

Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 12
Lecture No: 71

Real Time Neural Network Based Control of a Robotic Manipulator (Part 5)

(Refer Slide Time: 00:17)



Hello everyone. Welcome to our final session of Nonlinear and Adaptive Control. I am Srikant Sukumar from Systems and Control, IIT Bombay. I am very very thankful to all those who have attended this course. And I really hope I have been able to generate enough interest in the topic of nonlinear adaptive control, it is a rather rich area. And it is of course a rather powerful set of tools in nonlinear control, which is what makes it one of the most applied areas in nonlinear control, for fighter jets like what you see in the background here and also for spacecrafts. These have even been tested in actual flight conditions.

So, and of course there are also companies that specialize in adaptive control for drones. So, a lot of applications again I focus mostly on the aerospace applications or aero-mechanical applications, because my own background. But, there is quite a bit of work out there on electrical, biological, chemical systems also using adaptive control.

So, what we have been looking at in this last week is a little bit of what you see in the bottom right of our motivating image in the background, which is on neural networks. So, as you can see

here in the image, this is like a deep learning network, so essentially a multi layered neural network. So, it is typical practice in neural networks to do to have several layers now, because again, we have a lot more computational power and lot more data available to train these networks.

So, the typical approach is to have an offline training, where you input a lot of training data on the left of this network, and you get the output on the right of the network. And use this input-output in order to tune the weights that appear in this sort of nonlinear function approximator. And then, you use these weights in order to do actual experiments. And when you go to the actual run, you design yours controllers based on these weights. So, this is what we have been looking at.

(Refer Slide Time: 02:48)

The slide content includes:

- for all $T \geq 0$ and some $\gamma \geq 0$.
- We say the system is dissipative if it is passive and in addition $\int_0^{\infty} y^T(r)u(r) dr \neq 0$ implies $\int_0^{\infty} g(r) dr > 0$. (6)
- A special sort of dissipativity occurs if $g(t)$ is a monic quadratic function of $[x]$ with bounded coefficients, where $x(t)$ is the internal state of the system. We call this state-strict passivity, and are not aware of its use previously in the literature (although cf. [11]). Then the L_2 norm of the state is overbounded in terms of the L_2 inner product of output and input (i.e., the power delivered to the system). This we use to advantage to conclude some internal boundedness properties of the system without the usual assumptions of observability (e.g., persistence of excitation), stability, etc.
- C. Robot Arm Dynamics**

$$\ddot{q}_i + \ddot{q}_i = \ddot{q}_i + \ddot{q}_i$$

$$M\ddot{q} + Y(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau$$

The dynamics of an n -link robot manipulator may be expressed in the Lagrange form [17]

with $q \in \mathbb{R}^n$ the joint variable vector, $M(q)$ the inertia matrix, $Y(q, \dot{q})\dot{q}$ the Coriolis/centrifugal matrix, $G(q)$ the gravity vector, $F(\dot{q})$ the friction vector, and τ_d the desired joint acceleration.
- where the functional estimation error is given by $\hat{f} = f - \hat{f}$. (15)
- This is an error system wherein the filtered tracking error is driven by the functional estimation error.
- The control r_c incorporates a proportional-plus-derivative (PD) term in $K_c r = K_p(e + \Lambda e)$.
- In the remainder of the paper we shall use (14) to focus on selecting NN tuning algorithms that guarantee the stability of the filtered tracking error $r(t)$. Then, since (9), with the input considered as $r(t)$ and the output as $e(t)$ describes a stable system, standard techniques [23], [41] guarantee that $e(t)$ exhibits stable behavior. In fact, $\|e\|_2 \leq \beta \|r\|_2 / \sigma_{\min}(\Lambda)$, $\|e\|_2 \leq \beta \|r\|_2$, with $\sigma_{\min}(\Lambda)$ the minimum singular value of Λ . Generally Λ is diagonal, so that $\sigma_{\min}(\Lambda)$ is the smallest element of Λ .
- The following standard properties of the robot dynamics are required [17] and hold for any revolute rigid serial robot arm.
- Property 1: $M(q)$ is a positive definite symmetric matrix bounded by $m_1 I \leq M(q) \leq m_2 I$
- Property 2: $Y(q, \dot{q})\dot{q}$ is bounded by $\eta_1(q)\|\dot{q}\|$

Handwritten annotations include:

- Red circle around the equation $M(q)\ddot{q} + Y(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau$.
- Red arrows pointing to \dot{q} and \ddot{q} in the equation, labeled "joint q-dot" and "world frame".
- Red arrows pointing to $Y(q, \dot{q})\dot{q}$ and $F(\dot{q})$, labeled "joint q-dot" and "world frame".

150 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REEE_Workshop_Slides_L... Adaptive_Control_Week02 Multilayer_neural-net_Lewis_Liu Survey-adaptivem-Archiv...

$\dot{x} = f(x, u, t), y = h(x, t)$

with state $x(t) \in \mathbb{R}^n$. We say the solution is uniformly ultimately bounded (UUB) if there exists a compact set $U \subset \mathbb{R}^n$ such that for all $x(t_0) = x_0 \in U$, there exists an $\epsilon > 0$ and a number $T(\epsilon, x_0)$ such that $\|x(t)\| < \epsilon$ for all $t \geq t_0 + T$. As we shall see in the proof of the theorems, the compact set U is related to the compact set on which NN approximation property (3) holds. Note that U can be made larger by selecting more hidden-layer neurons.

Some aspects of passivity will subsequently be important [11], [16], [17], [41]. A system with input $u(t)$ and output $y(t)$ is said to be passive if it verifies an equality of the so-called "power form"

$$L(t) = y^T u - g(t) \quad (4)$$

with $L(t)$ lower bounded and $g(t) \geq 0$. That is

$$\int_0^T y^T(\tau)u(\tau) d\tau \geq \int_0^T g(\tau) d\tau - \gamma^2 \quad (5)$$

for all $T \geq 0$ and some $\gamma \geq 0$. We say the system is dissipative if it is passive and in addition

$$\int_0^\infty y^T(\tau)u(\tau) d\tau \neq 0 \text{ implies } \int_0^\infty g(\tau) d\tau > 0. \quad (6)$$

A special sort of dissipativity occurs if $g(t)$ is a monic quadratic function of $\|x\|$ with bounded coefficients, where $x(t)$ is the internal state of the system. We call this state-strict passivity, and are not aware of its use previously in the

dynamics may be written in terms of the filtered tracking error as

$$M\dot{r} = -V_m r - \tau + \dot{d} + \tau_d \quad (10)$$

where the nonlinear robot function is

$$\dot{x} = M(y)\dot{q} + A\dot{q} + V_m(y, \dot{q})x + A\dot{q} \quad (11)$$

and, for instance, we may select

$$\dot{x} = [x^T \dot{x}^T \ddot{x}^T]^T$$

Define now a control input torque as

$$\tau_c = \dot{f} + K_v r \quad (12)$$

gain matrix $K_v = K_v^T > 0$ and $\hat{f}(x)$ an estimate of $f(x)$ provided by some means not yet disclosed. The closed-loop system becomes

$$M\dot{r} = -(K_v + V_m)r + \tau_c - \dot{f} \quad (14)$$

where the functional estimation error is given by

$$\tilde{f} = \dot{f} - \hat{f}$$

This is an error system wherein the filtered tracking error is driven by the functional estimation error. The control τ_c incorporates a proportional (PD) term in $K_v r = K_v(\dot{q} + \tau)$.

In the remainder of the paper we shall use selecting NN tuning algorithms that guarantee the tracking error $r(t)$. Then, since (9),

NPTEL

150 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REEE_Workshop_Slides_L... Adaptive_Control_Week02 Multilayer_neural-net_Lewis_Liu Survey-adaptivem-Archiv...

For notational convenience define the matrix of all the weights as

$$Z = \begin{bmatrix} W & 0 \\ 0 & -V \end{bmatrix} \quad (16)$$

A. Some Bounding Assumptions and Facts

Some required mild bounding assumptions are now stated. The two assumptions will be true in every practical situation, and are standard in the existing literature. The facts are easy to prove given the assumptions.

Assumption 1: The ideal weights are bounded by known positive values so that $\|V\|_F \leq V_M, \|W\|_F \leq W_M$, or

$$\|Z\|_F \leq Z_M \quad (17)$$

with Z_M known.

Assumption 2: The desired trajectory is bounded in the sense, for instance, that

$$\|y_d\| \leq Q_d \quad (18)$$

where $Q_d \in \mathbb{R}$ is a known constant.

we take the following

Fact 4: For sigmoid, RBF, and tanh activation functions, the higher-order terms in the Taylor series are bounded by

$$|O(\hat{V}^T x)| \leq c_3 + c_4 Q_d \|\hat{V}\|_F + c_5 \|\hat{V}\|_F |r|$$

where c_i are computable positive constants.

Fact 4 is direct to show using (19), some standard norm inequalities, and the fact that $\sigma(\cdot)$ and its derivative are bounded by constants for RBF, sigmoid, and tanh.

The extension of these ideas to nets with greater than three layers is not difficult, and leads to composite function terms in the Taylor series (giving rise to backpropagation filtered error terms for the multilayer net case—see Theorem 3.1).

B. Controller Structure and Error System Dynamics

Define the NN functional estimate of (11) by

$$\hat{f}(x) = \hat{W}^T \sigma(\hat{V}^T x) \quad (25)$$

with \hat{V}, \hat{W} the current (estimated) values of the ideal NN weights V, W as provided by the tuning algorithms subsequently to be discussed. With τ_c defined in (13), select the control input

$$\tau = \tau_c - v = \hat{W}^T \sigma(\hat{V}^T x) + K_v r + v \quad (26)$$

NPTEL

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY BOMBAY. Downloaded on June 07, 2022 at 05:39:43 UTC from IEEE X

We of course, are looking at rather special situation where there are a couple of things, the weight or the learning itself of the weights is happening online along with the control. And secondly, in order to make sure that this learning plus control mix is stable. We use adaptive control principles. So, I hope you have seen by now that the adaptive control is essentially very very closely connected to deep learning. And we also saw that deep learning is now prevalent also in reinforcement learning and other classification approaches. So, over and all you understand, I hope you understand that these adaptive control, stable adaptive control methods are of significant use in learning.

So, we are looking at a very specific case of this three-layer neural network being applied as a function approximator to a multi-link robot. And because it is like a neural network approximator, we do not really care to write out these mass inertia functions, and the Coriolis functions by hand for every single robot. If you bring in any robot, this is like a plug and play controller. So, once you you could, if you have a control module, you can plug it into any multi-link robot, which has this kind of Euler-Langrange system structure, like you see here in equation-7.

And and, you can start controlling it, because your neural network using this adaptive law will automatically learn all these nonlinear robot function. So, this is, this is what is the cool aspect, that it is agnostic to what kind of M, V, G, F you have, this function, the neural network function approximator is going to learn it. And how the learning happens via the adaptive update laws is what we saw last time, so, we sort of added a robustification term.

(Refer Slide Time: 05:09)

The screenshot shows a presentation slide titled "III. NN CONTROLLER". The slide text is as follows:

III. NN CONTROLLER

In this section we derive a NN controller for the robot dynamics in Section II. We propose various weight-tuning algorithms, including standard backpropagation. It is shown that with backpropagation tuning the NN can only be guaranteed to perform suitably in closed loop under unrealistic ideal conditions (which require, e.g., $f(x)$ linear). A modified tuning algorithm is subsequently proposed so that the NN controller performs under realistic conditions.

Thus, assume that the nonlinear robot function (11) is given by an NN as in (3) for some constant "ideal" NN weights W and V , where the net reconstruction error $\tilde{e}(x)$ is bounded by a known constant ϵ_N . Unless the net is "minimal," suitable "ideal" weights may not be unique [1], [42]. The "best" weights may then be defined as those which minimize the supremum norm over S of $\tilde{e}(x)$. This issue is not of major concern here, as we only need to know that such ideal weights exist; their actual values are not required.

According to Theorem 2.1, this mild approximation assumption always holds for continuous functions. This is in stark contrast to the case for adaptive control, where approximation assumptions such as the Erzberger or linear-in-the-parameters assumptions may not hold. The mildness of this assumption is the main advantage to using multilayer nonlinear nets over linear two-layer nets.

For notational convenience define the matrix of all the

NN's. Proper use of these Taylor series-based results gives a requirement for new terms in the weight tuning algorithms for nonlinear NN's that do not occur in linear NN's.

Let \hat{V}, \hat{W} be some estimates of the ideal weight values, as provided for instance by the weight tuning algorithms to be introduced. Define the weight deviations or weight estimation errors as

$$\tilde{V} = V - \hat{V}, \quad \tilde{W} = W - \hat{W}, \quad \tilde{Z} = Z - \hat{Z} \quad (20)$$

and the hidden-layer output error for a given x as

$$\tilde{\sigma} = \sigma(V^T x) - \hat{\sigma}(V^T x) \quad (21)$$

The Taylor series expansion for a given $\tilde{\sigma}$ may be written as

$$\hat{\sigma}(V^T x) = \sigma(\hat{V}^T x) + \sigma'(\hat{V}^T x) \tilde{V}^T x + O(\tilde{V}^T x)^2 \quad (22)$$

with $\sigma'(z) \equiv d\sigma(z)/dz$, and $O(z)^2$ denoting terms of order two. (Compare to [33] where a different Taylor series was used for identification purposes only.) Denoting $\hat{\sigma}' = \sigma'(\hat{V}^T x)$, we have

$$\tilde{\sigma} = \sigma'(V^T x) \tilde{V}^T x + O(\tilde{V}^T x)^2 - \hat{\sigma}' V^T x + O(\tilde{V}^T x)$$

Different bounds may be put on the Taylor order terms depending on the choice for $\sigma(\cdot)$. Note that

$$O(\tilde{V}^T x)^2 = [\sigma(V^T x) - \sigma(\hat{V}^T x)] - \sigma'(\hat{V}^T x) \tilde{V}^T x$$

Handwritten annotations in red include circles around $\hat{\sigma}(V^T x)$ in equation (21) and $\hat{\sigma}' = \sigma'(\hat{V}^T x)$ in the text below equation (22). There are also some scribbles and arrows pointing to the equations.

The key step is the use now of the Taylor series approximation (23) for δ , according to which the closed-loop error system is

$$M\dot{r} = -(K_v + V_m)r + \tilde{W}^T \delta + \tilde{W}^T \delta' \tilde{V}^T x + w_1 + v \quad (28)$$

where the disturbance terms are

$$w_1(t) = \tilde{W}^T \delta' \tilde{V}^T x + W^T O(\tilde{V}^T x)^2 + (\epsilon + \tau_d) \quad (29)$$

Unfortunately, using this error system does not yield a compact set outside which a certain Lyapunov function derivative is negative. Therefore, write finally the error system

$$M\dot{r} = -(K_v + V_m)r + \tilde{W}^T (\delta - \delta' \tilde{V}^T x) + \tilde{W}^T \delta' \tilde{V}^T x + w + v \quad (30)$$

$$\equiv -(K_v + V_m)r + \zeta_1$$

where the disturbance terms are

$$w(t) = \tilde{W}^T \delta' \tilde{V}^T x + W^T O(\tilde{V}^T x)^2 + (\epsilon + \tau_d) \quad (31)$$

It is important to note that the NN reconstruction error $e(x)$, the robot disturbances τ_d , and the higher-order terms in the Taylor series expansion of $f(x)$ all have exactly the same influence as disturbances in the error system. The next key bound is required. Its importance is in allowing one to overbound $w(t)$ at each time by a known computable function; it follows from Fact 4 and some standard norm inequalities.

and any constant positive definite (design) matrices F, G . Then the tracking error $r(t)$ goes to zero with t and the weight estimates \tilde{W}, \tilde{V} are bounded.

Proof: Define the Lyapunov function candidate

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W}) + \frac{1}{2} \text{tr}(\tilde{V}^T G^{-1} \tilde{V}) \quad (35)$$

Differentiating yields

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + \text{tr}(\tilde{W}^T F^{-1} \dot{\tilde{W}}) + \text{tr}(\tilde{V}^T G^{-1} \dot{\tilde{V}})$$

whence substitution from (28) (with $w_1 = 0, v = 0$) yields

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r + \text{tr}(\tilde{W}^T (F^{-1} \dot{\tilde{W}} + \delta r^T) + \text{tr}(\tilde{V}^T (G^{-1} \dot{\tilde{V}} + \epsilon r^T \tilde{W}^T \delta'))$$

The skew symmetry property makes the second term zero, and since $\dot{W} = W - \dot{W}$ with W constant, so that $d\tilde{W}/dt = -d\dot{W}/dt$ (and similarly for V), the tuning rules yield

$$\dot{L} = -r^T K_v r.$$

Handwritten notes: $\delta = F \delta r^T$ (33), $\tilde{V} = G \delta' (\tilde{V}^T \tilde{W} r)$ (34), and "only possible \tilde{V}^T ".

where the disturbance terms are

$$w_1(t) = \tilde{W}^T \delta' \tilde{V}^T x + W^T O(\tilde{V}^T x)^2 + (\epsilon + \tau_d) \quad (29)$$

Unfortunately, using this error system does not yield a compact set outside which a certain Lyapunov function derivative is negative. Therefore, write finally the error system

$$M\dot{r} = -(K_v + V_m)r + \tilde{W}^T (\delta - \delta' \tilde{V}^T x) + \tilde{W}^T \delta' \tilde{V}^T x + w + v \quad (30)$$

$$\equiv -(K_v + V_m)r + \zeta_1$$

where the disturbance terms are

$$w(t) = \tilde{W}^T \delta' \tilde{V}^T x + W^T O(\tilde{V}^T x)^2 + (\epsilon + \tau_d) \quad (31)$$

It is important to note that the NN reconstruction error $e(x)$, the robot disturbances τ_d , and the higher-order terms in the Taylor series expansion of $f(x)$ all have exactly the same influence as disturbances in the error system. The next key bound is required. Its importance is in allowing one to overbound $w(t)$ at each time by a known computable function; it follows from Fact 4 and some standard norm inequalities.

and any constant positive definite (design) matrices F, G . Then the tracking error $r(t)$ goes to zero with t and the weight estimates \tilde{W}, \tilde{V} are bounded.

Proof: Define the Lyapunov function candidate

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W}) + \frac{1}{2} \text{tr}(\tilde{V}^T G^{-1} \tilde{V}) \quad (35)$$

Differentiating yields

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + \text{tr}(\tilde{W}^T F^{-1} \dot{\tilde{W}}) + \text{tr}(\tilde{V}^T G^{-1} \dot{\tilde{V}})$$

whence substitution from (28) (with $w_1 = 0, v = 0$) yields

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r + \text{tr}(\tilde{W}^T (F^{-1} \dot{\tilde{W}} + \delta r^T) + \text{tr}(\tilde{V}^T (G^{-1} \dot{\tilde{V}} + \epsilon r^T \tilde{W}^T \delta'))$$

The skew symmetry property makes the second term zero, and since $\dot{W} = W - \dot{W}$ with W constant, so that $d\tilde{W}/dt = -d\dot{W}/dt$ (and similarly for V), the tuning rules yield

$$\dot{L} = -r^T K_v r.$$

And of course, we sort of wrote the dynamics in a rather nice way, either in the in form 28 or in the form of equation-30. So, it was either in the form of equation 28 with the disturbance terms w_1 , or in the form of equation 30 with disturbance terms labeled as W . So, the first result then that we saw was on backpropagation, which was rather ideal situation, where we assume that these disturbances are 0. We saw that it is not very easy to sort of claim w_1 is 0, it is obvious if you assume that the second order terms are 0. So, your function is a linear function. And we do not have these, it is an exact estimator and the disturbance is 0.

So, these terms will become 0 rather easily, but we realize that this term will become 0, only if you know you have this kind of W tilde, V tilde type of a thing. So, this is a sort of linear in

parameter type of assumption of the function is, if you want these to be 0. And if this does happen, we did have this backpropagation-based tuning of weights.

So, this sort of maps, why it is given this name is because this sort of maps to the sort of weight tuning algorithms that standard nonlinear neural networks use offline. So, this is the update law, I mean, and if you notice that, you have to remember that this sigma hat was essentially, sigma hat is just defined as sigma of V hat transpose x.

So, so remember, this is just notation, so, sigma hat is just V hat transpose x. So, this was a notation that was used. And so that notation, so you can see that W hat dot depends on V hat, and V hat dot also depends on W hat. So, they are sort of inter dependent here. And we also saw how the Lyapunov analysis goes because of this ideal circumstance, where you do not have this w1 and v. We just make them 0 here, and you look at this equation.

So, all you have is, so this term goes to 0 because of the skew symmetry, then this term contributes to a nice negative term. And these two terms are of course cancelled using the update law. And then you have of course, this L function, this Lyapunov candidate function L, its derivative is negative semi-definite. And then you can use your LaSalle invariance or Barbalat's Lemma signal chasing to prove r goes to 0.

(Refer Slide Time: 07:52)

1:53 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

EE333_Workshop_Slides_Level... Adaptive_Control_Workshop Multilayer_neural-net_Lewis... Survey-adapt-learn-Necessity

Lecture 12-6

Since $L > 0$ and $\dot{L} \leq 0$ this shows stability in the sense of Lyapunov so that r, \hat{V} , and \hat{W} (and hence \hat{V}, \hat{W}) are bounded. Moreover

$$\int_0^{\infty} -\dot{L} dt < \infty. \quad (36)$$

LaSalle's extension [17], [41] is now used to show that $r(t)$ in fact goes to zero. Boundedness of r guarantees the boundedness of e and \dot{e} , whence boundedness of the desired trajectory shows q, \dot{q}, x are bounded. Property 2 then shows boundedness of $V_m(q, \dot{q})$. Now, $L = -2r^T K_v \dot{r}$, and the boundedness of $M^{-1}(q)$ and of all signals on the right-hand side of (28) verify the boundedness of \dot{L} , and hence the uniform continuity of \dot{L} . This allows one to invoke Barbalat's Lemma [17], [41] in connection with (36) to conclude that \dot{L} goes to zero with t , and hence that $r(t)$ vanishes. ■

Note that the problem of net weight initialization occurring in other approaches in the literature does not arise. In fact, selecting the initial weights $\hat{W}(0), \hat{V}(0)$ as zero takes the NN out of the circuit and leaves only the outer tracking loop in Fig. 2. It is well known that the PD term $K_v \dot{r}$ in (26) can then stabilize the plant on an interim basis. A formal proof reveals that K_v should be large enough and the initial filtered error $r(0)$ small enough. The exact value of K_v needed for initial stabilization is given in [9], though for practical purposes it is only necessary to select K_v large.

or when $f(x)$ is nonlinear, cannot be guaranteed to yield bounded weights in the closed-loop system.

General Case: To confront the stability and tracking performance of a NN robot arm controller in the thorny general case, we require: 1) the modification of the weight tuning rule and 2) the addition of a robustifying term $v(t)$. The problem in this case is that, though it is not difficult to conclude that $r(t)$ is bounded, it is impossible without these modifications to show that the NN weights are bounded in general. Boundedness of the weights is needed to verify that the control input $\tau(t)$ remains bounded.

The next theorem relies on an extension to Lyapunov theory. The disturbance τ_d , the NN reconstruction error ϵ , and the nonlinearity of $f(x)$ make it impossible to show that the Lyapunov derivative \dot{L} is nonpositive for all $r(t)$ and weight values. In fact, it is only possible to show that \dot{L} is negative outside a compact set in the state space. This, however, allows one to conclude boundedness of the tracking error and the neural net weights. In fact, exp during the proof. The required Theorem 1.5-6], the last portion of the proof used in [26].

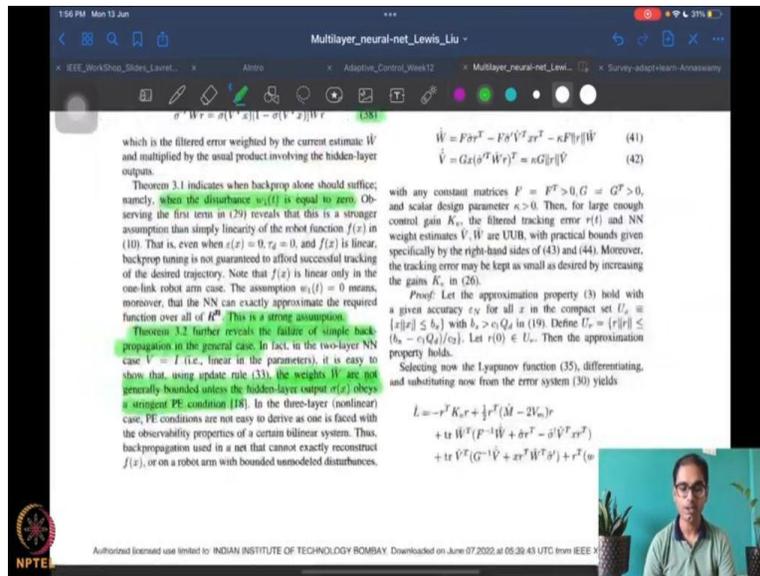
Theorem 3.2: Let the desired τ_d and the initial filtered error $r(0)$ be small enough. Take the control input for (7) as

$$\tau(t) = -K_v(t)\dot{r}(t) + \tau_d(t) + \epsilon(t)$$

update goes away, so you will stay at \hat{W} and \hat{V} to be 0 for all time. So basically, it takes the neural network out of the circuit and has the only has only the outer tracking.

So, so of course, basically, if you I mean you can do some kind of a local stabilization using this K_v term. So, if you have a large enough K_v , and if the filtered error r is already small enough, then you are fine, so, so anyway. So, so of course it is not possible to choose its large K_v for several practical reasons. So, this is so this is the connection to the backpropagation algorithm. So, this is equation-33 and 34 are matched with 37 and 38, which is what is known in the neural network theory as the back-propagation results.

(Refer Slide Time: 09:49)

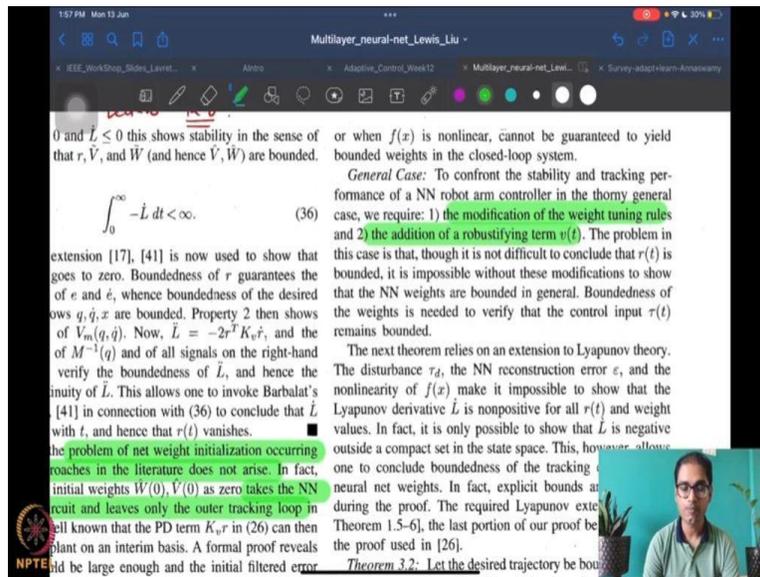


So, theorem is when it reveals that when, when only this back propagation alone will suffice. And this is when this term is exactly equal to 0. And and this is what I was saying that this first term in 29 reveals that it is a rather strong assumption than just some linearity of f_x . So, even if ϵ is 0, τ_d is 0, f is linear, this is not guaranteed. So, and of course, f_x is of course, only linear in the one link arm case. So, there is a lot of assumptions, a lot of assumptions. So, this is a I mean, w_1 being equal to 0 is a very strong assumption. So, so of course, then there is additional results that are in this article also.

So, that is what is there in theorem 3.2, it reveals the failure of simple back propagation in the general case. Basically, it can be shown the author's themselves, I am not sure if they are showing it themselves, so we will not worry about that right now. But, the point is it can be

shown that the weights become unbounded. I mean, all of this is because of the assumption that w_1 is equal to 0, w_1 is not really 0, even for the linear f_x case. And even the linear case happens, only if f_x is only the robot the single link rotate. So, this is what the authors say are generally not bounded, unless the hidden layer obeys a stringent persistence condition.

(Refer Slide Time: 11:26)



Now, therefore, there is a need to look at the more general case. So, so of course, there is need to modify the tuning weight, there is the need to add a robustifying term, so, both of these are there. So, we already added v , because in the previous result, we assumed that v was exactly 0. And we gave a set of weight tuning rules. Now of course, the authors will give a slightly different weight tuning. So, so this is what is going to be detailed in the second theorem.

(Refer Slide Time: 12:09)

The next theorem relies on an extension to Lyapunov theory. The disturbance τ_d , the NN reconstruction error ε , and the nonlinearity of $f(x)$ make it impossible to show that the Lyapunov derivative \dot{L} is nonpositive for all $r(t)$ and weight values. In fact, it is only possible to show that \dot{L} is negative outside a compact set in the state space. This, however, allows one to conclude boundedness of the tracking error and the neural net weights. In fact, explicit bounds are discovered during the proof. The required Lyapunov extension is [17, Theorem 1.5-6], the last portion of our proof being similar to the proof used in [26].

Theorem 3.2: Let the desired trajectory be bounded by (18). Take the control input for (7) as (26) with robustifying term

$$v(t) = -K_2(\|\hat{Z}\|_F + Z_M)r \quad (39)$$

and gain

$$K_2 > C_2$$

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{W}r = \sigma(\hat{V}^T x)[1 - \sigma(\hat{V}^T x)]\hat{W}r \quad (38)$$

and gain

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad (37)$$

with C_2 the known constant in (32). Let NN weight tuning be provided by

So, of course, this I mean the disturbance τ_d and the reconstruction error, make it impossible to show that \dot{L} is non-positive for all r . So, therefore, you can only show that it is \dot{L} is negative outside a compact set, so this is what is as usual our residual set, type result, you are used to this. Because, if there is disturbance, there is an epsilon error in the function approximator, then, you are used to the, I mean you have to have some kind of residual set performance zone, you cannot have exact convergence. So, one should not even expect any exact convergence. So, this is the first important thing to remember.

(Refer Slide Time: 12:53)

Theorem 3.2: Let the desired trajectory be bounded by (18). Take the control input for (7) as (26) with robustifying term

$$v(t) = -K_2(\|\hat{Z}\|_F + Z_M)r \quad (39)$$

and gain

$$K_2 > C_2 \quad (40)$$

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{W} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W} \quad (41)$$

$$\dot{V} = Gx(\hat{\sigma}'\hat{W}r)^T = \kappa G\|r\|\hat{V} \quad (42)$$

with any constant matrices $F = F^T > 0$, $G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_2 , the filtered tracking error $r(t)$ and weight estimates \hat{V}, \hat{W} are UUB, with practical bounds specifically by the right-hand sides of (43) and (44). The tracking error may be kept as small as desired by the gains K_2 in (26).

Proof: Let the approximation property (3) be given accuracy ε_x for all x in the compact set Ω_x . This is a strong assumption.

1:58 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 7, NO. 2, MARCH 1996

Fig. 2. NN control structure.

with $\hat{r}(t)$ a function to be detailed subsequently that provides robustness in the face of higher-order terms in the Taylor series. The proposed NN control structure is shown in Fig. 2, where $g \equiv [g^T \hat{q}^T]^T$, $e \equiv [e^T \hat{e}^T]^T$.

Using this controller, the closed-loop filtered error dynamics become

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + W^T\sigma(\hat{V}^T x) - \hat{W}^T\sigma(\hat{V}^T x) + (e + \tau_d) + v.$$

Adding and subtracting $W^T\hat{\sigma}$ yields

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + W^T\hat{\sigma} + \hat{W}^T\hat{\sigma} + (e + \tau_d) + v.$$

Fact 5: The disturbance term (31) is bounded according to

$$\|w(t)\| \leq (c_N + b_N + c_N Z_M) \|z\| + c_N Z_M \|z\| + c_N \|r\|$$

or

$$\|w(t)\| \leq c_0 + c_1 \|z\| + c_2 \|z\| + c_3 \|z\| \quad (32)$$

with C_i computable known positive constants.

C. Weight Updates for Guaranteed Tracking Performance

We give here some NN weight-tuning algorithms that guarantee the tracking stability of the closed-loop system under various assumptions. It is required to demonstrate that the tracking error $r(t)$ is suitably small and that the NN weights \hat{V}, \hat{W} remain bounded, for then the control $\tau(t)$ is bounded. The key features of all our algorithms are that stability is guaranteed, there is no off-line learning phase so that NN control begins immediately, and the NN weight initialization does not require the requirement for "initial weights."

Ideal Case—Backpropagation Tuning of \hat{W}

result details the closed-loop behavior in a case that demands 1) no net functional reconstruction error, 2) no unmodeled disturbances in the robot arm dynamics, and 3) no higher-order Taylor series terms. The last amounts to the assumption that $f(x)$ in (10) is linear. In this case the tuning rules are straightforward and familiar. Our contribution lies in the proof and the conditions thus determined showing when the algorithm works, and when it cannot be relied on.

So see, so, this is the sort of robustifying term that the authors introduce. So, this is essentially the robustification term, it contains of the I mean honestly not a smooth term. It is a non-smooth term if you may, and this term is like a norm of Z hat, Frobenius norm of Z hat. And then of course, you have some state dependent terms. And then end with some gain which is larger than some constant C2. And this constant C2 is known in equation 32, it is coming from the bound-on W. So, the idea is sort of to take care of this, sort of terms, so we will see how that happens.

(Refer Slide Time: 13:38)

Adding and subtracting $W^T\hat{\sigma}$ yields

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + W^T\hat{\sigma} + \hat{W}^T\hat{\sigma} + (e + \tau_d) + v$$

with $\hat{\sigma}$ and σ defined in (21). Adding and subtracting now $\hat{W}^T\hat{\sigma}$ yields

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + \hat{W}^T\hat{\sigma} + \hat{W}^T\hat{\sigma} + (e + \tau_d) + v. \quad (27)$$

The key step is the use now of the Taylor series approximation (23) for $\hat{\sigma}$, according to which the closed-loop error system is

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + \hat{W}^T\hat{\sigma} + \hat{W}^T\hat{\sigma} + w_1 + v \quad (28)$$

where the disturbance terms are

$$w_1(t) = \hat{W}^T\hat{\sigma}(\hat{V}^T x) + W^T O(\hat{V}^T x)^2 + (e + \tau_d). \quad (29)$$

Unfortunately, using this error system does not yield a compact set outside which a certain Lyapunov function derivative is negative. Therefore, write finally the error system

$$\dot{M}\hat{r} = -(K_v + V_m)\hat{r} + \hat{W}^T(\hat{\sigma} - \hat{\sigma}(\hat{V}^T x)) + \hat{W}^T\hat{\sigma}(\hat{V}^T x) + v + w_1$$

$$\equiv -(K_v + V_m)\hat{r} + \zeta_1 \quad (30)$$

where the disturbance terms are

$$w(t) = \hat{W}^T\hat{\sigma}(\hat{V}^T x) + W^T O(\hat{V}^T x)^2 + (e + \tau_d). \quad (31)$$

It is important to note that the NN reconstruction error $\hat{e}(x)$, the robot disturbances τ_d , and the higher-order terms

result details the closed-loop behavior in a certain idealized case that demands: 1) no net functional reconstruction error, 2) no unmodeled disturbances in the robot arm dynamics, and 3) no higher-order Taylor series terms. The last amounts to the assumption that $f(x)$ in (10) is linear. In this case the tuning rules are straightforward and familiar. Our contribution lies in the proof and the conditions thus determined showing when the algorithm works, and when it cannot be relied on.

Theorem 3.1. Let the desired trajectory be bounded and suppose the disturbance term $w(t)$ in (28) is equal to zero. Let the control input for (7) be given by (26) with $v(t) = 0$ and weight tuning provided by

$$\dot{W} = F\hat{\sigma} \quad (33)$$

$$\dot{V} = G\hat{\sigma}(\hat{\sigma}^T W\hat{\sigma})^T \quad (34)$$

and any constant positive definite (design) matrices F, G . Then the tracking error $r(t)$ goes to zero with t and the weight estimates \hat{V}, \hat{W} are bounded.

Proof. Define the Lyapunov function candidate

$$L = \frac{1}{2}r^T M r + \frac{1}{2}\text{tr}(\hat{W}^T F^{-1} \hat{W}) + \frac{1}{2}\text{tr}(\hat{V}^T G^{-1} \hat{V}). \quad (35)$$

Differentiating yields

$$\dot{L} = r^T \dot{M} r + \frac{1}{2}r^T \dot{M} r + \text{tr}(\hat{W}^T F^{-1} \dot{\hat{W}}) + \text{tr}(\hat{V}^T G^{-1} \dot{\hat{V}}) + r^T W^T \sigma^2$$

whence substitution from (28) (with $w_1 = 0$),

$$\dot{L} = -r^T K_v r + \frac{1}{2}r^T (\dot{M} - 2V_m) r + \text{tr}(\hat{W}^T F^{-1} \dot{\hat{W}}) + \text{tr}(\hat{V}^T G^{-1} \dot{\hat{V}}) + r^T W^T \sigma^2$$

are nothing but the continuous propagation algorithm. In the scalar and gain

$$\sigma(z)(1 - \sigma(z)) \quad (37) \quad K_2 > C_2 \quad (40)$$

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{W} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W} \quad (41)$$

$$\dot{V} = Gx(\hat{\sigma}'\hat{W}r)^T = \kappa G\|r\|\hat{V} \quad (42)$$

weighted by the current estimate \hat{W} product involving the hidden-layer

backprop alone should suffice; since $w_1(t)$ is equal to zero. Observation (29) reveals that this is a stronger linearity of the robot function $f(x)$ in $\tau_d = 0$, $\tau_d = 0$, and $f(x)$ is linear, need to afford successful tracking error that $f(x)$ is linear only in the assumption $w_1(t) = 0$ means, NPTE exactly approximate the required

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_v , the filtered tracking error $r(t)$ and NN weight estimates \hat{V}, \hat{W} are UUB, with practical bounds specifically by the right-hand sides of (43) and (44). the tracking error may be kept as small as desired by the gains K_v in (26).

Proof: Let the approximation property (3) a given accuracy ϵ , for all x in the compact

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{W} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W} \quad (41)$$

$$\dot{V} = Gx(\hat{\sigma}'\hat{W}r)^T = \kappa G\|r\|\hat{V} \quad (42)$$

error weighted by the current estimate \hat{W} usual product involving the hidden-layer

ates when backprop alone should suffice; disturbance $w_1(t)$ is equal to zero. Observation (29) reveals that this is a stronger linearity of the robot function $f(x)$ in $\tau_d = 0$, $\tau_d = 0$, and $f(x)$ is linear, guaranteed to afford successful tracking error. Note that $f(x)$ is linear only in the case. The assumption $w_1(t) = 0$ means, NN can exactly approximate the required f^* . This is a strong assumption. her reveals the failure of simple back-propagation. In fact, in the two-layer NN linear in the parameters), it is easy to update rule (33), the weights \hat{W} are not unless the hidden-layer output $\sigma(x)$ obeys linearity [18]. In the three-layer (nonlinear) are not easy to derive as one is faced with series of a certain bilinear system. Thus

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_v , the filtered tracking error $r(t)$ and NN weight estimates \hat{V}, \hat{W} are UUB, with practical bounds given specifically by the right-hand sides of (43) and (44). Moreover, the tracking error may be kept as small as desired by increasing the gains K_v in (26).

Proof: Let the approximation property (3) hold with a given accuracy ϵ_N for all x in the compact set $U_x \equiv \{x\|x\| \leq b_x\}$ with $b_x > c_1 Q_d$ in (19). Define $U_r = \{r\|r\| \leq (b_x - c_1 Q_d)/c_2\}$. Let $r(0) \in U_r$. Then the approximation property holds.

Selecting now the Lyapunov function (35), differential and substituting now from the error system (30) yields

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r$$

And then the neural network weight tuning is changed a little bit. If you see the second term is not very different from what we had. So, you had $Gx \sigma' \hat{V}^T x r^T$, W hat r transpose. And this is exactly the same, $Gx \sigma' \hat{V}^T x r^T$ W hat r whole transpose, this is exactly the same. But, the first term was $F \sigma' \hat{V}^T x r^T$ in the back-propagation term case, and here it is modified to include two more terms. Again, there is a non-smooth term here, but, it is a nice continuous term, but a non-smooth term, because it is a norm r type of a term.

And this term is also written in terms of norm r type of term, I think this is not equal to, this is what I need to. Let us see, let us just a second please, just give me a moment, please. I think that

term is not quite right. I believe this is not equal to this is actually a minus term, and this is not an equal to, but this is a minus term. That is a typo error in the paper, so that is fine.

So, this also the V hat dot is also changed, and the W hat dot also changes. So, you have the ideal version and then you have these new terms. And now, how you and essentially what you can prove is that for large enough K_v and the filter tracking error r , these become uniform ultimately bounded with practical bounds given specifically by 43 44. So, this is what you sort of obtain, this is what you sort of obtain.

(Refer Slide Time: 15:49)

Theorem 3.1 indicates when backprop alone should suffice; namely, when the disturbance $w_1(t)$ is equal to zero. Observing the first term in (29) reveals that this is a stronger assumption than simply linearity of the robot function $f(x)$ in (10). That is, even when $e(x) = 0$, $\tau_d = 0$, and $f(x)$ is linear, backprop tuning is not guaranteed to afford successful tracking of the desired trajectory. Note that $f(x)$ is linear only in the one-link robot arm case. The assumption $w_1(t) = 0$ means, moreover, that the NN can exactly approximate the required function over all of R^n . This is a strong assumption.

Theorem 3.2 further reveals the failure of simple backpropagation in the general case. In fact, in the two-layer NN case $V = I$ (i.e., linear in the parameters), it is easy to show that, using update rule (33), the weights \hat{W} are not generally bounded unless the hidden-layer output $\sigma(x)$ obeys a stringent PE condition [18]. In the three-layer (nonlinear) case, PE conditions are not easy to derive as one is faced with the observability properties of a certain bilinear system. Thus, backpropagation used in a net that cannot exactly reconstruct $f(x)$, or on a robot arm with bounded unmodeled disturbances,

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_v , the filtered tracking error $r(t)$ and NN weight estimates \hat{V}, \hat{W} are UUB with practical bounds given specifically by the right-hand sides of (43) and (44). Moreover, the tracking error may be kept as small as desired by increasing the gains K_v in (26).

Proof: Let the approximation property (3) hold with a given accuracy ϵ_N for all x in the compact set $U_x = \{x | \|x\| \leq b_x\}$ with $b_x > c_1 Q_d$ in (19). Define $U_r = \{r | \|r\| \leq (b_x - c_1 Q_d)/c_2\}$. Let $r(0) \in U_r$. Then the approximation property holds.

Selecting now the Lyapunov function (35), differentiating, and substituting now from the error system (30) yields

$$\begin{aligned} \dot{L} = & -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r \\ & + \text{tr } \hat{W}^T (F^{-1} \dot{\hat{W}} + \hat{\sigma} r^T - \delta' \hat{V}^T z r^T) \\ & + \text{tr } \hat{V}^T (G^{-1} \dot{\hat{V}} + z r^T \hat{W}^T \delta') + r^T (w \end{aligned}$$

Authorized licensed use limited to: INDIAN INSTITUTE OF TECHNOLOGY BOMBAY. Downloaded on June 07, 2022 at 05:39:43 U

selecting the initial weights $\hat{W}(0), \hat{V}(0)$ as zero takes the NN out of the circuit and leaves only the outer tracking loop in Fig. 2. It is well known that the PD term $K_v r$ in (26) can then stabilize the plant on an interim basis. A formal proof reveals that K_v should be large enough and the initial filtered error $r(0)$ small enough. The exact value of K_v needed for initial stabilization is given in [9], though for practical purposes it is only necessary to select K_v large.

Note next that (33) and (34) are nothing but the continuous-time version of the backpropagation algorithm. In the scalar sigmoid case, for instance

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad (37)$$

so that

$$\delta^T \hat{W} r = \sigma(\hat{V}^T x) [1 - \sigma(\hat{V}^T x)] \hat{W} r \quad (38)$$

which is the filtered error weighted by the current estimate \hat{W} and multiplied by the usual product involving the hidden-layer outputs.

Theorem 3.1 indicates when backprop alone should suffice; namely, when the disturbance $w_1(t)$ is equal to zero. Observing the first term in (29) reveals that this is a stronger assumption than simply linearity of the robot function $f(x)$ in (10). That is, even when $e(x) = 0$, $\tau_d = 0$, and $f(x)$ is linear, tuning is not guaranteed to afford successful tracking of the desired trajectory. Note that $f(x)$ is linear only in the

neural net weights. In fact, explicit bounds are discovered during the proof. The required Lyapunov extension is [17, Theorem 1.5-6], the last portion of our proof being similar to the proof used in [26].

Theorem 3.2: Let the desired trajectory be bounded by (18). Take the control input for (7) as (26) with robustifying term

$$U = -K_x (\|\hat{Z}\| r + Z_M) r \quad (39)$$

and gain

$$K_x > C_2 \quad (40)$$

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{\hat{W}} = F \hat{\sigma} r^T - F \delta' \hat{V}^T z r^T - \kappa F \|r\| \hat{W} \quad (41)$$

$$\dot{\hat{V}} = G z (\delta^T \hat{W} r)^T + \kappa G \|r\| \hat{V} \quad (42)$$

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for control gain K_x , the filtered tracking error weight estimates \hat{V}, \hat{W} are UUB, with practical bounds given specifically by the right-hand sides of (43) and (44). Moreover, the tracking error may be kept as small as desired by increasing the gains K_x in (26).

2:03 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REE_WorkShop_Slides_Lehrl... Adaptive_Control_Week12 Multilayer_neural-net_Lewis_Liu Survey-adaptHearn-Assessory

$$M\dot{r} = -(K_v + V_m)r + \dot{W}^T \hat{\sigma} + \dot{W}^T \hat{\sigma}' \hat{V}^T x + w_1 + v \quad (28)$$

here the disturbance terms are

$$w_1(t) = \dot{W}^T \hat{\sigma}' \hat{V}^T x + W^T O(\hat{V}^T x)^2 + (\epsilon + \tau_d) \quad (29)$$

Unfortunately, using this error system does not yield a compact set outside which a certain Lyapunov function derivative is negative. Therefore, write finally the error system

$$M\dot{r} = -(K_v + V_m)r + \dot{W}^T (\hat{\sigma} - \hat{\sigma}' \hat{V}^T x) + \dot{W}^T \hat{\sigma}' \hat{V}^T x + w + v$$

$$\equiv -(K_v + V_m)r + \zeta_1 \quad (30)$$

here the disturbance terms are

$$w(t) = \dot{W}^T \hat{\sigma}' \hat{V}^T x + W^T O(\hat{V}^T x)^2 + (\epsilon + \tau_d) \quad (31)$$

It is important to note that the NN reconstruction error $\epsilon(x)$, the robot disturbances τ_d , and the higher-order terms in the Taylor series expansion of $f(x)$ all have exactly the same influence as disturbances in the error system. The next step is to bound $w(t)$ at each time by a known computable function; follows from Fact 4 and some standard norm inequalities.

and any constant positive definite (design) matrices F, G . Then the tracking error $r(t)$ goes to zero with t and the weight estimates \hat{V}, \hat{W} are bounded.

Proof: Define the Lyapunov function candidate

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\hat{W}^T F^{-1} \hat{W}) + \frac{1}{2} \text{tr}(\hat{V}^T G^{-1} \hat{V}) \quad (35)$$

Differentiating yields

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + \text{tr}(\hat{W}^T F^{-1} \dot{\hat{W}}) + \text{tr}(\hat{V}^T G^{-1} \dot{\hat{V}})$$

whence substitution from (28) (with $w_1 = 0, v = 0$) yields

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r + \text{tr}(\hat{W}^T (F^{-1} \dot{\hat{W}} + \hat{\sigma}' r^T) + \text{tr}(\hat{V}^T (G^{-1} \dot{\hat{V}} - r^T \hat{W}^T \hat{\sigma}'))$$

The skew symmetry property makes the second term zero, and since $\dot{W} = W - \dot{W}$ with W constant, so that $dW/dt = -\dot{W}/dt$ (and similarly for V), the tuning rates

$$\dot{L} = -r^T K_v r \leq 0$$

2:03 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REE_WorkShop_Slides_Lehrl... Adaptive_Control_Week12 Multilayer_neural-net_Lewis_Liu Survey-adaptHearn-Assessory

with σ and σ' defined in (21). Adding and subtracting now $\dot{W}^T \hat{\sigma}$ yields

$$M\dot{r} = -(K_v + V_m)r + \dot{W}^T \hat{\sigma} + \dot{W}^T \hat{\sigma}' \hat{V}^T x + \dot{W}^T \hat{\sigma} + (\epsilon + \tau_d) + v \quad (27)$$

The key step is the use now of the Taylor series approximation (23) for $\hat{\sigma}$, according to which the closed-loop error system is

$$M\dot{r} = -(K_v + V_m)r + \dot{W}^T \hat{\sigma} + \dot{W}^T \hat{\sigma}' \hat{V}^T x + w_1 + v \quad (28)$$

where the disturbance terms are

$$w_1(t) = \dot{W}^T \hat{\sigma}' \hat{V}^T x + W^T O(\hat{V}^T x)^2 + (\epsilon + \tau_d) \quad (29)$$

Unfortunately, using this error system does not yield a compact set outside which a certain Lyapunov function derivative is negative. Therefore, write finally the error system

$$M\dot{r} = -(K_v + V_m)r + \dot{W}^T (\hat{\sigma} - \hat{\sigma}' \hat{V}^T x) + \dot{W}^T \hat{\sigma}' \hat{V}^T x + w + v$$

$$\equiv -(K_v + V_m)r + \zeta_1 \quad (30)$$

where the disturbance terms are

$$w(t) = \dot{W}^T \hat{\sigma}' \hat{V}^T x + W^T O(\hat{V}^T x)^2 + (\epsilon + \tau_d) \quad (31)$$

It is important to note that the NN reconstruction error $\epsilon(x)$, the robot disturbances τ_d , and the higher-order terms in the Taylor series expansion of $f(x)$ all have exactly the same influence as disturbances in the error system. The next step is to bound $w(t)$ at each time by a known computable function; follows from Fact 4 and some standard norm inequalities.

the proof and the conditions thus determined showing when the algorithm works, and when it cannot be relied on.

Theorem 3.1: Let the desired trajectory be bounded and suppose the disturbance term $w_1(t)$ in (28) is equal to zero. Let the control input for (7) be given by (26) with $v(t) = 0$ and weight tuning provided by

$$\dot{W} = F \hat{\sigma} r^T \quad (33)$$

$$\dot{V} = G x (\hat{\sigma}' \hat{W} r)^T \quad (34)$$

only possible if \dot{W}

and any constant positive definite (design) matrices F, G . Then the tracking error $r(t)$ goes to zero with t and the weight estimates \hat{V}, \hat{W} are bounded.

Proof: Define the Lyapunov function candidate

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\hat{W}^T F^{-1} \hat{W}) + \frac{1}{2} \text{tr}(\hat{V}^T G^{-1} \hat{V}) \quad (35)$$

Differentiating yields

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + \text{tr}(\hat{W}^T F^{-1} \dot{\hat{W}}) + \text{tr}(\hat{V}^T G^{-1} \dot{\hat{V}})$$

whence substitution from (28) (with $w_1 = 0, v = 0$) yields

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r + \text{tr}(\hat{W}^T (F^{-1} \dot{\hat{W}} + \hat{\sigma}' r^T) + \text{tr}(\hat{V}^T (G^{-1} \dot{\hat{V}} + r^T \hat{W}^T \hat{\sigma}'))$$

The skew symmetry property makes the second term zero, and since $\dot{W} = W - \dot{W}$ with W constant, so that $dW/dt = -\dot{W}/dt$ (and similarly for V), the tuning rates

So, so you of course, assume these nice approximation properties hold, which is this for the function approximator with some epsilon N, in and we assume that this also holds not everywhere, but in this nice set. This is nice set. And again you have Ur, which is a set, which is correspondingly defined for the r variable. You typically assume that these bounds hold within a certain set. So, then of course this approximation property holds here, for this set r in Ur, 0 in Ur. So, now as usual you do the Lyapunov derivative, so this is the nice term, then you have this skew symmetry term which is going to go away.

Then, this first term is coming just from the derivative of our same L. The L is not changed actually, the L is still the same, so this, so this is the term. So, the first term here, so this term is

coming from this guy, and then this is coming from your r dot. And so, these terms are coming from r dot. So, in fact, these, this term and this term is not different from what you had in the previous case, except for of course, the fact that you have introduced a V . And, so that sort of also has to show up here, those terms also show up corresponding to the v here, because you have this kind of a term.

So, and then then you have this term which was again same as before. So, so the additional terms if you see are, you had a \hat{r} transpose and this term already, so the additional terms here are, so, these two terms were already there. So, the additional term is sort of this guy. Let us see this an additional term, but this two was already there. Yes, this is an additional term because we are using equation number-30, so we are using this guy. So, the, so this guy results in this term, and the V term is coming from this, so this guy accounts for the first term here. So, everything else is the same, in fact, my my my mistake, the V is not even stated until now.

(Refer Slide Time: 18:37)

The tuning rules give

$$\begin{aligned} \dot{L} &= -r^T K_w r + \kappa \|r\| \text{tr} \{ \dot{W}^T (W - \hat{W}) \\ &\quad + \kappa \|r\| \text{tr} \{ \dot{V}^T (V - \hat{V}) + r^T (w + v) \} \\ &= -r^T K_w r + \kappa \|r\| \text{tr} \{ \dot{Z}^T (Z - \hat{Z}) \} + r^T (w + v) \end{aligned}$$

Since $\text{tr} \{ \dot{Z}^T (Z - \hat{Z}) \} = \langle \dot{Z}, Z \rangle_F - \|\dot{Z}\|_F^2 \leq \|\dot{Z}\|_F \|Z\|_F - \|\dot{Z}\|_F^2$, there results

$$\begin{aligned} \dot{L} &\leq -K_{v \min} \|r\|^2 + \kappa \|r\| \|\dot{Z}\|_F (Z_M - \|\hat{Z}\|_F) \\ &\quad - K_Z (\|\dot{Z}\|_F + Z_M) \|r\|^2 + \|r\| \|w\| \\ &\leq -K_{v \min} \|r\|^2 + \kappa \|r\| \|\dot{Z}\|_F (Z_M - \|\hat{Z}\|_F) \\ &\quad - K_Z (\|\dot{Z}\|_F + Z_M) \|r\|^2 \\ &\quad + \|r\| (C_0 + C_1 \|\dot{Z}\|_F + C_2 \|\hat{Z}\|_F \|r\|) \\ &\leq -\|r\| [K_{v \min} \|r\| + \kappa \|\dot{Z}\|_F (\|\hat{Z}\|_F - Z_M) \\ &\quad - C_0 - C_1 \|\dot{Z}\|_F] \end{aligned}$$

where $K_{v \min}$ is the minimum singular value of K_v and the last inequality holds due to (40). Thus, \dot{L} is negative as long as the term in braces is positive.

Defining $C_3 = Z_M + C_1/\kappa$ and completing the square yields

$$\begin{aligned} K_{v \min} \|r\| + \kappa \|\dot{Z}\|_F (\|\hat{Z}\|_F - C_2) - C_0 \\ = \kappa (\|\dot{Z}\|_F - C_3/2)^2 - \kappa C_3^2/4 + K_{v \min} \|r\| - C_0 \end{aligned}$$

which is guaranteed positive as long as either

Fig. 3. Neural net closed-loop error system.

weight errors are fundamentally bounded by Z_M (through $\|\hat{Z}\|_F$). The parameter κ offers a design trade-off between the relative magnitudes of $\|\dot{r}\|$ and $\|\dot{Z}\|_F$.

The first terms of (41) and (42) are nothing but the standard backpropagation algorithm. The last terms correspond to e-modification [26] in standard use in adaptive control to guarantee bounded parameter estimates. The forgetting term in the second term in (41) is novel and bears design significance. It can be thought of as being a nonlinear "backprop" network. The second term in (41) sees a traveling wave in the backpropagation algorithm.

2:05 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REB_Shock_Slides_Lewis_Liu... Adaptive_Control_Week12 Multilayer_neural-net_Lewis_Liu... Survey-adapt+learn+Anomaly

K_v should be large enough and the initial filtered error small enough. The exact value of K_v needed for initialization is given in [9], though for practical purposes it is necessary to select K_v large.

The next that (33) and (34) are nothing but the continuous-version of the backpropagation algorithm. In the scalar case, for instance

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \quad (37)$$

$$\sigma^T \dot{W} r = \sigma(\dot{V}^T x)[1 - \sigma(\dot{V}^T x)] \dot{W} r \quad (38)$$

is the filtered error weighted by the current estimate \dot{W} multiplied by the usual product involving the hidden-layer outputs.

Theorem 3.1 indicates when backprop alone should suffice; namely, when the disturbance $w_1(t)$ is equal to zero. Observing the first term in (29) reveals that this is a stronger condition than simply linearity of the robot function $f(x)$ in x . That is, even when $\varepsilon(x) = 0$, $\tau_d = 0$, and $f(x)$ is linear, backprop tuning is not guaranteed to afford successful tracking of the desired trajectory. Note that $f(x)$ is linear only in the link robot arm case. The assumption $w_1(t) = 0$ means, however, that the NN can exactly approximate the required function over all of \mathbb{R}^n . This is a strong assumption.

Theorem 3.2 further reveals the failure of simple backprop.

Theorem 3.2: Let the desired trajectory be bounded by (18). Take the control input for (7) as (26) with robustifying term

$$v(t) = -K_v(\|\dot{Z}\|_F + Z_M)r \quad (39)$$

and gain

$$K_v > C_2 \quad (40)$$

with C_2 the known constant in (32). Let NN-weight tuning be provided by

$$\dot{W} = F\sigma r^T - F\sigma^T \dot{V}^T x r^T - \kappa F\|r\|\dot{W} \quad (41)$$

$$\dot{V} = Gx(\sigma^T \dot{W} r)^T - \kappa G\|r\|\dot{V} \quad (42)$$

with any constant matrices $F = F^T > 0$, $G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_v , the filtered tracking error $r(t)$ and NN weight estimates \dot{W} are UUB, with practical bounds specified by the right-hand sides of (43) and (44). Moreover, the tracking error may be kept as small as desired by increasing the gains K_v in (26).

Proof: Let the approximation property (3) hold to a given accuracy ε_N for all x in the compact set $\{x \mid \|x\| \leq b_x\}$ with $b_x > c_3 Q_d$ in (19). Define $U_c = \{r \mid \|r\| \leq U_c\}$. Then the approxi-

2:07 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

REB_Shock_Slides_Lewis_Liu... Adaptive_Control_Week12 Multilayer_neural-net_Lewis_Liu... Survey-adapt+learn+Anomaly

$$\begin{aligned} & + \|r\|C_0 + C_1\|\dot{Z}\|_F + C_2\|\dot{Z}\|_F\|r\| \\ & \leq -\|r\|[K_{v,\min}\|r\| + \kappa\|\dot{Z}\|_F(\|\dot{Z}\|_F - Z_M) \\ & \quad - C_0 - C_1\|\dot{Z}\|_F] \end{aligned}$$

where $K_{v,\min}$ is the minimum singular value of K_v , and the last inequality holds due to (40). Thus, \dot{L} is negative as long as the term in braces is positive.

Defining $C_3 = Z_M + C_1/\kappa$ and completing the square yields

$$\begin{aligned} & K_{v,\min}\|r\| + \kappa\|\dot{Z}\|_F(\|\dot{Z}\|_F - C_2) - C_0 \\ & = \kappa(\|\dot{Z}\|_F - C_3/2)^2 - \kappa C_3^2/4 + K_{v,\min}\|r\| - C_0 \end{aligned}$$

which is guaranteed positive as long as either

$$\|r\| > \frac{\kappa C_3^2/4 + C_0}{K_{v,\min}} \equiv b_r \quad (43)$$

or

$$\|\dot{Z}\|_F > C_3/2 + \sqrt{C_3^2/4 + C_0/\kappa} \equiv b_Z \quad (44)$$

where

$$C_3 = Z_M + C_1/\kappa \quad (45)$$

Thus, \dot{L} is negative outside a compact set. The form of the right-hand side of (43) shows that the control gain K_v can be selected large enough so that $b_r < (b_x - c_1 Q_d)/c_3$. Then, any trajectory $r(t)$ beginning in U_c evolves completely within U_c . According to a standard Lyapunov theorem, the error

weight errors are fundamentally bounded by Z_M (through The parameter κ offers a design trade-off between the relative magnitudes of $\|r\|$ and $\|\dot{Z}\|_F$.

The first terms of (41) and (42) are nothing but the standard backpropagation algorithm. The last terms correspond to κ -modification [26] in standard use in adaptive control to guarantee bounded parameter estimates; they form a sort of forgetting term in the weight updates. The second term in (41) is novel and bears discussion. The standard backpropagation terms can be thought of as backward propagating signals through a nonlinear "backprop" network [27] that contains multiple hidden units. The second term in (41) seems to correspond to a first-order correction to the weight tuning for \dot{W} .

Note that there is design freedom in the degree of complexity (e.g., size) of the NN. For a more complex NN (e.g., more hidden units), the bounding constants will decrease, resulting in smaller tracking errors. On the other hand, a simplified NN with fewer hidden units will result in larger error bounds. This degradation can be compensated for, as long as bounds are known, by selecting a larger signal $v(t)$, or for Λ in (9).

An alternative to guaranteeing bounded NN weights for the two-layer case presented in [32], [33], and [34] is used for tuning \dot{W} . In the algorithm is given in [22].

So, once you do that, the authors apply the tuning rule, and they do a bunch of simplifications. I am not going into the complete details of this. I would ask you to verify the steps here, in order to get to this, I will ask you to verify the steps. So, these nice negative terms still remains. And then you have this kind of W minus W tilde, V minus V tilde, and r transpose w plus v type of term. So, V is still not substituted notice. And then of course, there is a you do this kind of nice trace equalities, and then you bound these trace equalities. And once you do that, you start working with the norms everywhere, the vector norms and the matrix norms here.

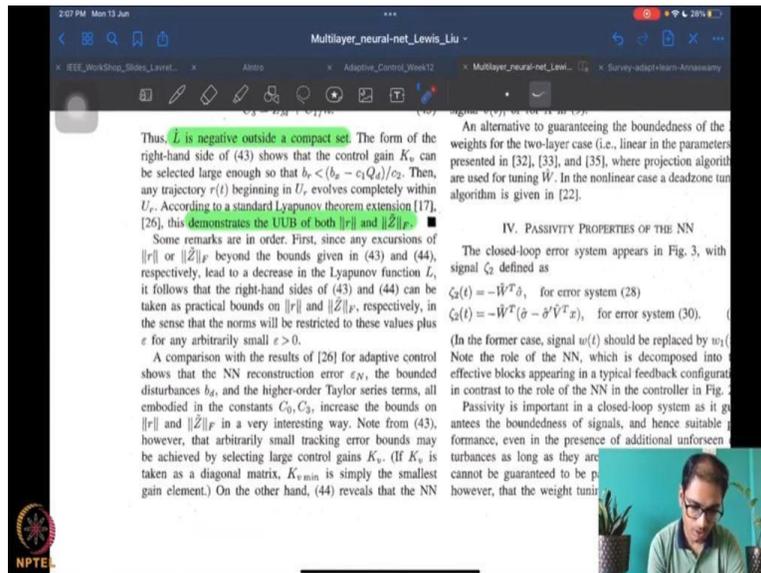
And this is where of course you introduce the expression for V also, and that expression for V gets substituted here. So again, I am not going into the details, I will ask you, I will request you

to sort of verify the details. These are simply just using standard norm inequalities. And then this particular equality and inequality right here, and after substitution of v , you get this kind of final expression, this kind of a final expression. Here $K_v \min$ is of course, minimum singular value of K_v , and the last inequality holds due to 40. So, so basically 40 is this guy. So, a lot of lot of the earlier inequalities that were mentioned have been used here, in order to arrive at these bounds, at this nice bound here.

So now, if you do a nice completion of squares after defining this kind of a this kind of a new variable C_3 . So, what is the completion of square? The completion of square is being done here on this term, so this term is this guy, and here is what you have here. And and you you sort of nicely break it into these pieces.

And, of course, this is where you sort of, you sort of do a standard completion of squares in order to get your residual set bound. And that is essentially what is being done in these steps, that is essentially what is being done in these sets. So, you get two different sets. So, one set is obtained from here, this guy, that is here C_3 . And this, and then you have another set on r . So, once it is on the Z tilde, which is a parameter error, and other set is on the r , which is the state errors.

(Refer Slide Time: 21:58)



And it is what you get is that L dot will be negative outside this compact set, and which is defined by these two inequalities 43 and 44. And therefore, you have standard uniform ultimate boundedness for both r and Z tilde F . And because Z tilde contains both V tilde and W tilde, you

have what you need. So, one thing is obvious that any excursion beyond these 43 44 leads to decrease in Lyapunov function, which means that you will get them back into the bounds. So, Z and Z tilde and r of course, restricted within this residual set. So, so anyway, I mean these bounds are dictated by number of higher order terms in the Taylor series, and magnitudes of the Taylor series bounds et-cetera.

(Refer Slide Time: 22:51)

Fig. 3. Neural net closed-loop error system.

weight errors are fundamentally bounded by Z_M (through C_3). The parameter κ offers a design trade-off between the relative eventual magnitudes of $\|r\|$ and $\|\tilde{Z}\|_F$.

The first terms of (41) and (42) are nothing but the standard backpropagation algorithm. The last terms correspond to the ϵ -modification [26] in standard use in adaptive control to guarantee bounded parameter estimates; they form a special sort of forgetting term in the weight updates. The second term in (41) is novel and bears discussion. The standard backprop terms can be thought of as backward propagating signals in a nonlinear "backprop" network [27] that contains multipliers. The second term in (41) seems to correspond to a forward travelling wave in the backprop net that provides a second-order correction to the weight tuning for \dot{W} .

Note that there is design freedom in the degree of complexity (e.g., size) of the NN. For a more complex NN (e.g., more hidden units), the bounding constants will decrease, resulting in smaller tracking errors. On the other hand, a simplification with fewer hidden units will result in larger error bounds. Degradation can be compensated for, as long as known, by selecting a larger value for K_v in the robot signal $v(t)$, or for Λ in (9).

An alternative to guaranteeing the boundedness of the NN weights for the two-layer case (i.e., linear in the parameters) is given in [13], where projection

$$\dot{\tau} \leq -K_v \min\|\tau\|^2 + \kappa\|\tau\|\|\tilde{Z}\|_F(Z_M - \|\tilde{Z}\|_F) - K_x(\|\tilde{Z}\|_F + Z_M)\|\tau\|^2 + \|\tau\|(C_0 + C_1\|\tilde{Z}\|_F + C_2\|\tilde{Z}\|_F\|\tau\|) \leq -\|\tau\|[K_v \min\|\tau\| + \kappa\|\tilde{Z}\|_F(\|\tilde{Z}\|_F - Z_M) - C_0 - C_1\|\tilde{Z}\|_F]$$

where $K_v \min$ is the minimum singular value of K_v , and the last inequality holds due to (40). Thus, L is negative as long as the term in braces is positive. Defining $C_3 = Z_M + C_1/\kappa$ and completing the square yields

$$K_v \min\|\tau\| + \kappa\|\tilde{Z}\|_F(\|\tilde{Z}\|_F - C_2) - C_0 = \kappa(\|\tilde{Z}\|_F - C_3/2)^2 - \kappa C_3^2/4 + K_v \min\|\tau\| - C_0$$

which is guaranteed positive as long as either

$$\|\tau\| \geq \frac{\kappa C_3^2/4 + C_0}{K_v \min} \equiv b_r \quad (43)$$

$$\|\tilde{Z}\|_F > C_3/2 + \sqrt{C_3^2/4 + C_0/\kappa} \equiv b_Z \quad (44)$$

here

$$C_3 = Z_M + C_1/\kappa. \quad (45)$$

Thus, L is negative outside a compact set. The form of the right-hand side of (43) shows that the control gain K_v

with C_2 the known constant in (32). Let NN weight tuning be provided by

$$\dot{W} = F\delta r^T - F\delta^T \dot{V}^T x r^T - \kappa F\|\tau\|\dot{W} \quad (41)$$

$$\dot{V} = Gx(\delta^T \dot{W} r)^T + \kappa G\|\tau\|\dot{V} \quad (42)$$

with any constant matrices $F = F^T > 0, G = G^T > 0$, and scalar design parameter $\kappa > 0$. Then, for large enough control gain K_v , the filtered tracking error $\tau(t)$ and NN weight estimates \tilde{V}, \tilde{W} are UUB, with practical bounds given specifically by the right-hand sides of (43) and (44). Moreover, the tracking error may be kept as small as desired by increasing the gains K_v in (26).

Proof: Let the approximation property (3) hold with a given accuracy ϵ_N for all x in the compact set $U_x \equiv \{x\|\tau\| \leq b_x\}$ with $b_x > c_1 Q_d$ in (19). Define $U_r \equiv \{r\|\tau\| \leq b_r\}$ with $b_r > c_1 Q_d/c_2$. Let $\tau(0) \in U_r$. Then the approximation property holds.

Selecting now the Lyapunov function (35), differentiating it and substituting now from the error system (30)

And of course, as you know that the first terms are nothing but the backpropagation algorithm itself. So, this is what is the standard backpropagation, and the last terms look like the epsilon modification. So, if you look at these terms, these are sort of like, so, these are like the standard

backpropagation. And these terms look like your epsilon modification, because this contains a \hat{V} term and this is basically a \hat{W} term. So, this is like an epsilon modification. And this why it is an epsilon modification instead of a sigma modification? Because the additional terms are scaled by the state remember.

In the sigma modification, the \hat{W} and \hat{V} term would be scaled by a constant, here they are not scaled by a constant, they are scaled by the state value itself. So, why it is better is because, if r becomes close to 0, you do not stop learning the parameter. Because, in the constant gain case, if r goes close to 0, these terms are dead.

These terms, if r goes close to 0, these terms become 0. But, this term continues to dominate and push \hat{W} and \hat{V} will continue to dominate and push \hat{W} and \hat{V} towards 0. So, whatever parameter value that was learned is lost. So, this is like an epsilon modification, because it is also scaled by norm r . So, that is the whole idea. So, this is like an epsilon modification. So, as I mentioned, we will not look into the passivity properties of the neural network right now.

(Refer Slide Time: 24:27)

Unfortunately, though dissipative, the closed-loop system is not SSP so, when disturbance $w_1(t)$ is nonzero, this does not yield boundedness of the internal states of the weight blocks (i.e., \hat{W}, \hat{V}) unless those blocks are observable, that is persistently exciting (PE). Unfortunately, this does not yield a convenient method for defining PE in a three-layer NN, as the two weight-tuning blocks are coupled, forming in fact a bilinear system. By contrast, PE conditions for the two-layer case $V = J$ (i.e., linear NN) are easy to deduce [18].

The next result shows why a PE condition is not needed with the modified weight update algorithm of Theorem 3.2; it is in the context of error system (30).

Theorem 4.2: The modified weight tuning algorithms (41), (42) make the map from $r(t)$ to $-\hat{W}^T(\delta - \delta^T V^T x)$, and the map from $r(t)$ to $-\hat{W}^T \delta^T V^T x$, both SSP maps.

Proof: The revised dynamics relative to \hat{W}, \hat{V} are given by

$$\dot{\hat{W}} = -F\delta r^T + F\delta^T V^T x r^T + \kappa F \|\epsilon\| \hat{W} \quad (49)$$

$$\dot{\hat{V}} = -Gx(\delta^T \hat{W} r)^T + \kappa G \|\epsilon\| \hat{V} \quad (50)$$

1) Selecting the nonnegative function

$$L = \frac{1}{2} \text{tr } \hat{W}^T F^{-1} \hat{W}$$

bounded in terms of the power delivered to each block. Then, boundedness of input-output signals assures state boundedness without any sort of observability requirement.

We define an NN as passive if, in the error formulation, it guarantees the passivity of the weight-tuning subsystems. Then, an extra PE condition is needed to guarantee boundedness of the weights [18]. We define an NN as robust if, in the error formulation, it guarantees the SSP of the weight-tuning subsystem. Then, no extra PE condition is needed for boundedness of the weights. Note that 1) SSP of the open-loop plant error system is needed in addition for tracking stability and 2) the NN passivity properties are dependent on the weight tuning algorithm used.

V. ILLUSTRATIVE DESIGN AND SIMULATION

A planar two-link arm used extensively in literature for illustration purposes appears in Fig. 4. The dynamics are given, for instance in [17]; no friction term was used in this example. This arm is simple enough to simulate conveniently, yet contains all the nonlinear terms arising in general n -link manipulators. The joint variable is $q = [q_1, q_2]^T$. We should like to illustrate the NN control scheme derived herein, which will require no knowledge of the dynamics, no structure which is needed for adaptive control.

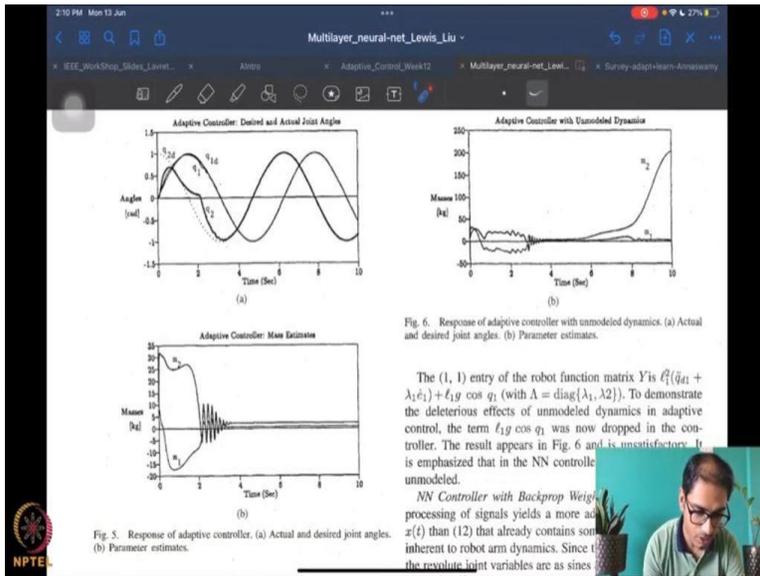
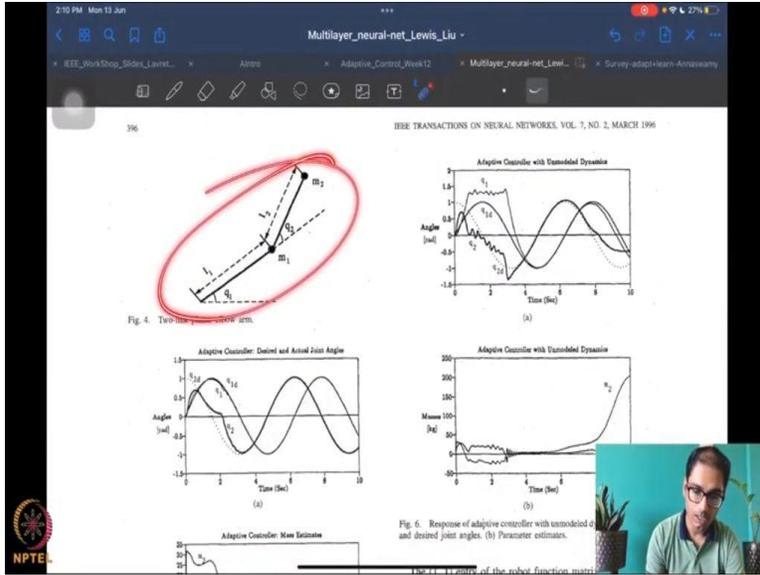


Fig. 5. Response of adaptive controller. (a) Actual and desired joint angles. (b) Parameter estimates.

Adaptive Controller—Baseline Design: For comparison, a standard adaptive controller is given by [40]

$$\tau = Y\hat{\psi} + K_v r \quad (51)$$

$$\dot{\hat{\psi}} = F Y^T r \quad (52)$$

with $F = F^T > 0$ a design parameter matrix, $Y(e, \dot{e}, q_d, \dot{q}_d, \ddot{q}_d)$ a fairly complicated matrix of robot functions that must be explicitly derived from the dynamics for each arm, and Ψ the vector of unknown parameters, in this case simply the link masses m_1, m_2 .

We took the arm parameters as $\ell_1 = \ell_2 = 1\text{ m}$, $m_1 = 0.8\text{ kg}$, $m_2 = 2.3\text{ kg}$, and selected $q_{1d}(t) = \sin t$, $q_{2d}(t) = \cos t$, $K_v = \text{diag}\{20, 20\}$, $F = \text{diag}\{10, 10\}$, $\Lambda = \text{diag}\{5, 5\}$. The response with this controller when $\hat{q}(0) = 0$, $\dot{\hat{q}}(0) = 0$, $\hat{m}_1(0) = 0$, $\hat{m}_2(0) = 0$ is shown in Fig. 5.

Note the good behavior, which obtains since there are only two unknown parameters, so that the single mode (e.g., two poles) of $q_d(t)$ guarantees PE [11].

Fig. 6. Response of adaptive controller with unmodeled dynamics. (a) Actual and desired joint angles. (b) Parameter estimates.

The (1, 1) entry of the robot function matrix Y is $\ell_1^2(\ddot{q}_{d1} + \lambda_1 \dot{e}_1) + \ell_1 g \cos q_1$ (with $\Lambda = \text{diag}\{\lambda_1, \lambda_2\}$). To demonstrate the deleterious effects of unmodeled dynamics in adaptive control, the term $\ell_1 g \cos q_1$ was now dropped in the controller. The result appears in Fig. 6 and is unsatisfactory. It is emphasized that in the NN controller all the dynamics are unmodeled.

NN Controller with Backprop Weight Tuning: Some preprocessing of signals yields a more advantageous choice for $x(t)$ than (12) that already contains some of the nonlinearities inherent to robot arm dynamics. Since the only occurrences of the revolute joint variables are as sines and cosines, the vector x can be taken as

$$x = [\zeta_1^T \zeta_2^T \cos(q)^T \sin(q)^T \ddot{q}^T \text{sgn}(\dot{q})^T]^T \quad (53)$$

where $\zeta_1 = \ddot{q}_d + \Lambda \dot{e}$, $\zeta_2 = \dot{q}_d + \Lambda e$ are as needed in the friction terms. The NN controller appears in Fig. 2.

The response of the controller (26) with backprop weight tuning (e.g., Theorem 3.1) appears in Fig. 7. The sigmoid activation functions were used, and 10 hidden-layer neurons. The values for $q_d(t)$, Λ , F , K_v were the same as before, and we selected $G = \text{diag}\{10, 10\}$. In this case the NN weights appear to remain bounded, though this cannot in general be guaranteed.

The choice of 10 hidden-layer neurons was made as follows. Three simulations were performed, using five, 10, then 15 hidden-layer neurons. It was observed that going from five-10 neurons significantly improved the performance, but going from 10-15 neurons made no perceptible difference.

Fig. 5. Response of adaptive controller. (a) Actual and desired joint angles. (b) Parameter estimates.

Adaptive Controller—Baseline Design: For comparison, a standard adaptive controller is given by [40]

$$\tau = Y\hat{\psi} + K_v r \quad (51)$$

$$\dot{\hat{\psi}} = F Y^T r \quad (52)$$

with $F = F^T > 0$ a design parameter matrix, $Y(e, \dot{e}, q_d, \dot{q}_d, \ddot{q}_d)$ a fairly complicated matrix of robot functions that must be explicitly derived from the dynamics for each arm, and Ψ the vector of unknown parameters, in this case simply the link masses m_1, m_2 .

We took the arm parameters as $\ell_1 = \ell_2 = 1\text{ m}$, $m_1 = 0.8\text{ kg}$, $m_2 = 2.3\text{ kg}$, and selected $q_{1d}(t) = \sin t$, $q_{2d}(t) = \cos t$, $K_v = \text{diag}\{20, 20\}$, $F = \text{diag}\{10, 10\}$, $\Lambda = \text{diag}\{5, 5\}$. The response with this controller when $\hat{q}(0) = 0$, $\dot{\hat{q}}(0) = 0$, $\hat{m}_1(0) = 0$, $\hat{m}_2(0) = 0$ is shown in Fig. 5.

Note the good behavior, which obtains since there are only two unknown parameters, so that the single mode (e.g., two poles) of $q_d(t)$ guarantees PE [11].

Fig. 6. Response of adaptive controller with unmodeled dynamics. (a) Actual and desired joint angles. (b) Parameter estimates.

The (1, 1) entry of the robot function matrix Y is $\ell_1^2(\ddot{q}_{d1} + \lambda_1 \dot{e}_1) + \ell_1 g \cos q_1$ (with $\Lambda = \text{diag}\{\lambda_1, \lambda_2\}$). To demonstrate the deleterious effects of unmodeled dynamics in adaptive control, the term $\ell_1 g \cos q_1$ was now dropped in the controller. The result appears in Fig. 6 and is unsatisfactory. It is emphasized that in the NN controller all the dynamics are unmodeled.

NN Controller with Backprop Weight Tuning: Some preprocessing of signals yields a more advantageous choice for $x(t)$ than (12) that already contains some of the nonlinearities inherent to robot arm dynamics. Since the only occurrences of the revolute joint variables are as sines and cosines, the vector x can be taken as

$$x = [\zeta_1^T \zeta_2^T \cos(q)^T \sin(q)^T \ddot{q}^T \text{sgn}(\dot{q})^T]^T \quad (53)$$

where $\zeta_1 = \ddot{q}_d + \Lambda \dot{e}$, $\zeta_2 = \dot{q}_d + \Lambda e$ and the signum function is needed in the friction terms. The NN controller appears in Fig. 2.

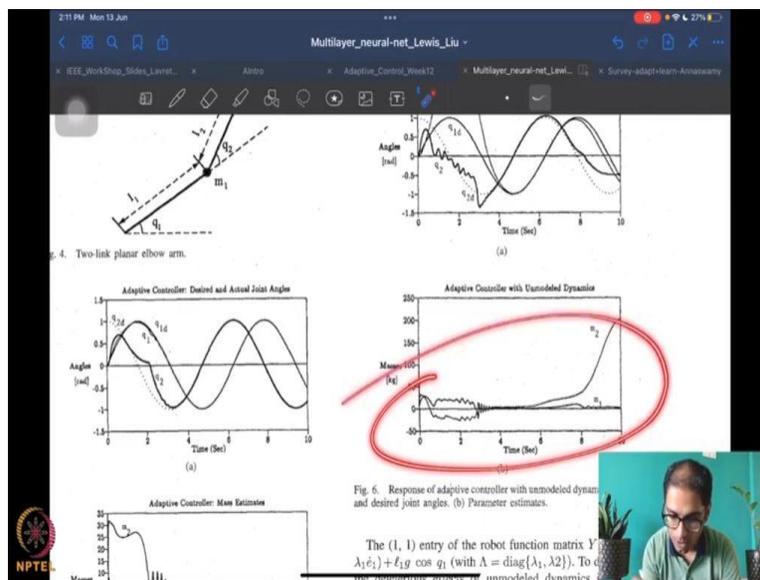
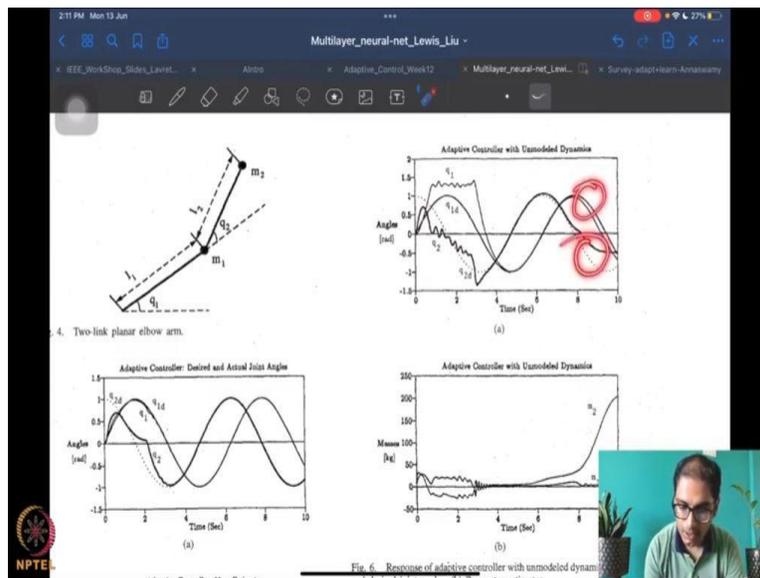
The response of the controller (26) with backprop weight tuning (e.g., Theorem 3.1) appears in Fig. 7. The sigmoid activation functions were used, and 10 hidden-layer neurons. The values for $q_d(t)$, Λ , F , K_v were the same as before, and we selected $G = \text{diag}\{10, 10\}$. In this case the NN weights appear to remain bounded, though this cannot in general be guaranteed.

The choice of 10 hidden-layer neurons was made as follows. Three simulations were performed, using five, 10, then 15 hidden-layer neurons. It was observed that going from five-10 neurons significantly improved the performance, but going from 10-15 neurons made no perceptible difference.

We want to just quickly look at like design. So, so this is like a standard adaptive control for a like a two-link robot, this is what planer two-link arm. So, this is like a standard setup for a lot of experiments in adaptive control. So, then you have the desired and actual joint angles here, so, you have q1, q1 desired and q2, q2 desired.

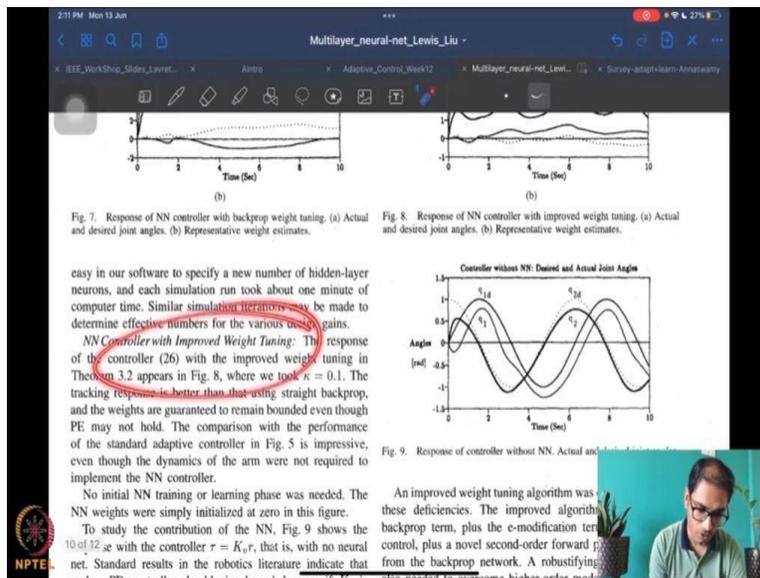
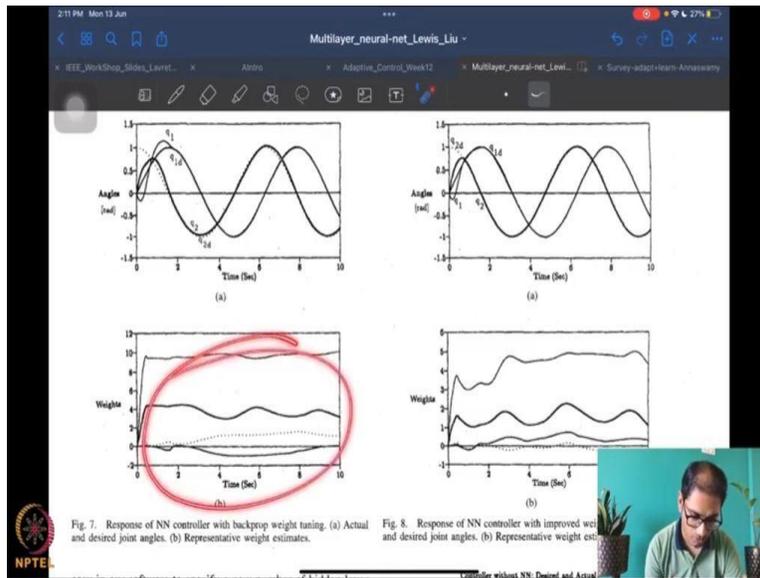
And then you also have mass estimates. The estimates of course do not converge you see, do not look like they converge to true values. But, whatever, I mean there is not what you guarantee in adaptive control anyway. So, note the good behavior which obtains, since there are only two unknown parameters. So that means, so you have nice persistence type results.

(Refer Slide Time: 25:22)



Now, if you add some unmodeled dynamics, so, some unmodeled terms, dynamical terms in the simulation you add, and you still implement an adaptive controller, not a neural network-based controller. You start to see significant deterioration in the performance. So q_1 is like this you can see, and q_1 desired is this, and similarly, q_2 is this. You know, this guy and the q_2 desired is this dotted line. So, you see that things sort of match for some time, but they do not match for a while, and so on and so forth. And the estimates of course, look rather bad, the estimates start to look rather bad.

(Refer Slide Time: 26:00)



2:12 PM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

Fig. 7. Response of NN controller with backprop weight tuning. (a) Actual and desired joint angles. (b) Representative weight estimates.

Fig. 8. Response of NN controller with improved weight tuning. (a) Actual and desired joint angles. (b) Representative weight estimates.

easy in our software to specify a new number of hidden-layer neurons, and each simulation run took about one minute of computer time. Similar simulation iterations may be made to determine effective numbers for the various design gains.

NN Controller with Improved Weight Tuning: The response of the controller (26) with the improved weight tuning in Theorem 3.2 appears in Fig. 8, where we took $\kappa = 0.1$. The tracking response is better than that using straight backprop, and the weights are guaranteed to remain bounded even though PE may not hold. The comparison with the performance of the standard adaptive controller in Fig. 5 is impressive, even though the dynamics of the arm were not required to implement the NN controller.

No initial NN training or learning phase was needed. The NN weights were simply initialized at zero in this figure.

To study the contribution of the NN, Fig. 9 shows the response with the controller $\tau = K_v r$, that is, with no neural net. Standard results in the robotics literature indicate that such a PD controller should give bounded errors if K_v is large enough. This is observed in the figure. It is very clear

Controller without NN: Desired and Actual Joint Angles

Fig. 9. Response of controller without NN. Actual and desired joint angles.

An improved weight tuning algorithm was derived to overcome these deficiencies. The improved algorithm contains a backprop term, plus the e-modification term for control, plus a novel second-order forward propagation from the backprop network. A robustifying controller was also needed to overcome higher-order modeling errors. The improved tuning algorithm makes the NN

Then, you implement like similarly, like a backdrop backpropagation-based weight tuning to this kind of a problem. So, and you basically look at this responsible neural network. And you see that you get nice better tracking, because q_1 converges to q_1 desired, even with this unknown unmodeled dynamics. And similarly, q_2 converges to q_2 desired.

The weights do not seem to do nice things still, and then, we add this improved weight tuning. And you see that you get slightly quicker tracking, better tracking performance. And mass convergence also lot of sort of looks similar. So, and then of course, we see the controller performance with and without without the neural networks. So, that that that is not matching at all, as you can see.

(Refer Slide Time: 27:07)

of the controller (26) with the improved weight tuning in Theorem 3.2 appears in Fig. 8, where we took $\kappa = 0.1$. The tracking response is better than that using straight backprop, and the weights are guaranteed to remain bounded even though PE may not hold. The comparison with the performance of the standard adaptive controller in Fig. 5 is impressive, even though the dynamics of the arm were not required to implement the NN controller.

No initial NN training or learning phase was needed. The NN weights were simply initialized at zero in this figure.

To study the contribution of the NN, Fig. 9 shows the response with the controller $\tau = K_v \dot{r}$, that is, with no neural net. Standard results in the robotics literature indicate that such a PD controller should give bounded errors if K_v is large enough. This is observed in the figure. It is very clear, however, that the addition of the NN makes a very significant improvement in the tracking performance.

VI. CONCLUSION

A multilayer (nonlinear) NN controller for a serial-link robot arm was developed. The NN controller has a structure derived from robot control theory passivity notions and offers guaranteed tracking behavior. Backpropagation tuning was shown to yield a passive NN, so that it only performs well under ideal conditions that require linear robot dynamics, no NN reconstruction error, and no robot unmodeled disturbances.

An improved weight tuning algorithm was derived to correct these deficiencies. The improved algorithm consists of a backprop term, plus the e-modification term from adaptive control, plus a novel second-order forward propagating wave from the backprop network. A robustifying control term is also needed to overcome higher-order modeling error terms. The improved tuning algorithm makes the NN strictly state passive, so that bounded weights are guaranteed in practical nonideal situations.

No NN off-line learning or training phase was needed; simply initializing the NN weights at zero made for fast convergence and bounded errors. Structured or partitioned NN's can be used to simplify the control as make for faster weight updates [21].

REFERENCES

[1] F. Albertini and E. D. Sontag, "For neural form," in *Proc. IEEE Conf. Decision Contr.*

So, this is sort of what the, so, so that is what this is sort of highlighted to study the contribution of the neural network. Figure-9 shows the response of the controller that with no neural network. And standard results in the robotics indicate PD controller should give bounded errors, if K_v is large enough. This is observed is however the addition of the neural networks has made a significant improvement in the tracking performance. So, this is the idea, so how they sort of illustrated value of this neural network type method is that they started with this nice baseline robotic model.

(Refer Slide Time: 27:48)

Fig. 6. Response of adaptive controller with unmodeled dynamics. (a) Actual and desired joint angles. (b) Parameter estimates.

The (1, 1) entry of the robot function matrix Y is $\dot{q}_1(\dot{q}_1 + \lambda_1 \dot{e}_1) + \dot{e}_1 g$ with $A = \text{diag}\{\lambda_1, \lambda_2\}$. To demonstrate the deleterious effects of unmodeled dynamics in adaptive control, the term $\dot{e}_1 g$ was now dropped in the controller. The result appears in Fig. 6 and is unsatisfactory. It is emphasized that in the NN controller all the dynamics are unmodeled.

NN Controller with Backprop Weight Tuning: Some pre-processing of signals yields a more advantageous choice for $x(t)$ than (12) that already contains some of the nonlinearities inherent to robot arm dynamics. Since the only occurrences of the revolute joint variables are as sines and cosines, x can be taken as

$$x = [\zeta_1^T \zeta_2^T \cos(q)^T \sin(q)^T \ddot{q}^T \text{sqn}(\dot{q})^T]^T \quad (51)$$

where $\zeta_1 = \ddot{q}_d + \lambda \dot{e}$, $\zeta_2 = \dot{q}_d + \lambda e$ and the signum function $\text{sqn}(\cdot)$ is needed in the friction terms. The NN controller

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 7, NO. 2, MARCH 1996

Adaptive Controller with Unmodeled Dynamics

Fig. 4. Two-link planar elbow arm.

Adaptive Controller: Desired and Actual Joint Angles

Adaptive Controller with Unmodeled Dynamics

Adaptive Controller—Baseline Design: For comparison, a standard adaptive controller is given by [40]

$$\tau = Y\hat{\psi} + K_v r \quad (51)$$

$$\dot{\hat{\psi}} = F Y^T r \quad (52)$$

with $F = F^T > 0$ a design parameter matrix, $Y(e, \dot{e}, q_d, \dot{q}_d)$ a fairly complicated matrix of robot functions that must be explicitly derived from the dynamics for each arm, and $\hat{\psi}$ the vector of unknown parameters, in this case simply the link masses m_1, m_2 .

We took the arm parameters as $\ell_1 = \ell_2 = 1$ m, $m_1 = 0.8$ kg, $m_2 = 2.3$ kg, and selected $q_{1d}(t) = \sin t$, $q_{2d}(t) = \cos t$, $K_v = \text{diag}\{20, 20\}$, $F = \text{diag}\{10, 10\}$, $\Lambda = \text{diag}\{5, 5\}$. The response with this controller when $q(0) = 0$, $\dot{q}(0) = 0$, $\hat{m}_1(0) = 0$, $\hat{m}_2(0) = 0$ is shown in Fig. 5.

Note the good behavior, which obtains since there are only two unknown parameters, so that the single mode (e.g., two poles) of $q_d(t)$ guarantees PE [11].

Adaptive Controller with Unmodeled Dynamics

Fig. 5. Response of adaptive controller. (a) Actual and desired joint angles. (b) Parameter estimates.

...the result appears in Fig. 6 and is unsatisfactory. It is emphasized that in the NN controller all the dynamics are unmodeled.

NN Controller with Backprop Weight Tuning: Some preprocessing of signals yields a more advantageous choice for $x(t)$ than (12) that already contains some of the nonlinearities inherent to robot arm dynamics. Since the only occurrences of the revolute joint variables are as sines and cosines, the vector x can be taken as

$$x = [\zeta_1^T \zeta_2^T \cos(q)^T \sin(q)^T \dot{q}^T \text{sgn}(\dot{q})^T]^T \quad (53)$$

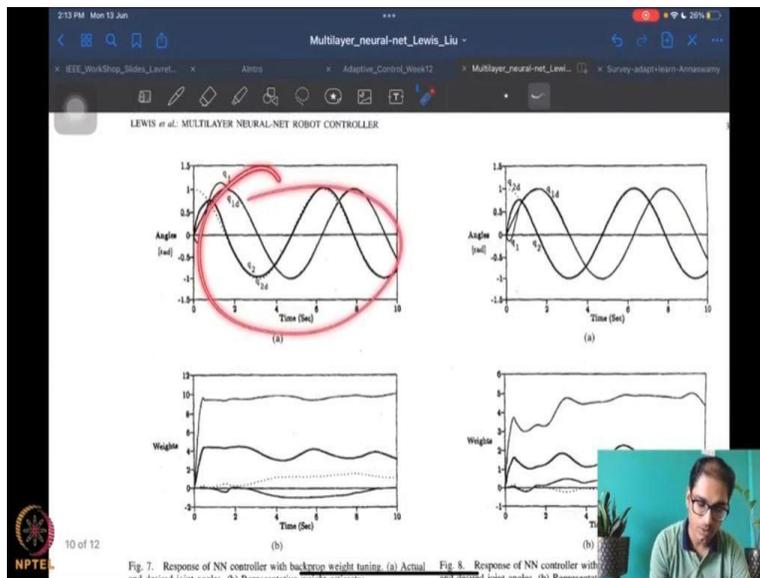
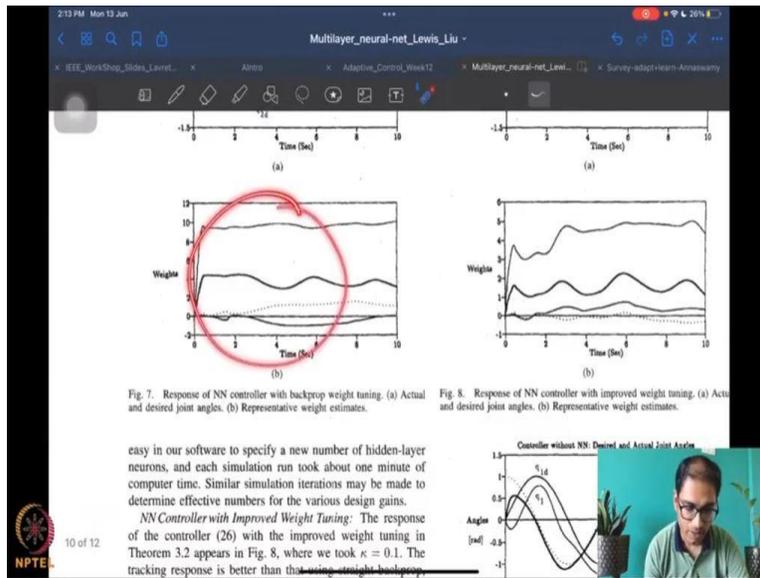
where $\zeta_1 = \ddot{q}_d + \Lambda \dot{e}$, $\zeta_2 = \dot{q}_d + \Lambda e$ and the signum function is needed in the friction terms. The NN controller appears in Fig. 2.

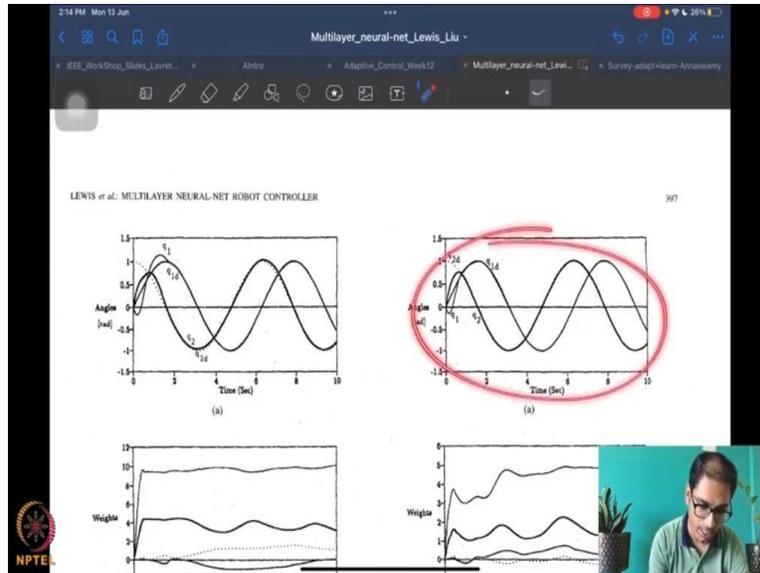
The response of the controller (26) (with $v(t) = 0$) with backprop weight tuning (e.g., Theorem 2.1) appears in Fig. 7. The sigmoid activation functions were used, and 10 hidden-layer neurons. The values for $q_d(t)$, Λ , F , K_v were the same as before, and we selected $\Lambda = \text{diag}\{10, 10\}$. In this case the NN weights appear to remain bounded, though this cannot in general be guaranteed.

The choice of 10 hidden-layer neurons was made. Three simulations were performed, using five, hidden-layer neurons. It was observed that going neurons significantly improved the performance from 10-15 neurons made no perceptible improvement.

And then they, they added some terms into this the system matrices like the M, Vm and G and F and all that, so they added just one term. So, so, so basically, they see that there is a deteriorated performance here, if you do not have a neural network type controller. Then, so basically is this is like, you see the performance here using the backpropagation is the idealized weight tuning also, when you assume some disturbance terms are exactly 0, well they are not. And then all these quantities are still the same, which is this, there is an additional term.

(Refer Slide Time: 28:30)





And and here, you see that, you sort of have relatively better results in terms of the tracking performance. A weight convergence anyway is not completely guaranteed even here, unless there is some kind of a persistence, but you do have very nice tracking performance. And then of course, you also show improved tracking and quicker tracking by using the modified weight tuning method. And that is what is sort of illustrated here. So, in conclusion, I really hope that you sort of got a good feel for how adaptive control is very closely connected to deep learning.

And the only difference here in adaptive control is that we look to prove stability, which, and therefore analytically, especially. And which may make designing all sorts of arbitrary tuning laws difficult. But, that is essentially what we do in adaptive control. Also, we designed tuning laws and we prove stability with that. And in this case, stability would mean some kind of good function function approximation, and therefore good tracking. So, the other thing is, because we were doing this adaptive control-based learning, we also were doing everything online.

Of course, you can use the same laws and do things offline also, because we saw that this backpropagation is essentially the best tuning that we did in adaptive control law, with an adaptive control law is same as what we have for the standard case, like like, your offline learning type of situation.

So, we are sort of at the backend of our course, and I really hope that this last week did give you a very good idea of how relevant adaptive control is even today, in the sense that it has very very close ties to deep learning. And hence to other forms of learning, like reinforcement learning and

classification problems, which are still connected to some kind of parameter estimation, and parameter learning.

I really hope all of you enjoyed the course as much as I enjoyed delivering it. And I am always open for feedback, comments, applications, cool new results, cool cool new experiments. So, I hope you will stay in touch through this course and learn more about the avenues that nonlinear adaptive control affords us. Thank you. And I am very glad for all of you, for all of your learning, and the fact that you listened to what I had to say for these learnings. Thanks a lot.