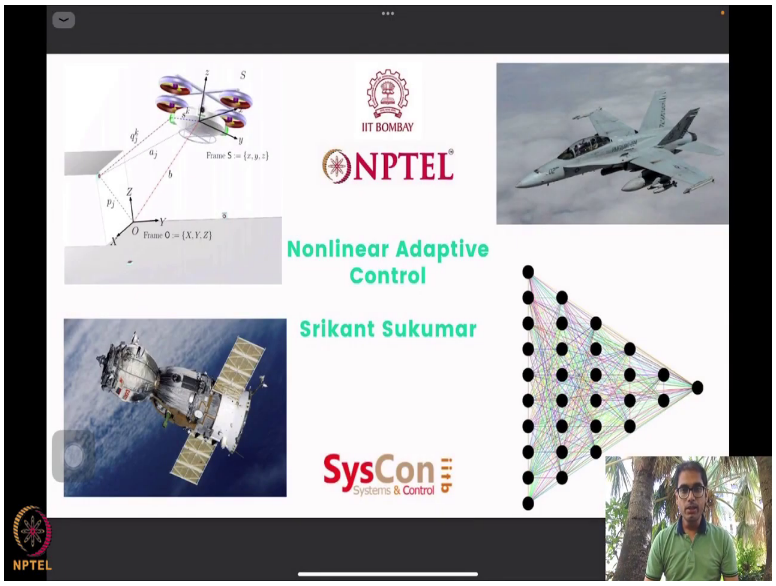
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 12
Lecture No: 68
Real Time Neural Network Based Control of a Robotic Manipulator (Part 2)

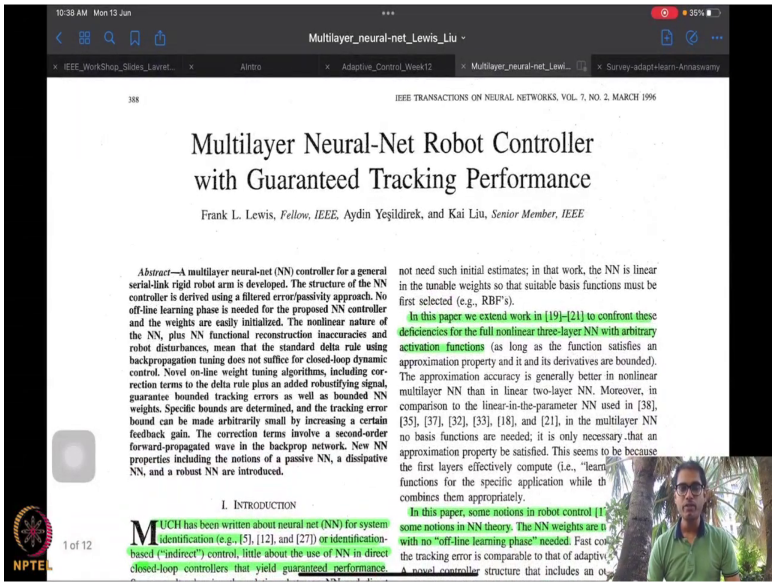
(Refer Slide Time: 00:17)



Hello, everyone. Welcome to yet another session of our NPTEL on Nonlinear and Adaptive Control. I am Srikant Sukumar from Systems and Control, IIT Bombay. So, we are in this very last week of this course on nonlinear adaptive control, and we are talking about some exciting things, and I hope you found the course, in general, as a rather interesting course.

So, right now we are going into slightly more, say, advanced and research topics and we have been discussing about learning. So, we started the week by trying to create a chronology of how things evolved in adaptive control and parallelly, in stochastic and discrete systems in the form of self-tuning regulators, and also another parallel in the form of learning classification kind of problems. So, I do hope that you sort of are relatively convinced that whatever algorithms that you have learned through the curriculum will help you drive autonomous systems such as what you see in the background.

(Refer Slide Time: 01:36)


10:38 AM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

IEEE_WorkShop_Slides_Lavret... | AIntro | Adaptive_Control_Week12 | Multilayer_neural-net_Lewi... | Survey-adapt+learn-Annaswamy

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

Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance

Frank L. Lewis, *Fellow, IEEE,* Aydin Yeşildirek, and Kai Liu, *Senior Member, IEEE*

***Abstract*—A multilayer neural-net (NN) controller for a general serial-link rigid robot arm is developed. The structure of the NN controller is derived using a filtered error/passivity approach. No off-line learning phase is needed for the proposed NN controller and the weights are easily initialized. The nonlinear nature of the NN, plus NN functional reconstruction inaccuracies and robot disturbances, mean that the standard delta rule using backpropagation tuning does not suffice for closed-loop dynamic control. Novel on-line weight tuning algorithms, including correction terms to the delta rule plus an added robustifying signal, guarantee bounded tracking errors as well as bounded NN weights. Specific bounds are determined, and the tracking error bound can be made arbitrarily small by increasing a certain feedback gain. The correction terms involve a second-order forward-propagated wave in the backprop network. New NN properties including the notions of a passive NN, a dissipative NN, and a robust NN are introduced.**

I. Introduction

MUCH has been written about neural net (NN) for system identification (e.g., [5], [12], and [27]) or identification-based ("indirect") control, little about the use of NN in direct closed-loop controllers that yield guaranteed performance.

not need such initial estimates; in that work, the NN is linear in the tunable weights so that suitable basis functions must be first selected (e.g., RBF's).

In this paper we extend work in [19]–[21] to confront these deficiencies for the full nonlinear three-layer NN with arbitrary activation functions (as long as the function satisfies an approximation property and it and its derivatives are bounded). The approximation accuracy is generally better in nonlinear multilayer NN than in linear two-layer NN. Moreover, in comparison to the linear-in-the-parameter NN used in [38], [35], [37], [32], [33], [18], and [21], in the multilayer NN no basis functions are needed; it is only necessary that an approximation property be satisfied. This seems to be because the first layers effectively compute (i.e., "learn functions for the specific application while th combines them appropriately.

In this paper, some notions in robot control [1 some notions in NN theory. The NN weights are t with no "off-line learning phase" needed. Fast co the tracking error is comparable to that of adaptiv A novel controller structure that includes an o

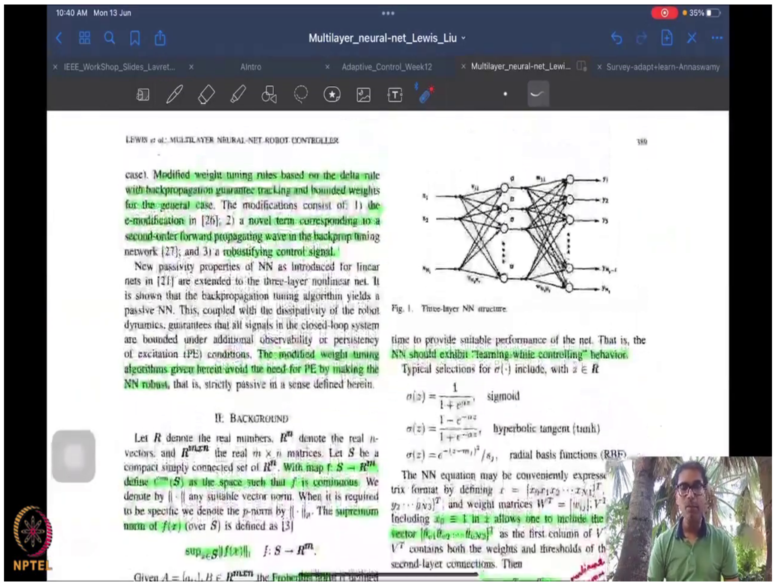
1 of 12

So, in this paper that we were looking at we were looking at a very specific learning setting, which is now a very popular setting and that is the one on deep learning. So, deep learning is essentially learning algorithms that make use of deep neural networks, that is multilayer neural networks.

And that is the sort of application that we are looking at here. One of the, sort of differences from what you typically are used to, in learning algorithms is that here we have everything to be online, that is you are doing learning also in real time. So, we of course, established a parallel of deep learning or neural networks with parameter learning.

So, essentially that is what it is because you have weights and then you are trying to learn some weights of the different layers of the neural network. So, we did establish that target. And the difference here lies in that we are trying to do the learning online, that is real time learning and control simultaneously. So, we talked a little bit about where the literature is, in the beginning, and we also sort of tried to understand that standard learning rules may not do a great job.

(Refer Slide Time: 03:10)



10:40 AM Mon 13 Jun
35%
Multilayer_neural-net_Lewis_Liu
IEEE_WorkShop_Slides_Lavret...
AIntro
Adaptive_Control_Week12
Multilayer_neural-net_Lewi...
Survey-adapt+learn-Annaswamy
NPTEL

LEWIS et al.: MULTILAYER NEURAL-NET ROBOT CONTROLLER 389

case). Modified weight tuning rules based on the delta rule with backpropagation guarantee tracking and bounded weights for the general case. The modifications consist of: 1) the e-modification in [26]; 2) a novel term corresponding to a second-order forward propagating wave in the backprop tuning network [27]; and 3) a robustifying control signal.

New passivity properties of NN as introduced for linear nets in [21] are extended to the three-layer nonlinear net. It is shown that the backpropagation tuning algorithm yields a passive NN. This, coupled with the dissipativity of the robot dynamics, guarantees that all signals in the closed-loop system are bounded under additional observability or persistency of excitation (PE) conditions. The modified weight tuning algorithms given herein avoid the need for PE by making the NN robust, that is, strictly passive in a sense defined herein.

II. BACKGROUND

Let R denote the real numbers, R^n denote the real n-vectors, and R^{mxn} the real $m \times n$ matrices. Let S be a compact simply connected set of R^n. With map f: $S \to R^m$, define $C^m(S)$ as the space such that f is continuous. We denote by $\|\cdot\|$ any suitable vector norm. When it is required to be specific we denote the p-norm by $\|\cdot\|_p$. The supremum norm of $f(x)$ (over S) is defined as [3]

$$\sup_{x\in S}\|f(x)\|, \quad f: S \to R^m.$$

Given $A = [a_{ij}], B \in R^{mxn}$ the Frobenius norm is defined

Fig. 1. Three-layer NN structure.

time to provide suitable performance of the net. That is, the NN should exhibit "learning-while controlling" behavior.

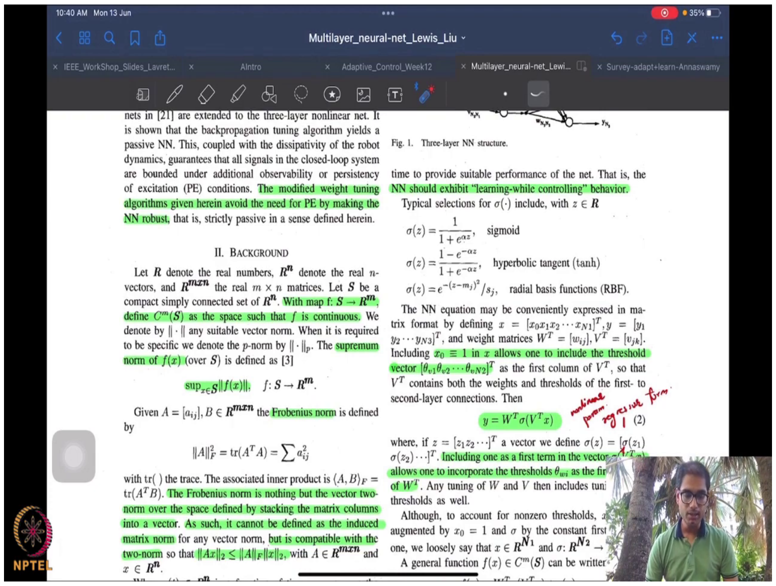
Typical selections for $\sigma(\cdot)$ include, with $z \in R$

$$\sigma(z) = \frac{1}{1+e^{\alpha z}}, \quad \text{sigmoid}$$

$$\sigma(z) = \frac{1-e^{-\alpha z}}{1+e^{-\alpha z}}, \quad \text{hyperbolic tangent (tanh)}$$

$$\sigma(z) = e^{-(z-m_j)^2}/s_j, \quad \text{radial basis functions (RBF).}$$

The NN equation may be conveniently express trix format by defining $x = [x_0x_1x_2\cdots x_{N1}]^T$, $y_2\cdots y_{N3}]^T$, and weight matrices $W^T = [w_{ij}], V^T$ Including $x_0 \equiv 1$ in x allows one to include the vector $[\theta_{v1}\theta_{v2}\cdots\theta_{vN2}]^T$ as the first column of V V^T contains both the weights and thresholds of th second-layer connections. Then



10:40 AM Mon 13 Jun
35%
Multilayer_neural-net_Lewis_Liu
IEEE_WorkShop_Slides_Lavret...
AIntro
Adaptive_Control_Week12
Multilayer_neural-net_Lewi...
Survey-adapt+learn-Annaswamy
NPTEL

nets in [21] are extended to the three-layer nonlinear net. It is shown that the backpropagation tuning algorithm yields a passive NN. This, coupled with the dissipativity of the robot dynamics, guarantees that all signals in the closed-loop system are bounded under additional observability or persistency of excitation (PE) conditions. The modified weight tuning algorithms given herein avoid the need for PE by making the NN robust, that is, strictly passive in a sense defined herein.

II. BACKGROUND

Let R denote the real numbers, R^n denote the real n-vectors, and R^{mxn} the real $m \times n$ matrices. Let S be a compact simply connected set of R^n. With map f: $S \to R^m$, define $C^m(S)$ as the space such that f is continuous. We denote by $\|\cdot\|$ any suitable vector norm. When it is required to be specific we denote the p-norm by $\|\cdot\|_p$. The supremum norm of $f(x)$ (over S) is defined as [3]

$$\sup_{x\in S}\|f(x)\|, \quad f: S \to R^m.$$

Given $A = [a_{ij}], B \in R^{mxn}$ the Frobenius norm is defined by

$$\|A\|_F^2 = \mathrm{tr}(A^TA) = \sum a_{ij}^2$$

with tr( ) the trace. The associated inner product is $\langle A,B\rangle_F = \mathrm{tr}(A^TB)$. The Frobenius norm is nothing but the vector two-norm over the space defined by stacking the matrix columns into a vector. As such, it cannot be defined as the induced matrix norm for any vector norm, but is compatible with the two-norm so that $\|Ax\|_2 \le \|A\|_F\|x\|_2$, with $A \in R^{mxn}$ and $x \in R^n$.

Fig. 1. Three-layer NN structure.

time to provide suitable performance of the net. That is, the NN should exhibit "learning-while controlling" behavior.

Typical selections for $\sigma(\cdot)$ include, with $z \in R$

$$\sigma(z) = \frac{1}{1+e^{\alpha z}}, \quad \text{sigmoid}$$

$$\sigma(z) = \frac{1-e^{-\alpha z}}{1+e^{-\alpha z}}, \quad \text{hyperbolic tangent (tanh)}$$

$$\sigma(z) = e^{-(z-m_j)^2}/s_j, \quad \text{radial basis functions (RBF).}$$

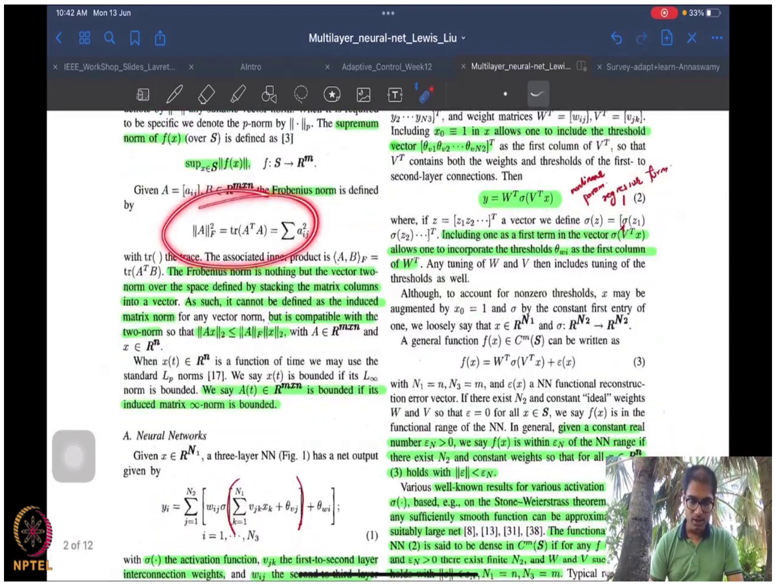
The NN equation may be conveniently expressed in matrix format by defining $x = [x_0x_1x_2\cdots x_{N1}]^T$, $y = [y_1 y_2\cdots y_{N3}]^T$, and weight matrices $W^T = [w_{ij}]$, $V^T = [v_{jk}]$. Including $x_0 \equiv 1$ in x allows one to include the threshold vector $[\theta_{v1}\theta_{v2}\cdots\theta_{vN2}]^T$ as the first column of V^T, so that V^T contains both the weights and thresholds of the first- to second-layer connections. Then

$$y = W^T\sigma(V^Tx) \qquad \text{nonlinear param.} \quad \text{regressive form} \qquad (2)$$

where, if $z = [z_1z_2\cdots]^T$ a vector we define $\sigma(z) = [\sigma(z_1)\ \sigma(z_2)\cdots]^T$. Including one as a first term in the vector allows one to incorporate the thresholds θ_{wi} as the fir of W^T. Any tuning of W and V then includes tuni thresholds as well.

Although, to account for nonzero thresholds, augmented by $x_0 = 1$ and σ by the constant first one, we loosely say that $x \in R^{N_1}$ and $\sigma: R^{N_2} \to$

A general function $f(x) \in C^m(S)$ can be writter


10:42 AM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

IEEE_WorkShop_Slides_Lavret... AIntro Adaptive_Control_Week12 Multilayer_neural-net_Lewi... Survey-adapt+learn-Annaswamy

to be specific we denote the p-norm by $\|\cdot\|_p$. The supremum norm of $f(x)$ (over S) is defined as [3]

$$\sup_{x\in S}\|f(x)\|, \quad f: S \to R^m.$$

Given $A = [a_{ij}]$, $B \in R^{mxn}$ the Frobenius norm is defined by

$$\|A\|_F^2 = \mathrm{tr}(A^TA) = \sum a_{ij}^2$$

with tr( ) the trace. The associated inner product is $\langle A,B\rangle_F = \mathrm{tr}(A^TB)$. The Frobenius norm is nothing but the vector two-norm over the space defined by stacking the matrix columns into a vector. As such, it cannot be defined as the induced matrix norm for any vector norm, but is compatible with the two-norm so that $\|Ax\|_2 \le \|A\|_F\|x\|_2$, with $A \in R^{mxn}$ and $x \in R^n$.

When $x(t) \in R^n$ is a function of time we may use the standard L_p norms [17]. We say $x(t)$ is bounded if its L_∞ norm is bounded. We say $A(t) \in R^{mxn}$ is bounded if its induced matrix ∞-norm is bounded.

A. Neural Networks

Given $x \in R^{N_1}$, a three-layer NN (Fig. 1) has a net output given by

$$y_i = \sum_{j=1}^{N_2}\left[w_{ij}\sigma\left(\sum_{k=1}^{N_1} v_{jk}x_k + \theta_{vj}\right) + \theta_{wi}\right]; \quad i = 1, \cdots, N_3 \qquad (1)$$

with $\sigma(\cdot)$ the activation function, v_{jk} the first-to-second layer interconnection weights, and w_{ij} the se~~cond-to-third layer~~

2 of 12

$y_2\cdots y_{N3}]^T$, and weight matrices $W^T = [w_{ij}], V^T = [v_{jk}]$. Including $x_0 \equiv 1$ in x allows one to include the threshold vector $[\theta_{v1}\theta_{v2}\cdots\theta_{vN2}]^T$ as the first column of V^T, so that V^T contains both the weights and thresholds of the first- to second-layer connections. Then

$$y = W^T\sigma(V^Tx) \qquad (2)$$

nonlinear param. regressive form

where, if $z = [z_1z_2\cdots]^T$ a vector we define $\sigma(z) = [\sigma(z_1)\ \sigma(z_2)\cdots]^T$. Including one as a first term in the vector $\sigma(V^Tx)$ allows one to incorporate the thresholds θ_{wi} as the first column of W^T. Any tuning of W and V then includes tuning of the thresholds as well.

Although, to account for nonzero thresholds, x may be augmented by $x_0 = 1$ and σ by the constant first entry of one, we loosely say that $x \in R^{N_1}$ and $\sigma: R^{N_2} \to R^{N_2}$.

A general function $f(x) \in C^m(S)$ can be written as

$$f(x) = W^T\sigma(V^Tx) + \varepsilon(x) \qquad (3)$$

with $N_1 = n, N_3 = m$, and $\varepsilon(x)$ a NN functional reconstruction error vector. If there exist N_2 and constant "ideal" weights W and V so that $\varepsilon = 0$ for all $x \in S$, we say $f(x)$ is in the functional range of the NN. In general, given a constant real number $\varepsilon_N > 0$, we say $f(x)$ is within ε_N of the NN range if there exist N_2 and constant weights so that for all
(3) holds with $\|\varepsilon\| < \varepsilon_N$.

Various well-known results for various activation $\sigma(\cdot)$, based, e.g., on the Stone–Weierstrass theorem any sufficiently smooth function can be approxima suitably large net [8], [13], [31], [38]. The functiona NN (2) is said to be dense in $C^m(S)$ if for any f and $\varepsilon_N > 0$ there exist finite N_2, and W and V suc ~~holds with $\|\varepsilon\| < \varepsilon_N$~~, $N_1 = n, N_3 = m$. Typical r

And so, the, this article talks about some modifications to the learning rules. Now, we also then of course discussed a little bit of the background. Let us see, I want to see, where I have marked these, it does not look like I marked the lecture. Just a second. I think the first lecture, of course started from the, this paper here. So, I think that is fine, that is marked in this paper.

So, let us see where we were. I need to find it. Just give me a second. Not here either. There is the STR. I just want to find where I marked, lecture number 12.2. Yes, that is here in the discrete and deterministic systems. And I believe we started, I am going to mark it here, as lecture, and let us see. Now, this is fine. So, this is, this is all of lecture 12.2, I believe. Yeah, I believe this is all of lecture 12.2. Yes, yes, this is fine. So, we do have all the lectures marked. My apologies. So, then we started looking a little bit at the background of what is the problem set up, pretty standard in most articles that you read.

Here, we talk about the type of norms on matrices and signals and vectors, then we do the most important definition, which is that of how the neural network looks which is with one input and one output layer, or, and then one hidden layer in between. So, this is how you have the three-layer neural network structure. So, you have the inputs going in and then they are scaled by some weights and offset. Then you get a summation over them, and that is the second layer where you have this activation function. And then you have sort of the connection with the third layer which is via this weight w i j and the theta w i offsets.

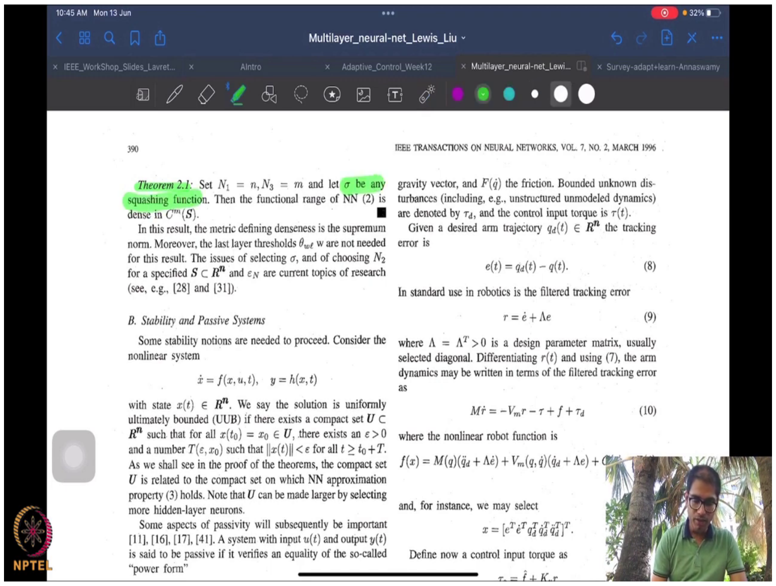
So, the aim is of course, for the neural network to identify all these quantities. And if you do, then you have a very clear relation between functional relationship between x and y, which is the input and the output. Do not think of these inputs and outputs as a control et cetera. This is a more general setting.

So, we also saw that there are different possible choices of these activation functions, they can be sigmoidal, hyperbolic, tangent or radial basis functions. And with some, sort of flexibility in terms of how we choose, how we define our x and V and W and so on, we can account for both the weights and the offsets in this sort of a structure, which is this nonlinear regressor parameter form.

Why do I say it is nonlinear, is because you have this W and also a V here. So, this is what makes it nonlinear because both W and V here are unknowns. So, usually, you would probably just have something like this, in a regressor parameter form. But here you have, first two different unknowns and then you also have this activation function which is also potentially nonlinear.

So, then of course, we talked about what it means for something, some function could be in the functional range of a neural network and that was defined using this kind of an expression, that if you have the neural network, it gives you something like this, and then if you have a small epsilon offset from the true value of the function, then it is somewhere in the functional range. And there are some nice results for specific cases of sigma, we, that, we were talking about these activation functions.

(Refer Slide Time: 08:15)


Multilayer_neural-net_Lewis_Liu

Theorem 2.1: Set $N_1 = n, N_3 = m$ and let σ be any squashing function. Then the functional range of NN (2) is dense in $C^m(\boldsymbol{S})$. ■

In this result, the metric defining denseness is the supremum norm. Moreover, the last layer thresholds $\theta_{w\ell}$ w are not needed for this result. The issues of selecting σ, and of choosing N_2 for a specified $\boldsymbol{S} \subset \boldsymbol{R}^n$ and ε_N are current topics of research (see, e.g., [28] and [31]).

B. Stability and Passive Systems

Some stability notions are needed to proceed. Consider the nonlinear system

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

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

gravity vector, and $F(\dot{q})$ the friction. Bounded unknown disturbances (including, e.g., unstructured unmodeled dynamics) are denoted by τ_d, and the control input torque is $\tau(t)$.

Given a desired arm trajectory $q_d(t) \in \boldsymbol{R}^n$ the tracking error is

$$e(t) = q_d(t) - q(t). \quad (8)$$

In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \quad (9)$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

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

where the nonlinear robot function is

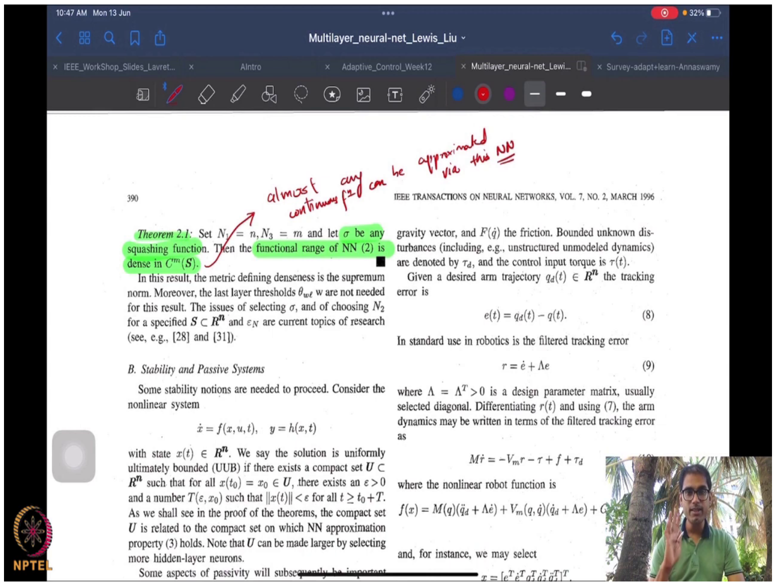
$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \Lambda e) + G$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T.$$

Define now a control input torque as

$$\tau_0 = \hat{f} + K_v r$$


Multilayer_neural-net_Lewis_Liu

almost any continuous f^n can be approximated via this $\underline{\underline{NN}}$

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

Theorem 2.1: Set $N_1 = n, N_3 = m$ and let σ be any squashing function. Then the functional range of NN (2) is dense in $C^m(\boldsymbol{S})$. ■

In this result, the metric defining denseness is the supremum norm. Moreover, the last layer thresholds $\theta_{w\ell}$ w are not needed for this result. The issues of selecting σ, and of choosing N_2 for a specified $\boldsymbol{S} \subset \boldsymbol{R}^n$ and ε_N are current topics of research (see, e.g., [28] and [31]).

B. Stability and Passive Systems

Some stability notions are needed to proceed. Consider the nonlinear system

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

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

Some aspects of passivity will subsequently be important

gravity vector, and $F(\dot{q})$ the friction. Bounded unknown disturbances (including, e.g., unstructured unmodeled dynamics) are denoted by τ_d, and the control input torque is $\tau(t)$.

Given a desired arm trajectory $q_d(t) \in \boldsymbol{R}^n$ the tracking error is

$$e(t) = q_d(t) - q(t). \quad (8)$$

In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \quad (9)$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

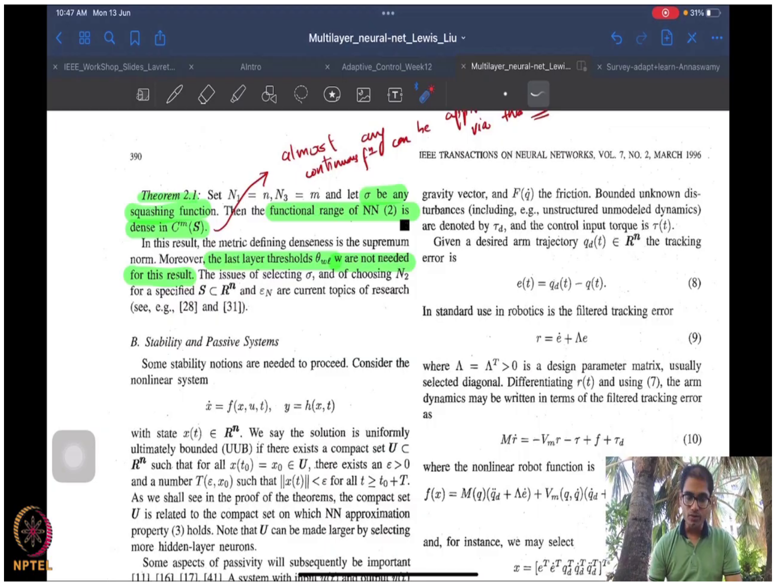
$$M\dot{r} = -V_m r - \tau + f + \tau_d$$

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \Lambda e) + G$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T.$$



Multilayer_neural-net_Lewis_Liu

almost any continuous f^n can be appr via the

Theorem 2.1: Set $N_1 = n, N_3 = m$ and let σ be any squashing function. Then the functional range of NN (2) is dense in $C^m(S)$. ∎

In this result, the metric defining denseness is the supremum norm. Moreover, the last layer thresholds $\theta_{w\ell}$ w are not needed for this result. The issues of selecting σ, and of choosing N_2 for a specified $S \subset R^n$ and ε_N are current topics of research (see, e.g., [28] and [31]).

B. Stability and Passive Systems

Some stability notions are needed to proceed. Consider the nonlinear system

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

with state $x(t) \in R^n$. We say the solution is uniformly ultimately bounded (UUB) if there exists a compact set $U \subset R^n$ such that for all $x(t_0) = x_0 \in U$, there exists an $\varepsilon > 0$ and a number $T(\varepsilon, x_0)$ such that $\|x(t)\| < \varepsilon$ 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.

gravity vector, and $F(\dot{q})$ the friction. Bounded unknown disturbances (including, e.g., unstructured unmodeled dynamics) are denoted by τ_d, and the control input torque is $\tau(t)$.

Given a desired arm trajectory $q_d(t) \in R^n$ the tracking error is

$$e(t) = q_d(t) - q(t). \tag{8}$$

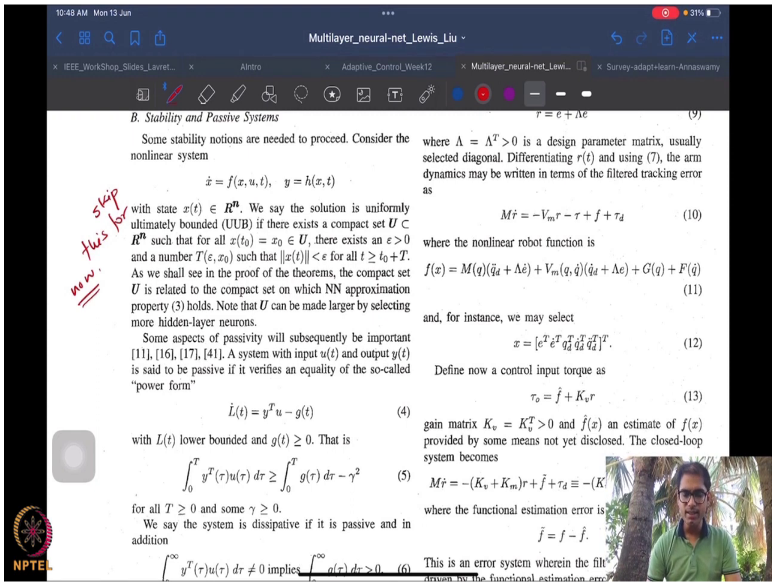
In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \tag{9}$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

$$M\dot{r} = -V_m r - \tau + f + \tau_d \tag{10}$$

where the nonlinear robot function is



Multilayer_neural-net_Lewis_Liu

B. Stability and Passive Systems

Some stability notions are needed to proceed. Consider the nonlinear system

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

skip this for now.

with state $x(t) \in R^n$. We say the solution is uniformly ultimately bounded (UUB) if there exists a compact set $U \subset R^n$ such that for all $x(t_0) = x_0 \in U$, there exists an $\varepsilon > 0$ and a number $T(\varepsilon, x_0)$ such that $\|x(t)\| < \varepsilon$ 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"

$$\dot{L}(t) = y^T u - g(t) \tag{4}$$

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

$$\int_0^T y^T(\tau)u(\tau)\,d\tau \geq \int_0^T g(\tau)\,d\tau - \gamma^2 \tag{5}$$

for all $T \geq 0$ and some $\gamma \geq 0$.

We say the system is dissipative if it is passive and in addition

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

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

$$M\dot{r} = -V_m r - \tau + f + \tau_d \tag{10}$$

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \tag{11}$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T. \tag{12}$$

Define now a control input torque as

$$\tau_o = \hat{f} + K_v r \tag{13}$$

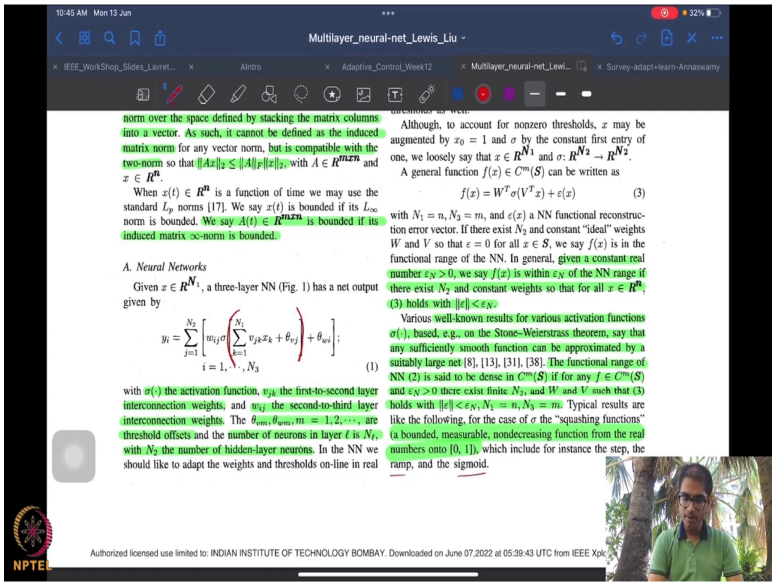
gain matrix $K_v = K_v^T > 0$ and $\hat{f}(x)$ an estimate of $f(x)$ provided by some means not yet disclosed. The closed-loop system becomes

$$M\dot{r} = -(K_v + K_m)r + \tilde{f} + \tau_d \equiv -(K$$

where the functional estimation error is

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

This is an error system wherein the filt



norm over the space defined by stacking the matrix columns into a vector. As such, it cannot be defined as the induced matrix norm for any vector norm, but is compatible with the two-norm so that $\|Ax\|_2 \le \|A\|_F \|x\|_2$, with $A \in \boldsymbol{R}^{mxn}$ and $x \in \boldsymbol{R}^n$.

When $x(t) \in \boldsymbol{R}^n$ is a function of time we may use the standard L_p norms [17]. We say $x(t)$ is bounded if its L_∞ norm is bounded. We say $A(t) \in \boldsymbol{R}^{mxn}$ is bounded if its induced matrix ∞-norm is bounded.

A. Neural Networks

Given $x \in \boldsymbol{R}^{N_1}$, a three-layer NN (Fig. 1) has a net output given by

$$y_i = \sum_{j=1}^{N_2} \left[w_{ij} \sigma \left(\sum_{k=1}^{N_1} v_{jk} x_k + \theta_{vj} \right) + \theta_{wi} \right]; \quad i = 1, \cdots, N_3 \tag{1}$$

with $\sigma(\cdot)$ the activation function, v_{jk} the first-to-second layer interconnection weights, and w_{ij} the second-to-third layer interconnection weights. The $\theta_{vm}, \theta_{wm}, m = 1, 2, \cdots,$ are threshold offsets and the number of neurons in layer ℓ is N_ℓ, with N_2 the number of hidden-layer neurons. In the NN we should like to adapt the weights and thresholds on-line in real

Although, to account for nonzero thresholds, x may be augmented by $x_0 = 1$ and σ by the constant first entry of one, we loosely say that $x \in \boldsymbol{R}^{N_1}$ and $\sigma: \boldsymbol{R}^{N_2} \to \boldsymbol{R}^{N_2}$.

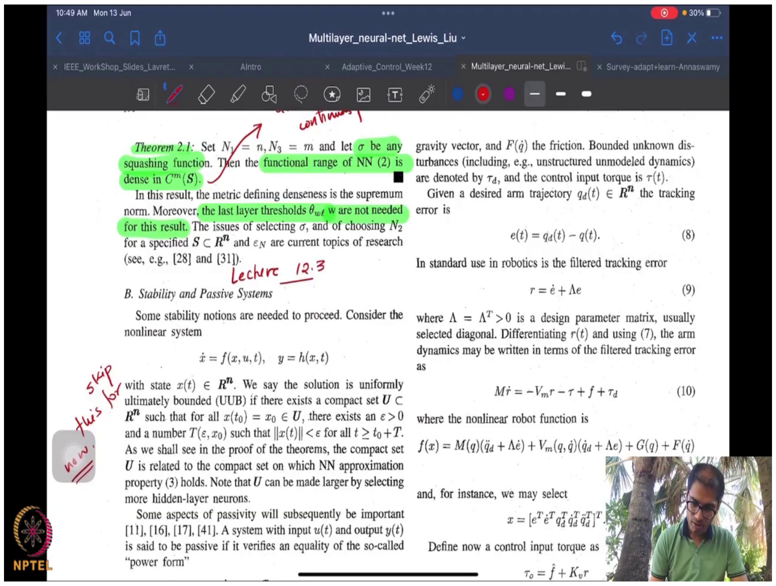
A general function $f(x) \in C^m(\boldsymbol{S})$ can be written as

$$f(x) = W^T \sigma(V^T x) + \varepsilon(x) \tag{3}$$

with $N_1 = n, N_3 = m$, and $\varepsilon(x)$ a NN functional reconstruction error vector. If there exist N_2 and constant "ideal" weights W and V so that $\varepsilon = 0$ for all $x \in \boldsymbol{S}$, we say $f(x)$ is in the functional range of the NN. In general, given a constant real number $\varepsilon_N > 0$, we say $f(x)$ is within ε_N of the NN range if there exist N_2 and constant weights so that for all $x \in \boldsymbol{R}^n$, (3) holds with $\|\varepsilon\| < \varepsilon_N$.

Various well-known results for various activation functions $\sigma(\cdot)$, based, e.g., on the Stone–Weierstrass theorem, say that any sufficiently smooth function can be approximated by a suitably large net [8], [13], [31], [38]. The functional range of NN (2) is said to be dense in $C^m(\boldsymbol{S})$ if for any $f \in C^m(\boldsymbol{S})$ and $\varepsilon_N > 0$ there exist finite N_2, and W and V such that (3) holds with $\|\varepsilon\| < \varepsilon_N, N_1 = n, N_3 = m$. Typical results are like the following, for the case of σ the "squashing functions" (a bounded, measurable, nondecreasing function from the real numbers onto [0, 1]), which include for instance the step, the ramp, and the sigmoid.

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



continuous

Theorem 2.1: Set $N_1 = n, N_3 = m$ and let σ be any squashing function. Then the functional range of NN (2) is dense in $C^m(\boldsymbol{S})$. ■

In this result, the metric defining denseness is the supremum norm. Moreover, the last layer thresholds $\theta_{w\ell}$ w are not needed for this result. The issues of selecting σ, and of choosing N_2 for a specified $\boldsymbol{S} \subset \boldsymbol{R}^n$ and ε_N are current topics of research (see, e.g., [28] and [31]).

Lecture 12.3

B. Stability and Passive Systems

Some stability notions are needed to proceed. Consider the nonlinear system

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

skip this for now.

with state $x(t) \in \boldsymbol{R}^n$. We say the solution is uniformly ultimately bounded (UUB) if there exists a compact set $\boldsymbol{U} \subset \boldsymbol{R}^n$ such that for all $x(t_0) = x_0 \in \boldsymbol{U}$, there exists an $\varepsilon > 0$ and a number $T(\varepsilon, x_0)$ such that $\|x(t)\| < \varepsilon$ for all $t \ge t_0 + T$. As we shall see in the proof of the theorems, the compact set $\boldsymbol{U}$ is related to the compact set on which NN approximation property (3) holds. Note that $\boldsymbol{U}$ 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"

gravity vector, and $F(\dot{q})$ the friction. Bounded unknown disturbances (including, e.g., unstructured unmodeled dynamics) are denoted by τ_d, and the control input torque is $\tau(t)$.

Given a desired arm trajectory $q_d(t) \in \boldsymbol{R}^n$ the tracking error is

$$e(t) = q_d(t) - q(t). \tag{8}$$

In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \tag{9}$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

$$M\dot{r} = -V_m r - \tau + f + \tau_d \tag{10}$$

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda \dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q})$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T.$$

Define now a control input torque as

$$\tau_o = \hat{f} + K_v r$$

And for some specific cases of sigma, you do have results which say that the functional range of the neural network is dense in the set of continuous functions. What, we will try to interpret this result, of course. First of all, there is the term squashing function. So, what is a squashing function. So, basically it is defined here. A squashing function is a bounded measurable non-decreasing function from the real numbers on to 0, 1. So, basically, this is this kind of function includes the step function, the ramp function, sigmoid function et cetera. So, squashing functions are rather relatively large class of functions.

So, for these kind of activation functions in your neural network, what this theorem claims is that the functional range of the neural network is dense and continuous. So, what does it mean? This statement, for those who have not seen analysis courses, essentially means that almost, almost any continuous function can be approximated via this neural network.

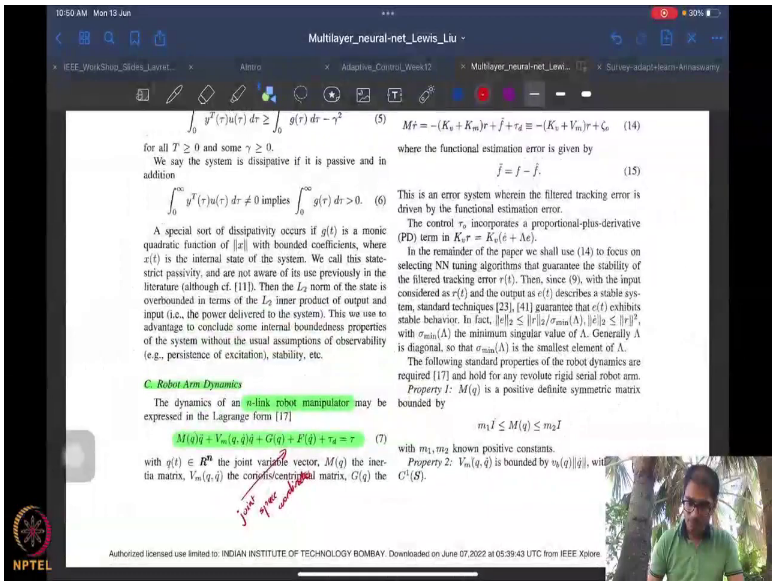
So, this is a very specific neural network. By the way, it is exactly a three-layer neural network. So, one input, one output and one hidden layer. So, with the specific neural network, you have a result which says that you can approximate almost any continuous function quite well with this neural network. So, this is what it means.

So, of course the dense is the in terms of the supremum norm. So, this is the whole idea of how you define dense. In fact, what the authors state here is that the last layer threshold values, that is the offsets, were not even required. The last layer thresholds were not needed for this result, you do not even need the last thresholds for you to have a good approximation. So, of course how to select these sigma and then choosing this N2 and a particular, for a particular set S are of course topics of research.

So, then there is the section on stability and passivity, which I am going to skip for now. For now, I am going to skip it. If I do see the need, I will get back to it. We have not discussed passivity, which is a very standard notion for a non-linear control. And if you have a system which is passive, designing controllers becomes sufficiently easy. This is this is usually covered in a more detailed nonlinear control course, which we do not aspire to be, we are more of a nonlinear adaptive control course.

So, we will skip this for now, and we will see. If we need it later on, we will talk about it in passing. So, then in this next section, the authors are starting to look at, oh by the way, I was supposed to start, to mark the lectures. So, I am going to mark it here, lecture 12.3. So, the third lecture of our last week.

(Refer Slide Time: 12:12)



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

for all $T \geq 0$ and some $\gamma \geq 0$.

We say the system is dissipative if it is passive and in addition

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

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

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

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \qquad (7)$$

with $q(t) \in \boldsymbol{R}^n$ the joint variable vector, $M(q)$ the inertia matrix, $V_m(q,\dot{q})$ the coriolis/centripetal matrix, $G(q)$ the

joint space coordinates

$$M\dot{r} = -(K_v + K_m)r + \tilde{f} + \tau_d \equiv -(K_v + V_m)r + \zeta_o \qquad (14)$$

where the functional estimation error is given by

$$\tilde{f} = f - \hat{f}. \qquad (15)$$

This is an error system wherein the filtered tracking error is driven by the functional estimation error.

The control τ_o incorporates a proportional-plus-derivative (PD) term in $K_v r = K_v(\dot{e} + \Lambda e)$.

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, $\|e\|_2 \leq \|r\|_2/\sigma_{\min}(\Lambda)$, $\|\dot{e}\|_2 \leq \|r\|^2$, 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.

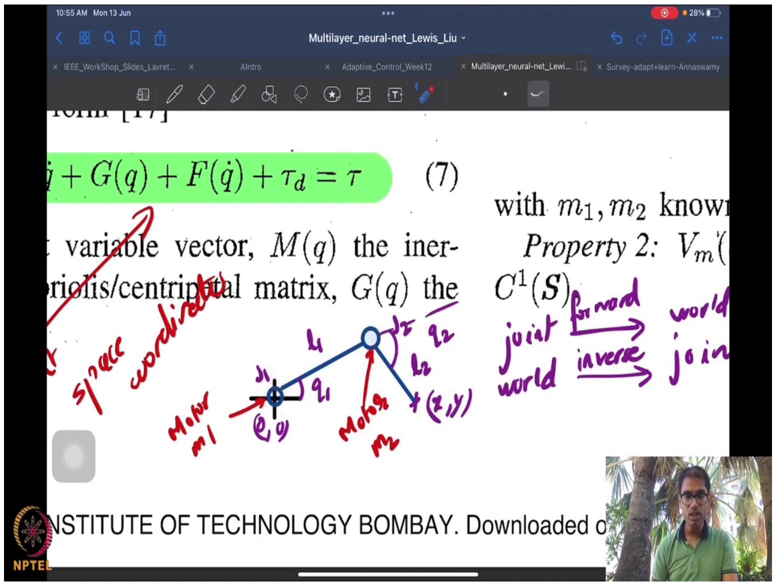
Property 1: $M(q)$ is a positive definite symmetric matrix bounded by

$$m_1 I \leq M(q) \leq m_2 I$$

with m_1, m_2 known positive constants.

Property 2: $V_m(q,\dot{q})$ is bounded by $v_b(q)\|\dot{q}\|$, wit

$C^1(S)$.



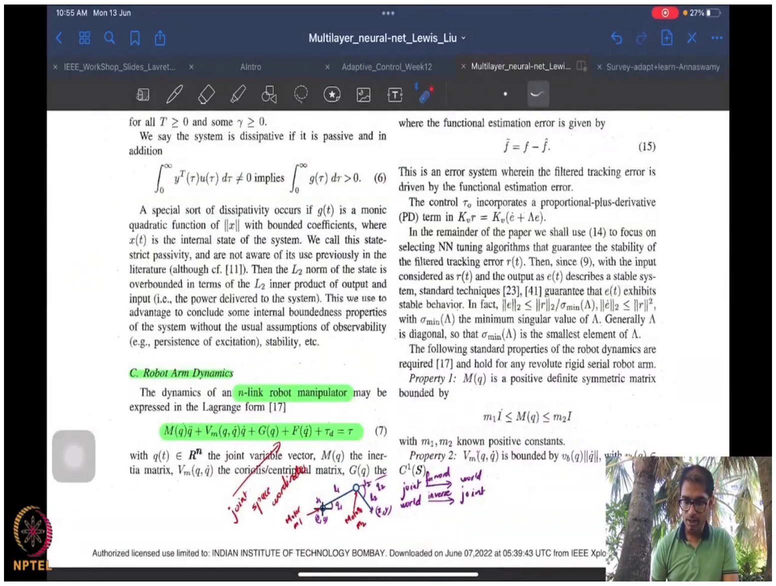
space coordinates
l_1
J_1
q_1
$(0,0)$
Motor m1
J_2
q_2
l_2
Motor m_2
(x,y)
joint $\xrightarrow{\text{forward}}$ world
world $\xrightarrow{\text{inverse}}$ join

$+ G(q) + F(\dot{q}) + \tau_d = \tau$ (7)

variable vector, $M(q)$ the iner-
riolis/centripetal matrix, $G(q)$ the

with m_1, m_2 known
Property 2: $V_m($
$C^1(S)$

NSTITUTE OF TECHNOLOGY BOMBAY. Downloaded o


10:55 AM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

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 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

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

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \quad (7)$$

with $q(t) \in \mathbf{R}^n$ the joint variable vector, $M(q)$ the inertia matrix, $V_m(q,\dot{q})$ the coriolis/centripetal matrix, $G(q)$ the

joint space coordinate

Motor #1 Motor #2 l_1 l_2 q_1 q_2 (x,y)

where the functional estimation error is given by

$$\tilde{f} = f - \hat{f}. \quad (15)$$

This is an error system wherein the filtered tracking error is driven by the functional estimation error.

The control τ_o incorporates a proportional-plus-derivative (PD) term in $K_v r = K_v(\dot{e} + \Lambda e)$.

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, $\|e\|_2 \leq \|r\|_2/\sigma_{\min}(\Lambda)$, $\|\dot{e}\|_2 \leq \|r\|^2$, 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

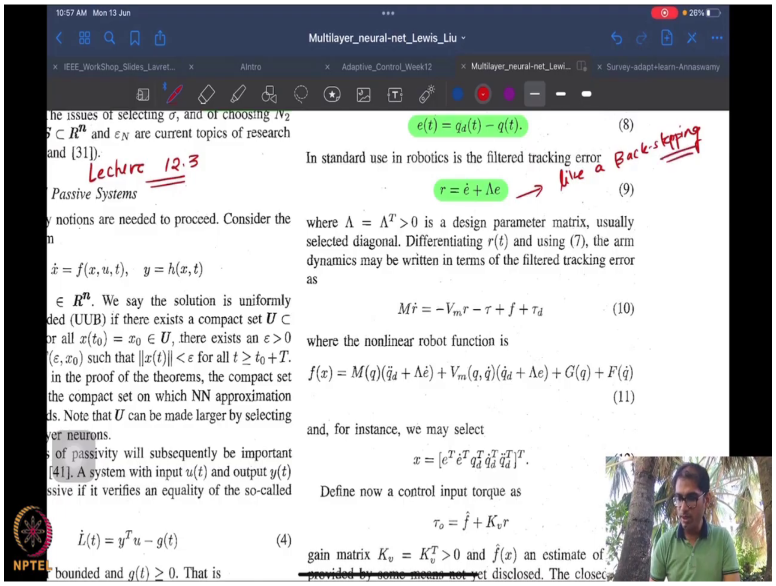
$$m_1 I \leq M(q) \leq m_2 I$$

with m_1, m_2 known positive constants.

Property 2: $V_m(q,\dot{q})$ is bounded by $v_b(q)\|\dot{q}\|$, with

$C^1(S)$

$$\text{joint} \xrightarrow{\text{forward}} \text{world}$$
$$\text{world} \xrightarrow{\text{inverse}} \text{joint}$$


10:57 AM Mon 13 Jun

Multilayer_neural-net_Lewis_Liu

he issues of selecting σ, and of choosing N_2 $S \subset \mathbf{R}^n$ and ε_N are current topics of research and [31]).

Lecture 12.3

Passive Systems

y notions are needed to proceed. Consider the m

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

$\in \mathbf{R}^n$. We say the solution is uniformly ded (UUB) if there exists a compact set $\boldsymbol{U} \subset$ or all $x(t_0) = x_0 \in \boldsymbol{U}$, there exists an $\varepsilon > 0$ (ε, x_0) such that $\|x(t)\| < \varepsilon$ for all $t \geq t_0 + T$. in the proof of the theorems, the compact set the compact set on which NN approximation ds. Note that U can be made larger by selecting er neurons.

of passivity will subsequently be important [41]. A system with input $u(t)$ and output $y(t)$ ssive if it verifies an equality of the so-called

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

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

$$e(t) = q_d(t) - q(t). \quad (8)$$

In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \quad (9)$$

→ like a Back-stepping

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

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

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \quad (11)$$

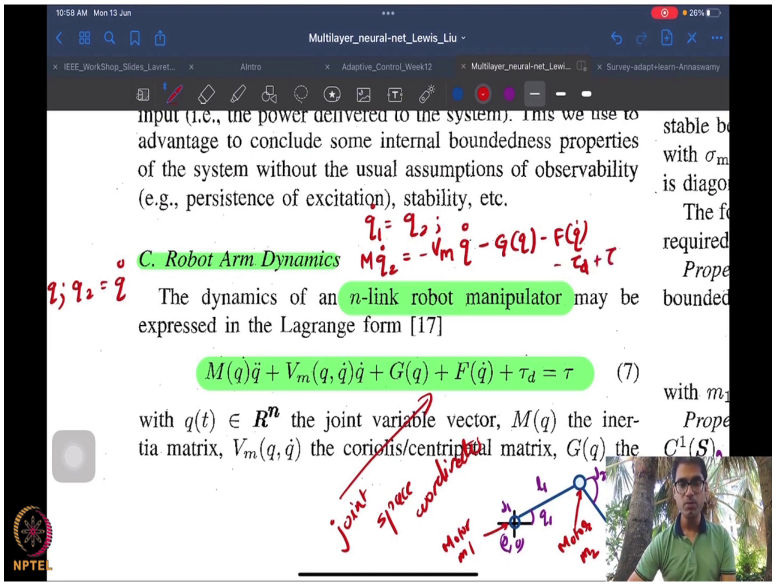
and, for instance, we may select

$$x = [e^T\, \dot{e}^T\, q_d^T\, \dot{q}_d^T\, \ddot{q}_d^T]^T.$$

Define now a control input torque as

$$\tau_o = \hat{f} + K_v r$$

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



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.

$\dot q_1 = q_2$; $M\dot q_2 = -V_m \dot q - G(q) - F(\dot q) - \tau_d + \tau$

q; $q_2 = \dot q$

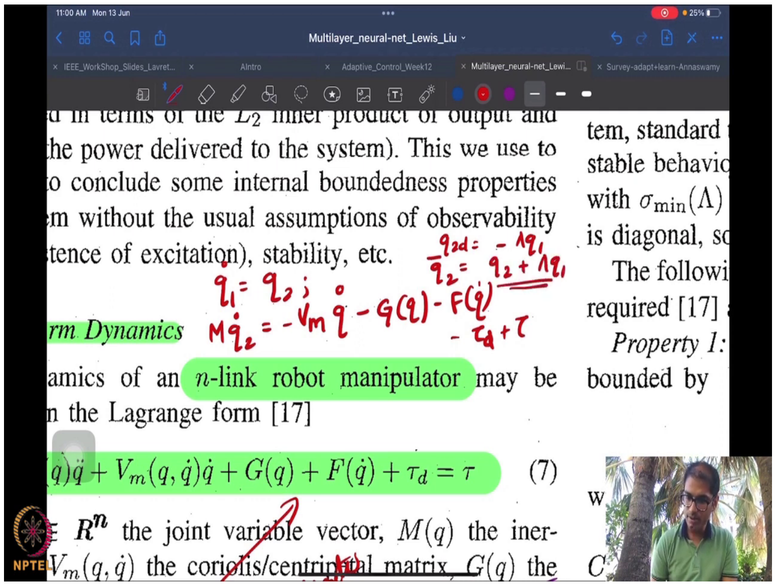
C. Robot Arm Dynamics

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

$$M(q)\ddot q + V_m(q,\dot q)\dot q + G(q) + F(\dot q) + \tau_d = \tau \qquad (7)$$

with $q(t) \in \boldsymbol{R}^n$ the joint variable vector, $M(q)$ the inertia matrix, $V_m(q,\dot q)$ the coriolis/centripetal matrix, $G(q)$ the

stable b
with σ_m
is diago
The f
required
Prope
bounded
with m_1
Prope
$C^1(S)$



d in terms of the L_2 inner product of output and the power delivered to the system). This we use to o conclude some internal boundedness properties m without the usual assumptions of observability tence of excitation), stability, etc.

$q_{2d} = -\Lambda q_1$
$\bar q_2 = q_2 + \Lambda q_1$

$\dot q_1 = q_2$; $M\dot q_2 = -V_m \dot q - G(q) - F(\dot q) - \tau_d + \tau$

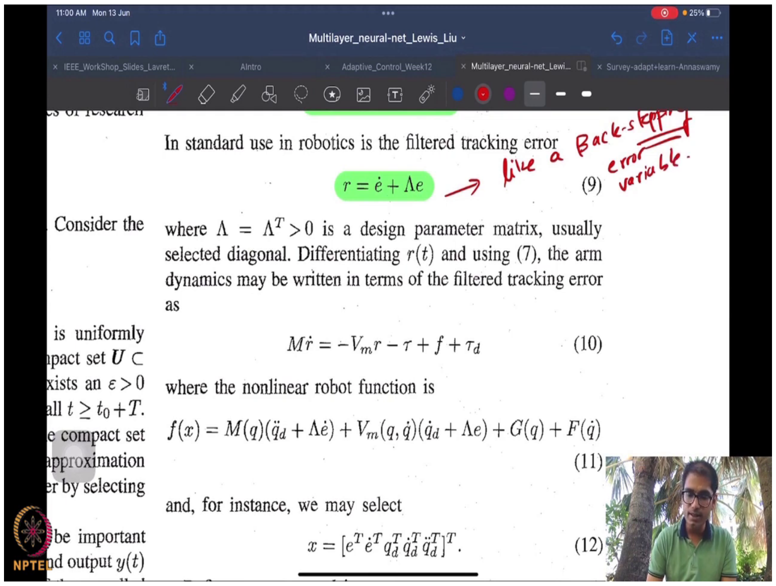
rm Dynamics

amics of an n-link robot manipulator may be n the Lagrange form [17]

$$q)\ddot q + V_m(q,\dot q)\dot q + G(q) + F(\dot q) + \tau_d = \tau \qquad (7)$$

$\boldsymbol{R}^n$ the joint variable vector, $M(q)$ the iner- $V_m(q,\dot q)$ the coriolis/centripetal matrix, $G(q)$ the

tem, standard
stable behavio
with $\sigma_{\min}(\Lambda)$
is diagonal, so
The followi
required [17]
Property 1:
bounded by



11:00 AM Mon 13 Jun
Multilayer_neural-net_Lewis_Liu
IEEE_WorkShop_Slides_Lavret...
AIntro
Adaptive_Control_Week12
Multilayer_neural-net_Lewi...
Survey-adapt+learn-Annaswamy
NPTEL

Consider the

is uniformly
pact set $U \subset$
xists an $\varepsilon > 0$
all $t \geq t_0 + T$.
e compact set
approximation
er by selecting

be important
nd output $y(t)$

In standard use in robotics is the filtered tracking error

$$r = \dot{e} + \Lambda e \quad \rightarrow \text{like a Back-stepping error variable.} \tag{9}$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as

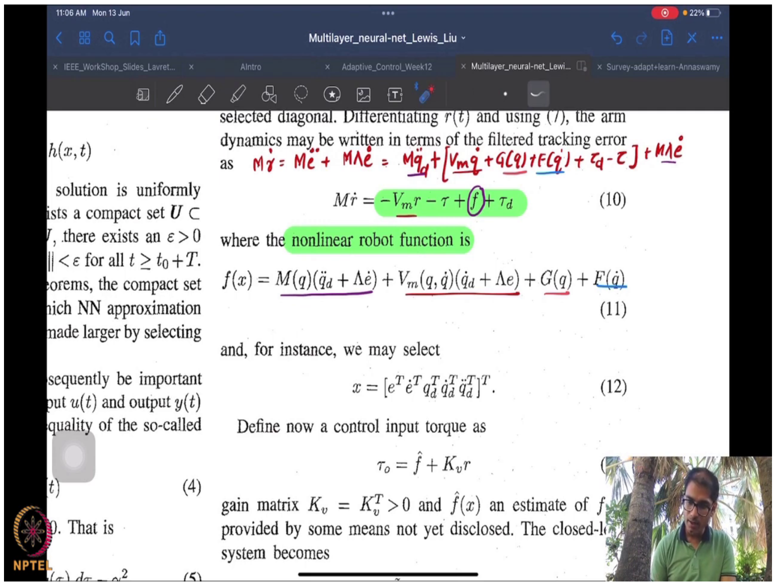
$$M\dot{r} = -V_m r - \tau + f + \tau_d \tag{10}$$

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \tag{11}$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T. \tag{12}$$



11:06 AM Mon 13 Jun
Multilayer_neural-net_Lewis_Liu
IEEE_WorkShop_Slides_Lavret...
AIntro
Adaptive_Control_Week12
Multilayer_neural-net_Lewi...
Survey-adapt+learn-Annaswamy
NPTEL

$h(x,t)$

solution is uniformly
ists a compact set $U \subset$
, there exists an $\varepsilon > 0$
$\| < \varepsilon$ for all $t \geq t_0 + T$.
eorems, the compact set
hich NN approximation
made larger by selecting

sequently be important
put $u(t)$ and output $y(t)$
quality of the so-called

$t)$ (4)

). That is

selected diagonal. Differentiating $r(t)$ and using (7), the arm dynamics may be written in terms of the filtered tracking error as $M\dot{r} = M\ddot{e} + M\Lambda\dot{e} = \underline{M\ddot{q}_d} + [\underline{V_m\dot{q}} + \underline{G(q)} + \underline{F(\dot{q})} + \tau_d - \tau] + \underline{M\Lambda\dot{e}}$

$$M\dot{r} = -\underline{V_m}r - \tau + \boxed{f} + \tau_d \tag{10}$$

where the nonlinear robot function is

$$f(x) = \underline{M(q)(\ddot{q}_d + \Lambda\dot{e})} + \underline{V_m(q,\dot{q})(\dot{q}_d + \Lambda e)} + \underline{G(q)} + \underline{F(\dot{q})} \tag{11}$$

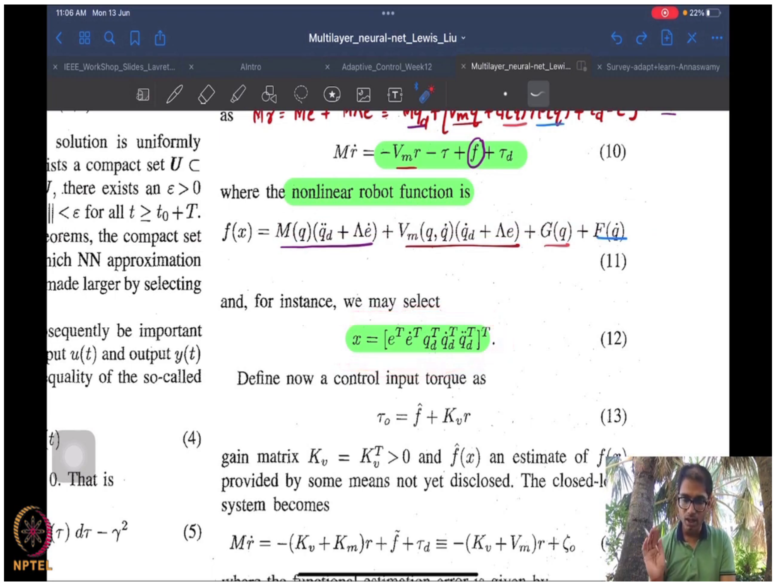
and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T. \tag{12}$$

Define now a control input torque as

$$\tau_o = \hat{f} + K_v r$$

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


Multilayer_neural-net_Lewis_Liu

solution is uniformly
ists a compact set $U \subset$
, there exists an $\varepsilon > 0$
$\| < \varepsilon$ for all $t \geq t_0 + T$.
orems, the compact set
ich NN approximation
made larger by selecting

sequently be important
put $u(t)$ and output $y(t)$
quality of the so-called

$t)$ (4)

0. That is

$\tau)\, d\tau - \gamma^2$ (5)

as

$$M\dot{r} = -V_m r - \tau + f + \tau_d \tag{10}$$

where the nonlinear robot function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \tag{11}$$

and, for instance, we may select

$$x = [e^T \dot{e}^T q_d^T \dot{q}_d^T \ddot{q}_d^T]^T. \tag{12}$$

Define now a control input torque as

$$\tau_o = \hat{f} + K_v r \tag{13}$$

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

$$M\dot{r} = -(K_v + K_m)r + \tilde{f} + \tau_d \equiv -(K_v + V_m)r + \zeta_o$$

So, in this section C, the authors start to look at the robotic arm dynamics. Why? Because this is the application that the authors are focusing on. I mean, if you focus on another application, you can write the dynamics for that in a particular form. So, this is the standard form for robot dynamics, this is called the Euler-Lagrange form.

So, the Euler, E-L equations, very standard to call them E-L equations. Here, this is, this is a structure which holds for all any n-link robotic manipulator. So, it is generally enough, if you have an n-link robotic manipulator, you can use this. And this is written, these equations are written in what is called joint space coordinates.

So, if you have any, I mean, I usually make this picture. I will try to make this picture here for you folks. So, if you have any sort of axis here, and then you have a one joint here, then you have another joint here. And then you have another axis here. Sorry, another sort of link here, joint. So, this is, so, this is, say, joint 1. And this is say joint 1, this is link 1, joint 2, link 2.

And then of course, you can mark it. So, now if you look at some coordinate system, say this is a 0 0, this point, then this coordinate in the world frame is, usually has some coordinates in, say two dimensional frame x y, it is a planar manipulator. If it is not a planer manipulator, then it will have all three coordinates. But let us keep things simple and say it is a planar manipulator.

And so, these are the coordinates in the world frame. But then, there is another way to specify these coordinates. So, this is, let us see, if you look at this angle, this is say q 1, and you look at this angle, this is say q 2. So, there are two ways to look at robot dynamics or two ways to write the robot dynamics model. One is in the joint frame, these q 1s and q 2s, and another is in the world frame, which is with this x and y.

You see that both have equal number in this case. In this particular case, the number of joint coordinates and number of world coordinates are the same, but that is not necessarily true because I could have 20 joints in a 2-dimensional frame. So, I have a lot of redundancy. And then, of course there is many more joint coordinates than there are world coordinates.

So, and of course there is a forward and inverse kinematics for those we have seen robotics. There is forward kinematics to move from the joint coordinates to the world coordinates. So, that is again something standard, joint to world is the forward kinematics. And the world to the joint is inverse kinematics.

So, you have these kinematics which means that it is just a relationship between the joint coordinates and the world coordinates and vice versa. So, that is what is your kinematics equation. So, now what these authors are doing there, in this particular problem, is they are writing the, everything in the joint coordinates. So, that is why you have these q.

These are joint coordinates or they are also in Euler-Lagrange dynamical notions called generalized coordinates. Why are they generalized coordinates, because they could represent angle or also linear motion, because you could have joints which are just lengthening or shortening. So, you could have linear actuators also, you could have angular actuators also, in a robot.

So, this q could represent both length and angle. So, that is why q is also called generalized coordinates. But most important to remember is that it is in the joint frame, it is not in the world frame. So, then you have this M, which is your inertia matrix. So, M is the inertia matrix, it is a function of your q, which is the joint coordinates.

Then you have V m, which is the coriolis and centripetal matrix, which is a function of both your q and q dot. Then, you have G q, which is usually the gravity matrix. Then, G q is the gravity vector, and the gravity, sorry, not the matrix but the gravity vector. And finally, F, which depends on q dot is the friction. Then you have these terms tau and tau d, and tau d is basically any kind of external disturbance. So, any kind of external disturbance is sort of put into the tau d.

We have already seen how disturbance gets into adaptive systems or any nonlinear system for that matter. And we usually assume that these are bounded, even though we may not know what the values are at each instant in time. And then you have tau which is the actual torque that I can exert. So, when you talk about this tau, for example, if you look at this manipulator, how would you apply this tau?

You would assume there is a, there is usually a motor m 1, and there is another motor m 2. So, you would have some motor here on this joint and another motor here on this joint, and this tau is essentially obtained by rotating the motor. It could be whatever, I mean, you could have a servo drive or you could have a, I mean, DC motor, whatever. I mean you could have different kinds of motors but the point is that is how you get the torque tau.

So, once you understand how this robot works and how the model looks like, of course, we are not giving any structure for this M, notice, if you have a specific robotic model, you will have to actually write the dynamics either your Newtonian formulation or with the Lagrangian formulation or Hamiltonian formulation, but the point is you will have to write the dynamics. And then once you put it in this structure and you will be able to do it in this structure, you will get the expressions for M, V m, G, F so on.

So, suppose, how is the problem formulated? I mean, any control problem is formulated with some kind of a desired reference trajectory. So, this is, we are already used to this kind of a notion. So, this is how any control problem is formulated. So, here you see that your desired trajectory some is qd of t. And this helps us define an error, which is the true value of the joint, sorry, which is, they have defined it in an opposite way, the desired value minus the actual value of the joint variable. So, if you have only angles, then the desired value of the angle minus the actual value of the angle.

Now, it is also well known in robotics to use this kind of a filtered tracking error in order to do the analysis for this system. This is actually, for you folks, this is essentially like a, like a backstepping variable, like a backstepping variable. Why? Because if you look at these dynamics, and if I did write it as, this can be written as a double integrator sort of dynamic side.

So, if I take q 1 as q and q 2 as q dot, then what would be my dynamics? My dynamics would be q 1, that is q 2. And q 2 dot or M q 2 dot is equal to minus V m q dot minus G q minus F q dot minus tau d plus tau. So, this is what would be my state space equations. And if you see, if I think of some kind of a backstepping type of algorithm to control design, because this is essentially a non-linear double integrator, some kind of nonlinear double integral.

So, we want to use backstepping to design, then I would choose q 2 as some minus lambda q 1. q 2 desired would be some minus lambda q 1. And so, R is like the backstepping error. So, I mean, of course, I am writing everything in the error. If q d is 0, if q d was not there, then e is just q, and forget the sign. The sign is irrelevant. If q d was not there, then e is just q.

So, R becomes q dot plus lambda q. So, that is what we said. We will choose q 2 as minus lambda q 1 and the backstepping error would be, so let me, q 2 desired will be something like minus lambda q 1, if I was just looking at the stabilization problem, that is, I want to send q 1 and q 2 to 0. And the backstepping error would be what, q 2 plus lambda q 1.

And you see, this is, and lambda is some positive definite matrix. This is exactly what you have here. Sorry, like a q 2 plus lambda q 1, if there was no desired, if there is desired, it is the same. I mean, if I write it in terms of the desired variable, I will get something very similar because if I wrote this as e 1 dot, e 1 and e 2, it will be e 1 dot equal to e 2 and M e 2 dot will be some equation on the right hand side. I will still get something very similar.

It does not matter if it is tracking or stabilization, but this is like the back stepping, this is like a backstepping error, variable. It is very standard to use this in robotics. Why? It is obvious, because it is a just a non-linear double integrator and if you want to get some kind of strict Lyapunov function or get some good Lyapunov candidate, this is the, what makes sense.

So, of course lambda is some positive definite symmetric matrix. I mean, you can of course have it as a scalar. Usually you select it as some diagonal matrix. You can either have a scalar a diagonal matrix. You can also have a full matrix, just positive definite symmetric. It is just a design parameter equivalent.

Now, if I take the derivative of r, and I plug in from my equations here, it is not difficult to see that I will get this kind of an equation. It is not difficult to see that I will get this kind of an equation because it should be evident to you that M r dot is equal to M e double dot plus M lambda e dot.

And M e double dot, so this is just M times e double dot, sorry, this M e double dot is M q d double dot, M q d double dot minus M q double dot, so minus M q double dot is this guy, minus M q double dot is essentially V m q dot, V q V m q dot plus G q plus F q dot plus tau d minus tau plus M lambda e dot. So, this is this big thing that you get.

So, what you do is you can write this, what you will do is you will write this whole thing, you want to write this whole thing as a, this guy. You want to write this whole thing as this guy. And how do you do that? You simply, you add and subtract minus V m r, and you just add and subtract minus V m r and then you combine it with this guy.

So, what do you have? You will have this, so this f is essentially to encapsulate all the remaining terms. So, you have these two terms here, then you have, from here and from here, you have these guys. Then you have this term, this term here, then you have this F q dot coming from here. And of course, we have retained the, the tau d and tau have been retained as it is.

So, I just need to check the sign, I, think there is some sign issue here or did I miss the same thing. Somehow, the sign seems off. If I take M r dot M e double dot minus M (lamda), sorry, M e double dot plus M lambda e dot, is just fine and. So, from M e double dot I have M q d double dot minus M q double dot, minus M q double dot. I see. I see. So, I think I messed up the sign. So, this has to be, all of this is with a plus, all of this actually has a plus sign. So, now it is all right, this is fine. All of this has a plus sign on it.

So, this, this is all fine. So, then I have tau d minus tau. So, this f is what? Encapsulates this kind of a nonlinear robot function. So, why do we call it a nonlinear robot function is because this is what sort of encapsulates the dynamics of the robot. Of course, there is also V m, but we are not going to worry about that term. This is what encapsulate, what encapsulates what the dynamics looks like.

So, and this is this f x, where, where of course we have chosen x to include all these quantities. e, e dot, q d, q d dot, q d double dot. Basically, everything that shows up here, you have q d double dot dot, q d dot, e and so on. So, you have all the variables appearing here and so it is a function of all these variables x. So, you have this, how you sort of put it.

Of course, it is also a function of q which is not written here but then that is already inside e. So, that is fine. So, great. So, this is the sort of simple, relatively simpler structure that we will work with. We will try to drive r to 0. We look at what that means in the subsequent section. So, we will look at what that means in the subsequent section.

So, essentially what we have done today is we have sort of continuing our discussion on learning based or adaptive control-based learning, and we are using it for a planar robotic manipulator. And so today, we spent some time understanding the dynamics of the manipulator, and in the subsequent session we hope to look at the control design part of the manipulator. So, I hope you are enjoying our sessions, and I hope to see you again soon.