We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

Application of Neural Networks to Modeling and Control of Parallel Manipulators

Ahmet Akbas
Marmara University
Turkey

1. Introduction

There are mainly two types of the manipulators: serial manipulators and parallel manipulators. The serial manipulators are open-ended structures consisting of several links connected in series. Such a manipulator can be operated effectively in the whole volume of its working space. However, as the actuator in the base has to carry and move the whole manipulator with its links and actuators, it is very difficult to realize very fast and highly accurate motions by using such manipulators. As a consequence, there arise the problems of bad stiffness and reduced accuracy.

Unlike serial manipulators their counterparts, parallel manipulators, are composed of multiple closed-loop chains driving the end-effector collectively in a parallel structure. They can take a large variety of form. However, most common form of the parallel manipulators is known as platform manipulators having architecture similar to that of flight simulators in which two special links can be distinguished, namely, the base and moving platform. They have better positioning accuracy, higher stiffness and higher load capacity, since the overall load on the system is distributed among the actuators.

The most important advantage of parallel manipulators is certainly the possibility of keeping all their actuators fixed to base. Consequently, the moving mass can be much higher and this type of manipulators can perform fast movements. However, contrary to this situation, their working spaces are considerably small, limiting the full exploitation of these predominant features (Angeles, 2007).

Furthermore, for the fast and accurate movements of parallel manipulators it is required a perfect control of the actuators. To minimize the tracking errors, dynamical forces need to be compensated by the controller. In order to perform a precise compensation, the parameters of the manipulator's dynamic model must be known precisely.

However, the closed mechanical chains make the dynamics of parallel manipulators highly complex and the dynamic models of them highly non-linear. So that, while some of the parameters, such as masses, can be determined, the others, particularly the firiction coefficients, can't be determined exactly. Because of that, many of the control methods are not efficient satisfactorly. In addition, it is more difficult to investigate the stability of the control methods for such type manipulators (Fang et al., 2000).

Under these conditions of uncertainty, a way to identify the dynamic model parameters of parallel manipulators is to use a non-linear adaptive control algorithm. Such an algorithm

Source: Parallel Manipulators, New Developments, Book edited by: Jee-Hwan Ryu, ISBN 978-3-902613-20-2, pp. 498, April 2008, I-Tech Education and Publishing, Vienna, Austria

can be performed in a real-time control application so that varying parameters can continuously be updated during the control process (Honegger et al., 2000).
Another way to identify the dynamic system parameters may be using the artificial intelligence (AI) techniques. This approach combines the techniques from the fields of AI with those of control engineering. In this context, both the dynamic system models and their controller models can be created using artificial neural networks (ANN).
This chapter is mainly concerned with the possible applications of ANNs that are contained within the AI techniques to modeling and control of parallel manipulators. In this context, a practical implementation, using the dynamic model of a conventional platform type parallel manipulator, namely Stewart manipulator, is completed in MATLAB simulation environment (www.mathworks.com).

2. ANN based modeling and control

Intelligent control systems (ICS) combine the techniques from the fields of AI with those of control engineering to design autonomous systems. Such systems can sense, reason, plan, learn and act in an intelligent manner, so that, they should be able to achieve sustained desired behavior under conditions of uncertainty in plant models, unpredictable environmental changes, incomplete, inconsistent or unreliable sensor information and actuator malfunction.
An ICS comprises of perception, cognition and actuation subsystems. The perception subsystem collects information from the plant and the environment, and processes it into a form suitable for the cognition subsystem. The cognition subsystem is concerned with the decision making process under conditions of uncertainty. The actuation subsystem drives the plant to some desired states.
The key activities of cognition systems include reasoning, using knowledge-based systems and fuzzy logic; strategic planning, using optimum policy evaluation, adaptive search, genetic algorithms and path planning; learning, using supervised or unsupervised learning in ANNs, or adaptive learning (Burns, 2001).
In this chapter it is mainly concerned with the application of ANNs that are contained within the cognition subsystems to modeling and control of parallel manipulators.

2.1 ANN overwiev

ANN is a network of single neurons jointed together by synaptic connections. Such that they are organized as neuronal layers. Each neuron in a particular layer is connected to neurons in the subsequent layer with a weighted synaptic connection. They attempt to emulate their biological counterparts.

2.1.1 Perceptrons

McCulloch and Pitts was started first study on ANN in 1943. They proposed a simple model of neuron. In 1949 Hebb described a technique which became known as Hebbian learning. In 1961 Rosenblatt devised a single layer of neurons, called a perceptron that was used for optical pattern recognition (Burns, 2001)
Perceptrons are early ANN models, consisting of a single layer and simple threshold functions. The architecture of a perceptron consisting of multiple neurons with *Nx1* inputs and *Mx1* outputs is shown in Fig. 1. As seen in this figure, the output vector of the

perceptron is calculated by summing the weighted inputs coming from its input links, so that

$$\mathbf{u} = \mathbf{W}\,\mathbf{p} + \mathbf{b} \tag{1}$$

$$\mathbf{q} = \mathbf{f}(\mathbf{u}) \tag{2}$$

where **p** is $Nx1$ input vector (p_1, p_2, ... p_N), **W** is MxN weighting coefficients matrix (w_{11}, w_{12},... w_{1N} ; .....,w_{j1}, w_{j2}, ..., w_{jN}; ....; w_{M1}, w_{M1},...,w_{MN}), **b** is $Mx1$ bias factor vector, **u** is $Nx1$ vector including the sum of the weighted inputs (u_1, u_2, ... u_M) and bias vector, **q** is $Mx1$ output vector (q_1, q_2, ... q_M)„ and **f(.)** is the activation function.

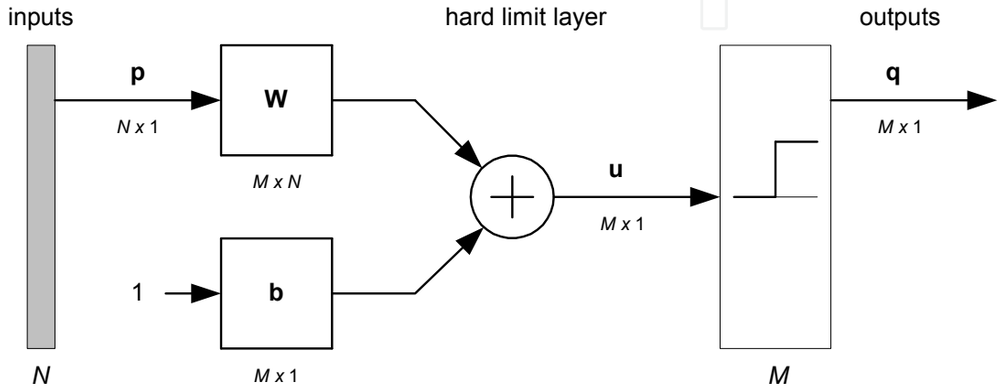


Fig. 1. The architecture of a perceptron

In early perceptron models, the activation function was selected as hard-limiter (unit step) given as follows:

$$q_i = \begin{matrix} 0 & , f(u_i) < 0 \\ 1 & , f(u_i) \geq 0 \end{matrix} \tag{3}$$

where $i = 1,2,\ldots,M$ denotes the number of neuron in the layer, u_i weighted sum of its particular neuron, and q_i its output. However, in any ANN the activation function $f(u_i)$ can take many forms, such as, linear (ramp), hyperbolic tangent and sigmoid forms. The equation for sigmoid function is:

$$f(u_i) = 1 / (1 + e^{-u_i}) \tag{4}$$

The sigmoid activation function given in Equation (4) is popular for ANN applications since it is differantiable and monolithic, both of which are a requirement for training algorithms like as the backpropagation algorithm.

Perceptrons must include a training rule for adjusting the weighting coefficients. In the training process, it compares the actual network outputs to the desired network outputs for each epoch to determine the actual weighting coefficients:

$$\mathbf{e} = \mathbf{q}^d - \mathbf{q} \tag{5}$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\,\mathbf{p}^T \tag{6}$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} \tag{7}$$

where **e** is $Mx1$ error vector, $\mathbf{q}^d$ is $Mx1$ target (desired) vector, the upscripts T , *old* and *new* denotes the transpose, the actual and previous (old) representation of the vector or matrix, respectively (Hagan et al., 1996).

2.1.2 Network architectures

There are mainly two types of ANN architectures: feedforward and recurrent (feedback) architectures. In the feedforward architecture, all neurons in a particular layer are fully connected to all neurons in the subsequent layer. This generally called a fully connected multilayer network. Recurrent networks are based on the work of Hopfield and contain feedback paths. A recurrent network having two inputs and three outputs is shown in Fig. 2. In Fig. 2, the inputs occur at time (kT) and the outputs are predicted at time (k+1)T, where k is discrete time index and T is sampling time, respectively.

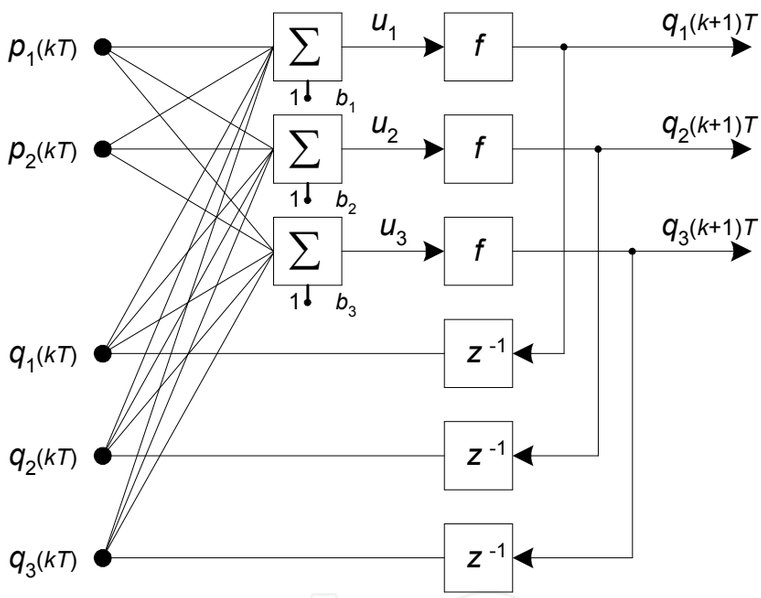


Fig. 2. Recurrent neural network architecture

Then the network can be represented in matrix form as:

$$\mathbf{q}(k+1)T = \mathbf{f}\,(\mathbf{W_1}\,\mathbf{p}(kT) + \mathbf{W_2}\,\mathbf{q}(kT) + \mathbf{b}) \tag{8}$$

where **b** is bias vector, **f(.)** is activation function, $\mathbf{W_1}$ and $\mathbf{W_2}$ are weight matrix for inputs and feedback paths, respectively.

2.1.3 Learning

Learning in the context of ANNs is the process of adjusting the weights and biases in such a manner that for given inputs, the correct responses, or outputs are achieved. Learning algorithms include supervised learning and unsupervised learning.

In the supervised learning the network is presented with training data that represents the range of input possibilities, together with associated desired outputs. The weights are adjusted until the error between the actual and desired outputs meets some given minimum value.

Unsupervised learning is an open-loop adaption because the technique does not use feedback information to update the network's parameters. Applications for unsupervised learning include speech recognition and image compression.
Important unsupervised learning include the Kohonen self-organizing map (KSOM), which is a competitive network, and the Grossberg adaptive resonance theory (ART), which can be for on-line learning.
There are multitudes of different types of ANN models for control applications. The first one of them was by Widrow and Smith (1964). They developed an Adaptive LINear Element (ADLINE) that was taught to stabilize and control an inverted pendulum. Kohonen (1988) and Anderson (1972) investigated similar areas, looking into associative and interactive memory, and also competitive learning (Burns, 2001).
Some of the more popular of ANN models include the multi-layer perceptron (MLP) trained by supervised algorithms in which backpropagation algorihm is used.

2.1.4 Backpropagation

The backpropagation algorithm was investigated by Werbos (1974) and futher developed by Rumelhart (1986) and others, leading to the concept of the MLP. It is a training method for multilayer feedforward networks. Such a network including N inputs, three layers of perceptrons, each has $L1$, $L2$, and M neurons, respectively, with bias adjustment is shown in Fig. 3.

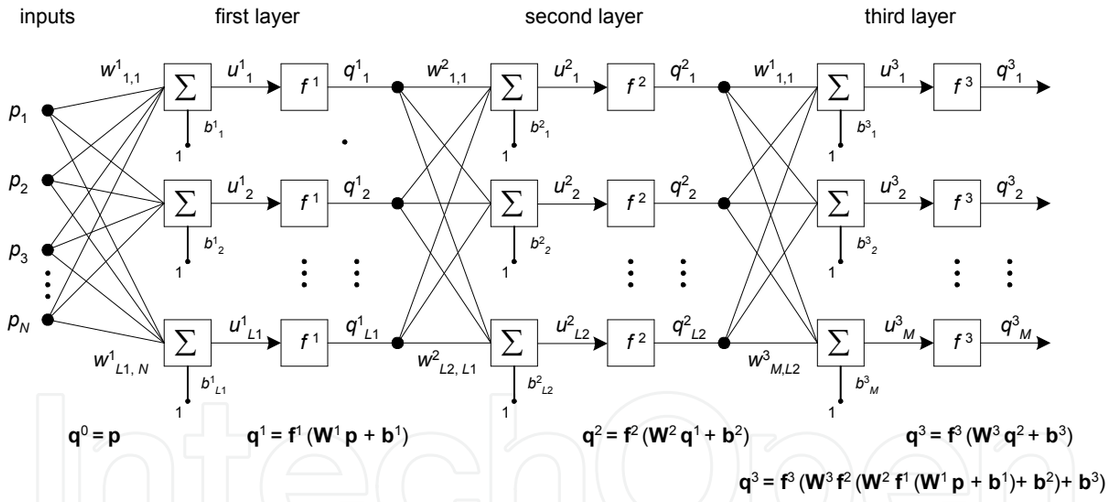


Fig. 3. Three-layer feedforward network

First step in backpropogation is propagating the inputs towards the forward layers through the network. For L layer feedforward network, training process is stated from the output layer:

$$\mathbf{q}^0 = \mathbf{p}$$

$$\mathbf{q}^{l+1} = \mathbf{f}^{l+1} (\mathbf{W}^{l+1} \mathbf{q}^l + \mathbf{b}^{l+1}) \ , \quad l = 0 , 1, 2,\ldots., L\text{-}1 \tag{9}$$

$$\mathbf{q} = \mathbf{q}^L$$

where l is particular layer number; $\mathbf{f}^l$ and $\mathbf{W}^l$ represent the activation function and weighting coefficients matrix related to the layer l, respectively.
Second step is propagating the sensivities ($\mathbf{s}$) from the last layer to the first layer through the network: $\mathbf{s}^L$, $\mathbf{s}^{L-1}$, $\mathbf{s}^{L-2}$,…, $\mathbf{s}^l$…, $\mathbf{s}^2$, $\mathbf{s}^1$. The error calculated for output neurons is propagated to the backward through the weighting factors of the network. It can be expressed in matrix form as follows:

$$\mathbf{s}^L = -2\dot{\mathbf{F}}^L(\mathbf{u}^L)\,(\mathbf{q}^d - \mathbf{q}) \quad , \quad \mathbf{s}^l = \dot{\mathbf{F}}^l(\mathbf{u}^l)\,(\mathbf{W}^{l+1})^T\mathbf{s}^{l+1} \ , \ \text{for } l = L\text{-}1,\ldots, 2, 1 \tag{10}$$

where $\dot{\mathbf{F}}^l(\mathbf{u}^l)$ is Jacobian matrix which is described as follows

$$\dot{\mathbf{F}}^l(\mathrm{u}^l) = \begin{bmatrix} \dfrac{\partial f^l(u_1^l)}{\partial u_1^l} & 0 & \cdots & 0 \\ 0 & \dfrac{\partial f^l(u_2^l)}{\partial u_2^l} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dfrac{\partial f^l(u_N^l)}{\partial u_N^l} \end{bmatrix} \tag{11}$$

Here N denotes the number of neurons in the layer l. The last step in backpropagation is updating the weighting coefficients. The state of the network always changes in such a way that the output follows the error curve of the network towards down:

$$\mathbf{W}^l(k+1) = \mathbf{W}^l(k) - \alpha\,\mathbf{s}^l\,(\mathbf{q}^{l-1})^T \tag{12}$$

$$\mathbf{b}^l(k+1) = \mathbf{b}^l(k) - \alpha\,\mathbf{s}^l \tag{13}$$

where α represents the training rate, k represents the epoch number (k=1,2,…,K). By the algorithmic approach known as gradient descent algorithm using approximate steepest descent rule, the error is decreased repeatedly (Hagan, 1996).

2.2 Applications to parallel manipulators

ANNs can be used for modeling various non-linear system dynamics by learning because of their non-linear system modelling capability. They offer highly parallel, adaptive models that can be trained by using system input-output data.
ANNs have the potential advantages for modeling and control of dynamic systems, such that, they learn from experience rather than by programming, they have the ability to generalize from given training data to unseen data, they are fast, and they can be implemented in real-time.
Possible applications using ANN to modeling and control of parallel manipulators may include:

- Modeling the manipulator dynamics,
- Inverse model of the manipulator,
- Controller emulation by modeling an existing controller,
- Various intelligent control applications using ANN models of the manipulator and/or its controller. Such as, ANN based internal model control (Burns, 2001).

2.2.1 Modeling the manipulator dynamics

Providing input/output data is available, an ANN may be used to model the dynamics of an unknown parallel manipulator, providing that the training data covers whole envelope of the manipulator operation (Fig. 4).

However, it is difficult to imagine a useful non-repetitive task that involves making random motions spanning the entire control space of the manipulator system. This results an intelligent manipulator concept, which is trained to carry out certain class of operations rather than all virtually possible applications. Because of that, to design an ANN model of the chosen parallel manipulator training process may be implemented on some areas of the working volume, depending on the structure of chosen manipulator (Akbas, 2005). For this aim, the manipulator(s) may be controlled by implementation of conventional control algorithms for different trajectories.

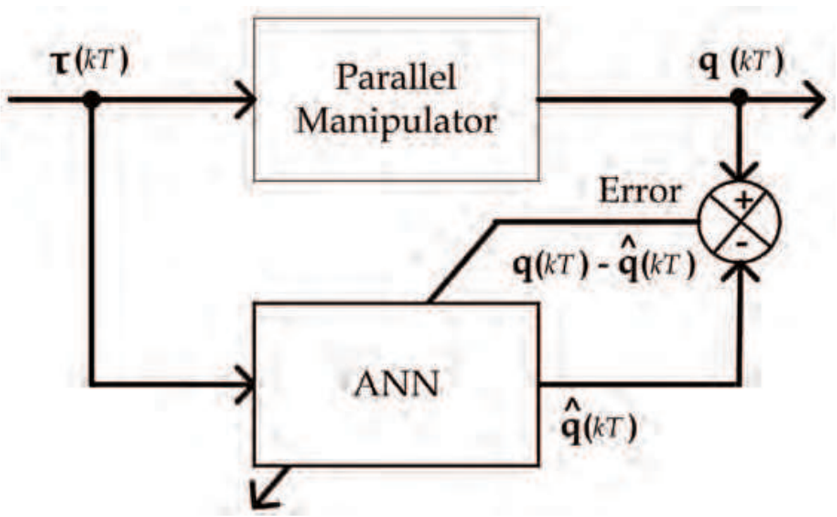


Fig. 4. Modelling the forward dynamics of a parallel manipulator

If the ANN in Fig. 4 is trained using backpropagation, the algorithm will minimize the following performance index:

$$PI = \sum_{n=1}^{N} \left(\left(q\left(kT\right) - \hat{q}\left(kT\right) \right)^{t} \left(q\left(kT\right) - \hat{q}\left(kT\right) \right) \right) \tag{14}$$

where $\mathbf{q}$ and $\hat{\mathbf{q}}$ denote the output vector of the manipulator and ANN model, respectively.

2.2.2 Inverse model of the manipulator

The inverse model of a manipulator provides a control vector $\boldsymbol{\tau}(kT)$, for a given output vector $\mathbf{q}(kT)$ as shown in Fig. 5. So, for a given parallel manipulator model, the inverse model could be trained with the parameters reflecting the forward dynamic characteristics of the manipulator, with time.

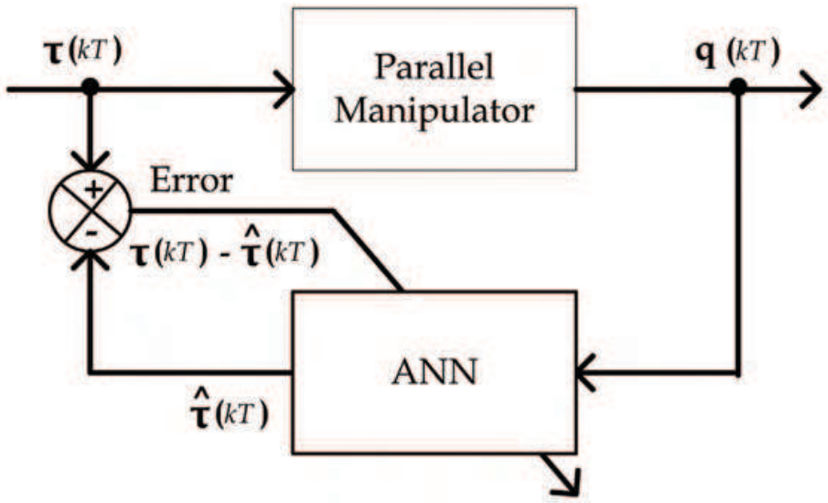


Fig. 5. Modelling the inverse dynamics of parallel manipulator

As indicated above, the training process may be implemented using input-output data obtained by manipulating certain class of operations on some areas of the working volume depending on the structure of chosen manipulator.

2.2.3 Controller emulation

A simple application in control is the use of ANNs to emulate the operation of existing controllers (Fig. 6).

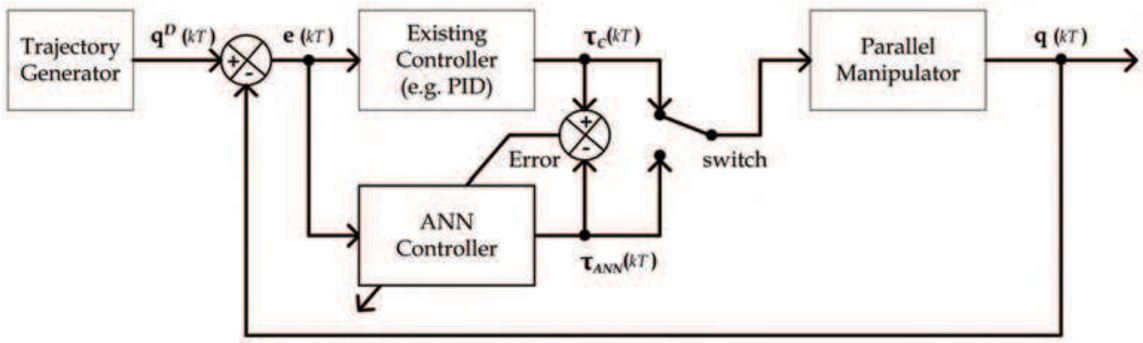


Fig. 6. Training the ANN controller and its implementation to the control system

It may be require several tuned PID controllers to operate over the constrained range of control actions. In this context, some manipulators may be required more than one emulated controllers that can be used in parallel form to improve the reliability of the control system by error minimization approach.

2.2.4 IMC implementation

ANN control can be implemented in various intelligent control applications using ANN models of the manipulator and/or its controller. In this context the internal model control

(IMC) can be implemented using ANN model of parallel manipulataor and its inverse model (Fig. 7).

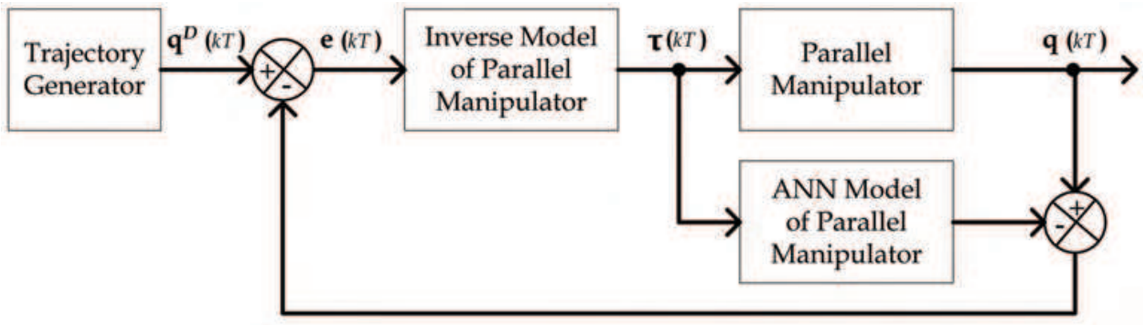


Fig. 7. IMC application using ANN models of parallel manipulator

In this implementation an ANN model model replaces the manipulator model, and an inverse ANN model of the manipulator replaces the controller as shown in Fig. 7.

2.2.5 Adaptive ANN control

All closed-loop control systems operate by measuring the error between desired inputs and actual outputs. This does not, in itself, generate control action errors that may be backpropagated to train an ANN controller. However, if an ANN of the manipulator exists, backpropagation through this network of the system error will provide the necessary control action errors to train the ANN controller as shown in Fig.8.

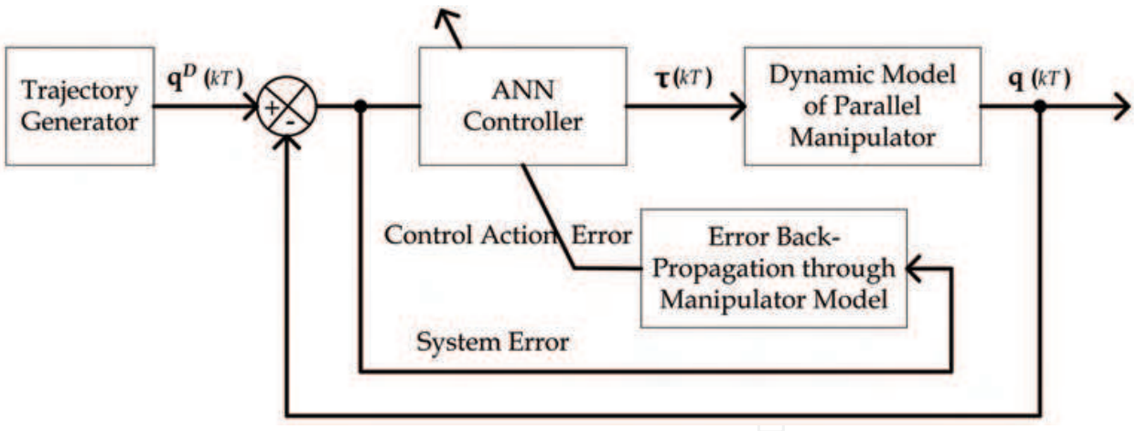


Fig. 8. Control action generated by adaptive ANN controller

3. The structure of Stewart manipulator

Six degrees of freedom (6-dof) simple and practical platform type parallel manipulator, namely Stewart manipulator, is sketched in Fig. 9. These type manipulators were first introduced by Gough (1956-1957) for testing tires. Stewart (1965) suggested their use as flight simulators (Angeles, 2007).

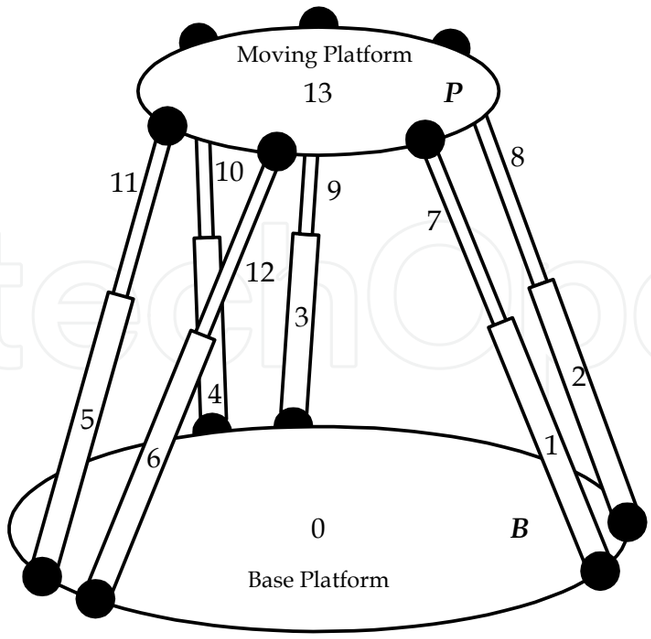


Fig. 9. A sketch of the 6-dof Stewart manipulator

In Fig. 9, the upper rigid body forming the moving platform, ***P***, is connected to the lower rigid body forming the fixed base platform, ***B***, by means of six legs. Each leg in that figure has been represented with a spherical joint at each end. Each leg has upper and lower rigid bodies connected with a prismatic joint, which is, in fact, the only active joint of the leg. So, the manipulator has thirteen rigid bodies all together, as denoted by 1,2…..13 in Fig. 9.

3.1 Kinematics

Motion of the moving platform is generated by actuating the prismatic joints which vary the lengths of the legs, q^{L_i} , i = 1….6. So, trajectory of the center point of moving platform is adjusted by using these variables.

For modeling the Stewart manipulator, a base reference frame F_B ($O_B x_B y_B z_B$) is defined as shown in Fig. 10. A second frame F_P ($O_P x_P y_P z_P$) is attached to the center of the moving platform, O_P , and the points linking the legs to the moving platform are noted as Q_i , i = 1….6, and each leg is attached to the base platform at the point B_i , i = 1….6.

The pose of the center point, O_P , of moving platform is represented by the vector

$$\mathbf{x} = [x_B \; y_B \; z_B \; \alpha \;\; \beta \;\; \gamma]^T \tag{15}$$

where x_B, y_B, z_B are the cartesian positions of the point O_P relative to the frame F_B and a, β, γ are the rotation angles, namely Euler angles, representing the orientation of frame F_P relative to the frame F_B by three successive rotations about the x_P, y_P and z_P axes, given by the matrices $R_x(a)$, $R_y(\beta)$, $R_z(\gamma)$ respectively (Spong & Vidyasagar, 1989). Thus, the rotation matrix between the F_B and F_P frames is given as follows:

$$R_B^P = R_x(a)\, R_y(\beta)\, R_z(\gamma) \tag{16}$$

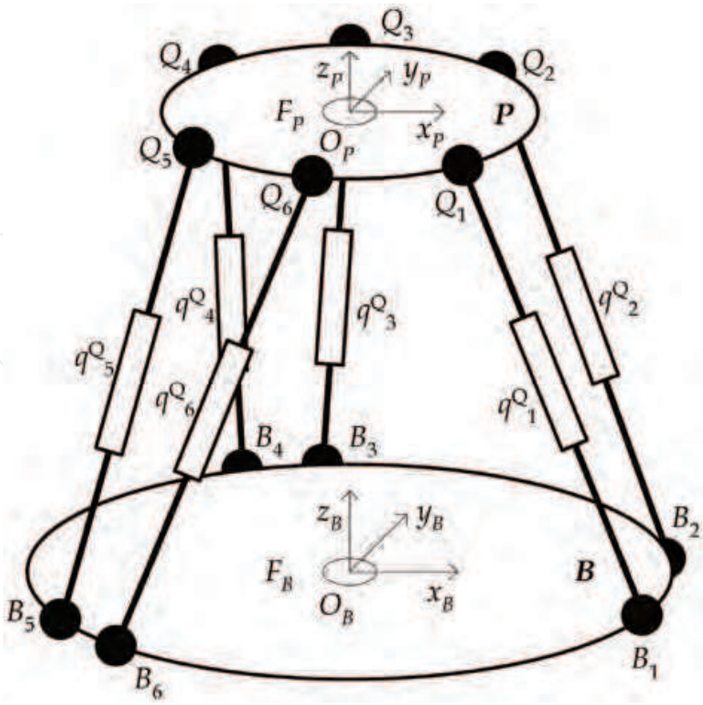


Fig. 10. Assignments for kinematic analysis of the Stewart manipulator

Then we can analyze the inverse kinematics of Stewart manipulator by the representation of any one of its legs. For a given pose of the center point of moving platform, O_P, the defining vectors are shown in Fig. 11.

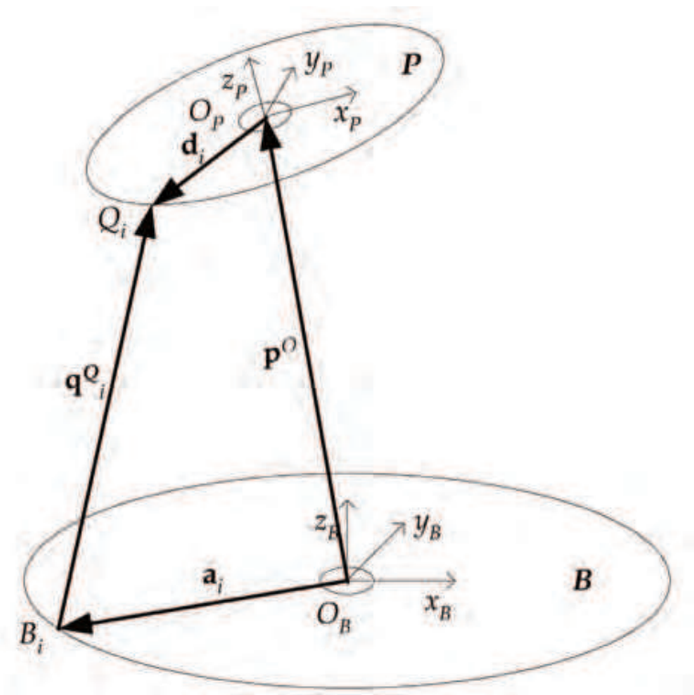


Fig. 11. Defining the vectors for a given pose of the moving platform

By using the rotation matrix given by equation (16), the position vector of the upper joint position, Q_i, connecting the moving platform to the leg i, $\mathbf{q}^{Q}{}_{i}$ can be transformed to the frame F_B as follows:

$$\mathbf{q}_i^Q = \mathbf{p}^O + R_B^P \, \mathbf{d}_i \qquad i = 1....6 \tag{17}$$

where $\mathbf{p}^O$ represents the position vector of the center point of moving platform, O_P , relative to the frame F_B , $\mathbf{d}_i$ is the position vector of the point Q_i , $i = 1....6$, relative to the frame F_P . Then the vector $\mathbf{q}^{A}{}_{i}$ representing the leg legths between the joint points B_i and Q_i can be transformed to the frame F_B as follows:

$$\vec{B_i Q_i} = \mathbf{q}_i^A = -\mathbf{a}_i + \mathbf{q}_i^Q \qquad i = 1....6 \tag{18}$$

where $\mathbf{a}_i$ represents the position vector of the point B_i , $i = 1....6$, relative to the frame F_B . The leg lengths $q^{A}{}_{i}$, $i = 1....6$, is then obtained by Euclidean norm of the leg vector given above. So, using equation (17) and (18) we can write (Zanganeh et al., 1997)

$$(q_i^A)^2 = (\mathbf{a}_i + \mathbf{p}^O + R_B^P \, \mathbf{d}_i)^T (\mathbf{a}_i + \mathbf{p}^O + R_B^P \, \mathbf{d}_i) \quad , \; i = 1....6 \tag{19}$$

The leg lengths related to a given pose of moving platform can be obtained for a trajectory defined by the pose vector, $\mathbf{x}$, given in equation (15). Considering a circular motion depicted as in Fig. 12, the trajectory of moving platform with zero rotation angles ($[a \; \beta \; \gamma] = [0 \; 0 \; 0]$) is given as follows:

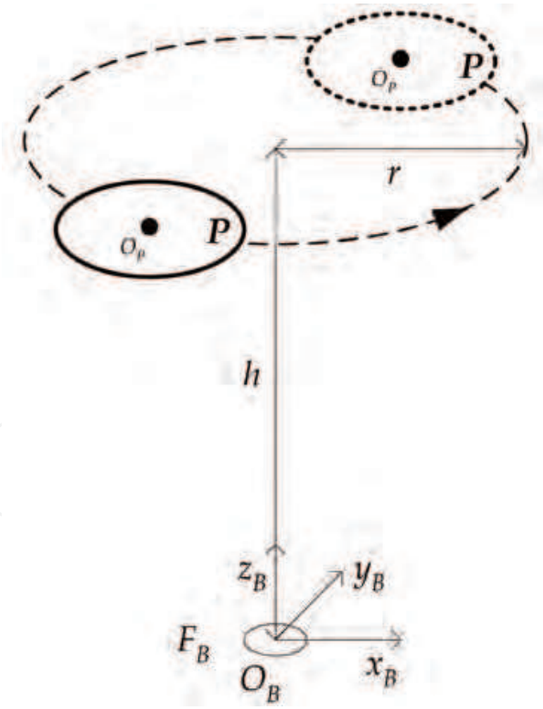


Fig. 12. A circular motion trajectory of the moving platform

$$\mathbf{x} = [(\mathbf{p}^O)^T \; 0 \; 0 \; 0]^T = \mathbf{A}(t) \, \mathbf{x}_0 \tag{20}$$

where $\mathbf{p}^O = [x_B \;\; y_B \;\; z_B \;]^T$ denotes the 3x1 position vector of the center point of moving platform, $\mathbf{A}(t)$ is a 6x6 matrix and $\mathbf{x}_0$ is a 6x1 coeeficient vector given as below

$$A(t)=\begin{bmatrix} \cos[\theta(t)] & -\sin[\theta(t)] & 0 & \\ \sin[\theta(t)] & \cos[\theta(t)] & 0 & 0 \\ 0 & 0 & 1 & \\ & 0 & & 0 \end{bmatrix} \tag{21}$$

$$\mathbf{x}_0 = [0\ r\ h\ 0\ 0\ 0]^T \tag{22}$$

where **O** denotes the 3x3 zero matrix, h is the hight of the center point of moving platform with respect to base frame, and r is the radius of the circle.
The Jacobian matrix that gives the relation between the prismatic joint velocities and the velocity of the center point of moving platform, O_P, can be derived using the partial differentiation of the inverse geometric model of the manipulator given in equation (19).

3.2 Dynamics

As descripted in Fig. 9, Stewart manipulator has thirteen rigid bodies. The Newton-Euler equations of the manipulator can be derived in a more compact form as described below (Fang et al., 2000; Khan et al., 2005):
Let the 6x6 matrix $\mathbf{M}_i$, denoting the mass and moment of inertia properties of the rigid body i be

$$M_i = \begin{bmatrix} I_i & 0 \\ 0 & m_i \times 1 \end{bmatrix} \quad , \ i = 1....13 \tag{23}$$

where **O** and **1** denote the 3x3 zero and identity matrices; $\mathbf{I}_i$ is inertia matrix defined with respect to the mass center, C_i, of the body i ; m_i is the mass of the body i. Let $\mathbf{c}_i$ and $\dot{\mathbf{c}}_i$ denote the position and velocity vectors of C_i, and $\boldsymbol{\omega}_i$ denote the angular velocity vector of C_i. Then the wrench vector $\mathbf{t}_i$ is defined in terms of the angular and linear velocities as follows:

$$t_i = \begin{bmatrix} \omega_i \\ \cdot \\ c_i \end{bmatrix} \quad , \ i = 1....13 \tag{24}$$

Let the 6x6 matrix $\boldsymbol{\Omega}_I$, denoting the angular velocity of the rigid body i be

$$\Omega_i = \begin{bmatrix} \omega_i & 0 \\ 0 & 0 \end{bmatrix} \quad , \ i = 1....13 \tag{25}$$

where, **O** denotes the 3x3 zero matrix. The generalized matrices given in equation (23) and (25) are block symmetrical, as follows:

$$M = diag(M_1, M_2, ..., M_{13}), \Omega = diag(\Omega_1, \Omega_2, ..., \Omega_{13}) \tag{26}$$

Then, the generalized wrench matrix **t** can be expressed as follows

$$t=[t_1^T t_2^T ..... t_{13}^T]^T \tag{27}$$

For the system having constraint on velocity, the constraint of velocity can be expressed by following equation:

$$Dt=0 \tag{28}$$

Let **T** be the natural orthogonal complement (NOC) of the coefficient matrix **D** related to the constraint equation (28) of velocity. Hence, employing the joint coordinates $\mathbf{q} \in R^6$ as

generalized coordinate vector, we can get the dynamic model of system, which don't contain the constraint forces.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} \tag{29}$$

where $\mathbf{M}(\mathbf{q})$ is a symmetrical and positive definition matrix as given below;

$$\mathbf{M}(\mathbf{q}) = \mathbf{T}^T\mathbf{M}\,\mathbf{T} \quad \in R^{6x6} \tag{30}$$

$\mathbf{C}$ is the coefficient matrix of the vectors of Coriolis and centripetal force as given below;

$$\mathbf{C} = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{T}^T\mathbf{M}\,\dot{\mathbf{T}} + \mathbf{T}^T\Omega\mathbf{M}\mathbf{T} \tag{31}$$

$\mathbf{q}$ is the generalized coordinate vector, $\boldsymbol{\tau} \in R^6$ is the generalized force (driving force) vector, respectively. $\mathbf{G}(\mathbf{q})$ is the gravity vector as given below;

$$\mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}^g(\mathbf{q}) = \mathbf{T}^T\mathbf{W}^g \tag{32}$$

where $\mathbf{W}^g$ are wrenches vector due to gravity:

$$\mathbf{W}^g = [\mathbf{W}_1{}^{g\,T}, \mathbf{W}_2{}^{g\,T}, \ldots\ldots \mathbf{W}_{13}{}^{g\,T}]^T = [\mathbf{0}, m_1\mathbf{g}^T, \ldots\ldots \mathbf{0}, m_{13}\mathbf{g}^T]^T \tag{33}$$

where $\mathbf{0}$ is 3x1 zero vector, $\mathbf{g}$ is the vector of acceleration of gravity.

4. Controller emulation by using Elman networks

In this stage, it is aimed to implement an application of ANN to emulate the operation of an existing PID controller in a Stewart manipulator control system. This system is given as a control system example for MATLAB applications (www.mathworks.com). The block diagram of the control system is given in Fig. 13.

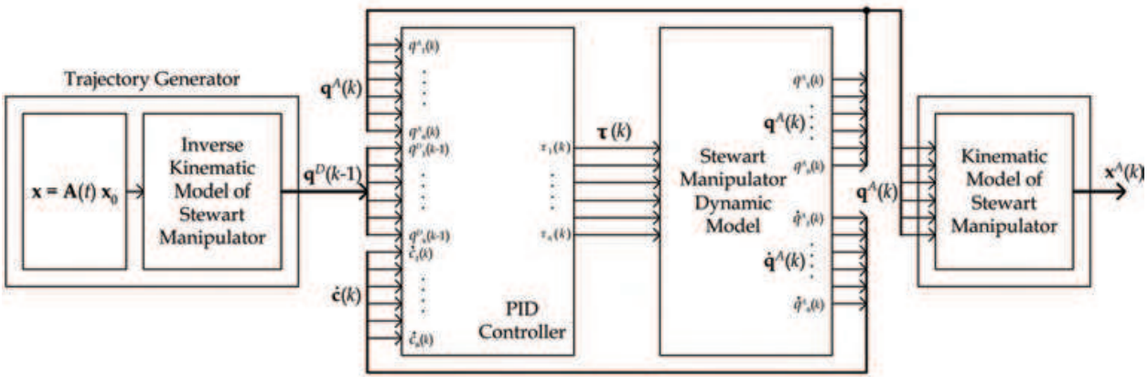


Fig. 13. Srewart manipulator control system using PID controller

As shown in this figure, trajectory generator calculates the leg lengths, which are desired leg lengths formed as a 6x1 $\mathbf{q}^D$ vector feeding the PID controller input, by using the inverse kinematic model of Stewart manipulator. PID controller produces a 6x1 control vector, $\boldsymbol{\tau}$, consisting of the leg forces applied to the prismatic joint actuators of the manipulator. In response, the dynamic model of the manipulator produces two 6x1 output vectors, $\mathbf{q}^A$ and $\dot{\mathbf{c}}=\dot{\mathbf{q}}^A$, which include actual leg lengths and actual linear leg velocities, respectively. These are fed back to the controller. So, the controller has 18 inputs and 6 outputs totally. PID

controller compares the actual and desired leg lengths to generate the error vector feeding its proportional and integral inputs. In the same time, the velocity feedback vector feeds the derivative input of the controller.
Designing an ANN emulation of controller generalized for the whole area of working space is more difficult task. It is also difficult to imagine a useful non-repetitive task that involves making random motions spanning the entire control space of the manipulator system. This results an intelligent manipulator concept, which is trained to carry out certain class of operations rather than all virtually possible applications (Akbas, 2005).
On the other hand, since the parallel manipulators have more complex dynamic structures, training process may be required much more data then other type plants. So, it can be taught to design more than one ANN controller trained by different input-output data sets, and use them in a parallelly formed controller structure instead of unique ANN controller structure. This can improve the reliability of the controller. Because of that, three ANN controllers are trained and they are used in parallel form in this case study.

4.1 Training

Due to its profound impact on the learning capability and networking performance, Elman network having recurrent structure is selected for training. Three of them, each have 18 inputs and 6 outputs, are trained by using PID controller input-output data. For this aim, input-output data are prepared during the implementation of the PID controller to the Stewart manipulator.
During the data log phase, manipulator is operated in a constrained area of its working space. For this aim, the manipulator is controlled by implementation of different trajectories selected uniformly in a planar sub-space, created as given example in equations (21) and (22) also as given in Fig. 12. Load variations are taken into consideration to generate the training data.
Three sets of input-output data each have 5000 vectors are generated by MATLAB simulations for each of Elman networks. MATLAB ANN toolbox is used for off-line training of Elman networks. Conventional backpropagation algorithm, which uses a threshold with a sigmoidal activation function and gradient descent error-learning, is used. Learning and momentum rates are selected optimally by MATLAB program. The numbers of neurons in the hidden layers are selected experimentally during the training. These are used as 40, 30 and 50, respectively for each network.

4.2 Implementation

After the off-line training, three of Elman networks are prepared as embedded Simulink blocks with obtained synaptic weights. To improve the reliability of the controller by error minimization approach, they are used in a parallel structure and embedded to the control system block diagram (Fig. 14). In this figure, parallely-implemented Elman ANN controller is represented in a block form. Its detailed representation is given in Fig. 15.
In this implementation, the force values generated by three Elman networks are applied to the inputs of the corresponding manipulator's dynamic model. Error vector is computed for each of the ANN by using the difference between the actual leg lengths generated by manipulator's dynamic model and the desired leg lengths. The results are evaluated to select the network generating the best result. Then it is assigned as the ANN controller for actual time step, and its output is assigned as the force output of the parallely-implemented Elman ANN controller output driving the manipulator's dynamic model (instead of a real manipulator, in this case).

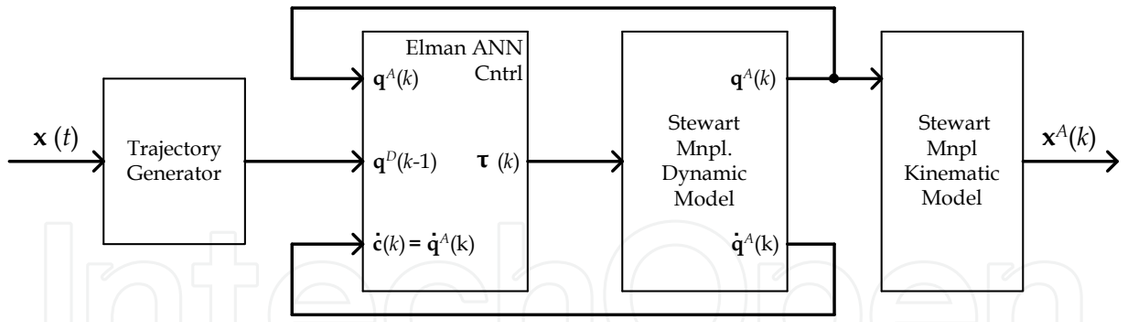


Fig. 14. ANN controller implementation to the manipulator control system

4.3 Simulation results

To compare the performance of the created ANN controller, the Srewart manipulator control system is operated both by the PID controller, and the parallelly-implemented Elman ANN controller for T=4 s. simulations. For these operations, a trajectory like as given with equations (21) and (22) is created with the parameter assignments: h = 2 m, r = 0.02 m. Also $\theta(t)$ parameter is used as follows:

$$\theta(t) = \frac{2\pi}{\mathrm{T}} t \quad , \quad 0 \text{ ¡Ü } t \text{ ¡Ü T} \tag{34}$$

During the simulations, the sampling period is chosen, as 0.001 s. So, totally 4000 steps are included in each simulation.

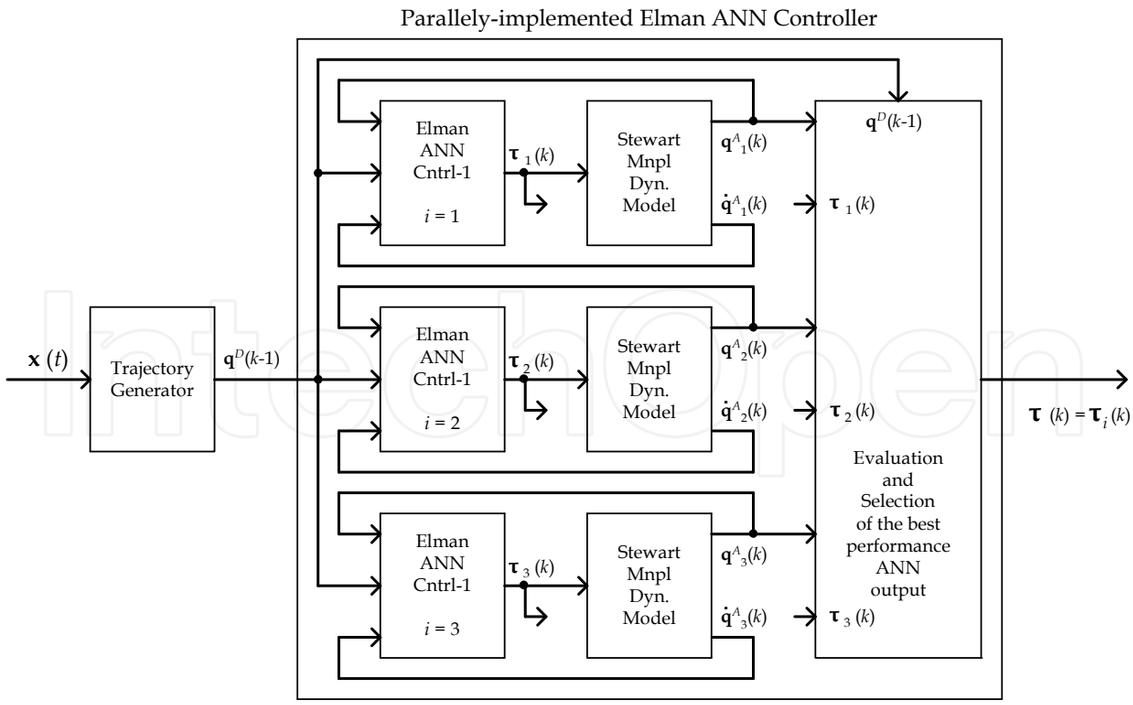
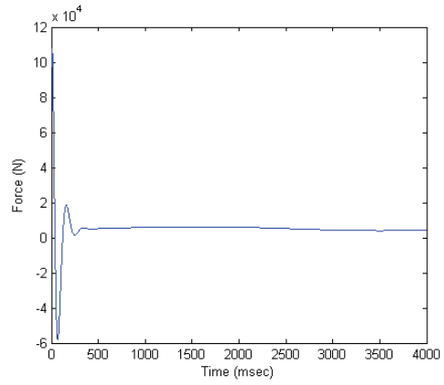
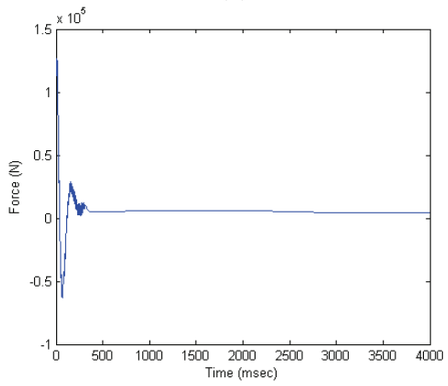


Fig. 15. The structure of parallelly-implemented Elman ANN controller

An example of the variations of the force outputs generated by both controllers is shown in Fig. 16, for the first leg of the manipulator. Fig. 16a and Fig. 16b show the force output of the PID controller and parallely-implemented Elman ANN controller, respectively. In these simulations, it has been observed that, the error between the two controller outputs is a little more at the starting phase of the simulations then the remaining times.
However, it can be said that, ANN controller emulates the PID controller successfully as a whole for the given trajectory.



(a)



(b)

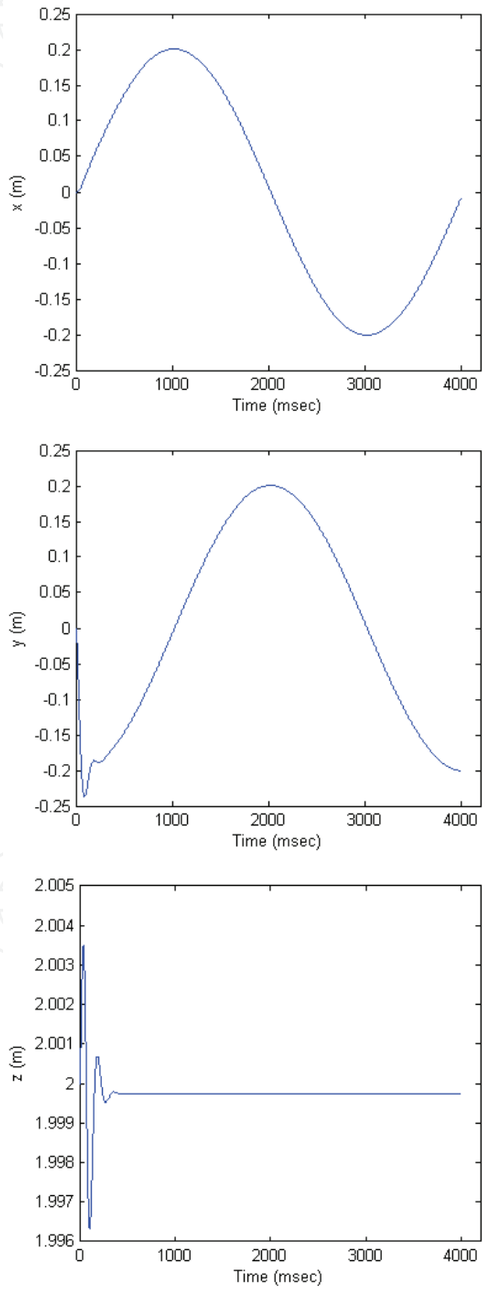
Fig.16. Force outputs of the controllers applied to the first leg of the Stewart manipulator (a)-PID controller output, (b)-ANN controller output

Similar adaptations are obtained for the control system output. For the given trajectory, position errors obtained by averaging the sum of the square errors relative to the desired position of the center point of moving platform both for the PID controller and ANN controller is given in Table 1. As seen in this table obtained position error values due to the x_B, y_B and z_B variations have too small changes.

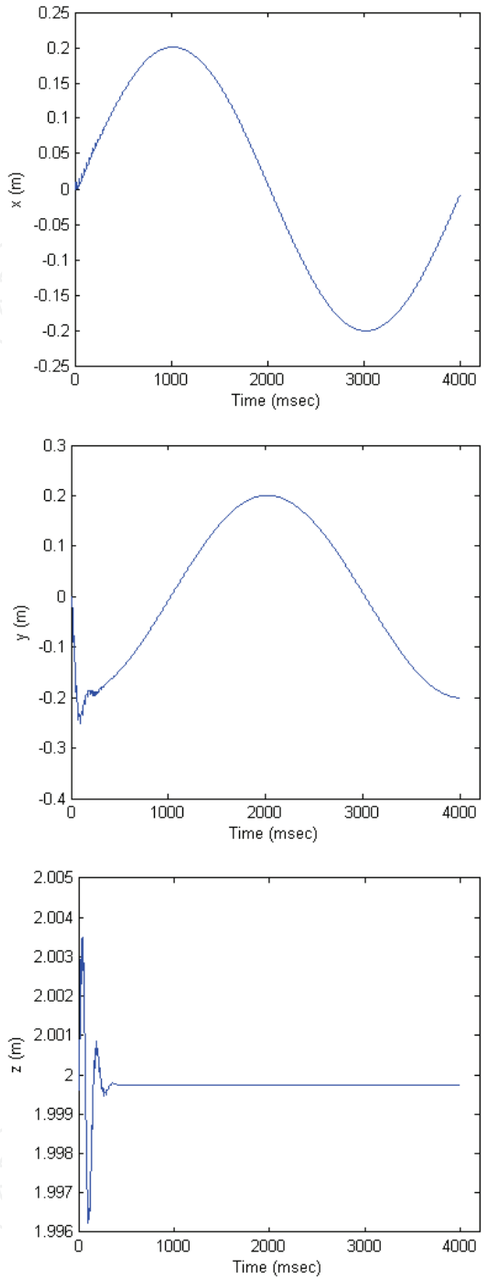
Axis	Average Position Errors (m)	
	PID controller	ANN controller
x_B	0.0000799	0.0000807
y_B	0.0002594	0.0002618
z_B	0.0000087	0.0000086

Table 1. The sum of the squares of the position errors obtained by PID and ANN

During simulations, variations of the x_B, y_B and z_B positions of the center point of moving platform are given in Fig. 17, so that, Fig. 17a and Fig. 17b show the variation of actual x_B, y_B and z_B positions obtained simulations using PID controller and parallely-implemented Elman ANN controller, respectively. As seen, the tracing error between the two control modes is a little more at the starting phase only. This is due to instantaneous big difference between the desired y_B position and its starting value. However, tracing the desired positions by PID controller is well emulated by parallely-implemented Elman ANN controller, as a whole.



(a)



(b)

Fig.16. Variation of actual position of the center point of moving platform, in simulations (a)-Obtained by the PID controller, (b)- Obtained by the ANN controller

5. Conclusion

This chapter is mainly concerned with the application of ANNs to modeling and control of parallel manipulators. A practical implementation is completed to emulate the operation of

an existing PID controller in a Stewart manipulator control system. It can be said that, excepted results has been achieved for this case study.
Since the parallel manipulators have more complex dynamic structures, depending on the chosen type of applications training process it may be required much more data then in this case. So, designing an ANN for applications including the whole area of working space is more difficult task. It is also difficult to imagine a useful non-repetitive task that involves making random motions spanning the entire control space of the manipulator system.
However, for a succesfull study, it may have an important role selecting the type and structure of ANN by experience, depending on the requirements of the chosen application.

6. References

Akbas, A. (2005). Intelligent predictive control of a 6-dof robotic manipulator with reliability based performance improvement, *Proceedings of the 6th International Conference on Intelligent Data Engineering and Automated Learning*, LNCS Vol. 3578, pp. 272-279, ISBN: 3-540-26972-X, Brisbane, Australia, July 2005, Springer

Angeles, J. (2007). *Fundamentals of Robotic Mechanical Systems, Theory, Methods, and Algorithms*, Springer Science+Business Media, LLC, ISBN: 0 387 29412 0, NY, USA

Burns, R.S. (2001). *Advanced Control Engineering*, Butterworth-Heinemann, ISBN: 0 7506 5100 8, Oxford

Fang, H.; Zhou, B.; Xu, H. & Feng, Z. (2000). Stability analysis of trajectory tracing contro1 of 6-dof parallel manipulator, *Proceedings of the 3d World Congress on Intelligent Control and Automation*, *IEEE*, Vol. 2, pp. 1235-1239, ISBN: 0-7803-5995-X, Hefei, China, June 28-July 2, 2000

Hagan, M.T.; Demuth, H.B. & Beale, M. (1996). *Neural Network Design*, PWS Publishing Company- Thompson Learning, ISBN: 7 111 10841 8, USA

Honegger, M.; Brega, R. & Schweitzer, G. (2000). Application of a nonlinear adaptive controller to a 6 dof parallel manipulator, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, *ICRA 2000*, pp. 1930-1935, ISBN: 0-7803-5889-9, San Francisco, CA, USA, April 24-28, 2000

Khan, W.A.; Krovi, V.N.; Saha, S.K. & Angeles, J. (2005). Modular and recursive kinematics and dynamics for parallel manipulators. *Multibody System Dynamics*, Vol. 14, No.3-4, Nov. 2005, pp. 419-455, ISSN: 1384-5640, Springer

Spong, M.W. & Vidyasagar, M. (1989). Robot Dynamics and Control. pp. 39-50, John Wiley & Sons

Zanganeh, K.E.; Sinatra, R. & Angeles, J. (1997). Kinematics and dynamics of a six-degree-of-freedom parallel manipulator with revolute legs. *Robotica*, Vol. 15, No.04, Jule 1997, pp. 385-394, ISSN: 0263-5747, Cambridge University Press

Parallel Manipulators, New Developments

Edited by Jee-Hwan Ryu

ISBN 978-3-902613-20-2

Hard cover, 498 pages

Publisher I-Tech Education and Publishing

Published online 01, April, 2008

Published in print edition April, 2008

Parallel manipulators are characterized as having closed-loop kinematic chains. Compared to serial manipulators, which have open-ended structure, parallel manipulators have many advantages in terms of accuracy, rigidity and ability to manipulate heavy loads. Therefore, they have been getting many attentions in astronomy to flight simulators and especially in machine-tool industries.The aim of this book is to provide an overview of the state-of-art, to present new ideas, original results and practical experiences in parallel manipulators. This book mainly introduces advanced kinematic and dynamic analysis methods and cutting edge control technologies for parallel manipulators. Even though this book only contains several samples of research activities on parallel manipulators, I believe this book can give an idea to the reader about what has been done in the field recently, and what kind of open problems are in this area.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen