A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robot Manipulators

Loo Chu Kiong, Mandava Rajeswari, Wong Eng Kiong & M. V. C. Rao

Faculty of Engineering and Technology, Multimedia University (Melaka), 75450 Melaka, Malaysia
E-mail: ckloo@mmu.edu.my

School of Industrial Technology, Universiti Sains Malaysia, Penang, Malaysia
E-mail: mandava@cs.usm.my

ABSTRACT

This paper proposes a nonlinear variable gain Proportional-Derivative (PD) controller that exhibits self-constructing and self-learning capabilities. In this method, the conventional linear PD controller is augmented with a nonlinear variable PD gain control signal using a dynamic structural network. The dynamic structural network known as Growing Multi-Experts Network grows in time by placing hidden nodes in regions of the state space visited by the system during operation. This results in a network that is "economic" in terms of network size. The proposed approach enhances the adaptability of conventional PD controller while preserving its linear structure. Based on the simulation study on variable load and friction compensation, the fast adaptation is shown to be able to compensate the non-linearity and the uncertainty in the robotic system.

Keywords: Nonlinear PD Controller, feedback error learning, dynamic structure neural network, friction compensation, variable load compensation

INTRODUCTION

Since their inception over thirty years ago, industrial robots have been widely used in industry. Despite a large number of sophisticated robot control methods reported in the robotics and control literature, linear PD controller is still one of the commonly used controllers in commercial industrial robots. Robot manipulators have highly nonlinear dynamics. Since the robot dynamics is nonlinear while PD control is a linear-control, applying PD control to trajectory-following problems would require a gain scheduling approach using local models; that is the robot dynamics is linearized about some operating points so that the PD gains can be selected to achieve certain performance specifications. The trajectory, however, varies with time (as opposed to a set-point). Therefore, the gains so selected may not be appropriate due to the linearization. Such a predesign needs significant effort and a priori robot model information, and cannot adapt itself to model error or time variation in an on-line manner. The limitations mentioned above have inspired the idea of model free self-adaptive independent joint controller proposed in this paper. The basic challenge of designing a model free self-adaptive controller is how to provide an on-line learning algorithm that does not require preliminary off-line learning. Instead, the controller is able to automatically improve its performance on the fly even when the robot model is unknown or partially known. This shares the idea of universal model-free controllers such as neural networks. Therefore, a neural network based feedforward controller that exhibits self-constructing and self-learning capabilities is presented in this paper.

** Corresponding author
Problem

Until the beginning of the nineties, the principal type of neural networks used for system identification and control problems was the multilayer perceptron (MLP) with sigmoidal hidden units in Psaltis et al. (1988); Khalid et al. (1992). However, slow learning and local minima problem of MLP have led to the development of modular neural networks such as the radial basis function (RBF) neural networks Kadirkamanathan et al. (1993). The RBF network has gained increasing attention among researchers owing to its modular structure and fast learning capability. With the use of Gaussian functions, the RBF network forms a local representation. Each basis function responds only to inputs in the neighborhood of a reference vector known as basis function center. The spread of the neighborhood is determined by the variance of basis function. In using RBF networks as neural network based controller, one typically initializes the center of the basis functions on regular points of a square mesh covering a relevant region of space where the input state is known to be contained, and denoted by a compact set $\mathcal{X}_{\mu}$. This approach is widely used in robotic control application Ge (1996); Ge et al. (2001); Ge et al. (2002a); Ge et al. (2002b). This situation is also encountered in system identification problems, where the state components represent delayed versions of the plant output response. Therefore, state variables move along specific trajectories within the state space, $\mathcal{X}_{\mu}$ depending on the excitation conditions of the model. This implies that placing basis functions on all mesh points within $\mathcal{X}_{\mu}$ results in a lot of redundancy. Fig. 1 illustrates this problem.

Nevertheless, even assuming that the basis functions are placed along the trajectory, they may be positioned close to each other owing to network adaptation. It is illustrated in Fig. 2. Fig. 3 indicates that the basis function A appears to be redundant since basis function A may produce a response amplitude similar to basis function B for an input signal x. Under this situation, either basis function A or B alone is able to perform the approximation sufficiently well for the covered region.
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

Fig. 2: Two basis functions are placed close to each other. One of them appears to be redundant

Overall, the applicability of modular neural network with static structure and using basis function is limited by the following issues:

- For a system identification or control problem, the initialization of Gaussian basis function based on mesh grid method may result in redundancy.
- Gaussian basis functions that are located close to each other may result in redundancy since they may generate similar activation values corresponding to an input signal near to their center.

In this paper, it is proposed to adaptively place the basis functions in the strategic regions in order to cope with the problem of a large number of redundant basis
functions. This results in a much smaller network in structure. In order to differentiate two closely positioned Gaussian basis functions, the concepts of “expertise level” and “expertise domain” are introduced into the network design. It is illustrated in Fig. 4. Based on this idea, two closely positioned Gaussian basis functions may share similar expertise domain but they may have different expertise levels. The expertise level indicates the significance of their contribution to approximate the region they recover. It is, therefore, possible to detect the redundancy by monitoring their expertise level value. Low expertise level indicates the basis function redundancy.

Fig. 4: Gaussian basis functions gated by a scaling factor to indicate their expertise level

In this paper, it is proposed to augment the conventional PD controller with a feedforward control signal using a dynamic structural network known as Growing Multi-Experts Network Loo et al. (2000); Loo et al. (2003). The dynamic structural network grows in time by placing hidden nodes in regions of the state space visited by the robot during operation. This results in a network that is “economic” in terms of network size. The attractive feature is that using dynamic network structure in neural control design results in a much smaller network and effective utilization of basis function with reduced storage and computational requirements. The adaptation capability of the network is shown to be able to compensate for the uncertainty of the robot dynamic model.

Growing Multi-Experts Network (GMN)

Control and modeling problem involves approximation of an unknown nonlinear function. In nonlinear systems the function complexity varies throughout the input space. Instead of trying to approximate the whole function globally, the input space is partitioned into subspaces that are easier to handle. This may be achieved by effectively using ‘divide and conquer’ strategy where the problem space is decomposed into overlapping regions and local experts approximate the function in every region. In this context, the problem space decomposition requires a gating unit that may clearly define the zone of influence of local experts.

The local experts are associated with a weighting factor that indicates the importance of each local expert. Finally, the network overall output is computed by combining the outputs from all the local experts.
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

Overall, GMN consists of three main building modules namely, expertise domain, expertise level, and local experts. Local experts may have a nonlinear function, linear function or even a mixture of both. In this paper, a polynomial function, mainly a linear model, is considered. Expertise domain is used to define the influence regions where the local experts perform local approximation. The competency of local experts is indicated by expertise level. High value of expertise level means the local expert is making significant contribution to the overall network output. Fig. 5 shows the Growing Multi-Experts Network architecture. The design of these three modules and the learning algorithm of GMN are described in detail in the following sections.

**Expertise Domain**

Gaussian basis functions have localized receptive fields. With this property, the Gaussian basis function is obviously a suitable choice for an expertise domain gating function to indicate the validity of input data with respect to the local expert model. The distance metric $d(\bar{x}; \tilde{\sigma}, \sigma)$ for Gaussian basis function $k$ as defined in equation (1) can scale the spread of the Gaussian basis function relative to its center $\tilde{\sigma}$ and the Gaussian basis function takes the distance metric as input and $\sigma$ as diagonal matrix.

$$d(\bar{x}; \tilde{\sigma}, \sigma) = \| \bar{x} - \tilde{\sigma} \|_\sigma = \sqrt{\bar{x}^T \sigma^{-1} \bar{x}}$$

$$\mu_k(d(\bar{x}; \tilde{\sigma}, \sigma)) = \exp(-d(\bar{x}; \tilde{\sigma}, \sigma))$$

**Expertise Level**

When two expertise domains (Gaussian basis functions) are located close to each other, there will be a high degree of overlapping. As a result, one of the local experts associated with the expertise domains may appear to be redundant. In such a case, a single local expert is sufficient to approximate the desired function within the expertise domain. In order to avoid such a redundancy, expertise level gating function is introduced to denote the importance of the local expert model so that the unimportant local expert may be deleted. Each expertise domain is multiplied by a weighting factor $\alpha$, known as
expertise level function. It is noted that $\alpha_i$ must also be a non-decreasing function of an adjustable parameter $\beta_i$. Smoothness of expertise function is also required because it is necessary for the gradient-based learning method. In addition, function $\alpha_i$ must be nonlinear because otherwise, it may be absorbed by local model function. To satisfy these requirements, the weighting factor $\alpha_i$ is defined as a logistic sigmoid function so that $\alpha_i$ is bounded ($0 < \alpha_i < 1$) for any real value of $\beta_i$ where $\beta_i$ is an adjustable parameter.

$$\alpha_i = \frac{1}{1 + e^{-\beta_i}}$$

The expertise level function is then multiplied by expertise domain function. Since the value of $\alpha_i$ is bounded ($0 < \alpha_i < 1$), the peak value of expertise domain (Gaussian basis function) $\mu_i(x; c_i, \sigma_i)$ is gated by a weighting factor $\alpha_i$. Therefore, the peak value of expertise domains varies according to the changes of $\beta_i$ in accordance with the importance of their contribution to the overall network output.

Fig. 7 shows the effect of varying control parameter $\beta_i$ on the peak value of expertise domain. In GMN, the initial value of $\beta_i$ is always set to zero and it is subject to error-based learning. The slope of expertise level function $\alpha(\beta)$ is the greatest in the mid-range in the vicinity of $\beta = 0.5$. Near both extremes, the slope diminishes asymptotically towards zero as depicted in Fig. 8. Therefore, the expertise level value is sensitive to the onset of learning, which exhibits fast learning capability at the initial stage. This is closely similar to the learning curve of a human being.

**Partition of Unity**

For modeling tasks the expertise domain (Gaussian basis functions) should form a partition of unity for the input space, i.e. at any point in the input space, the sum of all expertise domains should be unity. This is an essential requirement for the network to be able to globally approximate any desired function as complex as the local expert model. The partition of unity also ensures that any variation in output over the input space is due only to the local expert model. Therefore, the gated expertise domains are normalized to achieve the partition of unity.
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

Fig. 7: $\beta$ is the control parameter of expertise level function, $a(\beta)$. The modifying value of $\beta$ has the effect of suppressing the expertise domain.

Fig. 8: The expertise level function of local experts

$$\Phi_i(\bar{x}) = \frac{\alpha_i \mu_i(d(\bar{x}; \bar{c}_i, \bar{\sigma}_i))}{\sum_{k=1}^{m} \alpha_k \mu_k(d(\bar{x}; \bar{c}_k, \bar{\sigma}_k))}$$

(4)

where $\mu_i(d(\bar{x}; \bar{c}_i, \bar{\sigma}_i))$ is the general unnormalised expertise domain, therefore the $m$ normalized expertise domains $\Phi_i(\bar{x})$ sum to unity.

$$\sum_{i=1}^{m} \Phi_i(\bar{x}) = 1$$

(5)
Normalization is very important for GMN, often making the GMN model less sensitive to the poor choice of Gaussian basis function parameters Werntges (1993). The normalization procedure is depicted as block 'N' in Fig. 5.

**Local Expert Model**

Growing Multi-experts Network utilizes a group of local expert models to perform function approximation. The approximation capability of GMN is dependent on the choice of local expert models. Local expert models may be a constant model, linear model, polynomial model or even a nonlinear model. Linear function is chosen as local expert models in GMN architecture. Linear models, although very restricted in their representational ability, are useful for a large range of problems. This is due to their simple representation, their ease of interpretability, and their robustness against noisy or missing data. It makes sense, therefore, to include the ability to at least be able to form a linear model within the expertise domain. The linear model is expressed as follows:

\[
\tilde{\theta}_e(x) = \tilde{x}^T \tilde{b}_e + \tilde{b}_{0,e} \tag{6}
\]

where \(\tilde{b}_e\) and \(\tilde{b}_{0,e}\) are adjustable parameters of the local expert and \(\tilde{x}\) is the input vector. Each local linear model is associated with the normalized gated expertise domain. The overall output of the GMN is described as given below:

\[
\hat{y} = \sum_{e=1}^{n} \Phi_e(\tilde{x}) \tilde{\theta}_e(x) \tag{7}
\]

The contribution of each local expert model is determined by the validity of the input data to the expertise domain. The magnitude of contribution is further gated by expertise level. In general, the expertise level may be an indicator of local expert redundancy. On the other hand, the overlapping of expertise domain allows the smooth combination of local expert outputs to produce a smooth approximation. The trained network structure is viewed as a decomposition of the complex, nonlinear system into a set of locally active sub-models which are then smoothly integrated by their associated expertise domain and expertise level. The next section will present the learning algorithm of GMN.

**Local Expertise Insertion based on Certainty Factor**

Certainty factor was introduced by Buchanan and Shortlife (1985) for a medical diagnostic system. Donald K. Wedding II and Krzysztof J. Gios conducted a comparative study of this method and the other technique for measuring RBF reliability Wedding et al. (1995). Certainty factor can be generated by GMN networks by using the \(\mu(d(\tilde{x};\tilde{c},\sigma))\) values from the expertise domain. Referring to Equation (2), it is obvious that this function returns values between 0.0 and 1.0 in the same range as allowed by certainty factors.

When the output \(\mu(d(\tilde{x};\tilde{c},\sigma))\) of an expertise domain is high (near 1.0), then this indicates that a point lies near the center of a cluster and therefore the network is confident about the prediction. A value of 1.0 exactly indicates that the data falls directly upon the center of an expertise domain. A value that is very small (near 0.0) will lie far outside the expertise domain, so the network is most likely to perform extrapolation. In
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

this scheme, N values of $\mu(d(\bar{x};\bar{e},\bar{\sigma}))$ are generated, one for each of the local experts in the hidden layer of the GMN network. The following equation is used to calculate the certainty factor using the recursive formula Wedding (1995).

$$C_{F_i} = C_{F_{i-1}} + (1 - C_{F_{i-1}}) \max_i \left( \mu(d(\bar{x};\bar{e},\bar{\sigma})) \right)$$

(8)

In this equation, the value of $i$ refers to the number of local experts that will be used to determine the certainty factor of the GMN network. In general, the value of $i$ should be set to N (the number of local experts in the hidden layer), but it can also be set to a lower value if computational time is critical to the application. The term $C_{F_{i-1}}$ refers to the certainty factor of the network using only $i-1$ nodes to calculate the confidence level. The value of $C_{F_0}$ (the certainty when local experts in the hidden layer are used) is defined to be 0.0. Lastly, the term $\max_i (\mu(d(\bar{x};\bar{e},\bar{\sigma})))$ is defined as the $ith$ largest value of $\mu(d(\bar{x};\bar{e},\bar{\sigma}))$. So, $\max_i (\mu(d(\bar{x};\bar{e},\bar{\sigma})))$ would mean the largest value of $\mu(d(\bar{x};\bar{e},\bar{\sigma}))$ and $\max_{i} (\mu(d(\bar{x};\bar{e},\bar{\sigma})))$ would be the second largest value of $\mu(d(\bar{x};\bar{e},\bar{\sigma}))$.

On-line GMN Identification Algorithm

For online identification application using RBF network, a recursive algorithm is required for the learning process of GMN network. For an effective adaptive identification scheme, the learning process requires allocation of new local experts depending on the inputs as well as network parameter adaptation. The algorithm is explained as follows:

The network begins with two local experts that are initialized randomly. As observations are received some of the input data will be taken as the new local experts. The following two criteria decide whether an input $x_n$ should be added to the hidden layer of the network:

$$C_{F_i}(n) \leq e_{df}$$

$$e_{man} = \sqrt{\frac{\sum_{i=M-1}^{M} [y(i) - \bar{y}(i)]^2}{M}} \geq e_{oldmax}$$

(9)

If the criteria above are fulfilled, then allocate a new local expert with

$$x_{new} = x_n$$

$$\sigma_{new} = \frac{\sum_{i=1}^{in-NB} \|x_i - \bar{c}_w\|}{NB}$$

$$a_{new} = 0$$

$$\psi_{new} = 0$$

$\bar{c}_w$ is the center of a local expert whose distance from the incoming input feature $x_n$ is the nearest among those of all the other local experts. The spread of new expertise domain is calculated as the average distance of NB neighborhood local experts from $\bar{c}_w$. 

If the criteria above are not fulfilled, then

\[
\tilde{e}_i^n = \tilde{e}_i^{n-1} + \eta_i \left[ \frac{\partial E^{n-1}}{\partial \tilde{e}_i^{n-1}} + \eta_{LM} (\tilde{e}_i^{n-1} - \tilde{e}_i^{n-2}) \right] \\
\tilde{\sigma}_i^n = \tilde{\sigma}_i^{n-1} + \eta_i \left[ \frac{\partial E^{n-1}}{\partial \tilde{\sigma}_i^{n-1}} + \eta_{LM} (\tilde{\sigma}_i^{n-1} - \tilde{\sigma}_i^{n-2}) \right] \\
\alpha_i^n = \alpha_i^{n-1} + \eta_i \left[ \frac{\partial E^{n-1}}{\partial \alpha_i^{n-1}} + \eta_{LM} (\alpha_i^{n-1} - \alpha_i^{n-2}) \right] \\
\bar{P}_i^n = \frac{1}{\lambda} \left[ \bar{P}_i^{n-1} - \bar{P}_i^{n-1} \bar{x}^n \bar{x}^n^T \bar{P}_i^{n-1} \right] \\
\tilde{\psi}_i^n = \tilde{\psi}_i^{n-1} + \Phi_i^n \bar{P}_i^n \bar{x}^n \bar{x}^n^T
\]

The expertise domain center \( \tilde{e}_i \), the expertise domain spread \( \tilde{\sigma}_i \) and the expertise level \( \alpha_i \) are learned using momentum-based gradient descent method where \( \eta_i, \eta_{LM} \) are learning rates and \( \eta_{LM} \) are momentum rates. The local expertise linear models are learned by Forgetting Weighted Recursive Least Square learning algorithm Ljung (1983). The initial parameter vector \( \Phi(0) \) is set as zero. The initial covariance matrix \( P(0) \) is assigned as one. To cope with changing environment, in general, \( \lambda \) is set a value within \( 0.9 < \lambda < 1 \). For \( \delta_{LM} \) consecutive observations, if the normalized expertise level of local expertise model is below the average expertise value, then prune the respective local expert.

\[
\frac{\gamma_i}{m} < \frac{1}{m} \\
\sum_{i=1}^{m} \gamma_i
\]

\( m \) is the total number of local experts.

**Robot Arm Dynamics**

A robot manipulator is typically modeled as a serial chain of \( n \) rigid bodies. In general, one end of the chain is fixed to some reference surface while the other end of the chain is free, thus forming an open kinematic chain of moving rigid bodies. The vector equation of motion of such a device is written in the form:

\[
D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = T_o
\]

This vector equation represents the dynamics of an interconnected chain of ideal rigid bodies, where is the vector of joint actuator torques and is the vector of generalized joint positions. It is a matrix, usually referred to as the manipulator mass matrix containing
the kinetic energy functions of the manipulator. It represents torques arising from centrifugal and Coriolis forces caused by gravitational effects when the manipulator is moving in its work space. In general, equation (17) is very complicated for all but the simplest manipulator configuration. The matrices in equation (17) have distinctive properties Lewis et al. (1993) that are the direct results of the application of the Euler-Lagrange method of formulating the dynamic equations of the manipulator system. Properties such as symmetry, boundedness and positive definiteness enable the exploitation of the manipulator model to design a model based control law for tracking control application properties Lewis et al. (1993). The resulting control law generates actuator torques that cause the manipulator to follow a desired trajectory in its workspace while maintaining small tracking errors. Ideal performance is achieved only when complete knowledge of manipulator model is available. However, in reality, model uncertainty is inevitable even when the best effort has been put into the analysis and derivation of the dynamic model.

**Nonlinear Variable Gain Proportional-Derivative (PD) Controllers**

In general, the linear discrete-time PD controller can be written as

$$T_p(t) = K_p(\theta_d(t) - \theta(t)) + K_d(\dot{\theta}_d(t) - \dot{\theta}(t))$$

(18)

Now, we rewrite equation (7) to represent the general GMN controller in the following manner:

$$u(t) = \sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x})$$

$$\sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x}) \left( b_{i1}(x)x_1(t) + b_{i2}(x)x_2(t) \right)$$

(19)

Without loss of generality, we assume $a_{ij} = 0$ for all $j$ in GMN controller, $x_1(t) = \theta_d(t) - \theta(t)$ and $x_2(t) = \dot{\theta}_d(t) - \dot{\theta}(t)$ and thus (18) can be written as

$$u(t) = \sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x})$$

$$\sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x}) \left( b_{i1}(x)(\theta_d(t) - \theta(t)) + b_{i2}(x)(\dot{\theta}_d(t) - \dot{\theta}(t)) \right)$$

(20)

From equation (20), one sees that the GMN controller is actually a nonlinear controller with variable gain changing with state of the input variables $(\theta_d(t) - \theta(t))$ and $(\dot{\theta}_d(t) - \dot{\theta}(t))$. Thus, the output from GMN controller becomes a nonlinear variable gain control signal written as

$$T_{pd}(t) = \sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x})$$

$$\sum_{i=1}^{m} \alpha_i \Phi_i(\bar{x}) \left( b_{i1}(x)(\theta_d(t) - \theta(t)) + b_{i2}(x)(\dot{\theta}_d(t) - \dot{\theta}(t)) \right)$$

(21)
The hybrid linear PD controller and GMN becomes a nonlinear adaptive PD controller and can be written as

\[ T_i(t) = T_{nl}(t) + T_{req}(t) \]

\[ = \left( K_p + \sum_{i=1}^{n} \alpha_i \Phi_i(x)b_{\theta_i}(x) \right) (\theta_i(t) - \theta(t)) + \]

\[ \left( K_d + \sum_{i=1}^{n} \alpha_i \Phi_i(x)b_{\dot{\theta}_i}(x) \right) (\dot{\theta}_i(t) - \dot{\theta}(t)) \]  

(22)

**Feedback-Error Learning**

Kawato and Mitsuo (1993) offer a complete solution to the inverse dynamics problems that use connectionist networks. Their work is based on the study of physiological information. They called this learning method feedback-error learning. In this paper, we adopt this learning scheme using feedback-error learning for a dynamic structural network applied to an adaptive nonlinear PD controller. In this learning scheme, the conventional PD controller serves the two purposes as the linear controller and as a stable trainer that provides error signal for the Growing Multi-Experts Network. It is shown in Fig. 9.

![Fig. 9: Block diagram of nonlinear adaptive PD controller](image)

**Simulation Studies – Variable Load Compensation**

The simulations are based on a situation that is often encountered when a 2-link robotic arm is used to transfer work pieces in a production line Song et al. (1994) as shown in Fig. 10. Suppose that the end-effector starts moving from position \( P_1 \) at \( t_0 = 0 \). It then picks up a container that contains \( n_p = 4 \) parts at time \( t_p^* = 2.0s \), moves to a workstation \( P_1 \) where \( n_{d_1} = 2 \) parts are removed from the container at time \( t_{d_1}^* = 4.0s \) and finally moves to workstation \( P_2 \) where the remaining parts \( n_{d_2} = 2 \) are removed from the container at
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

Fig. 10: The two-link robotic arm

The load transitions can suddenly change in robot dynamics and are unpredictable. So the control strategy developed in this paper is an attractive scheme in this application because it does not require exact values for the mass of the parts in the container, the pick-up time, or the drop-off times.

The payload variation model can be represented as

\[
m^*_p = \left( \sum_{i=1}^{n} m_i (t) + m_c \right) \sum_{i=1}^{n} m_i (t) U_i \left(t - t^*_i \right) - \sum_{i=1}^{n} m_i (t) U_i \left(t - t^*_i \right) - \sum_{i=1}^{n} m_i (t) U_i \left(t - t^*_i \right)
\]  

where \( m_p, m_c \) are weight of the parts and container. They are assigned as 1.2 kg and 0.5 kg respectively. In the simulation, a two-link robot is used with the dynamics model from equation (23). The two-link robot is illustrated in Fig. 10.

The payload variation model can be represented as

\[
D_0(q) = \begin{bmatrix}
(m_1 + m_2) l_1^2 + m_1 l_2^2 + 2 m_1 l_1 l_2 \cos(q_1) & m_1 l_2^2 + m_1 l_1 l_2 \cos(q_1) \\
2 m_1 l_1 l_2 \cos(q_1) + m_1 l_2^2 & m_1 l_2^2
\end{bmatrix}
\]  

\[
C_0(q, \dot{q}) = \begin{bmatrix}
(m_1 + m_2) l_2 \sin(q_2) \dot{q}_2 & -m_2 l_2 \sin(q_2) \dot{q}_1 - m_2 l_2 \sin(q_2) \dot{q}_2 \\
0 & m_2 l_2 \sin(q_2) \dot{q}_1
\end{bmatrix}
\]

\[
G_0(q) = \begin{bmatrix}
(m_1 + m_2) l_1 g \cos(q_1) + m_1 l_2 g \cos(q_1 + q_2) \\
m_2 l_2 g \cos(q_1 + q_2)
\end{bmatrix}
\]

where \( m_i \) and \( l_i \) represent the mass and length of link \( i (i = 1, 2) \), respectively; \( g = 9.81 \text{m/s}^2 \) is the gravitational constant; and \( q_1 \) and \( q_2 \) are the angles for the joints. Note that in deriving these equations, it is assumed that the mass of each link is a point mass, concentrated at the distal end of the link. Let the parts and the container be treated as a point-mass payload, which has the total mass \( m^*_p \).
From equations (17) and (27), the loaded robot dynamic model can be expressed as:

\[ T_l(m, q, \dot{q}, \ddot{q}) = m \begin{bmatrix} l_1^2 + l_2^2 + 2l_1l_2 \cos(q_1) - l_1^2 \cos(q_2) & \dot{q}_1 \\ l_1^2 \sin(q_2) \dot{q}_2 & l_1^2 \cos(q_2) \dot{q}_2 \end{bmatrix} + m_\ddot{q}_1 + m_\ddot{q}_2 = \frac{\ddot{l}}{\ddot{q}_1}, \frac{\ddot{l}}{\ddot{q}_2} \]

In this simulation study, the robot parameters are set as \( l_1 = 0.432 \text{ m}; \ l_2 = 0.432 \text{ m}; \ m_1 = 15.91 \text{ kg}; \ m_2 = 11.6 \text{ kg}; \). The main concern of the simulation is to test the effectiveness of the proposed control strategy; it is assumed that the desired path for the end-effector at each stage is given such that the corresponding paths for the joints are determined to be the Hermite polynomial trajectory \( Ge et al. (1998) \) of the third degree in \( t \) with continuous bounded position, velocity and acceleration. The general expression for this kind of desired trajectory is given as:

\[ q_d(t, t_d) = q_i + \left(-\frac{2.0}{t_d^3} + \frac{3.0}{t_d^2} \right) (q_f - q_i) \]  

where \( q_i \) and \( q_f \) are the arm initial and final positions, respectively, and \( t_d \) represents the time at which the desired arm trajectory reaches the desired final position. In this simulation example, the following values were chosen for the desired trajectories:

\[ t_d = 2.0 \text{ s}, \quad q_i(0) = [0.0 \ 0.0]^T \text{ rad}, \quad q_f(t_d) = [1.0 \ 2.0]^T \text{ rad} \]

The gains for the PD-controller were chosen:

\[ K_p = [900 \ 400], \quad K_d = [30 \ 20] \]

On-line GMN parameter settings are depicted in Table 1.

<table>
<thead>
<tr>
<th>Parameter setting of on-line GMN for nonlinear PD controller</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1st Joint</strong></td>
</tr>
<tr>
<td>Certainty Factor, ( \delta_p )</td>
</tr>
<tr>
<td>Pruning threshold, ( \delta_{max} )</td>
</tr>
<tr>
<td>Adaptation parameter, ( \eta_r )</td>
</tr>
<tr>
<td>Adaptation parameter, ( \eta_p )</td>
</tr>
<tr>
<td>Adaptation parameter, ( \eta_g )</td>
</tr>
<tr>
<td>Momentum term, ( \eta_m )</td>
</tr>
<tr>
<td>Forgetting factor, ( \lambda )</td>
</tr>
<tr>
<td>Number of adaptation steps before pruning, ( \delta_{adap} )</td>
</tr>
<tr>
<td>Edge removal threshold, ( a_{max} )</td>
</tr>
</tbody>
</table>
Case 1: Non-adaptive Control
When the GMN controller is not activated, the resulting control action is effectively a simple PD control. It is observed from these results that the non-adaptive PD control has a significant joint angle and joint velocity tracking error due to the dynamic effects of load transition. Figs. 11a and 11b show the desired trajectories and trajectories obtained from non-adaptive PD controller. Joint angle 2 has substantial tracking error as shown in Fig. 11b. Figs. 11c and 11d illustrate the joint velocity tracking profile for first joint and second joint trajectory respectively.

Case 2: Adaptive Control
Figs. 12a and 12b, show that the tracking error is much smaller than the non-adaptive case because of the "learning" mechanism. Joint velocity of first joint and second joint link also exhibit significant improvement as shown in Figs. 12c and 12d. The position tracking error of non-adaptive PD and adaptive PD controller is compared in Figs. 13a and 13b. It is clear that addition of GMN control signal makes a significant improvement in the tracking performance whereas Figs. 14a and 14b indicate substantial reduction in velocity tracking error.
Fig. 11: Fixed PD controller (a) first joint position tracking (b) second joint position tracking (c) first joint velocity tracking (d) second joint velocity tracking
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

Fig. 12: Adaptive PD controller (a) first joint position tracking (b) second joint position tracking (c) first joint velocity tracking (d) second joint velocity tracking
Fig. 13: Comparison of tracking error (a) joint one position (b) joint two position
A Self-learning Nonlinear Variable Gain Proportional Derivative (PD) Controller in Robots

DISCUSSION
The proposed adaptive PD controller is compared with conventional PD controller. The comparison of the two schemes is based on computer simulation. Comparison of the position tracking performance for the two control schemes shows a substantial improvement in steady state error. The augmented control signals from GMN networks have successfully reduced the steady state error as shown in Figs. 13a and 13b. The maximum tracking errors for joint one and two using GMN-based control scheme were found to be 0.10 and 0.05. In the case of conventional PD control scheme, the maximum tracking errors were 0.24 and 0.16 respectively. The reduction of tracking error is due to the efficacy of adaptive PD controller in compensating for the unstructured uncertainty caused by variable load in two-link robot.

CONCLUSION
We have presented a novel self-learning nonlinear PD controller using a class of dynamic structure network known as Growing Multi-Experts Network. This method results in a network that is “economic” in terms of network size and learns the required control behavior on the fly during the robot operation. The effectiveness of the nonlinear adaptive PD controller scheme was put to test with variable load and friction compensation in robot manipulators. The performance results demonstrate the learning capabilities and better tracking performance of nonlinear adaptive PD controller in comparison to conventional linear PD controller.

REFERENCES


