Soft Comput (2013) 17:431–444
DOI 10.1007/s00500-012-0918-1

ORIGINAL PAPER

Discrete-time weight updates in neural-adaptive control

D. Richert · K. Masaud · C. J. B. Macnab

Published online: 30 September 2012

Abstract Typical neural-adaptive control approaches update neural-network weights as though they were adaptive parameters in a continuous-time adaptive control. However, requiring fast digital rates usually restricts the size of the neural network. In this paper we analyze a delta-rule update for the weights, applied at a relatively slow digital rate. We show that digital weight update causes the neural network to estimate a discrete-time model of the system, assuming that state feedback is still applied in continuous time. A Lyapunov analysis shows uniformly ultimately bounded signals. Furthermore, slowing the update frequency and using the extra computational time to increase the size/accuracy of the neural network results in better performance. Experimental results achieving link tracking of a two-link flexible-joint robot verify the improved performance.

Keywords Neural-adaptive control · Fuzzy-adaptive control · Discrete-time control · Lyapunov stability · Robot manipulators · Flexible-joint robots

1 Introduction

Originally, researchers saw the potential of neural networks in control systems to perform discrete mappings. In robotics specifically, a neural network can learn the static discrete-time inverse dynamics: mapping desired positions/velocities at a future time step to the currently applied control torques as in Miller et al. (1990). However, adapting to changing dynamics like significant payloads requires on-line real-time weight updates. It is now well known that a continuous-time adaptive Lyapunov design can provide stable weight updates. In this paradigm, the neural-network basis functions take the place of linear-in-parameter (LP) regressor matrices, and weights take the place of adaptive parameters. When using neural networks with adaptive weight updates, i.e. neural-adaptive control, choices include backpropogation networks (Kim et al. 2004; Wang et al. 2007), radial basis function networks (Nakanishi and Schaal 2004), and the Cerebellar Model Arithmetic Computer (CMAC) (Kim and Lewis 2000). When using fuzzy approximators the centers of fuzzy membership functions become the adaptive parameters (Lee and Zak 2004). Various combinations and modifications of these basic types are often proposed to improve approximation abilities and adaptation rates, such as the output recurrent CMAC (Chiu 2010, 2011), generalized neurons (Chaturvedi and Malik 2007), and the feedback fuzzy neural controller (Frayman and Wang 2002).

In this paper we use the platform of a flexible-joint robot with large (exaggerated) elasticity to test the proposed method in an application where achieving both stability and performance is non-trivial. A survey of control techniques applied to flexible-joint robot appears in (Ozgoli and Taghirad 2006). Here, we look at adaptive control for link (output) tracking. Although traditional linear-in-parameters (LP) adaptive controls have been proposed for two-link flexible-joint robots, for example, (Huang and Chien 2009; Kim and Lee 2004), due to the size of the regressor matrices experimental validation is not trivial for multiple links; most experimental results utilize the single-link and/or single-flexible-joint, for example (Huang and Chen 2004). The only example of an experiment (for adaptive link-tracking of a 2-link flexible-joint robot)

D. Richert · K. Masaud · C. J. B. Macnab (✉)
Department of Electrical and Computer Engineering,
University of Calgary, Calgary, AB T2N-1N4, Canada
e-mail: cmacnab@ucaglary.ca

found by the authors was in (Dixon et al. 2000), which takes about ten repetitions of the trajectory before convergence due to the over-parameterized design, and requires an interm trajectory that slowly moves toward the desired trajectory due to the use of high-gain nonlinear-damping terms. On the other-hand, the use of neural-adaptive control offers a low-gain solution with the ability to avoid over-parameterization (using tuning functions) which results in faster convergence time without need of interm trajectories, with experimental verification on a 2-link flexible-joint robot in (Macnab 2010).

Mathematical treatment of a neural network as if it were a continuous-time signal may not constitute the most practical approach. Highly accurate, and thus large, neural networks can take significant time calculating outputs and applying weight updates. This implies a digital design of the weight updates. However, few papers appear in the robotics literature on this topic, especially for robots with elastic degrees of freedom. Purely digital designs for neural-adaptive control of a rigid robot appear in (Jagannathan 1999; Lei and Wu 2006; Suna et al. 2002). Robots with elastic degrees of freedom offer a challenge to the control designer, as the under-actuated nonminimum phase nature of the system makes achieving stability a challenge. Experimental validation becomes important, since stability proofs can rely on unrealistic assumptions about physical systems including unconstrained actuation, lack of unmodeled dynamics, and continuous signals. Violation of these assumptions in real elastic systems may lead to excitation of natural frequencies and instability. An indirect discrete-time adaptive scheme was proposed for a flexible-link robot was tested in experiment in (Rokui and Khorasani 2000) and more recently simulation results appear in (Tian et al. 2004). Non-adaptive digital control of a simulated flexible-joint robot was examined in (Ider and Zgren 2000).

Most neural-adaptive approaches use a Lyapunov-based design for controls and weight updates, straightforward where nonlinearities are matched by control signals. When the nonlinearities are not matched, Lyapunov backstepping is appropriate for systems in strict-feedback form. However, limitations to treating neural-networks in identical fashion to LP models include the following:

1. offering no advantages over adaptive control if the nonlinear model is LP,
2. requiring on-line integration of weight update laws, and
3. violating stability-proof assumptions when the neural network output is calculated at a relatively slow digital frequency.

In general, some of the unique and interesting features of neural networks are ignored or under-utilized when weights are treated as if they were adaptive parameters in LP models. Some notable differences between neural-networks and LP models, that may be exploited, are as follows:

1. a neural network can approximate a discrete-time inverse-dynamics mapping and
2. the greater the number of basis functions (and weights) the more accurate the approximation capability.

The present work stems from that in (Macnab and D'Eleuterio 2000), which used the above properties to reveal the inherent difference between LP adaptive and neural-adaptive approaches. Here we develop the methodology to the point where actual experiments with a flexible-joint robot are shown to be successful.

In LP adaptive control a parameter update law appears as a time derivative, and numerical integration provides the adaptive parameter estimates. If one applies a delta-rule (rectangular integration) at a relatively slow frequency instead of a more advanced numerical integration routine, very poor performance or instability is the result. Moreover, flexible-joint robots require a computationally complex LP-adaptive design, requiring a very slow frequency unsuitable for numerical integration. In contrast, we show here that updating neural network weights with a delta-rule, at a relatively slow frequency, is completely appropriate. Moreover, by slowing the update frequency (to a point) and using that extra computational time to add basis functions to the network, robot performance actually improves. In this paper we show the reason for this effect; with a delta-rule update the network approximates terms in a discrete-time model of the system, which is more accurate when more basis functions are added.

The main advantage to using continuous time control designs for robotic manipulators has been the ease of making the control adaptive. In this work, we maintain a continuous-time backstepping design of Kwan and Lewis (2000) only for the error-feedback (proportional-derivative) control signals. Only the neural networks appear as discrete-time signals. Practically speaking, the error-feedback signals are simple and easily applied at a rate several orders of magnitude above the highest natural frequency of interest. Using a slow digital frequency for the neural network allows more computational time and thus more basis functions in the neural network. We show the increased approximation abilities of the network results in significant gains in performance. Discrete-time Lyapunov functions establish the stability properties.

2 Background

In this section we consider the traditional design of a direct neural-adaptive control for a rigid robotic manipulator in

continuous time. The dynamics of an n-linked rigid robot manipulator are

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{u}, \tag{1}$$

where $\boldsymbol{\theta} \in \Re^n$ contains link angles, $\mathbf{M}(\boldsymbol{\theta}) \in \Re^{n \times n}$ is the positive definite inertia matrix, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \Re^n$ includes friction, centripetal and Coriolis terms, and $\mathbf{u} \in \Re^n$ contains control torques. Given desired trajectory $\boldsymbol{\theta}_d(t), \dot{\boldsymbol{\theta}}_d(t), \ddot{\boldsymbol{\theta}}_d(t)$ and using the auxiliary error

$$\mathbf{z} = \Lambda(\boldsymbol{\theta} - \boldsymbol{\theta}_d) + (\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{\theta}})_d = \Lambda \mathbf{e}_1 + \mathbf{e}_2, \tag{2}$$

with positive constant Λ results in error dynamics

$$\mathbf{M}\dot{\mathbf{z}} = \mathbf{M}\Lambda\mathbf{e}_2 - \mathbf{C} - \mathbf{M}\ddot{\boldsymbol{\theta}}_d + \mathbf{u}. \tag{3}$$

Consider a control containing neural network outputs and linear feedback

$$\mathbf{u} = -\phi(\mathbf{q})\hat{\mathbf{w}} - \mathbf{G}\mathbf{z}, \tag{4}$$

with positive gains in $\mathbf{G}$, matrix of m basis functions $\phi \in \mathcal{R}^{n \times m}$, and m weights in $\hat{\mathbf{w}} \in \mathcal{R}^m$. In this paper $\mathbf{G}$ contains positive control gains on the diagonal, and zeros off the diagonal. The network requires inputs $\mathbf{q} = [\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d]^T$. The error dynamics become

$$\mathbf{M}\dot{\mathbf{z}} = -\mathbf{G}\mathbf{z} + (\mathbf{M}\Lambda\mathbf{e}_2 - \mathbf{C} - \mathbf{M}\ddot{\boldsymbol{\theta}}_d) - \phi(\mathbf{q})\hat{\mathbf{w}}. \tag{5}$$

The error variable $\tilde{\mathbf{w}} = \mathbf{w} - \hat{\mathbf{w}}$ denotes the difference between (unknown) ideal weights $\mathbf{w}$ and the weight estimates (actual weights) $\hat{\mathbf{w}}$. Consider the Lyapunov function candidate

$$V = \frac{1}{2}\mathbf{z}^T\mathbf{M}\mathbf{z} + \frac{1}{2\gamma}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} \tag{6}$$

where γ is a positive adaptation gain. Taking the time-derivative gives

$$\dot{V} = \mathbf{z}^T(\mathbf{M}\Lambda\mathbf{e}_2 - \mathbf{C} - \mathbf{M}\ddot{\boldsymbol{\theta}}_d + \frac{d}{dt}(\mathbf{M}^{-1})\mathbf{z}/2 - \phi(\mathbf{q})\hat{\mathbf{w}} - \mathbf{G}\mathbf{z}) - \tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}}/\gamma. \tag{7}$$

$$\dot{V} = \mathbf{z}^T(\mathbf{f}(\mathbf{q}) - \phi(\mathbf{q})\hat{\mathbf{w}} - \mathbf{G}\mathbf{z}) - \tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}}/\gamma \tag{8}$$

An ideal neural network $\boldsymbol{\phi}(\mathbf{q})\mathbf{w}$ can uniformly approximate the nonlinear function $\mathbf{f}(\mathbf{q})$ so that

$$\mathbf{f}(\mathbf{q}) = \boldsymbol{\phi}(\mathbf{q})\mathbf{w} + \mathbf{d}(\mathbf{q}), \tag{9}$$

where approximation error magnitude $\|\mathbf{d}\|$ is bounded by a constant $d_{\max}$ in the region of approximation $\mathcal{D}$ (i.e. $\|\mathbf{d}\| \le d_{\max} \forall \mathbf{q} \in \mathcal{D}$). The time-derivative becomes

$$\dot{V} = \mathbf{z}^T[\boldsymbol{\phi}(\mathbf{q})\tilde{\mathbf{w}} + \mathbf{d} - \mathbf{G}\mathbf{z}] - \tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}}/\gamma \tag{10}$$

$$\dot{V} = \mathbf{z}^T(\mathbf{d} - \mathbf{G}\mathbf{z}) + \tilde{\mathbf{w}}^T(\boldsymbol{\phi}^T(\mathbf{q})\mathbf{z} - \dot{\hat{\mathbf{w}}}/\gamma). \tag{11}$$

Applying robust [σ-modification leakage (Ioannuou and Kokotovic 1984)] weight updates

$$\dot{\hat{\mathbf{w}}} = \gamma(\boldsymbol{\phi}^T(\mathbf{q})\mathbf{z} - v\hat{\mathbf{w}}), \tag{12}$$

allow one to bound the time-derivative

$$\dot{V} \le s(\|\mathbf{z}\|, \|\tilde{\mathbf{w}}\|), \tag{13}$$

where

$$s(\|\mathbf{z}\|, \|\tilde{\mathbf{w}}\|) = -G_{\min}\|\mathbf{z}\|^2 + d_{\max}\|\mathbf{z}\| + v\|\mathbf{w}\|\|\tilde{\mathbf{w}}\| - v\|\tilde{\mathbf{w}}\|^2, \tag{14}$$

where $G_{\min}$ is the minimum eigenvalue of symmetric $\mathbf{G}$ (Fig. 1). Thus $\dot{V} < 0$ outside a compact set on the $(\|\mathbf{z}\|, \|\tilde{\mathbf{w}}\|)$ plane defined by ellipse $s(\|\mathbf{z}\|, \|\tilde{\mathbf{w}}\|) = 0$. This guarantees that $\|\mathbf{z}\|$ and $\|\tilde{\mathbf{w}}\|$ are uniformly ultimately bounded (UUB). In particular $\dot{V} < 0$ when $\|\mathbf{z}\| > \delta_z$ or $\|\tilde{\mathbf{w}}\| > \delta_w$ where

$$\delta_z = \frac{d_{\max}}{2G_{\min}} + \sqrt{\frac{d_{\max}^2}{4G_{\min}^2} + \frac{v\|\mathbf{w}\|^2}{4G_{\min}}}, \tag{15}$$

$$\delta_w = \frac{\|\mathbf{w}\|}{2} + \sqrt{\frac{d_{\max}^2}{4vG_{\min}} + \frac{\|\mathbf{w}\|^2}{4}}. \tag{16}$$

Defining

$$V_b(\|\mathbf{z}\|, \|\tilde{\mathbf{w}}\|) = M_{\min}\|\mathbf{z}\|^2/2 + \|\tilde{\mathbf{w}}\|^2/(2\gamma),$$

where $M_{\min}$ is the smallest eigenvalue of $\mathbf{M}(\boldsymbol{\theta})$ over all values of $\boldsymbol{\theta}$, then the uniform ultimate bound on the error comes from solving for z_b in

$$V_b(\|\mathbf{z}\| = z_b, \|\tilde{\mathbf{w}}\| = 0) = V_b(\|\mathbf{z}\| = \delta_z, \|\tilde{\mathbf{w}}\| = \delta_w), \tag{17}$$

$$\frac{M_{\min}}{2}z_b^2 = \frac{M_{\min}}{2}\delta_z^2 + \frac{\delta_w^2}{2\gamma}. \tag{18}$$

The resulting uniform ultimate bound on $\|\mathbf{z}\|$ is

$$z_b = \sqrt{\delta_z^2 + \frac{\delta_w^2}{M_{\min}\gamma}}. \tag{19}$$

3 Proposed method

3.1 Discrete-time model

When implementing the control with a serial digital processor, the feedback part of the control law $-\mathbf{G}\mathbf{z}$ can run at a high frequency (typically in KHz or even MHz on a PC), but the neural network will take significantly more computational time. With our commercial control software, the additional custom neural network runs slower than 100 Hz, although efficient specially written software on contemporary multi-core processors are capable of much faster rates. We propose treating the feedback as a continuous

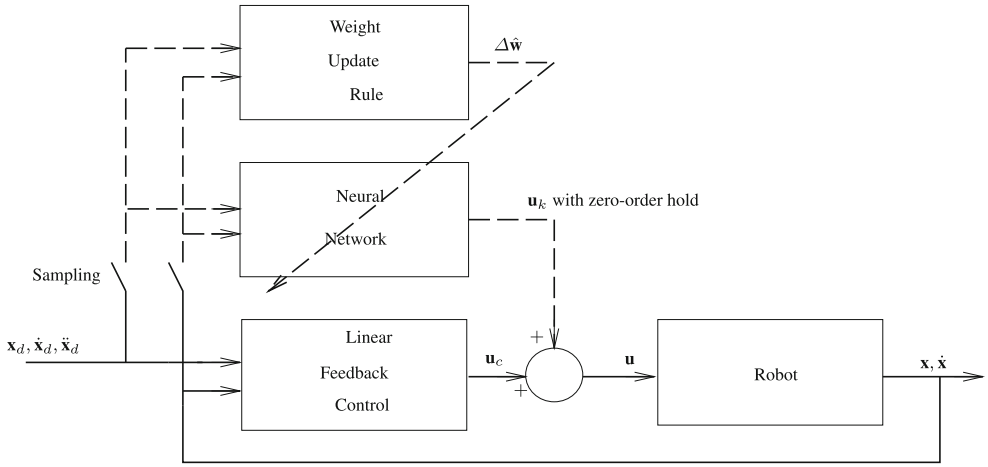

Fig. 1 Block diagram for multirate neural-adaptive control of a rigid robot

time signal in the control design, but the neural network as a discrete time signal

$$\mathbf{u}(t) = -\underbrace{\boldsymbol{\phi}(\mathbf{q}(kT))\hat{\mathbf{w}}(kT)}_{\text{discretetime}} - \underbrace{\mathbf{G}\mathbf{z}(t)}_{\text{continuoustime}}, \tag{20}$$

$$\mathbf{u} = \mathbf{u}_k - \mathbf{G}\mathbf{z}, \tag{21}$$

where k denotes the discrete sampling interval and$T$ is the period of the sampling interval. In this paper subscript k will mean at time kT e.g. $\mathbf{u}_k = \mathbf{u}(kT)$. The error dynamics become

$$\dot{\mathbf{z}} = \mathbf{A}(\boldsymbol{\theta})\mathbf{z} + \mathbf{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d) + \mathbf{B}(\boldsymbol{\theta})\mathbf{u}_k, \tag{22}$$

where

$$\mathbf{A} = -\mathbf{M}^{-1}\mathbf{G}, \qquad \mathbf{H} = \Lambda\mathbf{e}_2 - \mathbf{M}^{-1}\mathbf{C} - \ddot{\boldsymbol{\theta}}_d,$$
$$\mathbf{B} = \mathbf{M}^{-1}.$$

Assuming the resulting discrete-time signals follow from a sampler activated by a periodic discrete-time component, followed by zero-order hold, then the discrete-time transformation (see Appendix) results in difference equation

$$\mathbf{z}_{k+1} = \mathbf{a}_k\mathbf{z}_k + \mathbf{b}_k\mathbf{u}_k + \int_{\tau=kT}^{(k+1)T} [\mathbf{y}_1(\tau) + \mathbf{y}_2(\tau)\mathbf{u}_k]d\tau, \tag{23}$$

where the $\mathbf{a}_k$, $\mathbf{b}_k$ terms are the well-known linear discrete-time transformations

$$\mathbf{a}_k = e^{\mathbf{A}_k T}, \qquad \mathbf{b}_k = \int_{\tau=kT}^{(k+1)T} e^{\mathbf{A}_k((k+1)T-\tau)}d\tau\mathbf{B}_k, \tag{24}$$

and nonlinear terms [also noted in (Kanellakopoulos 1994; Rokui and Khorasani 1997; Yeh and Kokotovic 1995)] are derived in the Appendix

$$\mathbf{y}_1(t) = e^{\mathbf{A}_k((k+1)T-t)}(\Delta\mathbf{A}(t)\mathbf{z} + \mathbf{H}(t)),$$
$$\mathbf{y}_2(t) = e^{\mathbf{A}_k((k+1)T-t)}\Delta\mathbf{B}(t),$$

where $\Delta\mathbf{A}(t) = \mathbf{A}(t) - \mathbf{A}_k$ and $\Delta\mathbf{B}(t) = \mathbf{B}(t) - \mathbf{B}_k$ for $t \geq kT$. Note that $\mathbf{a}_k$ has positive eigenvalues less than one, and $\mathbf{b}_k$ has positive eigenvalues (both due to positive definite $\mathbf{M}$ and $\mathbf{G}$ in the continuous-time system).

Expressing $\mathbf{y}_1$ and $\mathbf{y}_2$ using Taylor series and using a trapezoidal approximation of the integral (noting that $y_1(kT) = e^{TA_k}\ \mathbf{H}(kT)$ and $\mathbf{y}_2(kT) = 0$ and $\Delta\mathbf{B}(kT) = 0$) as in (Macnab and D'Eleuterio 2000) results in

$$\mathbf{z}_{k+1} = \mathbf{a}_k\mathbf{z}_k + \mathbf{b}_k\mathbf{u}_k + T\mathbf{y}_{1,k} + \frac{T^2}{2}\frac{\partial}{\partial t}\mathbf{y}_1(t)\Big|_{t=kT} + \frac{T^2}{2}\frac{\partial}{\partial t}\mathbf{y}_2(t)\Big|_{t=kT}\mathbf{u}_k + \mathcal{O}(T^3), \tag{25}$$

$$\mathbf{z}_{k+1} = \mathbf{a}_k\mathbf{z}_k + \mathbf{b}_k\mathbf{u}_k + \mathbf{F}_0(\mathbf{q}) + \mathbf{F}_1(\mathbf{q}) + \mathbf{F}_2(\mathbf{q})\mathbf{u}_k + \mathcal{O}(T^3), \tag{26}$$

where

$$\mathbf{F}_0(\mathbf{q}) = Te^{T\mathbf{A}_k}\mathbf{H}_k, \qquad \mathbf{F}_1(\mathbf{q}) = T\mathbf{y}_{1,k} + \frac{T^2}{2}\frac{\partial}{\partial t}\mathbf{y}_1(t)\Big|_{t=kT},$$
$$\mathbf{F}_2(\mathbf{q}) = \frac{T^2}{2}\frac{\partial}{\partial t}\mathbf{y}_2(t)\Big|_{t=kT}\mathbf{u}_k$$

and $\mathcal{O}(T^3)$ consists of third order terms (multiplied by T^3) and higher. For adaptive control design purposes, it will be easier to use

$$\mathbf{m}_k = \mathbf{b}_k^{-1}, \quad \text{and} \quad \mathbf{g}_k = \mathbf{b}_k^{-1}\mathbf{a}_k. \tag{27}$$

Note that $\mathbf{m}_k$ has positive eigenvalues. The discrete-time error dynamics become

$$\mathbf{m}_k\mathbf{z}_{k+1} = \mathbf{g}_k\mathbf{z}_k + \mathbf{u}_k + \mathbf{m}_k(\mathbf{F}_0 + \mathbf{F}_1) + \mathbf{m}_k\mathbf{F}_2\mathbf{u}_k + \mathcal{O}(T^3). \tag{28}$$

Assuming the neural network sampling interval T is (chosen) small enough means that third-order terms can be ignored. The remaining second-order terms in (28) are multiplied by a first-order term in the control, and thus become third order as well. The network needs to approximate remaining terms

$$\boldsymbol{\phi}_k(\mathbf{q})\mathbf{w} + \mathbf{d}_k(\mathbf{q}) = \mathbf{m}_k(\mathbf{F}_0(\mathbf{q}) + \mathbf{F}_1(\mathbf{q})), \tag{29}$$

where $\mathbf{d}_k$ is the bounded approximation error in input region $\mathcal{D}$. The discrete part of the control consists of neural network output

$$\mathbf{u}_k = -\boldsymbol{\phi}_k\hat{\mathbf{w}}_k. \tag{30}$$

The neural network approximates a term of order T in (29); thus the term multiplied by $\mathbf{u}_k$ in (28) should now become order T^3 so that

$$\mathbf{m}_k\mathbf{z}_{k+1} = \mathbf{g}_k\mathbf{z}_k + \boldsymbol{\phi}_k\tilde{\mathbf{w}}_k + \mathbf{d}_k + \mathcal{O}(T^3). \tag{31}$$

We are now ready to prove the stability of the system and design the discrete weight update for the network using the Lyapunov approach.

3.2 Discrete weight update

Consider the positive definite discrete-time Lyapunov candidate

$$V_k = \mathbf{z}_k^T\mathbf{m}_{k-1}^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_{k-1}\mathbf{z}_k + \frac{1}{2\gamma}\tilde{\mathbf{w}}_k^T\tilde{\mathbf{w}}_k, \tag{32}$$

where γ is a positive constant adaptation gain. For clarity of presentation, first let us consider the case when $\mathbf{d}_k = 0$ (the final result in this section includes $\mathbf{d}_k$). The design ignores higher order terms in (31) since they are of order T^3. Utilizing (31) and writing $\tilde{\mathbf{w}}_{k+1} = \tilde{\mathbf{w}}_k + \Delta\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_k - \Delta\hat{\mathbf{w}}$, the time difference equation is

$$\begin{aligned}\Delta V &= V_{k+1} - V_k \\ &= (\mathbf{z}_k^T\mathbf{g}_k^T + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T)\mathbf{g}_k^{-1}(\mathbf{g}_k\mathbf{z}_k + \boldsymbol{\phi}_k\tilde{\mathbf{w}}_k) \\ &\quad + \frac{1}{\gamma}(\tilde{\mathbf{w}}_k^T - \Delta\hat{\mathbf{w}}^T)(\tilde{\mathbf{w}}_k - \Delta\hat{\mathbf{w}}) - \mathbf{z}_k^T\mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k\mathbf{z}_k - \frac{1}{\gamma}\tilde{\mathbf{w}}_k^T\tilde{\mathbf{w}}_k\end{aligned} \tag{33}$$

$$\begin{aligned}\Delta V = \mathbf{z}_k^T(\mathbf{g}_k^T - \mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k)\mathbf{z}_k + \mathbf{z}_k^T\mathbf{g}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{z}_k \\ + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k + \frac{1}{\gamma}\left(-2\tilde{\mathbf{w}}_k^T\Delta\hat{\mathbf{w}} + \Delta\hat{\mathbf{w}}^T\Delta\hat{\mathbf{w}}\right).\end{aligned} \tag{34}$$

A symmetric $\mathbf{G}$ ensures a symmetric $\mathbf{g}_k$ (since inertia matrix $\mathbf{M}$ is also symmetric) implying $\mathbf{g}_k^T\,\mathbf{g}_k^{-1} = \mathbf{I}$. Then

$$\begin{aligned}\Delta V = \mathbf{z}_k^T(\mathbf{g}_k^T - \mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k)\mathbf{z}_k + 2\tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{z}_k \\ + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k + \frac{1}{\gamma}\left(-2\tilde{\mathbf{w}}_k^T\Delta\hat{\mathbf{w}} + \Delta\hat{\mathbf{w}}^T\Delta\hat{\mathbf{w}}\right).\end{aligned} \tag{35}$$

Thus, choosing the discrete weight update

$$\Delta\hat{\mathbf{w}} = \gamma(\boldsymbol{\phi}_k^T\mathbf{z}_k - v\hat{\mathbf{w}}_k), \tag{36}$$

results in

$$\begin{aligned}\Delta V = \mathbf{z}_k^T(\mathbf{g}_k^T - \mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k)\mathbf{z}_k + 2\tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{z}_k + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k \\ + \frac{1}{\gamma}\left(-2\gamma\tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{z}_k + 2\gamma v\tilde{\mathbf{w}}_k^T\hat{\mathbf{w}}_k + \gamma^2(\mathbf{z}_k^T\boldsymbol{\phi}_k - v\hat{\mathbf{w}}_k^T)\right. \\ \left.\times\left(\boldsymbol{\phi}_k^T\mathbf{z}_k - v\hat{\mathbf{w}}_k\right)\right).\end{aligned} \tag{37}$$

with the $2\tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{z}_k$ terms cancelling to give

$$\begin{aligned}\Delta V = \mathbf{z}_k^T(\mathbf{g}_k^T - \mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k)\mathbf{z}_k + \tilde{\mathbf{w}}_k^T\boldsymbol{\phi}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k \\ + \frac{1}{\gamma}\left(2\gamma v\tilde{\mathbf{w}}_k^T\hat{\mathbf{w}}_k + \gamma^2(\mathbf{z}_k^T\boldsymbol{\phi}_k - v\hat{\mathbf{w}}_k^T)(\boldsymbol{\phi}_k^T\mathbf{z}_k - v\hat{\mathbf{w}}_k)\right).\end{aligned} \tag{38}$$

Substituting $\hat{\mathbf{w}}_k = \mathbf{w} - \tilde{\mathbf{w}}_k$ gives

$$\begin{aligned}\Delta V = \mathbf{z}_k^T(\mathbf{g}_k^T - \mathbf{m}_k^T\mathbf{g}_{k-1}^{-1}\mathbf{m}_k + \gamma\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T)\mathbf{z}_k \\ + \tilde{\mathbf{w}}_k^T\left((-2v + \gamma v^2)\mathbf{I} + \boldsymbol{\phi}_k^T\mathbf{g}_k^{-1}\boldsymbol{\phi}_k\right)\tilde{\mathbf{w}}_k \\ + 2v\gamma\mathbf{z}_k^T\boldsymbol{\phi}_k\tilde{\mathbf{w}}_k + (2v - \gamma v^2)\mathbf{w}^T\tilde{\mathbf{w}}_k \\ - 2\gamma v\boldsymbol{\phi}_k^T\mathbf{w}_k^T\mathbf{z}_k + \gamma v^2\mathbf{w}^T\mathbf{w}.\end{aligned} \tag{39}$$

The time difference can be bounded by

$$\Delta V \leq -\mathbf{Z}^T\mathbf{L}\mathbf{Z} + \mathbf{R}^T\mathbf{Z} + D, \tag{40}$$

where

$$\mathbf{Z} = \begin{bmatrix}\|\mathbf{z}_k\| \\ \|\tilde{\mathbf{w}}_k\|\end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix}2v\gamma\|\mathbf{w}\| \\ (v - v^2\gamma)\|\mathbf{w}\|\end{bmatrix},$$
$$D = \gamma v^2\|\mathbf{w}\|^2,$$

and

$$\mathbf{L} = -\begin{bmatrix}\lambda_{\max}(\mathbf{g}_k - \mathbf{m}_k^T\mathbf{g}_k^{-1}\mathbf{m}_k) + \gamma\|\boldsymbol{\phi}\|^2 & -v\gamma \\ -v\gamma & -2v + v^2\gamma + \lambda_{\max}(\mathbf{g}_k^{-1})\|\boldsymbol{\phi}\|^2\end{bmatrix}. \tag{41}$$

The condition for uniformly ultimately bounded signals is that eigenvalues of $\mathbf{L}$ must be positive

$$L_{\min} = \inf_{\boldsymbol{\theta}}(\lambda(\mathbf{L})) > 0. \tag{42}$$

Rewriting the bound (40) as

$$\Delta V \leq s_d(\mathbf{Z}), \tag{43}$$

with

$$s_d(\mathbf{Z}) = -L_{\min}\left(\|\mathbf{Z}\| - \frac{\|\mathbf{R}\|}{2L_{\min}}\right)^2 + \frac{\|\mathbf{R}\|^2}{4L_{\min}} + D. \tag{44}$$

leads us to conclude $\dot{V}<0$ when $\|\mathbf{Z}\| > \delta_Z$ where

$$\delta_Z = \frac{\|\mathbf{R}\|}{2L_{\min}} + \sqrt{\frac{\|\mathbf{R}\|^2}{4L_{\min}^2} + \frac{D}{L_{\min}}}. \tag{45}$$

Thus δ_Z constitutes a uniform ultimate bound on $\|\mathbf{Z}\|$. Consider

$$V_d = \mathbf{Z}^T \mathbf{P}_{\min} \mathbf{Z}, \tag{46}$$

where $\mathbf{P}_{\min}$ contains smallest possible eigenvalues of $\mathbf{m}_{k-1}^T \mathbf{g}_{k-1}^{-1} \mathbf{m}_{k-1}$ and $1/\gamma$

$$\mathbf{P}_{\min} = \begin{bmatrix} \inf_\theta \lambda(\mathbf{m}_{k-1}^T \mathbf{g}_{k-1}^{-1} \mathbf{m}_{k-1}) & 0 \\ 0 & 1/\gamma \end{bmatrix} = \begin{bmatrix} P_{1,\min} & 0 \\ 0 & 1/\gamma \end{bmatrix}. \tag{47}$$

Then a bound on the state error, when it is larger than δ_Z, follows from solving for z_b using

$$V_d(\|\mathbf{z}_k\| = z_b, \|\tilde{\mathbf{w}}_k\| = 0) = V_d(\|\mathbf{z}_k\| = 0, \|\tilde{\mathbf{w}}_k\| = \delta_Z), \tag{48}$$

$$P_{1,\min} z_b^2 = \delta_Z^2/\gamma, \tag{49}$$

$$z_b = \delta_Z / \sqrt{P_{1,\min}\gamma}. \tag{50}$$

Note that if $\delta_Z > z_b$ then δ_Z would become the bound on $\|\mathbf{z}_k\|$.

3.3 Robustness of weight update

Taking into account neural network approximation error, such that $\mathbf{d}_k \neq 0$, $\mathbf{L}$ remains the same, but now

$$\mathbf{R} = \begin{bmatrix} 2v\gamma\|\mathbf{w}\| + 2d_{\max} \\ (v - v^2\gamma)\|\mathbf{w}\| + 2\lambda_{\max}(\mathbf{g}_k^{-1}) d_{\max} \end{bmatrix},$$

$$D = \gamma v^2 \|\mathbf{w}\|^2 + d_{\max}^2 \lambda_{\max}(\mathbf{g}_k^{-1}).$$

Thus, the conditions for uniform ultimate bounded signals do not change; no redesign of parameters is necessary to ensure stability. The size of the bound does increase.

3.4 Extension to flexible-joint robots

For a flexible-joint robot, angles $\boldsymbol{\theta}$ are link angles and the actual rotor angles (after gear reduction) are denoted $\boldsymbol{\phi}$. The equations of motion for the flexible-joint robot are then (Spong and Vidyasagar 1989)

$$\mathbf{M}\ddot{\boldsymbol{\theta}} = \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{K}(\boldsymbol{\phi} - \boldsymbol{\theta}) \tag{51}$$

$$\mathbf{J}\ddot{\boldsymbol{\phi}} = -\mathbf{D}(\dot{\boldsymbol{\phi}}) + \mathbf{K}(\boldsymbol{\theta} - \boldsymbol{\phi}) + \mathbf{u} \tag{52}$$

where $\mathbf{K}$ is a diagonal matrix of joint stiffnesses, $\mathbf{J}$ is the inertia matrix of the rotors (after gear reduction) and $\mathbf{D}$ contains friction coefficients. Writing this in strict-feedback form suitable for backstepping, with $\boldsymbol{\mu}_1$ a desired value for $\boldsymbol{\phi}$, with $\boldsymbol{\mu}_2$ a desired value for $\dot{\boldsymbol{\phi}}$, and $\mathbf{z}_2 = \boldsymbol{\phi} - \boldsymbol{\mu}_1$, $\mathbf{z}_3 = \dot{\boldsymbol{\phi}} - \boldsymbol{\mu}_2$ results in

$$\mathbf{K}^{-1}\mathbf{M}\dot{\mathbf{z}}_1 = \mathbf{R}_1(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d) + \boldsymbol{\mu}_1 + \mathbf{z}_2 - \boldsymbol{\theta} \tag{53}$$

$$\dot{\mathbf{z}}_2 = \boldsymbol{\mu}_2 + \mathbf{z}_3 - \dot{\boldsymbol{\mu}}_1(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d, \boldsymbol{\theta}_d^{(3)}, \boldsymbol{\phi}) - \dot{\boldsymbol{\theta}} \tag{54}$$

$$\mathbf{J}\dot{\mathbf{z}}_3 = \mathbf{R}_2(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{u} - \mathbf{J}\dot{\boldsymbol{\mu}}_2(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d, , \boldsymbol{\theta}_d^{(3)}, \boldsymbol{\theta}_d^{(4)}, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}) \tag{55}$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ collect linear and nonlinear terms. Analogous to the rigid case, the virtual controls and control will have both continuous-time linear and digital network components. Choosing (virtual) controls

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_{1,k} + \boldsymbol{\theta} - \mathbf{G}_1\mathbf{z}_1 \tag{56}$$

$$\boldsymbol{\mu}_2 = \boldsymbol{\mu}_{2,k} + \dot{\boldsymbol{\theta}} - \mathbf{z}_1 - \mathbf{G}_2\mathbf{z}_2 \tag{57}$$

$$\mathbf{u} = \mathbf{u}_k - \mathbf{z}_2 - \mathbf{G}_3\mathbf{z}_3 \tag{58}$$

results in error dynamics

$$\mathcal{M}\dot{\boldsymbol{\zeta}} = -\mathcal{G}\boldsymbol{\zeta} + \mathcal{F}(\mathbf{q}_2) + \boldsymbol{\mu}_k \tag{59}$$

where $\mathbf{q}_2 = [\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d, \boldsymbol{\theta}_d^{(3)}, \boldsymbol{\theta}_d^{(4)}]^T$ and

$$\boldsymbol{\zeta} = \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\phi} - \boldsymbol{\mu}_1 \\ \dot{\boldsymbol{\phi}} - \boldsymbol{\mu}_2 \end{bmatrix} \qquad \boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_{1,k} \\ \boldsymbol{\mu}_{2,k} \\ \mathbf{u}_k \end{bmatrix}$$

$$\mathcal{M} = \begin{bmatrix} \mathbf{K}^{-1}\mathbf{M}(\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J} \end{bmatrix} \qquad \mathcal{G} = \begin{bmatrix} \mathbf{G} & -\mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{G} & -\mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{G} \end{bmatrix}$$

and

$$\mathcal{F}(\mathbf{q}_2) = [\mathbf{R}_1 - \dot{\mu}_1 \quad \mathbf{R}_2 - \mathbf{J}\dot{\mu}_2]^T \tag{60}$$

By seeing that the form of (59) is analogous to (3) one could calculate a new $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{m}_k$ and $\mathbf{g}_k$. This calculation would use $\mathcal{M}$ and $\mathcal{G}$ instead of $\mathbf{M}$ and $\mathbf{G}$, respectively, as well as $\mathcal{F}$ instead of nonlinear terms $\mathbf{M}^{-1}\mathbf{H}$. The analogy works because $\mathcal{M}$ and $\mathcal{G}$ are both positive definite. However, neither $\mathcal{M}$ nor $\mathcal{G}$ are symmetric; therefore, a new $\mathbf{g}_k$ would not be symmetric. To regain a symmetric $\mathbf{g}_k$ needed in the stability analysis consider that

$$\mathcal{M}\dot{\boldsymbol{\zeta}} = -\boldsymbol{\Gamma}\boldsymbol{\zeta} + 0.5(\mathcal{G}^T - \mathcal{G})\boldsymbol{\zeta} + \mathcal{F}(\mathbf{q}_{2,k}) + \boldsymbol{\mu}_k \tag{61}$$

where $\boldsymbol{\Gamma} = 0.5(\mathcal{G} + \mathcal{G}^T)$ is a diagonal matrix. Then

$$\dot{\boldsymbol{\zeta}} = \mathcal{A}\boldsymbol{\zeta} + \mathcal{M}^{-1}[0.5(\mathcal{G}^T - \mathcal{G})\boldsymbol{\zeta} + \mathcal{F}(\mathbf{q}_{2,k})] + \mathcal{B}\boldsymbol{\mu}_k \tag{62}$$

where $\mathcal{A} = -\mathcal{M}^{-1}\boldsymbol{\Gamma}$ and $\mathcal{B} = \mathcal{M}^{-1}$. Rewriting this as

$$\dot{\boldsymbol{\zeta}} = \mathcal{A}\boldsymbol{\zeta} + \mathcal{H} + \mathcal{B}\boldsymbol{\mu}_k \tag{63}$$

where $\mathcal{H}$ collects linear and nonlinear terms, puts the dynamics in the same form as (5). However, $\mathcal{A}$ and $\mathcal{B}$ will only be symmetric if joint stiffness are identical in

$\mathbf{K}$, ensuring $\mathbf{K}^{-1}\,\mathbf{M}$ and $\mathcal{M}$ are symmetric. Thus, we must deal with this equation with two cases.

3.4.1 Case 1: identical joint stiffnesses

Assuming identical joint stiffnesses makes $\mathbf{K}^{-1}\mathbf{M}$ symmetric leading to symmetric matrices $\mathcal{M}$, $\mathcal{A}$ and $\mathcal{B}$. Then discrete time matrices $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{m}_k$, $\mathbf{g}_k$ follow from (24) and (27) using $\mathcal{A}, \boldsymbol{\zeta}, \mathcal{H}, \mathcal{B}, \boldsymbol{\mu}_k$ instead of $\mathbf{A}$, $\mathbf{z}$, $\mathbf{H}$, $\mathbf{B}$, $\mathbf{u}_k$, respectively. The resulting discrete-time representation can be expressed like (25) as

$$\begin{aligned}\mathbf{m}_k\boldsymbol{\zeta}_{k+1} = \mathbf{g}_k\boldsymbol{\zeta}_k + \mathbf{u}_k + \mathbf{m}_k[\mathbf{F}_0(T, \mathbf{x}_k, \boldsymbol{\zeta}_k) + \mathbf{F}_1(T^2, \mathbf{x}_k, \boldsymbol{\zeta}_k) \\ + \mathbf{F}_2(T^2, \mathbf{x}, \boldsymbol{\zeta})\boldsymbol{\mu}_k] + \mathcal{O}(T^3),\end{aligned} \tag{64}$$

For the flexible-joint design, we will ignore terms of order T^2 rather than T^3. Even so, the neural networks in $\mu_{1,k}$ and $\mu_{2,k}$ cannot fully approximate the first two elements of $\mathbf{F}_0$ since virtual control must only rely on states and errors from previous backstepping steps, i.e we must have

$$\mu_1 = \mu_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z}) \tag{65}$$

$$\mu_2 = \mu_2(\mathbf{x}_2, \mathbf{x}_2, \mathbf{x}_3, \mathbf{z}, \mathbf{z}_2) \tag{66}$$

We must split $\mathbf{F}_0$ into parts

$$\mathbf{F}_0 = Te^{T\mathcal{A}(\theta_k)}\mathcal{H} \tag{67}$$

$$\mathbf{F}_0 = Te^{T\mathcal{A}(\theta_k}[\mathcal{M}^{-1}(\boldsymbol{\theta}_k)\mathcal{F}(\mathbf{q}_{2,k}) + \mathcal{M}^{-1}(\boldsymbol{\theta}_k)0.5(\mathcal{G}^T - \mathcal{G})\boldsymbol{\zeta}_k] \tag{68}$$

By replacing $\boldsymbol{\zeta}_k = \boldsymbol{\zeta}_{k-1} + \Delta\boldsymbol{\zeta}$ we can separate this expression into two constituent terms

$$\mathbf{F}_0 = T\mathbf{f}_{0,1}(\mathbf{q}_{2,k}, \boldsymbol{\zeta}_{k-1}) + T\mathbf{f}_{0,2}(\boldsymbol{\theta}_k, \Delta\boldsymbol{\zeta}) \tag{69}$$

Since $\Delta\boldsymbol{\zeta}$ is of order T the second term is of order T^2 and will be ignored. The first term can be estimated by the neural network, since $\boldsymbol{\zeta}_{k-1}$ is known (from a previous time step). Specifically, the three neural networks are provided with inputs

$$\boldsymbol{\phi}_1() = \boldsymbol{\phi}_1(\mathbf{q}_{2,k}, \mathbf{z}_k, \mathbf{z}_{2,k-1}, \mathbf{z}_{3,k-1}) \tag{70}$$

$$\boldsymbol{\phi}_2() = \boldsymbol{\phi}_2(\mathbf{q}_{2,k}, \mathbf{z}_k, \mathbf{z}_{2,k}, \mathbf{z}_{3,k-1}) \tag{71}$$

$$\boldsymbol{\phi}_3() = \boldsymbol{\phi}_3(\mathbf{q}_{2,k}, \mathbf{z}_k, \mathbf{z}_{2,k}, \mathbf{z}_{3,k}) \tag{72}$$

Then (28) for the flexible case becomes

$$\mathbf{m}_k\mathbf{z}_{k+1} = \mathbf{g}_k\mathbf{z}_k + \mathbf{u}_k + \boldsymbol{\phi}_k\mathbf{w} + \mathbf{d}_k + \mathcal{O}(T^2), \tag{73}$$

where the network $\boldsymbol{\phi}_k\mathbf{w}$ approximates $\mathbf{m}_k\ \mathbf{F}_0$ with approximation error $\mathbf{d}_k$. Thus, the flexible-joint design requires a faster frequency, one where terms of order T^2 can be ignored rather than just T^3 as in the rigid case. The analysis then follows in exactly the same way as (30) through (50).

3.4.2 Case 2: differing joint stiffnesses

For the case of different stiffnesses on the joints, the stiffnesses need to be known. The virtual control $\boldsymbol{\mu}_1$ must now be a desired value for $\mathbf{K}\boldsymbol{\phi}$ making

$$\mathbf{z}_2 = \mathbf{K}\boldsymbol{\phi} - \boldsymbol{\mu}_1 \tag{74}$$

and the first equation of the error dynamics (53) becomes

$$\mathbf{M}\dot{\mathbf{z}}_1 = \mathbf{R}_1(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d) + \boldsymbol{\mu}_1 + \mathbf{z}_2 - \mathbf{K}\boldsymbol{\theta} \tag{75}$$

Thus the continuous part of the first virtual control changes slightly to

$$\boldsymbol{\mu}_{1,c} = \mathbf{K}\boldsymbol{\theta} - \mathbf{G}_1\mathbf{z} \tag{76}$$

and then the top-left matrix in $\mathcal{M}$ becomes $\mathbf{M}$ instead of $\mathbf{K}^{-1}\,\mathbf{M}$. Since $\mathcal{M}$ is now symmetric as required, the rest of the stability analysis then follows in the same way as (61) through (73).

4 Results

4.1 Two-link planar flexible-joint robot

The experimental apparatus consists of a two-link planar flexible-joint robotic arm (Fig. 2). Harmonic drives have gear ratios of 100:1 and 80:1 for rotors 1 and 2, respectively. The control design ignores the harmonic drive dynamics, since their natural frequencies are more than an order of magnitude higher than the elastic (spring) joints. The manufacturer supplies some mechanical model parameters (Table 1) as well as WinCon interface/control software in a Matlab/Simulink-based environment.

External linear springs provide the flexibility in the joints, allowing large elastic deflections. The natural frequencies at $\theta_1 = \theta_2 = 0$ are 0.67 and 3.7 Hz (calculated from manufacturer data and confirmed by observation). In the experiments, the robot holds a 0.54 Kg payload during the experiments. The natural frequencies at $\theta_1 = \theta_2 = 0$ become 0.5 and 2.0 Hz, with the payload center-of-mass 17.5 cm from joint 2. In the experiment the desired trajectories are sinusoids in joint coordinates, with amplitudes of 23° and 11.5°, for joints one and two, respectively, each with period of 20 s. The experiment presents a challenging underactuated control problem.

These sinusoids provide achievable trajectories, implying the weight drift will not be severe. Thus, a relatively small leakage parameter ν will stop the drift, meaning it will not reduce performance significantly. Note this may not be the case for complex trajectories or when large external disturbances are present.

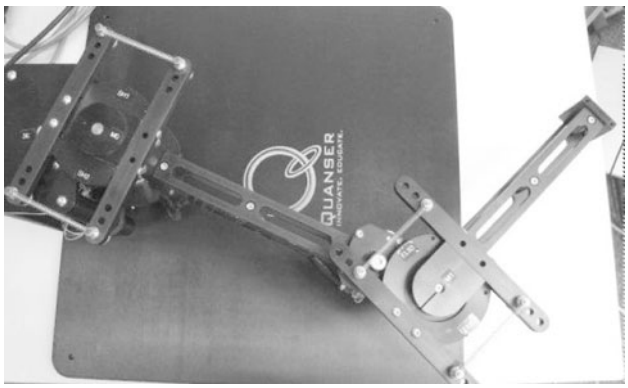

Fig. 2 Experimental two-link flexible-joint robot arm

Table 1 Flexible-joint robot parameters

	Link 1	Link 2
Length (m)	0.343	0.264
Center of mass (m)	0.159	0.055
Mass (kg)	1.51	0.87326
Inertia about center of mass (kg m^2)	39170.18e–6	8082.84e–6
	Joint 1	Joint 2
Equivalent motor inertia (kg m^2)	0.0111	0.0085
Stiffness (N m s)/rad	4.0	4.0

Four optical encoders measure the two rotor angles and the two link angles, i.e. the angles on each side of a flexible joint. The control software, running on a 3-GHz PC, samples the encoders at 2,000 Hz. Second-order filters, with cutoff frequency of 25 Hz, provide velocity estimates in the control software.

$$\frac{X_2(s)}{X_1(s)} = \frac{s\omega_f^2}{s^2 + s2\zeta_f\omega_f + \omega_f^2}$$

with $\omega_f = 25 \cdot \pi$ and $\zeta = 0.707$.

The supplied software from the manufacturer allows one to specify torque as a motor input, and appropriate voltage commands are automatically supplied to the motor. The actuators saturate quite easily; thus the provided control software truncates the control signal at ± 1 Nm of commanded torque. The state-feedback control that augments the neural network also produces control outputs to the motor at the 2,000-Hz frequency. For the purposes of the control design in this experiment the state-feedback is considered a continuous-time signal.

The neural network can only provide its outputs, and update its weights, at a much slower update due to the amount of computational overhead involved. The experiments use the discrete-time frequency of the neural network as the independent variable, testing the proposed multi-rate control formulation at both 100 and 50 Hz.

4.2 Tuned PID control

We compare the proposed method to a PID control of form

$$u_i = K_c\left(e_{1,i} + \frac{1}{T_i}\int e_{1,i}dt + T_d e_{2,i}\right), \tag{77}$$

where $i = 1, 2$ are the two links (i.e. a decentralized control). We tune the gains using Zieglar-Nichols methods: first using proportional term only with increasing gain K_u, then introducing an oscillation by creating a large initial spring deflection, and finally recording the value of K_u where a sustained or increasing oscillation is (just) observed and noting the period of oscillation P_u. The tuning is done for each joint separately. Values of K_c, T_i and T_d are read off the standard Zieglar-Nichols tuning tables for the given K_u and P_u, with results shown in Table 2. We first tune the PID control with no payload (although it is actually tested in experiment with the payload) and then create another PID control by tuning it with the 0.54 payload attached on the second link.

The PID control runs at 2,000 Hz, a frequency high enough above the physical natural frequencies (1 Hz) that the continuous-time design can be implemented without discrete-time modifications.

4.3 Estimating stability for proposed method

The matrices $\mathbf{A}_k, \mathcal{A}_k, \mathbf{a}_k, \boldsymbol{\alpha}_k, \mathbf{m}_K$ are all time-varying because of time-varying inertia matrix $\mathbf{M}(\theta_2)$. The smallest eigenvalues of $\mathbf{L}$ are found at $\theta_2 = 0$. We will show the calculations for one case, $\theta_2 = 0$, no payload, 50 Hz digital frequency, and put the other results in table form. In this first case the global inertia matrix using parameters in SI units supplied by the manufacturer is

$$\mathcal{M}(\theta_2 = 0, \text{no payload}) = \begin{bmatrix} 0.22 & 0.023 & 0 & 0 & 0 & 0 \\ 0.023 & 0.012 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.011 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0094 \end{bmatrix}, \tag{78}$$

Choosing diagonal unity feedback gains $\mathbf{G} = \mathbf{I}_{2x2}$ results in symmetric matrix $\boldsymbol{\Gamma} = 0.5\,(\mathbf{G} + \mathbf{G}^T) = \mathbf{I}_{6x6}$. Using $\mathcal{A} = -\mathcal{M}^{-1}\mathcal{G} = -\mathcal{M}^{-1}$ and $\mathbf{B} = \mathcal{M}^{-1}$ the first discrete-time linear transformation matrix at 50 Hz ($h = 1/50$) is

Table 2 PID gains

	$K_{u,1}$	$P_{u,1}$ (s)	$K_{c,1}$	$T_{i,1}$	$T_{d,i}$	$K_{u,2}$	$P_{u,2}$ (s)	$K_{c,2}$	$T_{i,2}$	$T_{d,2}$
PID tuned without payload	0.875	1.6	0.51	0.8	0.20	3	0.3	1.8	0.15	0.066
PID tuned with payload	0.875	2.2	0.51	1.1	0.28	3	4.2	1.8	0.21	0.053

$$\mathbf{a}_k(50\text{Hz}) = \begin{bmatrix} 0.58 & 0.073 & 0 & 0 & 0 & 0 \\ 0.073 & 0.0092 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.98 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.98 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.12 \end{bmatrix} \tag{79}$$

and the eigenvalues of $\mathbf{a}_k$ are 0, 0.59, 0.98, 0.98, 0.17, 0.12 (all within the unit circle) and the inverse of the second discrete-time linear transformation is

$$\mathbf{m}_k = \mathbf{b}_k^{-1} = \begin{bmatrix} 2.4142 & 0.1777 & 0 & 0 & 0 & 0 \\ 0.1777 & 1.0223 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50.5017 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50.5017 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1988 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.1367 \end{bmatrix} \tag{80}$$

with positive eigenvalues values 50, 50, 2.4, 1.2, 1.1, 1.0.

Calculating $\mathbf{g}_k = \mathbf{b}^{-1}\,\mathbf{a}_k$ gives the symmetric matrix

$$\mathbf{g}_k = \begin{bmatrix} 1.4142 & 0.1777 & 0 & 0 & 0 & 0 \\ 0.1777 & 0.0223 & 0 & 0 & 0 & 0 \\ 0 & 0 & 49.5017 & 0 & 0 & 0 \\ 0 & 0 & 0 & 49.5017 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1988 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1367 \end{bmatrix} \tag{81}$$

The parameters used for the neural network training in the experiments were $\gamma = 0.2$ and $\nu = 2$, with basis functions normalized such that $\|\boldsymbol{\phi}\| \leq 0.1$. Calculations for the minimum eigenvalue of $\mathbf{L}$ can then be done for different frequencies and payloads (Table 3). This confirms that gains $\gamma, \boldsymbol{\nu}$, normalization parameter for $\boldsymbol{\phi}$, and sampling period h can be chosen to ensure (42) holds for the experimental robot (and moreover becomes easier to satisfy the higher the sampling frequency). The experiments are done with a payload of 0.5 Kg and frequencies of 50 and 100 Hz, which fall within the stability range.

Table 3 $\lambda_{\min}(\mathbf{L})$ for different frequencies and payloads (must be greater than zero for stability)

Payload (Kg)	25 Hz	50 Hz	100 Hz	200 Hz
0	–3.2e7	–565	0.70	1.9
0.5	–78	1.7	1.9	1.9
1	–3.9	1.9	1.9	1.9

4.4 CMAC and controller parameters

The CMAC neural network performs the nonlinear approximation in this work, but the general method simply assumes a neural-adaptive backstepping design has been previously accomplished in continuous time which can also be accomplished with other types of neural networks like multi-layer perceptrons (Macnab 2007). The CMAC basis functions have hypercube domains, offering significantly easier implementation than other neural networks in real-time control systems (Albus 1975a, b). It avoids the curse-of-dimensionality (Kuo and Sloan 2005) found with radial-basis-function networks, allowing it to handle the 14 inputs (states and desired states) for the flexible-joint robot. It trains much faster than multilayer perceptron (backpropagation) networks, requiring many less repetitions of a trajectory for convergence (Miller III et al. 1990). Using a delta-rule update is especially advantageous in CMAC, as the local nature of the basis functions make them rather sharp and hard to integrate with accuracy.

Given n inputs, a CMAC consists of m layers of n-dimensional look-up tables. Each dimension of the look-up table is a normalized input, quantized into q discrete intervals. Thus, the CMAC input activates (indexes) one hypercube on each layer. In a binary CMAC, the sum of the weights associated with the activated hypercubes provides the output. Theoretically, there are a total of $m\,q^n$ hypercubes. Utilizing a hash-coding scheme allows storage of weights into a much smaller physical memory.

In this experiment the CMAC uses a physical memory with a size of 40,000 cells—allocated in the off-line memory as a 1D array. However, we only use at most $m = 400$ basis functions, which have to be calculated within sampling period h. Compare this with an RBF design which, assuming the desired trajectory was not

Table 4 CMAC parameters

Min, max (θ_1) = –143,143 deg	Min, max (θ_2) = –114,114 deg
Min, max ($\dot{\theta}_1$) = –57, 57 deg/s	Min, max ($\dot{\theta}_2$) = –28, 28 deg/s
Min, max ($\ddot{\theta}_1$) = –17,17 deg/{s^2	Min, max ($\ddot{\theta}_2$) = –11,11 deg/s^2

known in advance so that basis function centres must be placed on an N-per-input lattice, would require N^{14} basis functions since the flexible-joint robot has 14 (desired and actual) state inputs. Even using minimum $N = 2$, this is 16,384 basis functions for an RBF network which would have to be calculated within h seconds. For the CMAC, each input is quantized into ten units, based on a normalization of each input using maximum and minimum values for each joint (in degrees) shown in Table 4.

The desired angles and velocities have the same normalizations. Note that only desired $\ddot{\theta}_d$ is input for each joint; acceleration measurement is not required. There are six CMAC outputs needed in total (two at each step of the backstepping procedure), but we can provide the same inputs for each output and the hypercubes need only be indexed once per control cycle.

The parameters for the control and weight update laws used in all experiments where $\Lambda = 1, G_i = 1 \text{for} i = 1, 2, 3, \gamma = 0.2$, and $v = 2$. The first experiments are run with a period of 0.01 s (at 100 Hz) until performance and weights have converged, and the proposed discrete update law $\Delta\mathbf{w}$ is compared with numerical integration of the traditional update $\dot{\mathbf{w}}$. Using $\Delta\mathbf{w}$ it is possible to compute more layers in the CMAC since it does not require the computational overhead of numerical integration found with using $\dot{\mathbf{w}}$ (50 more in the 100 Hz case). A numerical integration routine based on the 5th-order Simpson's Rule provides a straightforward way to integrate the sampled data. At the 100 Hz frequency, it provides more than enough accuracy to integrate the signals coming from the (approximately) 1 Hz natural frequencies of the robot. The second experiment doubles the sampling period to 0.02 s (50 Hz). With this increased amount of computational time available, it is possible to double the number of layers in each CMAC. With twice as many layers, the CMAC is capable of greater approximation accuracy and therefore (potentially) achieving better performance.

4.5 Training time and convergence

In order to demonstrate the difficulty involved in controlling such a highly elastic system, we first look at the performance with some linear controls. The PID tuned without payload (pre-tuned) results in instability when trying to control the system with payload added (Fig. 3, dashed line). The PID tuned with the payload attached remains stable but does not result in high performance, with a root-mean-square (RMS) position error (the calculation includes the angle error in both links) in one trajectory oscillation of 8.2° (Fig. 3, dash-dot line). The backstepping technique without neural network compensation [i.e. Eqs. (56, 57, 58) with $\boldsymbol{\mu}_{k,1} = \boldsymbol{\mu}_{k,2} = \mathbf{u}_k = [0\ 0]^T$] results in RMS error of 5.0° (Fig. 3, solid line). Note that the performance of the backstepping without neural networks is approximately the same as that using the neural networks with continuous-time update on the first trial (after which the neural-adaptive scheme would and will start to reduce the error). The

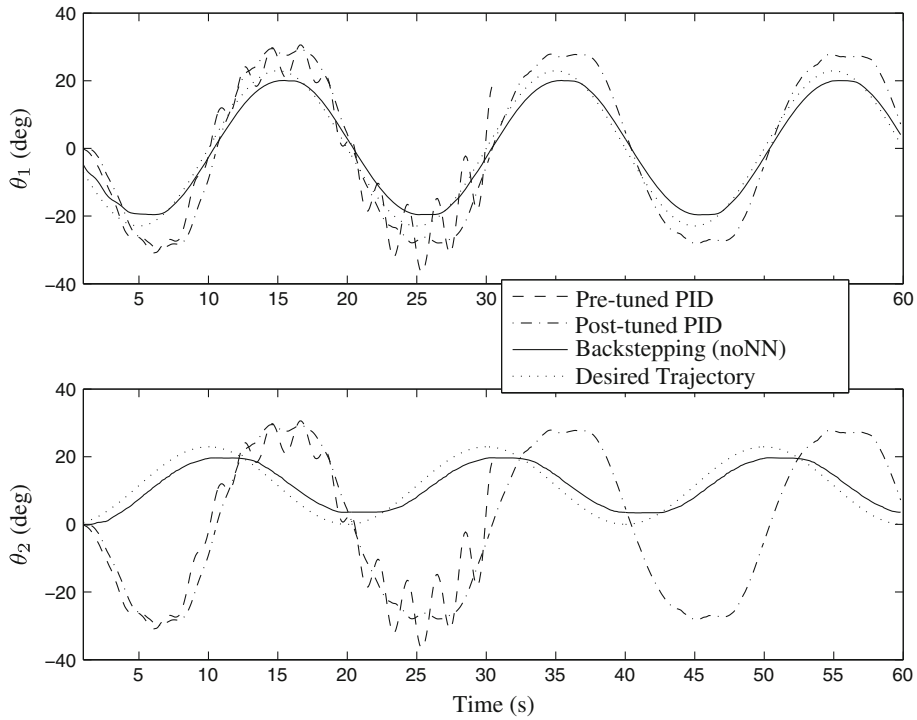

Fig. 3 The first three oscillations (trials) of the desired trajectory with linear controls

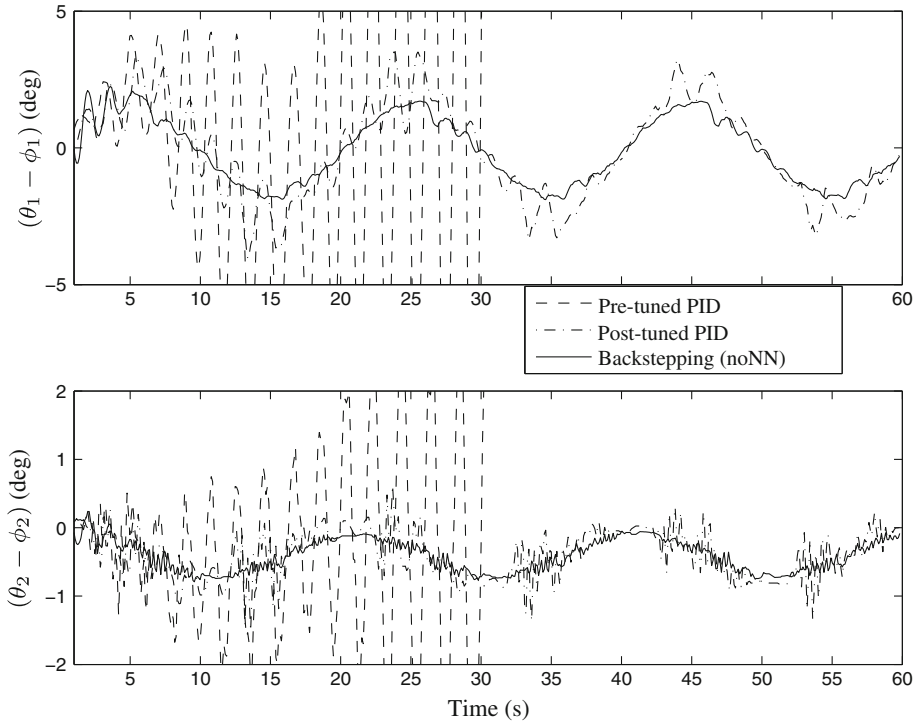

Fig. 4 Elastic deflections resulting with linear controls

Table 5 Experiment parameters

Experiment	Frequency (Hz)	Update method	Number of CMAC layers
1	100	$\dot{\hat{\mathbf{w}}}$	$m = 150$
2	100	$\Delta\hat{\mathbf{w}}$	$m = 200$
3	50	$\dot{\hat{\mathbf{w}}}$	$m = 300$
4	50	$\Delta\hat{\mathbf{w}}$	$m = 400$

amount of elastic deflection in the system is significant, illustrating the difficulty in controlling the system (Fig. 4).

The experiments compare the proposed $\Delta\hat{\mathbf{w}}$ for training a discrete-time model with integrating the update rule $\dot{\hat{\mathbf{w}}}$ for continuous-time model, at both 100 and 50 Hz (Table 5). The continuous time feedback control is the same for all experiments, and all parameters remain the same except for the number of CMAC layers. The maximum number of CMAC layers possible is used in each case that can be calculated during the 0.01 or 0.02 sec periods.

At 100 Hz, the discrete-time weight update $\Delta\hat{\mathbf{w}}$ causes the performance to converge in 50 trials, compared with 120 trials for the integrated weight update $\dot{\hat{\mathbf{w}}}$ when using the same adaptation rate constant γ (Fig. 5, left). Verifying that the largest weight magnitude over the training time has stopped growing allows us to conclude that the CMAC training has indeed converged (Fig. 6), and we are assured no surprising bursting effect will occur later on. Since one can simply increase γ in $\dot{\hat{\mathbf{w}}}$ to speed convergence, this result does not imply the proposed method trains faster in general. Rather, we are interested in comparing the final performance. After convergence, the RMS position error is 1.18° for $\Delta\hat{\mathbf{w}}$ and 1.36° for $\dot{\hat{\mathbf{w}}}$, with the slight improvement explained by the additional 50 CMAC layers used in the discrete design.

Note that this result is quite different than what is found in traditional adaptive control using a linear-in-parameters model. In that case, numerical integration is a must and a delta-rule update would result in a much poorer (or unstable) performance.

At 50 Hz, the result from integrating $\dot{\hat{\mathbf{w}}}$ results in an RMS error of 1.63° (Fig. 5, right). The performance is actually 20 % worse than the 100-Hz experiment, even though the number of layers in the CMAC is double. This is due to the numerical inaccuracies resulting from numerical integration at the lower frequency, lowering the ability of the CMAC to approximate the continuous-time model. For the discrete-time design the RMS error improves 26 % from the 100-Hz case, to 0.871°. In this case, the doubling of the CMAC size allows a more accurate approximation of the discrete time model.

Note that performance will improve only to a point by lowering the frequency and increasing CMAC size. For example, at 25 Hz both methods fail to produce an acceptable result, and the system appears to go unstable. This is consistent with the stability analysis for different frequencies performed in Table 3.

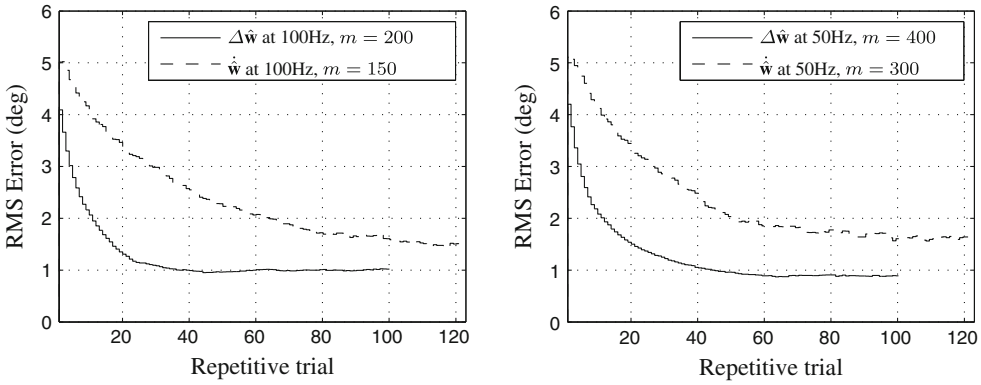

Fig. 5 Training over repetitive trials: A 100-Hz update frequency results in similar performance as the traditional method (*left*), but halving the update frequency while doubling the neural network size actually improves performance with the proposed discrete update method (*right*)

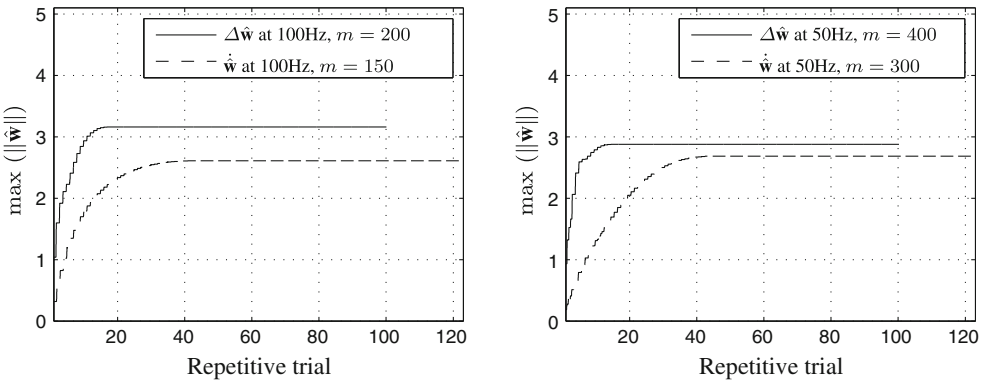

Fig. 6 Halting weight drift: The maximum weight converges in each case, justifying the value for ν in the weight update and the length of training time

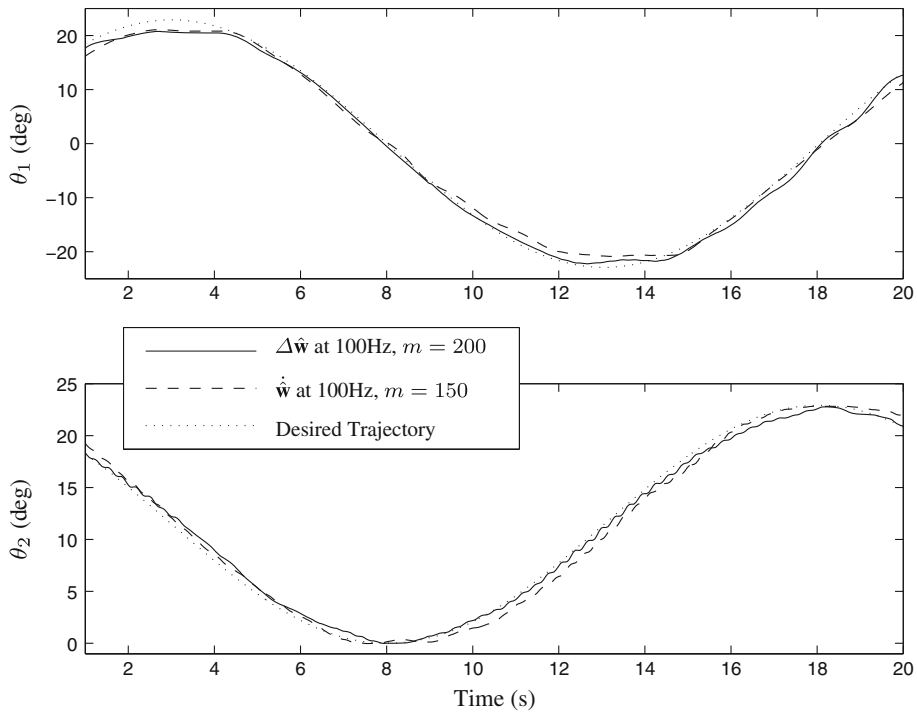

Fig. 7 Angle tracking with 100 Hz update: Comparing performance on the 100th repetitive trial

The mathematical form of the system and control is identical for known non-identical stifnesses according to (75, 76), but admittedly performance will degrade when there is uncertainty, nonlinearities, or time variations in the stiffnesses. We emphasize that the platform of a flexible-joint robot was chosen as generic non-minimum phase experiment to test the stability properties of discrete-time adaptive control in a non-trivial application. This is not a

paper that is geared specifically toward industrial flexible-joint robots; indeed, the elasticity is far greater than typical terrestrial robots (space robots do have large joint flexibilities, however).

5 Conclusion

When implementing continuous-time neural-adaptive designs for a robotic manipulator, digital implementation of control signals is typical. Assuming the digital frequency is high enough, these continuous-time designs of controls and weight updates result in high performance. However, some systems may have difficulty calculating required neural-network outputs fast enough. Alternatively, one may wish to improve approximation accuracy by increasing the size of the neural network and slowing the output rate. For addressing these problems, we propose analyzing the effects of the neural network in discrete time and applying delta-rule weight updates. The state feedback components of the control law remain in continuous time, as these can easily be implemented at a high frequency. Stability analysis follows from utilization of discrete Lyapunov functions. A non-minimum phase nonlinear system verifies stability. Specifically, an experimental two-link flexible-joint robot with highly elastic springs tracks a desired trajectory in the link angles, i.e. tip tracking. Experiments confirm that increasing the size of the neural network while slowing the digital update rate improves performance using the new method, whereas continuous-time designs do worse at the slower rate (Fig. 7).

Appendix: discrete-time transformation

We are interested in the discrete-time transformation of a continuous nonlinear system given by

$$\dot{\mathbf{z}} = \mathbf{A}(\boldsymbol{\theta})\mathbf{z} + \mathbf{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d) + \mathbf{B}(\boldsymbol{\theta})\mathbf{u}_k \tag{82}$$

We want to integrate the expression from time $t = t_1$ to $t = t_2$, where $t_2 - t_1 = T$. Let us denote terms like $\mathbf{A}(\boldsymbol{\theta}(t))$ simply as $\mathbf{A}(t)$ for ease of presentation. Now in anticipation of handling the time-varying matrix $\mathbf{A}(t)$, let us rewrite the equation as

$$\dot{\mathbf{z}} = \mathbf{A}(t_1)\mathbf{z} + [\mathbf{A}(t) - \mathbf{A}(t_1)]\mathbf{z} + \mathbf{H}(t) + \mathbf{B}(t)\mathbf{u}_k \tag{83}$$

Premultiply by $\exp(-\mathbf{A}(t_1)t)$, and let us define $\Delta\mathbf{A} = \mathbf{A}(t) - \mathbf{A}(t_1)$; then

$$e^{-\mathbf{A}(t_1)t}\dot{\mathbf{z}} - e^{-\mathbf{A}(t_1)t}\mathbf{A}(t_1)\mathbf{z} = e^{-\mathbf{A}(t_1)t}[\Delta\mathbf{A}\mathbf{z} + \mathbf{H}(t) + \mathbf{B}(t)\mathbf{u}_k] \tag{84}$$

$$\frac{de^{-\mathbf{A}(t_1)t}\mathbf{z}}{dt} = e^{-\mathbf{A}(t_1)t}[\Delta\mathbf{A}\mathbf{z} + \mathbf{H}(t) + \mathbf{B}(t)\mathbf{u}_k] \tag{85}$$

Let us integrate the left-hand side of (85).

$$\int_{t_1}^{t_2} \frac{de^{-\mathbf{A}(t_1)\tau}\mathbf{z}}{d\tau} d\tau = e^{-\mathbf{A}(t_1)t}\mathbf{z}(t)\Big|_{t_1}^{t_2} = e^{-\mathbf{A}(t_1)t_2}\mathbf{z}(t_2) - e^{-\mathbf{A}(t_1)t_1}\mathbf{z}(t_1)$$

By integrating and multiplying both sides by $\exp(\mathbf{A}(t_1)t_2)$ we have

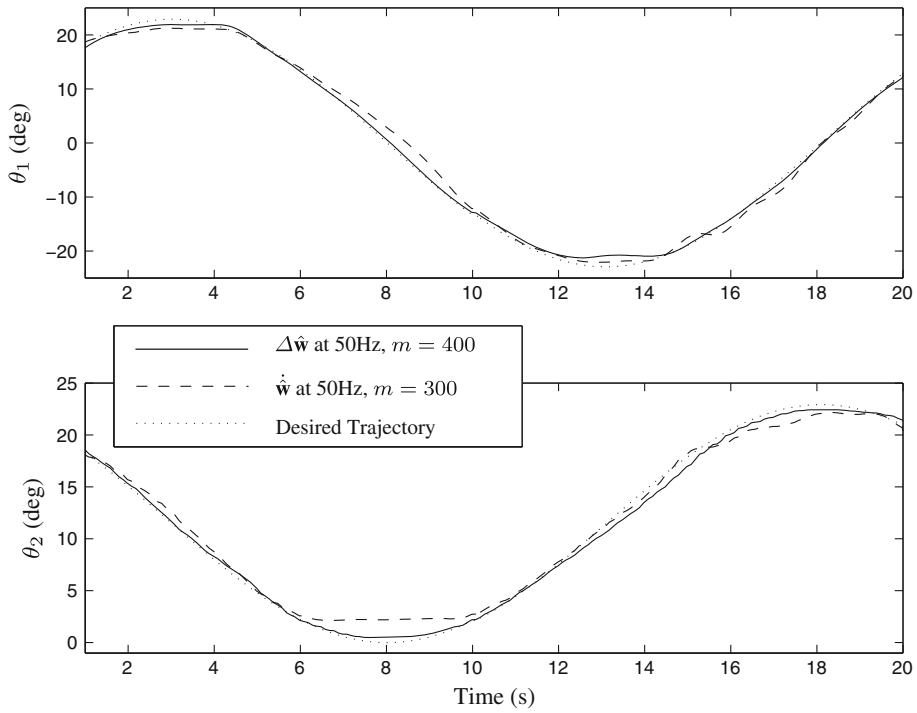

Fig. 8 Angle tracking with 50 Hz update: Comparing performance on the 100th repetitive trial

$$\mathbf{z}(t_2) - e^{\mathbf{A}(t_1)(t_2-t_1)}\mathbf{z}(t_1) = \int_{t_1}^{t_2} e^{\mathbf{A}(t_1)(t_2-\tau)}[\Delta\mathbf{A}(\tau)\mathbf{z} + \mathbf{H}(\tau) + \mathbf{B}(\tau)\mathbf{u}_k]d\tau \quad (86)$$

Choose $t_1 = kT$, $t_2 = (k+1)T$, and denote $\mathbf{z}_k = \mathbf{z}(kT)$, $\mathbf{A}_k = \mathbf{A}(kT)$, and $\mathbf{B}_k = \mathbf{B}(kT)$

$$\mathbf{z}_{k+1} = e^{T\mathbf{A}_k}\mathbf{z}_k + \int_{kT}^{(k+1)T} e^{\mathbf{A}_k((k+1)T-\tau)}[\Delta\mathbf{A}(\tau)\mathbf{z} + \mathbf{H}(\tau) + \mathbf{B}(\tau)]d\tau \quad (87)$$

By defining $\Delta\mathbf{B}(t) = \mathbf{B}(t) - \mathbf{B}_k$, then we can write

$$\mathbf{z}_{k+1} = \mathbf{a}_k\mathbf{z}_k + \mathbf{b}_k\mathbf{u}_k + \int_{kT}^{(k+1)T} e^{\mathbf{A}_k((k+1)T-\tau)}[\Delta\mathbf{A}(\tau)\mathbf{z} + \mathbf{H}(\tau) + \Delta\mathbf{B}(\tau)\mathbf{u}_k]d\tau \quad (88)$$

where (Fig. 8)

$$\mathbf{a}_k = e^{T\mathbf{A}_k} \quad (89)$$

$$\mathbf{b}_k = \int_{kT}^{(k+1)T} e^{\mathbf{A}_k((k+1)T-\tau)}d\tau\mathbf{B}_k \quad (90)$$

References

Albus J (1975) Data storage in the cerebellar model articulation controller (CMAC). J Dyn Syst Meas Contr 97:228–233

Albus J (1975) A new approach to manipulator control: the cerebellar model articulation controller (CMAC). J Dyn Syst Measur Contr 97:220–227

Chaturvedi DK, Malik OP (2007) Experimental studies of a generalized neuron based adaptive power system stabilizer. Soft Comput 11:149–155

Chiu CH (2010) Adaptive output recurrent cerebellar model articulation controller for nonlinear system control. Soft Comput 14:627–638

Chiu CH, Peng YF, Lin YW (2011) Robust intelligent backstepping tracking control for wheeled inverted pendulum. Soft Comput 15:2029–2040

Dixon W, Zergeroglu E, Dawson D, Hannan M (2000) Global adaptive partial state feedback tracking control of rigid- link flexible-joint robots. Robotica 18:325–336

Frayman Y, Wang L (2002) A dynamically constructed fuzzy neural controller for direct model reference adaptive control of multi-input-multi-output nonlinear processes. Soft Computing 6:244–253

Huang AC, Chen YC (2004) Adaptive sliding control for single-link flexible-joint robot with mismatched uncertainties. IEEE Trans. Contr. Syst. Technol 12(5):770–775

Huang AC, Chien MC (2009) Design of a regressor-free adaptive impedance controller for flexible-joint electrically driven robots. In: Proceedings of IEEE International Conference on Industrial Electronics and Applications, pp 17–22. Xi'an, China

Ider S, zgren M (2000) Trajectory tracking control of flexible-joint robots. Comput Struct 76(6):757–763

Ioannuou P, Kokotovic P (1984) Instability analysis and improvement of robustness of adaptive control. Automatica 20(5):583–594

Jagannathan S (1999) Discrete-time CMAC NN control of feedback linearizable nonlinear systems under a persistence of excitation. IEEE Trans Neural Netw 10:128–137

Kanellakopoulos I (1994) A discrete-time adaptive nonlinear system. IEEE Trans Automat Contr 39:2362–2365

Kim MS, Lee JS (2004) Adaptive tracking control of flexible-joint manipulators without overparametrization. J Robotic Syst 21(7): 369–379

Kim N, Calise A, Hovakimyan N (2004) Several extensions in methods for adaptive output feedback control. In: Proceedings of American Control Conference, pp 2421–2426, Boston

Kim Y, Lewis F (2000) Optimal design of CMAC neural-network controller for robot manipulators. IEEE Trans Syst Man Cybern C 30(1):22–30

Kuo F, Sloan I (2005) Lifting the curse of dimensionality. Am Math Soc 52:1320–1328

Kwan C, Lewis F (2000) Robust backstepping control of nonlinear systems using neural networks. IEEE Trans Syst Man Cybern A 30:753–766

Lee Y, Zak SH (2004) Uniformly ultimately bounded fuzzy adaptive tracking controllers for uncertain systems. IEEE Trans Fuzzy Syst 12:797–811

Lei Y, Wu H (2006) Tracking control of robotic manipulators based on the all-coefficient adaptive control method. Int J Control Automat Syst 4(2):139–145

Macnab C (2007) A new robust weight update for multilayer-perceptron adaptive control. Control Intell Syst 35(3):279–288

Macnab C (2010) Improved output tracking of a flexible-joint arm using neural networks. Neural Process Lett 32(2):201–218

Macnab C, D'Eleuterio G (2000) Discrete-time Lyapunov design for neuroadaptive control of elastic-joint robots. Int J Robot Res 19:511–525

Miller III W, Glanz F, Kraft L (1990) CMAC: an associative neural network alternative to backpropagation. Proc IEEE 78(10):1561–1567

Miller III W, Hewes P, Glanz F, Kraft L (1990) Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. IEEE Trans Robot Automat 6(1):1–9

Nakanishi J, Schaal S (2004) Feedback error learning and nonlinear adaptive control. Neural Netw 17:1453–1465

Ozgoli S, Taghirad HD (2006) A survey on the control of flexible joint robots. Asian J Control 8(4):1–15

Rokui M, Khorasani K (1997) An indirect adaptive control for fully feedback linearizable discrete-time non-linear systems. Int J Adapt Contr Sig Proc 11:665–680

Rokui MR, Khorasani K (2000) Experimental results on discrete-time nonlinear adaptive tracking control of a flexible link manipulator. IEEE Trans Syst Man Cybern B 30(1):151–164

Spong M, Vidyasagar M (1989) Robot dynamics and control. Wiley, New York

Suna FC, Lib HX, Lic L (2002) Robot discrete adaptive control based on dynamic inversion using dynamical neural networks. Automatica 38:1977–1983

Tian L, Wang J, Mao Z (2004) Constrained motion control of flexible robot manipulators based on recurrent neural networks. IEEE Trans Syst Man Cybern B 34(3):1541–1552

Wang HR, Yang L, Wei LX (2007) Fuzzy-neuro position/force control for robotic manipulators with uncertainties. Soft Comput 11:311–315

Yeh P, Kokotovic P (1995) Adaptive control of a class of nonlinear discrete-time systems. Int J Contr 62:303–324