Neural Networks
Lecture 9: Designing Controller Using Neural Networks

H.A. Talebi
Farzaneh Abdollahi

Department of Electrical Engineering

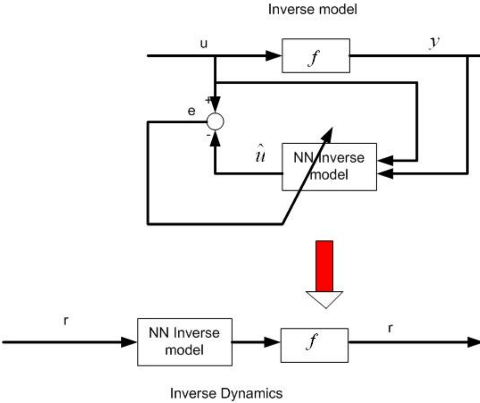
Amirkabir University of Technology

Open-Loop Inverse Dynamics

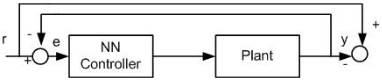
- The Inverse model obtained from identification is directly applied.
- $\therefore$ Considering reference signal r

$$y = f^{-1} f r = r$$

- This method can be considered as Indirect adaptive control



NN in Control Feedback



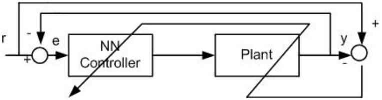
- Objective: Tacking reference signal r
- But: In this model, output of NN for training is not available $\rightsquigarrow$ BP can not be applied directly.

$$e = r - y, \quad E = \frac{1}{2}e^2$$

$$\triangle w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

- Only output of the plant, y is available.

Gradient Through Plant



- The plant can be considered as output layer of NN with fixed weights
- $\therefore$ desired output of the NN is available and BP algorithm can be employed.

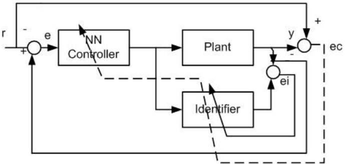
$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial e} \cdot \frac{\partial e}{\partial y} \cdot \frac{\partial y}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial e} = e, \quad \frac{\partial e}{\partial y} = -1$$

$$\frac{\partial y}{\partial w_{ij}} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_{ij}}$$

- To train the NN, $\frac{\partial y}{\partial u}$ is required, therefore, this method is so-called Gradient through plant

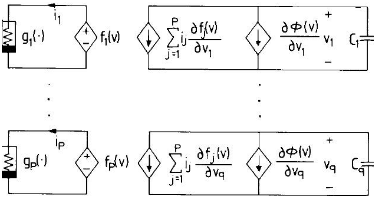
NN in Control Feedback



- If the plant dynamics is not known $\frac{\partial y}{\partial u}$ is not available!!
- Solution
  1. Using a NN identifier to identify the system dynamics directly.
     - Then apply $\frac{\partial \hat{y}}{\partial u}$ instead of $\frac{\partial y}{\partial u}$.
     - This method is so-called Gradient Through The Model of The Plant
  2. Approximate $\frac{\partial y}{\partial u}$ with $sign\{\frac{\partial y}{\partial u}\}$ which is usually available without knowing the dynamics
     - If the direction of the gradient is true, the magnitude of $\frac{\partial y}{\partial u}$ can be compensated by η

An Example of Optimal Control Using Hopfield [1]

- It can be shown that solution of the general nonlinear programming problem with inequality constraint: $\min_{v \in R^n} \phi(v)$ subject to $f_i(v) \geq 0 \quad i = 1, ..., q$ equivalents to solving the canonical nonlinear programming circuit.
- It is described by the following equation $C_i \frac{dv_i}{dt} = -\frac{\partial \phi}{\partial v_i} - \sum_{j=1}^{p} g(f(v)) \frac{\partial f_j}{\partial v_i}$.
- $\phi(.)$ and $f(.)$ and their 1^{st} and 2^{nd} order partial derivatives exist and are continuous.
- $g(.)$ is passive monotone nondecreasing fcn, continuous, but not necessarily differentiable, imposes the constraints in the circuit realization.



Canonical nonlinear programming circuit–dynamic model.

- Let us consider discrete time, linear quadratic regulator (LQR) with objective:

$$\min_{u_k, x_k} J = \frac{1}{2} x_N^T S x_N + \frac{1}{2} \sum_{k=0}^{N-1} \{x_k^T Q x_k + u_k^T R u_k\}$$

subject to the plant dynamics $x_{k+1} = A x_k + B u_k, \quad k = 0, ..., N-1$

- $x \in R^n, \;\; u \in R^p$
- Assume (A, B) is controllable
- Define $v = (x_0, x_1, ..., x_N, u) \in R^{(N+1)n+p}$,

$$\Gamma = \begin{bmatrix} -A & I & 0 & 0 & \dots & 0 & 0 & -B & 0 & \dots & 0 & 0 \\ 0 & -A & I & 0 & \dots & 0 & 0 & 0 & -B & \dots & 0 & 0 \\ 0 & 0 & -A & I & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I & 0 & 0 & 0 & \dots & -B & 0 \\ 0 & 0 & 0 & 0 & \dots & -A & I & 0 & 0 & \dots & 0 & -B \end{bmatrix}$$

- $\therefore$ The problem can be defined as $\min_v J$ s.t. $\Gamma v = 0$

- The following theorem help us to describe equality constraint by inequalities
- Theorem: m equality constraints $\sum_{j=1}^{n} a_{ij}x_j = c_i$ for $i = 1, ..., m$ is equivalent to inequality constraints: $\sum_{j=1}^{n} a_{ij}x_j \leq c_i$ and $\sum_{j=1}^{n}(\sum_{i=1}^{m} a_{ij})x_j \geq \sum_{i=1}^{m} c_i$
- $\therefore$ Our problem can be presented as
  $\min_v J$ subject to $f(v_i) = \sum_{j=1}^{(N+1)n+p} \bar{a}_{ij}v_j \geq 0$ for $i = 1, ..., Nn+1$
- $\bar{a}_{ij}$ are the elements of matrix $\bar{A}$:

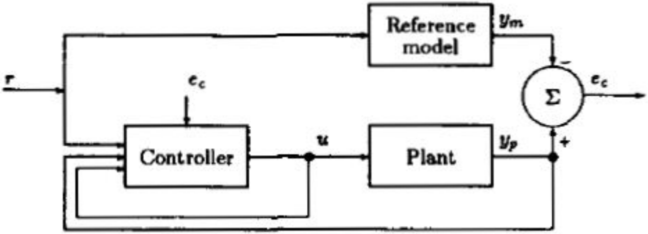
$$\begin{bmatrix} -A & I & 0 & 0 & \dots & 0 & 0 & -B & 0 & \dots & 0 & 0 \\ 0 & -A & I & 0 & \dots & 0 & 0 & 0 & -B & \dots & 0 & 0 \\ 0 & 0 & -A & I & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I & 0 & 0 & 0 & \dots & -B & 0 \\ 0 & 0 & 0 & 0 & \dots & -A & I & 0 & 0 & \dots & 0 & -B \\ \alpha^T & & \gamma & \gamma & \dots & \gamma & \underline{1} & \beta^T & \beta^T & \dots & \beta^T & \beta^T \end{bmatrix}$$

- $\gamma = (\alpha^T - \underline{1})$, $\alpha = (\sum_{i=1}^{n} a_{i1}, \ \sum_{i=1}^{n} a_{i2}, ..., \sum_{i=1}^{n} a_{in})^T$,
  $\underline{1} = (1, 1, \ldots, 1)^T, \beta = (\sum_{i=1}^{n} b_{i1}, \ \sum_{i=1}^{n} b_{i2}, ..., \sum_{i=1}^{n} b_{in})^T$

- To implement the NN, we should find:
  - $\frac{\partial f_i}{\partial v_i} = \bar{a}_{ij}$
  - $\frac{\partial J}{\partial v_i} = \begin{cases} \sum_{j=1}^{n} q_{i-ln,j} v_{ln+j} & i = ln+1, ..., (l+1)n, \\ & l = 0, ..., N-1 \\ \sum_{j=1}^{n} s_{i-Nn,j} v_{Nn+j} & i = Nn+1, ..., (N+1)n \\ \sum_{j=1}^{n} r_{i-(N+1)n,j} v_{(N+1)n+j} & i = (N+1)n+1, ..., (N+1)n+p \end{cases}$
  - q_{ij}, s_{ij}, r_{ij} are elements of Q, S, R respectively

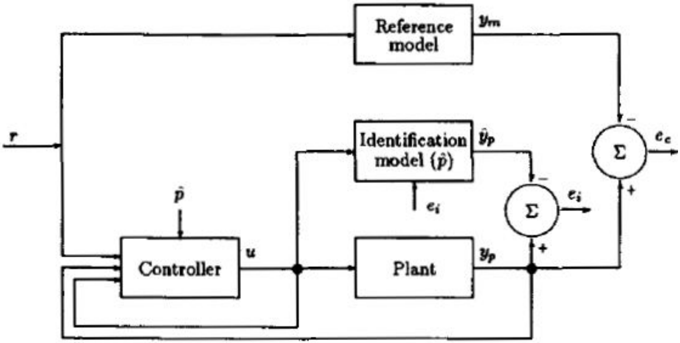
Adaptive Control Using Neural Networks

1. Direct Control
   - Parameters of the controller is directly adjusted to reduce the norm of output error

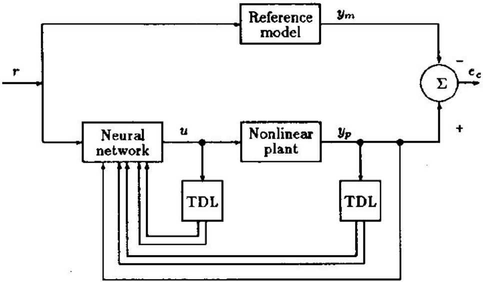


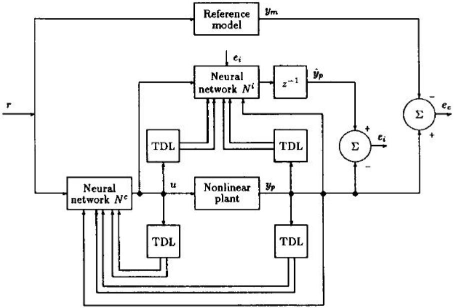
Adaptive Control Using Neural Networks

2. Indirect Control
   - The model of the plant is identified first and the parameters of the controller is defined based on identified model



- They can be controller y Neural networks





Example [2]

- Consider the difference equation:
  $y_p(k+1) = f[y_p(k), y_p(k-1)] + u(k)$
- $f(.)$ is unknown
- For the sake of simulation $f[y_p(k), y_p(k-1)] = \frac{y_p(k)y_p(k-1)[y_p(k)+2.5]}{1+y_p^2(k)+y_p^2(k-1)}$
- Reference model: $y_m(k+1) = 0.6y_m(k) + 0.2y_m(k-1) + r(k)$
- $r(k) = sin(\frac{2\pi k}{25})$: a bounded reference input
- Objective: Determine a bound control signal $u(k)$ s.t.
  $\lim_{k\to\infty} e_c(k) = y_p(k) - y_m(k) = 0$

Example Cont'd

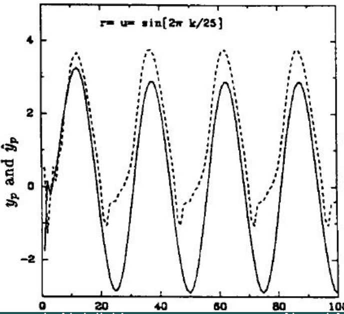
- If $f(.)$ was known the proper control signal would be $u(k) = -f[y_p(k), y_p(k-1)] + 0.6y_p(k) + 0.2y_p(k-1) + r(k)$ yields $e_c(k+1) = 0.6e_c(k) + 0.2e_c(k-1)$
  - $\therefore$ the reference model is a.s. since $\lim_{k\to\infty} e_c(k) = 0$
- Since the plant is unknown, assuming the unforced system is stable, $f(.)$ is estimated by series parallel NN identifier as $\hat{f}(.)$
- Hence $u(k) = -\hat{f}[y_p(k), y_p(k-1)] + 0.6y_p(k) + 0.2y_p(k-1) + r(k)$

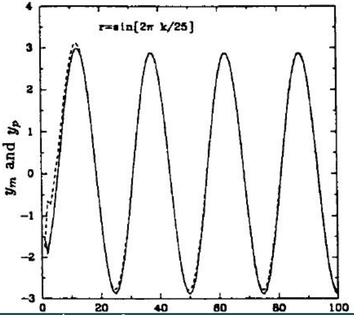
Example Cont'd

- Identification will be off-line
- Once the plant is identified in desired level of accuracy, control is initiated to make the plant output follow the reference model.
- Note that using the estimated function in fb loop may result in unbounded solution
- Hence for on-line control, identification and control should proceed simultaneously.
- The time interval T_i and T_c for updating the identification and control parameters should be chosen wisely.

Example Cont'd

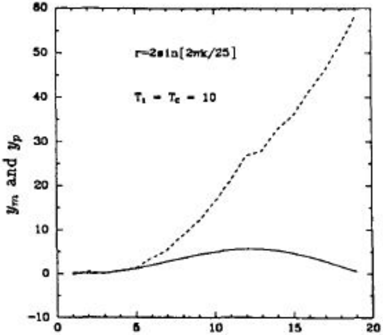
- a) Identified signal $\hat{y}_p$ (dashed) and output of the plant with no control action (solid)
- b) Response for $r = sin(\frac{2\pi k}{25})$ with control (dashed); reference signal (solid)
- $T_i = T_c = 1$





Example Cont'd

- Choose $T_i = T_c = 10$
- Response for $r = sin(\frac{2\pi k}{25})$ with control (dashed); reference signal (solid)
- $\therefore$ To have stable on-line control, the identification should be accurate enough before the control action is initiated!

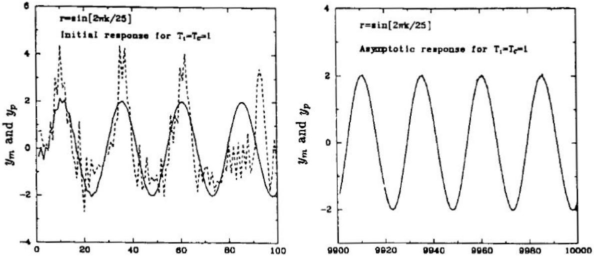


Example 2 [2]

- Consider the difference equation:
  $y_p(k+1) = f[y_p(k), y_p(k-1), ..., y_p(k-n+1)] + \sum_{j=0}^{m-1} \beta_j u(k-j) \; m \leq n$
- $f(.)$ and β_j are unknown; β_0 is nonzero with known sign
- For the sake of simulation
  $f[y_p(k), y_p(k-1)..., y_p(k-n+1)] = \frac{5y_p(k)y_p(k-1)}{1+y_p^2(k)+y_p^2(k-1)+y_p^2(k-2)}$;
  $\beta_0 = 1, \beta_1 = 0.8$
- Reference model:
  $y_m(k+1) = 0.32y_m(k) + 0.64y_m(k-1) - 0.5y_m(k-2) + r(k)$
- $r(k) = sin(\frac{2\pi k}{25})$: a bounded reference input
- Objective: Determine a bound control signal $u(k)$ s.t.
  $\lim_{k\to\infty} e_c(k) = y_p(k) - y_m(k) = 0$
- Assume $sgn(\beta_0) = +1; \beta_0 \geq 0.1$

Example 2 Cont'd

- The control signal is: $u(k) = \frac{1}{\hat{\beta}_0}[-\hat{f}_k[y_p(k), y_p(k-1), y_p(k-2)] - \hat{\beta}_1 u(k-1) + 0.32y_p(k) + 0.64y_p(k-1) - 0.5y_p(k-2) + r(k)]$
- Choose $T_i = T_c = 10$
- Response for $r = sin(\frac{2\pi k}{25})$ with control (dashed); reference signal (solid) left): first 100 sec; right) after 9900sec

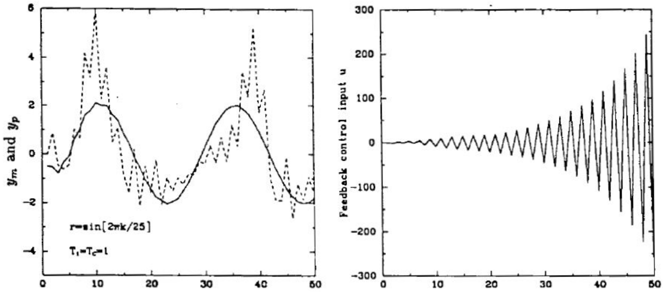


Example 3 [2]

- Consider dynamics similar to Example 2 but replace $0.8u(k-1)$ with $1.1u(k-1)$
- Apply similar controller
- The system is nonminimum phase (it has zero out of unit circle)
- $\therefore$ The output error is bounded but the control signal is unbounded

Example 3 Cont'd

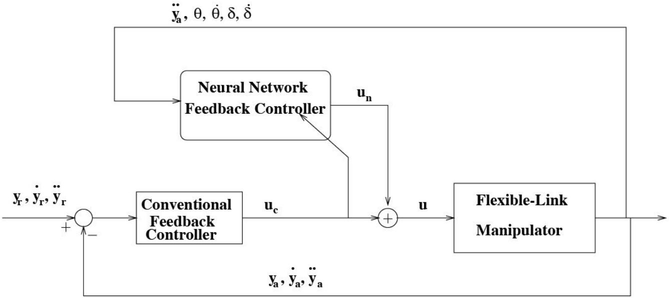
- left) Response for $r = sin(\frac{2\pi k}{25})$ with control (dashed); reference signal (solid)
- right) control signal $u(k)$



Inverse Dynamics Model Learning (IDML) [3]

- Consider the system dynamics $\ddot{x} = f(x, \dot{x}) + u,\ y = x$
- To track reference signal y_a, control signal can be defined $u = u_n + u_c$
  - $u_c = (-\ddot{y}_a) + K_1(\dot{y}_r - \dot{y}_a) + K_0(y_r - y_a)$ is conventional FB controller
  - $u_n = -f(x, \dot{x})$
- $f(.)$ is not known and is estimated by NN $\rightsquigarrow$ $u_n = \hat{f}$

- The error signal for training is $e_n = u - u_n = u_c$
- The learning rule is $\dot{w} = \eta \frac{\partial \hat{f}}{\partial w} u_c$



Example 4 [3]

- Consider one-link flexible arm:
  $$M(\delta)\begin{bmatrix}\ddot{\theta}\\ \ddot{\delta}\end{bmatrix}+\begin{bmatrix}h_1(\dot{\theta},\delta,\dot{\delta})+F_1\dot{\theta}+f_c\\ h_2(\dot{\theta},\delta)+K\delta+F_2\dot{\delta}\end{bmatrix}=\begin{bmatrix}u\\ 0\end{bmatrix}$$
  - θ: hub angle
  - δ: deflection variable
  - h_1 and h_2 are Coriolis and Centrifugal forces, respectively
  - $M(\delta)$: P.D. inertia matrix
  - u: torque
  - F_1: viscus damping; F_2 damping matrix;
  - f_c hub friction; K stiffness matrix

Example 4 Cont'd

- The nonlinear dynamics is assumed to be unknown
- For the sake of simulation, the numerical values are
  - $M(\delta) = \begin{bmatrix} m(\delta) & 1.0703 & -0.0282 \\ 1.0703 & 1.6235 & -0.4241 \\ -0.0282 & -0.4241 & 2.592 \end{bmatrix};$
    $m(\delta) = 0.9929 + 0.12(\delta_1^2 + \delta_2^2) - 0.24\delta_1\delta_2$
  - $K = \begin{bmatrix} 17.4561 & 0 \\ 0 & 685.5706 \end{bmatrix}$
  - $h_1(\dot{\theta}, \delta, \dot{\delta}) = 0.24\dot{\theta}[(\delta_1 - \delta_2)\dot{\delta}_1 - (\delta_1 - \delta_2)\dot{\delta}_2]$
  - $h_2(\dot{\theta}, \delta) = \begin{bmatrix} -0.12\dot{\theta}^2(\delta_1 - \delta_2) \\ -0.12\dot{\theta}^2(\delta_2 - \delta_1) \end{bmatrix}$
  - $f_c = C_{coul}(\frac{2}{1+e^{-10\dot{\theta}}} - 1);\ C_{coul} = \begin{cases} 4.74 & \dot{\theta} > 0 \\ 4.77 & \dot{\theta} < 0 \end{cases}$
- By output redefinition, the nonminimum phase problem is solved

Example 4 Cont'd

- The NN structure:
  - Three layer: 4 input; 5 hidden,1 output
- $K_0 = 1; K_1 = 2$

Example 4 Cont'd

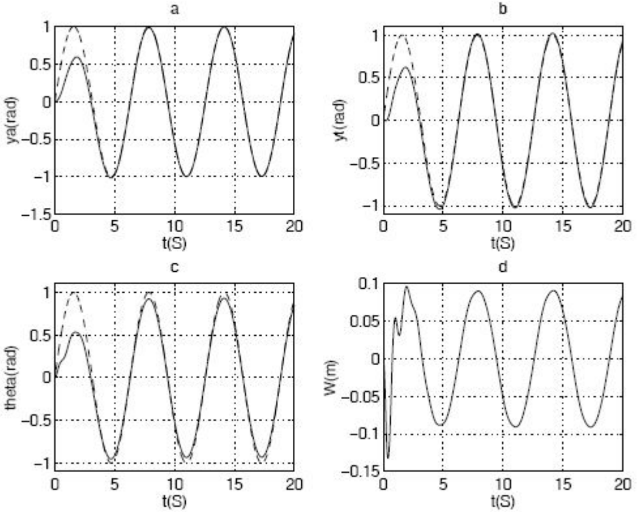
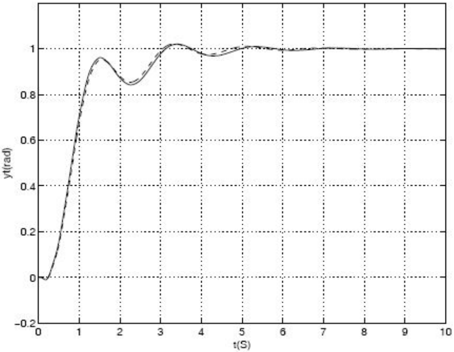


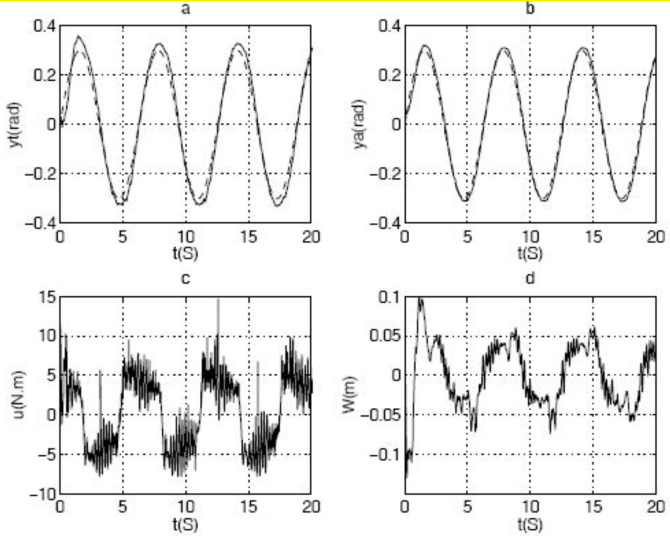
Figure 4.7: Output responses for System II to $sin(t)$ reference trajectory using the

Example 4 Cont'd



igure 4.8: Actual tip responses to step input for System II using the IDML neural etwork controller; (dashed line corresponds to model with Coulomb friction at the ub).

Example 4 Cont'd



References

M P. Kennedy and L. O. Chua, “Neural networks for nonlinear programming,” *IEEE Transactions of Circuit and Systems*, vol. 35, no. 5, pp. 554–562, 1988.

K.S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4–27, March 1990.

H.A. Talebi, .R.V Patel and K. Khorasani, *Control of Flexible-link Manipulators Using Neural Networks*.

Springer, 2001.