pairs. The link weights in the consequent part and the parameters of membership functions are adjusted to minimize the energy function associated to the inputs $X_k$ and the real (or desired) output corresponding to the $k^{th}$ joint torque.

$$E_k = \frac{1}{2}(o^k - \tau^k)^2 = \frac{1}{2}\sum_{j=1}^{N}(o^k_j(a_j, b_j Z_j) - \tau^k_j)^2$$  \hspace{1cm} (19)

is minimized according to a Levenberg-Marquard back-propagation algorithm. The consequent parameters are then adapted by a last square estimate.

In equation (19) $o^k$ is the computed output from the fuzzy system associated to the inputs $X_k$ and $\tau_k$ is the real (or desired) output corresponding to the $k^{th}$ joint torque.

VI. SIMULATION RESULTS

Simulation results are reported in this section. The robot numerical values for the link lengths are: $L_1 = 0.432$ m , $L_2 = 0.45$ m , $L_3 = 0.45$ m , and their masses are $m_1 = 1.0$ kg , $m_2 = 0.8$ kg , $m_3 = 1.0$ kg respectively. The gravity acceleration is $g = 9.81m/s^2$. The viscous friction coefficients are $K_{v1} = 0.3$, $K_{v2} = 0.5$, $K_{v3} = 0.5$, while the dry friction coefficients are $K_{f1} = 0.3$, $K_{f2} = 0.5$, $K_{f3} = 0.5$, and finally the extremum joint rates and accelerations values are $|\dot{q}_{\text{Max}}| = 2$ rad/s and $|\ddot{q}_{\text{Max}}| = 3$ rad/s$^2$ respectively. The simulated scenario is to move the robot end-effector from a starting position (0.1, 0.1) to a target position (0.5, 0.7) (all in meters). For FuNOTEC learning, a 100 samples trajectory is used, among which 80 are considered for training. Whereas testing and validation datasets, each of them is obtained using 10 trajectory samples.

A sketchy representation of FuNOTEC is given in Fig. 4.

Three simulation experiments had been realized with the same data and conditions. The first one is devoted to $\tau_1$, whereas the second and the third are performed for the outputs $\tau_2$, and $\tau_3$ respectively. Figures 6 gives capture screen of energy variations for the simulated trajectory. One notices the decreasing of energy variations from the figure.

VII. CONCLUSIONS

In this paper, a multi-level approach to manipulator robot trajectory planning has been developed. This approach allows a minimum time-energy trajectory planning under actuator and workspace limitations. It consists of four major steps. First a pre-processing step starting from reading data related to link lengths and masses, initial and target points, and limits on actuator torques, joint, joint rates, and joint acceleration. Second, a neural network converts the initial and target Cartesian coordinates to joint coordinates. This allows the setting of the minimum time trajectory parameters in the second step. Third, the inverse dynamic transformer gives us the corresponding torques to joint, joint rates, and joint acceleration variables computed beforehand. This allows setting up the training and testing datasets on which to build the neuro-fuzzy controller. Fourth and last step, one finds the torques corresponding to the minimum energy trajectory and satisfying actuators and workspace limitations. Preliminary simulation results are encouraging. An ongoing work is to achieve more simulation experiments by including obstacle avoidance within the fuzzy-neuro optimal time-energy trajectory planning system of the presented approach.
REFERENCES


