

**Robotics and Control: Theory and Practice**  
**Prof. N. Sukavanam**  
**Department of Mathematics**  
**Indian Institute of Technology, Roorkee**

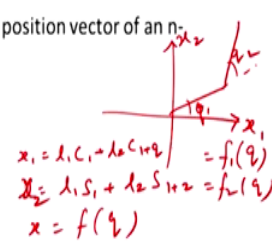
**Lecture – 20**  
**Neural Network Based Control for Robot Manipulators**

In this lecture we shall see how Neural Network Based Controllers can be designed for a Robot Manipulator to track desired trajectories under uncertainties and disturbances. So, the so, let us consider the kinematic model which can be written as  $x$  is equal to  $f$  of  $q$ .



(Refer Slide Time: 00:41)

### Kinematic Model

- The end-effector position and orientation in the task-space denoted by  $x = [x_1 \ x_2 \ \dots \ x_m]^T$ , is defined as
 
$$x = f(q) \dots \dots \dots (1)$$
 where  $f(q) \in R^m$  denotes the direct kinematics and  $q = [q_1 \ q_2 \ \dots \ q_n]^T \in R^n$  ( $m < n$ ) denotes the joint position vector of an  $n$ -link robot manipulator.



$x_1 = l_1 \cos q_1 + l_2 \cos(q_1 + q_2) = f_1(q)$   
 $x_2 = l_1 \sin q_1 + l_2 \sin(q_1 + q_2) = f_2(q)$   
 $x = f(q)$

 IIT ROORKEE
  NITEL ONLINE CERTIFICATION COURSE

2

So, here  $x$  denote the Cartesian position and orientation of the robot end effector and  $q$  which is equal to  $q_1 \ q_2 \ \dots \ q_n$  vector denotes the joint variables. So, for example, if we take a 2 omp manipulator with angles we can call it as  $q_1$  and  $q_2$  and the Cartesian axis are  $x_1$  and  $x_2$ .

We can write the kinematics equation as  $x_1$  is equal to  $l_1 \cos q_1$  plus  $l_2 \cos q_1 + q_2$   
 $x_2$  equal to  $l_1 \sin q_1$  and  $l_2 \sin q_1 + q_2$ .

So, here we can write  $x$  to the  $x_1, x_2$  and  $q$  to be  $q_1$  and  $q_2$  and this function we can call it as  $f_1$  of  $q$  and this is  $f_2$  of  $q$ . So, we can write it as  $x$  is equal to  $f$  of  $q$  where  $f$  equal to  $f_1 \ f_2$  vector and  $x$  is  $x_1 \ x_2$  vector. So, it is a kinematic model for the 2 omp manipulator.

(Refer Slide Time: 02:25)

### Kinematic Model

- Differentiating (1) with respect to time yields  

$$\dot{x} = J(q)\dot{q} \dots \dots \dots (2)$$
- The manipulator Jacobian, denoted by  $J(q) \in R^{m \times n}$  is defined as follows  

$$J(q) = \frac{\partial f(q)}{\partial q} \dots \dots \dots (3)$$
- One solution of equation (2) is  

$$\dot{q} = J^+(q)\dot{x} \dots \dots \dots (4)$$
- where  $J^+$  denotes the pseudo-inverse of  $J$  and it is defined in the following manner:  

$$J^+ = J^T (JJ^T)^{-1} \dots \dots \dots (5)$$
  
 such that:  

$$JJ^+ = I_m \dots \dots \dots (6)$$

*Handwritten notes:*  
 $\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$   
 $\dot{x}_1 = l_1(-\dot{q}_1)q_2 - l_2 \dot{q}_2 \sin(q_1 + q_2)$   
 $J = \begin{bmatrix} -l_1 q_2 & -l_2 \sin(q_1 + q_2) \\ l_1 & l_2 \cos(q_1 + q_2) \end{bmatrix}$   
 $J^+ = \begin{bmatrix} -l_1 q_2 & -l_2 \sin(q_1 + q_2) \\ l_1 & l_2 \cos(q_1 + q_2) \end{bmatrix}^T \begin{bmatrix} -l_1 q_2 & -l_2 \sin(q_1 + q_2) \\ l_1 & l_2 \cos(q_1 + q_2) \end{bmatrix}^{-1}$

*Handwritten note:*  $J \in R^{m \times n}$

And, now if we differentiate this equation with respect to time: we get the Cartesian velocity  $\dot{x}$  in terms of the joint velocity  $\dot{q}$ . So, here again  $\dot{x}$  represent  $\dot{x}_1 \ \dot{x}_2$  dot etcetera  $\dot{x}_n$  dot the velocity of the end effector it contains position and orientation of the end effector and  $\dot{q}$  denotes the joint velocity, where this  $J$  represent the Jacobian of the manipulator. So, in the previous example if we see that we can write  $\dot{x}_1$  dot is equal to  $l_1 \cos$

theta 1. If we differentiate will get minus sorry cos q 1 if we differentiate we will get sin q 1 into q 1 dot minus l 2 sin q 1 plus q 2 multiplied by q 1 dot plus q 2 dot.

Similarly,  $\dot{x}_2$  can be written and this can be written as  $\dot{x}_1$  dot  $\dot{x}_2$  vector can be written as a 2 by 2 matrix multiplied by  $\dot{q}_1$  dot  $\dot{q}_2$  dot, where the 2 by 2 matrix contains this coefficients minus l 1 sin q 1 minus l 2 sin q 1 plus q 2 that is multiplied by  $\dot{q}_1$  dot etcetera. So, the four coefficients can be written in the matrix which forms the Jacobian of the 2 omp manipulator. So, this is a general one for any robot manipulator we can write  $\dot{x}$  dot equal to Jacobian into  $\dot{q}$  dot form.

Now, if J is a 2 n by n matrix then J inverse can be calculated, but if J is not a, it is not a square matrix then we can find the pseudo inverse of the Jacobian. So, the J plus denote the pseudo inverse which can be calculated by this expression, this is especially in the case of if J is a matrix of size. It is n cross m where m is greater than n. So, in the case of redundant manipulators we can write and if m is equal to n J plus denotes the inverse of the matrix J, the usual inverse of the matrix.

So, the J plus is denote defined as J transpose multiplied by J J transpose whole inverse provided the inverse exist for the J J transpose. And, here because n is less than or equal to m. J J transpose is a n cross m matrix and we can find the inverse of the square matrix and we can calculate J plus. And, it can be easily verified that J into J J plus will give the identity matrix from the expression 5 itself we can see that J J plus is identity matrix.

(Refer Slide Time: 05:59)


## Dynamic Model

The dynamics model for an  $n$ -link robot manipulator can be described as:

$$M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) + F_r(\dot{q}) + T_d = \tau$$

where  $M(q) \in R^{n \times n}$  represents the inertia matrix,  $V(q, \dot{q}) \in R^{n \times n}$  represents the centripetal-coriolis matrix,  $G(q) \in R^n$  represents the gravity effects,  $F_r(\dot{q}) \in R^n$  represents the friction effects,  $T_d \in R^n$  is a vector of unknown but bounded external disturbances or unmodeled dynamics effects and  $\tau \in R^n$  represents the torque input vector. The robot dynamics given above has the following useful properties:

- **Property 1.** The inertia matrix  $M(q)$  is symmetric, positive definite and satisfies the following inequalities:
$$m_1 \|\xi\|^2 \leq \xi^T M(q) \xi \leq m_2 \|\xi\|^2 \quad \xi \in R^n$$
where  $m_1, m_2$  are positive constants, and  $\|\cdot\|$  denotes the standard Euclidean norm.

NPTEL ONLINE  
CERTIFICATION COURSE4

So, now we consider the dynamic equation of the robot manipulator for a given manipulator we have seen how to derive the dynamic equation using the Euler Lagrange equation. And, here  $M$  represents the inertia matrix,  $V$  is the centrifugal and Coriolis term,  $g$  represent the gravity term. And, the disturbance which occurs during the motion of the robot manipulators are denoted by  $F$  and the  $T$  suffix  $d$  the disturbance. External disturbance is denoted by  $T$  suffix  $d$  and the friction due to the manipulator itself is denoted by  $F_q$  function and the  $\tau$  denotes the torque applied at the joints of the manipulator.

So, this  $F$  term and the  $T$  suffix  $d$  terms are unknown because they are all disturbances or the friction terms, but we know that in the real life situation all these disturbances are in most of the time they are small in size. So, they are all bounded by certain constant. Here these are all vectors  $F$  is a vector and  $T$  suffix  $d$  is a vector the disturbance terms and the norm of those vectors can be bounded by certain constant. And, we also know that the matrix  $M$ , the matrix

which is the inertia matrix satisfies this property because the entries of the inertia matrices are all bounded below and above.


So, we can write the property that for any vector  $x_i$  in  $\mathbb{R}^n$   $x_i^T M x_i$  is a real number that is bounded above and below by this types of constants where  $m_1$  and  $m_2$  are suitable values we can easily find.

(Refer Slide Time: 08:16)

Dynamic Model

- Property 2.** The inertia and centripetal-coriolis matrices satisfy the following skew-symmetric relationship:
 
$$\underbrace{\xi^T}_{1 \times n} \underbrace{\left( \frac{1}{2} \dot{M}(q) - \underline{V(q, \dot{q})} \right)}_{n \times n} \underbrace{\xi}_{n \times 1} = 0 \quad \forall \xi \in \mathbb{R}^n$$

where  $\dot{M}(q)$  denotes the time derivatives of the inertia matrix.


NPTEL ONLINE CERTIFICATION COURSE
5

And, we can also see that the matrix  $M$  if we differentiate with respect to time and half  $dM/dt$  minus the centrifugal term. It is a skew symmetric matrix, in other words if we take any vector  $x_i$  in  $\mathbb{R}^n$   $x_i^T$  of this into  $x_i$  will always be equal to 0.

So, the  $x_i^T$  is  $1 \times n$  matrix and this whole thing is a  $n \times n$  matrix and this is  $n \times 1$ . So, this is a single number real number which is equal to 0, when we operate with

any arbitrary  $x_i$ . That is the property of the skew symmetric matrix and  $M \frac{1}{2} \dot{M} - V$  always satisfies this particular property.



(Refer Slide Time: 09:13)

### Error System Formulation

- Our aim is to develop a control torque input  $\tau(t)$  in such a way that the robot end-effector follows a desired trajectory as closely as possible.
- The task-space tracking error,  $e(t) \in R^m$  is defined as follows:
 
$$e = x_d - x \dots \dots \dots (7)$$
 where  $x_d \in R^m$  represents the desired task-space trajectory.
- Take the time derivative and then substituting  $\dot{x}$  from (2), we obtain:
 
$$\dot{e} = \dot{x}_d + \alpha e - \alpha e - J\dot{q} \dots \dots \dots (9)$$
 where  $\alpha \in R^{m \times m}$  represents a diagonal, positive definite gain matrix.

Simplifying we obtain

$$\dot{e} = -\alpha e + J(J^+(\dot{x}_d + \alpha e) - \dot{q}) \dots \dots \dots (10)$$

IIT BOMBAY  
NPTEL ONLINE  
CERTIFICATION COURSE

6

Now, let us consider the desired Cartesian trajectory is given by  $x_d$ . It is a function of  $t$  as  $t$  changes the  $x_d$  changes, it is a vector function and  $x$  is the current position of the end effector. So, the error is denoted by  $e$  and it is  $x_d$  minus the desired trajectory minus the current trajectory of the robot end effector. Now, if you differentiate  $e$  we will get  $\dot{x}_d$  minus  $\dot{x}$  and we add and subtract a term  $\alpha e$ , where  $\alpha$  is a positive constant, sorry  $\alpha$  is a  $m$  cross  $m$  matrix which is a positive definite matrix.

For example, we can take a diagonal matrix with positive numbers as the  $\alpha$  matrix and that is a positive definite matrix. Then  $\dot{e}$  can be written in this particular form. So, here we see that  $\dot{e}$  is equal to minus  $\alpha e$  is here and the remaining terms that is  $J(J^+(\dot{x}_d + \alpha e) - \dot{q})$

identity. So, the first term is  $\dot{x}_d$  plus  $\alpha e$  those terms are here  $\dot{x}_d$  plus  $\alpha e$  is here by multiplying this, and then minus  $J \dot{q}$  is here. So, all the terms are rewritten in this particular form. Now, if we denote this  $J \dot{x}_d + \alpha e$  as  $r$  vector.



(Refer Slide Time: 11:04)

### Error System Formulation

- We define a filtered tracking error signal,  $r(t) \in R^n$  as follows:  

$$r = J^+(\dot{x}_d + \alpha e) - \dot{q} \dots \dots \dots (11)$$
- Therefore, the closed loop task-space position tracking error system can now be written in the final form:  

$$\dot{e} = -\alpha e + J r \dots \dots \dots (12)$$


IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSE
7

Then we can write the equation 10 as simply, if we write  $r$  is equal to  $J$  plus into  $\dot{x}_d$  plus  $\alpha e$  minus  $\dot{q}$  sorry the whole thing as  $r$ , then we will get  $\dot{e}$  is written as  $-\alpha e + J r$ . So, that is very easy to verify from the previous.

(Refer Slide Time: 11:30)

### Error System Formulation

- $r = J^*(\dot{x}_d + \alpha e) - \dot{q} \dots \dots \dots (11)$   $\dot{q} = J^*(\dot{x}_d + \alpha e) - r$
- $M(q)\ddot{q} + V(q, \dot{q})\dot{q} + G(q) + F_r(\dot{q}) + T_d = \tau \dots \dots \dots (13)$
- On differentiating (11), substituting in (13) the robot dynamics can be written in terms of  $r$  as:  



$$M\dot{r} = -Vr + h(y) - \tau + T_d$$
 $h(y) = M \frac{d}{dt} [J^*(\dot{x}_d + \alpha e)] + V[J^*(\dot{x}_d + \alpha e)] + G(q) + F_r(\dot{q})$

where

$h(y) = M \frac{d}{dt} [J^*(\dot{x}_d + \alpha e)] + V[J^*(\dot{x}_d + \alpha e)] + G(q) + F_r(\dot{q})$   $y = \begin{bmatrix} x_d \\ \dot{x}_d \end{bmatrix}$

is termed as robot nonlinear function and we may choose  $y = [\ddot{x}_d^T \dot{x}_d^T x_d^T e^T \dot{e}^T]^T \in \mathbb{R}^{5n}$   $\dot{x}_d = \begin{bmatrix} \dot{x}_d \\ \ddot{x}_d \end{bmatrix}$

- During the controller development, we will make the assumption that the kinematic singularities are always avoided and the terms  $x_d, \dot{x}_d, \ddot{x}_d, M(q), V(q, \dot{q}), G(q), \tau_d, J(q)$  and  $J^*(q)$  are all bounded.



NPTEL ONLINE CERTIFICATION COURSE
8

Now, if you substitute, if you write the entire equation in terms of  $r$ . So,  $r$  is denoted by  $J$  plus  $\dot{x}_d$  plus  $\alpha e$  minus  $\dot{q}$  as we have seen earlier. And, then if we convert the dynamic equation is also written in this particular form as we have already seen. Now, what we do is we will write the dynamic equation in terms of  $r$  only, if we see that  $\dot{q}$  is the from this equation we will get  $\dot{q}$  is nothing, but  $J$  plus  $\dot{x}_d$  plus  $\alpha e$  minus  $r$ .

So, now, if we differentiate one more time we will get  $\ddot{q}$  from this equation and then if we substitute in this equation 13, the  $\ddot{q}$  value from here we can rewrite the whole equation in terms of  $r$  and its derivative. So, equation 13 can be converted into this equation  $M\ddot{q}$  into  $\dot{r}$  can be written as minus  $V$  time  $r$ . This  $V$  is there and we can convert it into this form plus  $h(y)$ .

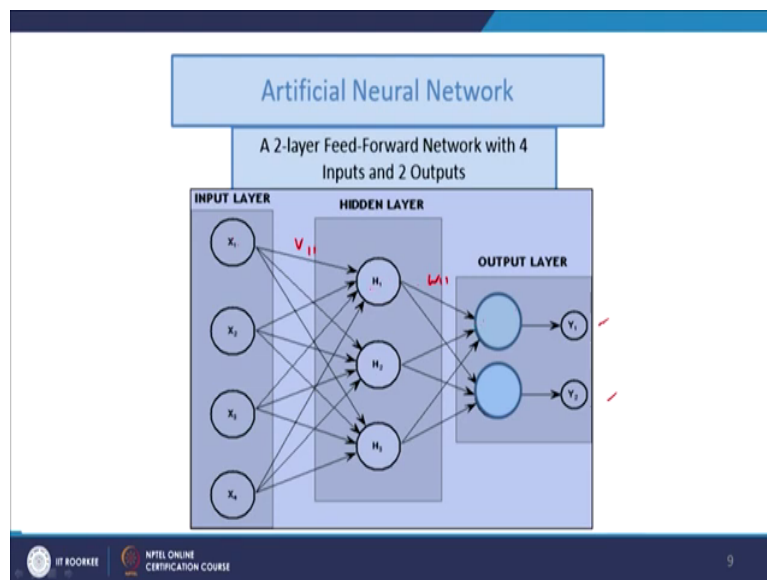
We are introducing a new term  $h(y)$  here minus  $\tau$  plus  $T$  suffix  $d$  which is already available here. This, if we take the  $\tau$  term in the left hand side we will get minus  $\tau$  plus  $T$  suffix  $d$ . So, this term can be verified directly by taking  $q$  double dot and substituting in this equation we get this arrangement, where the newly introduced term  $h$  of  $y$  is nothing, but this particular term.

So, how to check this particular thing? Substitute this  $h$  of  $y$  in this equation directly and then check that we will get the equation 13 directly. So, now this  $y$  variable is nothing, but a vector which contains  $x^d$  transpose  $x^d$  dot transpose  $x^d$  double dot transpose  $e$  transpose and  $e$  dot transpose. For example, if we write  $x^d$  it is a vector and  $x^d$  transpose is called a row vector.

So, whatever has been written here inside the bracket are all row vectors and then when we take the transpose of the whole thing, it becomes a column vector. So,  $y$  is a column vector whose size is because  $e$   $e$  dot  $x^d$  all of them have size  $n$ . If they are all belonging to  $\mathbb{R}^n$  space and there are 5 such components. So, it is nothing, but this belongs to  $\mathbb{R}^{5 \times n}$ .

The numbers the coordinates of  $y$  are of the size  $5n$  components are there for this thing. Now so,  $h$  of  $y$  is written in this particular expression, our aim is to because the error is to we want to make this error 0  $e$ . So, our aim is to make the error tending to 0 as  $T$  becomes larger and larger during the tracking.

(Refer Slide Time: 15:25)



So, how to use neural network for this purpose? We can design a controller if there is no disturbance term. This  $F$  term and  $T$  suffix  $d$  terms are not there in the equation or we say that it is a ideal situation. There is no disturbance then we know the dynamics of the equation and we can find a controller using some simple PD or PID controller as we have seen using the Lyapunov theory.

Here also we will use Lyapunov theory, but due to how to avoid this disturbance terms that is what we will see using artificial neural network. So, we have already seen what is artificial neural network in the previous lecture. So,  $x_i$ 's are the input and  $y_i$ 's are the output and the hidden layer is having this  $\mu_1 \mu_2$  this values. So, there are here 4 inputs and 3 hidden layer neurons and 2 output neurons.

(Refer Slide Time: 16:31)

## Artificial Neural Network

Consider a neural network with  $n$  – input  $\{x_1, x_2, \dots, x_n\}$ , with  $l$  neurons in the hidden layer and  $m$  – output  $\{y_1, y_2, \dots, y_m\}$ .  
Let  $u_{ij}$  be the weight connecting  $i^{th}$  input and  $j^{th}$  hidden neuron and  
 $v_{jk}$  be the weight connecting  $j^{th}$  hidden neuron and the  $k^{th}$  output neuron.


The value at the  $j^{th}$  hidden layer is given by


$$h_j = \sigma \left[ \sum_{i=1}^n u_{ij} x_i \right] \dots \dots \dots (A)$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

is a sigmoid function.

 IIT BOMBAY

 NPTEL ONLINE  
CERTIFICATION COURSE

10

So, and we have seen that how to write the relation between the input and the output. So, this sigma denotes the sigmoid function which we can take it one of the examples of sigmoid function is sigma of any variable  $z$  is  $1$  by  $1 + e$  power minus  $z$ ; its one of the sigmoid function. So, using this we can write the relation the  $u_{ij}$  are the  $u_{ij}$  for example,  $u_{11}$  is the weight joining this first neuron, first input to the first hidden layer neuron etcetera. So, similarly the  $v_{11}$  denotes the relation between the hidden layer to the output neuron.

(Refer Slide Time: 17:29)

## Artificial Neural Network

The value at the output  $y_k: k = 1, 2, \dots, m$  is given by:

$$y_k = \sum_{j=1}^l w_{jk} h_j$$

$$y_k = \sum_{j=1}^l w_{jk} \sigma \left[ \sum_{i=1}^n v_{ij} x_i \right] \dots \dots \dots (B)$$

Denoting  $X = [x_1 \ x_2 \ \dots \ x_n]'$  and  $Y = [y_1 \ y_2 \ \dots \ y_m]'$

$$V_{n \times l} = \{v_{ij}\}_{\substack{i=1,n \\ j=1,l}} \text{ and } W_{l \times m} = \{w_{jk}\}_{\substack{j=1,l \\ k=1,m}}$$

Equation (B) can be written as:

$$Y = W' \sigma(V'X) \dots \dots \dots (C)$$

So, that relation is written like this. So, output  $y_k$  has this particular term and where  $w_{jk}$  and  $v_{ij}$  all denotes the weights connecting various neurons. And, now this expression can be written in the matrix form like this if we denote the  $V$  matrix to be  $v_{ij}$  and  $W$  matrix to be  $w_{jk}$  as given here.

We can write capital  $Y$ , capital  $Y$  is nothing, but  $y_1, y_2$  etcetera  $y_m$ . This vector it is equal to  $W$  transpose this one into sigmoid function at the value  $V$  transpose  $X$  where capital  $X$  denotes the vector  $x_1, x_2, x_n$ . So, using this expression neural network expression we can approximate any given function of  $x_1, x_2, x_n$ .

(Refer Slide Time: 18:22)

### Function Approximation using Neural Network

Let  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$  be a continuous function defined on a closed and bounded set  $\Omega \subset \mathbb{R}^n$ , then there exists weight matrices  $W$  and  $V$  such that  $f(X)$  is approximated using neural network as in equation (C), i.e., for any given  $\epsilon > 0$ , there exists matrices  $U$  and  $V$  such that:

$$\|f(X) - Y\| < \epsilon$$

where  $Y$  is given by Equation (C).

NPTEL ONLINE CERTIFICATION COURSE 12

So, if  $f$  is a continuous function from  $\mathbb{R}^n$  to  $\mathbb{R}^m$ . So, function of  $n$  variables and its value is a vector in  $\mathbb{R}^m$ . Then that function can be approximated by a neural network of the form given in Equation C.

So, for any small value epsilon given we can find a neural network. So, that the difference between  $f$  of  $X$  and the neural network is very small; the given small value can be obtained. So, this is the famous theorem from the neural network approximation property for continuous functions.

(Refer Slide Time: 19:26)

### Neural Network Based Control Scheme with Adaptive Compensator



- Based on the above error system development, to accomplish the desired motion trajectory the following controller is proposed:
 
$$\tau = \hat{h}(y) + Kr + J^T e + v$$
- where  $\hat{h}(y)$  is an estimation of robot nonlinear function  $h(y)$ ,  $K$  is a positive definite gain matrix and  $v$  is an adaptive compensator defined later.
- With the controller the closed loop error dynamics becomes
 
$$M\dot{r} = -Vr - Kr + \tilde{h}(y) - J^T e - v + T_d$$
- where  $\tilde{h}(y) = h(y) - \hat{h}(y)$  is the functional estimation error. The functional estimation  $\hat{h}(y)$  with a feed-forward neural network (FFNN) may be given as:
 
$$\hat{h}(y) = \hat{W}^T \sigma(\hat{V}^T y)$$

Using this FFNN functional approximation, we can write

$$M\dot{r} = -Vr - Kr + \hat{W}^T \sigma(\hat{V}^T y) - \hat{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$

*h(y)*

*h = h - h*



13

Now, we can note that the disturbance which is occurring the friction or the external disturbance is during the tracking, they are assumed to be a continuous function. And we say, assume that it can be approximated by a neural network in this particular form as given in the expression C. So, now let us assume that the control tau, the torque tau is selected in this particular form where  $\hat{h}$  of  $y$  is the estimate of the  $h$  of  $y$ .

That is  $h$  of  $y$  is the disturbance which we have seen in the expression which we have introduced here. So, this contains the disturbance terms  $f$  the estimate of  $h$  of  $y$  using a neural network is written in the form like this,  $\hat{h}$  cap equal to  $\hat{W}$  cap transpose sigmoid function  $\hat{V}$  cap transpose  $y$ . So, here capital  $T$  denotes the transpose of the matrix  $\hat{W}$  cap. So, I think we can change here also here also we can write the  $\hat{W}^T$  denotes the transpose.

So,  $\hat{h}$  is the estimate of the  $h$  of  $y$  using a neural network expression. So, using the feed forward neural network function approximation we can write the equation as given here as  $\dot{r} = -Vr - Kr + \hat{h}$  but the  $\hat{h}$  value minus the estimate  $\hat{h}$ .

(Refer Slide Time: 21:39)

### Neural Network Based Control Scheme with Adaptive Compensator

- Adding and subtracting  $\hat{W}^T \sigma(\hat{V}^T y)$ , we get:
 
$$\dot{M}r = -Vr - Kr + \hat{W}^T (\sigma(V^T y) - \sigma(\hat{V}^T y)) + \hat{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$
- The Taylor series expansion of  $\sigma(V^T y)$  about  $\hat{V}^T y$  gives us:
 
$$\sigma(V^T y) = \sigma(\hat{V}^T y) + \sigma'(\hat{V}^T y) \tilde{V}^T y + O(\tilde{V}^T y)^2$$

With  $\sigma'(z) = \frac{d\sigma(z)}{dz} |_{z=z}$  and  $O(z)^2$  denoting terms of second order.

Denoting  $\tilde{\sigma} = \sigma'(\hat{V}^T y)$  we have:

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

$$\dot{M}r = -Vr - Kr + \hat{W}^T (\sigma(\hat{V}^T y) + \sigma'(\hat{V}^T y) \tilde{V}^T y - \sigma(\hat{V}^T y)) + \hat{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$

$$\dot{M}r = -Vr - Kr + \hat{W}^T (\sigma'(\hat{V}^T y) \tilde{V}^T y) + \hat{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$

So

$$\dot{M}r = -Vr - Kr + \hat{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y + \hat{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$

Now we can see that adding and subtracting  $\hat{W}^T \sigma(\hat{V}^T y)$  in this expression we note here  $\hat{W}$  and  $\hat{V}$  are appearing in one term,  $\hat{W}$  and  $\hat{V}$  are appearing in the second term in this 2 terms. Now, we are just making a combination  $\hat{W}^T \sigma(\hat{V}^T y)$ . This one we are adding and subtracting and then combining those terms we will get the next term like this.

Only we are changing the first and second terms are as it is and the last terms were also as it is we are because we are adding and subtracting we will get the terms to be in this particular

form. Now, we note that  $\sigma V^T y$ . So,  $\sigma V^T y$  it is a function of the  $\hat{v}_i$  and  $\hat{v}_j$  are the estimated values of the beats. So, if we write the  $\sigma V^T y$  as a Taylor series expansion about this estimated values.

So, we can write it as  $\sigma \hat{V}^T y$  plus the first derivative of this function at this point  $\hat{V}^T y$  multiplied by the  $V^T y$  minus we estimate this thing. So, for example, if we take a Taylor series expansion of a function  $f$  of  $x$  about a point  $x_0$ , we will get  $f(x)$  equal to  $f(x_0)$  plus  $f'(x_0)(x - x_0)$  plus etcetera. So, these are the higher order terms. So, similarly  $\sigma V$  is a function  $V$  is a matrix here and we are finding the Taylor series about the  $\hat{V}$  matrix plus its derivative term at the  $\hat{V}$  and the difference between the  $v$  matrix and the  $\hat{V}$  matrix terms.

So, we get  $\tilde{V}^T$  here and the higher order terms and we can omit the higher order terms and we can denote using this notations we can write the further terms. So,  $\dot{M}^T$  finally, can be converted into this particular form using this Taylor; if we substitute for example,  $\sigma V^T y$  for this term. So, this and this will get subtracted and we will get  $\sigma \dot{V}^T y$ , this is a matrix. When we differentiate a vector then we will get a matrix expression into  $\tilde{V}^T y$  as we have seen here plus the remaining terms are as it is.

(Refer Slide Time: 25:03)

## Neural Network Based Control Scheme with Adaptive Compensator

Finally adding and subtracting  $\tilde{W}^T \sigma'(\hat{V}^T y) \hat{V}^T y$  we get:

$$M\dot{r} = -Vr - Kr + \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y + \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y + \tilde{W}^T \sigma(\hat{V}^T y) - J^T e - v + T_d$$

$$M\dot{r} = -Vr - Kr + \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y + \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y - \tilde{W}^T \sigma'(\hat{V}^T y) \hat{V}^T y + w - J^T e - v$$

where modified disturbance term is:

$$w = \tilde{W}^T \sigma(\hat{V}^T y) + \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y + T_d$$

such that

$$\|w\| \leq \rho_1$$

So, further simplifying this expression we will get the terms  $M \dot{r}$  to be in this particular form. Now, what we want is to find the  $\tilde{W}$  cap and  $\tilde{V}$  cap,  $h$  of  $y$  itself is not known because it contains some disturbance. We want to estimate this unknown function using neural network. So, our aim is to find this  $\tilde{W}$  cap and  $\tilde{V}$  cap using some technique.

So, we have to follow a weight updation algorithm and as we have seen that all this functions which are involved in this disturbances and whatever expression we are finding they are all bounded. So, we can find a suitable constant  $\rho_1$  which is bounding this particular expression which is appearing in the equation.

(Refer Slide Time: 25:57)



### Neural Network Based Control Scheme with Adaptive Compensator

- In disturbance terms  $w$ , external disturbances or unmodeled dynamics  $T_d$  and the higher-order terms in the Taylor series expansion for  $\sigma$  all have exactly the same influence as disturbances. To eliminate the effect of these disturbances, we choose the stabilizing adaptive compensator as:

$$v = \frac{\rho_1^2 r}{\rho_1 \|r\| + \delta(t)}$$

where  $\dot{\delta} = -\gamma\delta$  with  $\delta(0) > 0$  and  $\gamma$  is a positive constant and  $\rho_1$  is suitable constant.

*Handwritten notes:*  
 $\delta(t) = e^{-\gamma t} \delta(0)$   
 $\rho_1$  is a bound



16

So, we take  $\rho_1$  as a bound, this can be taken to be a big number for example, we can take it to be 10 or 15, 20 any number depending on the situation. A particular robot manipulator or particular situation whether the situation has more disturbance or less disturbance etcetera; we can define or a priori estimate of this bound can be assumed to be a particular value.

Now, we will design this  $v$  expression which is appearing in the equation here. We have introduced this  $v$  expression in the equation before. So, everywhere we can see this equation in the dynamics of the, we have selected the control  $\tau$  to be like this  $\hat{h} \dot{y} - K r + J^T e + v$ . So, this is also called a design parameter which we want to design in such a way that the disturbances are removed. So, the design parameter  $v$  is selected to be like, this  $\rho_1$  constant we are assuming and in the previous slide this is a bound for this norm of  $w$ .

And,  $r$  is the vector which we have already introduced and the denominator we are introducing  $p-1$  into norm of this vector  $r$  plus  $\delta$ . So, this  $\delta$  is taken to be a differential equation  $\delta \dot{\phantom{x}}$  that is  $d\delta/dt$  equal to minus  $\gamma\delta$  with the assumption that  $\gamma$  this  $\delta$  of 0 is positive number and  $\gamma$  is a positive constant. So, we can easily see that if you solve this equation we will get  $\delta$  of  $t$  is nothing, but  $e$  to the power minus  $\gamma t$  into  $\delta$  of 0. So, if  $\delta$  of 0 is positive number as  $t$  becomes larger and larger  $e$  to the power minus  $\gamma t$  is becoming smaller and smaller it will tend to 0 as  $t$  tends to infinity.

So, that is the aim of selecting this  $\delta$ .  $\delta$  becomes 0 as  $t$  tends to these are all function of  $t$ . So, that is how we select the function  $\delta$ . Now, we substitute all this in the end. So, that the tracking is done in a proper way by avoiding all the disturbances by learning the disturbances using neural network. So, how to learn the weights? How to update the weights as the time progresses?

(Refer Slide Time: 29:12)



## Stability Analysis

- Here we give NN weight update algorithm that guarantees the stability of the system.
- It is required to show that the tracking errors are arbitrarily small and the estimates  $\hat{V}$ ,  $\hat{W}$  and  $\hat{\phi}$  remain bounded, so that the control  $\tau(t)$  is bounded.
- Theorem** : Consider the error dynamics model given previously for the robot dynamics. If control input torque  $\tau$  and adaptive compensator  $v$  are designed as given before and adaptive laws are selected as

$$\begin{cases} \dot{\hat{W}} = F_w \hat{\sigma} r^T - F_w \hat{\sigma}' \hat{V}^T y r^T \\ \dot{\hat{V}} = G_v y (\hat{\sigma}' \hat{W} r)^T \end{cases}$$

weight update algorithm

with some scalar positive definite symmetric matrices  $F_w, G_v$  and then the estimates  $\hat{V}, \hat{W}$  are bounded and task-space tracking error  $e$  asymptotically converges to zero.



17

So, we assume that the  $\hat{W}$  is updated in this particular way as given in this equation.  $\dot{\hat{W}}$  is the derivative of  $\hat{W}$  that is given by this expression and  $\dot{\hat{V}}$  is given by this expression; where this  $F$  matrix  $G$  matrix, they are all positive definite matrices. So, for simplicity we can take all these matrices to be a diagonal matrix with positive constants positive real numbers. And, this  $\hat{\sigma}$   $\hat{\sigma}'$  all these are defined already in this slide, here  $\hat{\sigma}$  is defined like this,  $\hat{\sigma}'$  is this and  $\hat{\sigma}'$  is denoted by this expression.

So, by substituting these expressions we see that there is a weight update algorithm, this is called weight update algorithm. So, as the time progresses we can keep on updating the weight using this particular procedure and how to prove that this particular procedure gives a correct weight for tracking the desired trajectory.

(Refer Slide Time: 31:00)

**Proof**

Consider the following Lyapunov Function Candidate:

$$L = \frac{1}{2} e^T e + \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T F_w^{-1} \tilde{W}) + \frac{1}{2} \text{tr}(\tilde{V}^T G_v^{-1} \tilde{V}) + \frac{1}{2} \text{tr}(\tilde{\phi}^T \Gamma_\phi^{-1} \tilde{\phi}) + \frac{\delta}{\gamma}$$

where  $\text{tr}(\cdot)$  represents the trace operator, differentiating above equation w.r.t t, we get:

$$\dot{L} = e^T \dot{e} + \frac{1}{2} r^T \dot{M} r + r^T M \dot{r} + \text{tr}(\tilde{W}^T F_w^{-1} \dot{\tilde{W}}) + \text{tr}(\tilde{V}^T G_v^{-1} \dot{\tilde{V}}) + \text{tr}(\tilde{\phi}^T \Gamma_\phi^{-1} \dot{\tilde{\phi}}) + \frac{\dot{\delta}}{\gamma}$$

Using (12) and (14)

$$\begin{aligned} \dot{L} = & -e^T \alpha e - r^T K r + e^T J r + \frac{1}{2} r^T (\dot{M} - 2V) r + r^T \tilde{W}^T \hat{\sigma}' \tilde{V}^T y \\ & + r^T (\tilde{W}^T (\sigma'(\tilde{V}^T y) \tilde{V}^T y - \sigma'(\tilde{V}^T y) \tilde{V}^T y)) + \text{tr}(\tilde{W}^T F_w^{-1} \dot{\tilde{W}}) + \text{tr}(\tilde{V}^T G_v^{-1} \dot{\tilde{V}}) \\ & + r^T W - r^T J^T e - r^T v - \delta \end{aligned}$$

*Handwritten notes on the slide:*  
 - Above the first equation:  $\|e\|^2$   
 - Above the second equation:  $\dot{e}, \dot{r}, \dot{\tilde{W}}, \dot{\tilde{V}}, \dot{\tilde{\phi}}, \dot{\delta}$   
 - Next to the last term of the second equation:  $\text{tr}(\Lambda)$  Sum diagonal  
 - Next to the last term of the third equation:  $r^T W$   $r^T J^T$   $r^T v$   $r^T W$   $r^T J^T$   $r^T v$   $r^T W$   $r^T J^T$   $r^T v$

So that can be seen here, we take a Lyapunov function L to be in this particular form, you can see that all of them are positive terms  $e^T e$  it is nothing, but norm of e square. And, this one is a positive term because M is a positive definite matrix and here F is a positive definite matrix therefore, this is a positive value. The trace, trace means the sum of the diagonal of this matrix all of them are positive values here also all these terms. Already we have seen that delta is a positive function and gamma is a positive number.

So, each and every term here all of them are positive values and at the 0 values, when we substitute all of them the e because it is a function of e r w tilde and v tilde etcetera. So, all the functions which are given here are positive and when we put all of them to be 0 we get the L value to be 0 at the origin. Now, we differentiate this with respect to time d L by d t and then substitute in the place of e dot in the place of r dot and then in the place of W cap dot and in

the place of  $V$  cap dot etcetera all the terms which we have already introduced in the previous slides.

So, substituting all the derivatives similarly  $\dot{\delta}$  also in the previous slide we have seen, all the substitutions and properly making use of the trace property. Trace of a matrix  $a$  is nothing, but sum of the diagonal elements now we can make use of one property. So, if we take  $r^T$  transpose  $s$ ; where  $r$  is let us say vector  $r_1, r_2, \dots, r_n$ . So,  $r^T$  transpose is a row vector and if  $s$  is  $s_1, s_2, \dots, s_n$ . So, we get a column vector  $s$  here. So,  $r^T s$  denotes  $r_1 s_1$  plus  $r_2 s_2$  etcetera. So, it is same as the trace of  $s$  and  $r^T$  transpose. If we take  $s$  is the column vector and  $r^T$  transpose is row vector, if we multiply will get a matrix.

And if we take the sum of the diagonal of this matrix it is same as nothing, but  $r^T$  transpose  $s$  itself.

(Refer Slide Time: 34:10)

**Proof**



Using Property 2 together with  $\tilde{\phi} = \phi - \hat{\phi}, \tilde{W} = -\hat{W}, \tilde{V} = -\hat{V}, \dot{\tilde{\phi}} = -\dot{\hat{\phi}}$  and adaptive laws, we get:

$$\begin{aligned} \dot{L} \leq & \underbrace{-e^T a e}_{\text{Property 2}} - \underbrace{r^T K r}_{\text{Property 2}} + \underbrace{r^T \tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y}_{\text{Property 2}} + \underbrace{r^T (\tilde{W}^T (\sigma'(\hat{V}^T y) \tilde{V}^T y - \sigma'(\hat{V}^T y) \hat{V}^T y))}_{\text{Property 2}} \\ & + \text{tr}(-\tilde{W}^T \sigma'(\hat{V}^T y) \tilde{V}^T y r^T + \tilde{W}^T \sigma'(\hat{V}^T y) \hat{V}^T y r^T) \\ & + \text{tr}(-\tilde{V}^T y r^T \tilde{W}^T \sigma'(\hat{V}^T y)) + \underbrace{\|r\| \rho_1}_{\text{Property 2}} - \underbrace{r^T v}_{\text{Property 2}} - \delta \end{aligned}$$

Using (15) and further simplifying, we get:

$$\dot{L} \leq \underbrace{-e^T a e}_{\text{Property 2}} - \underbrace{r^T K r}_{\text{Property 2}} + \underbrace{\|r\| \rho_1}_{\text{Property 2}} - \underbrace{r^T \left( \frac{\rho_1^2 r}{\rho_1 \|r\| + \delta} \right)}_{\text{Property 2}} - \delta$$

$r^T r = \|r\|^2$



NPTEL ONLINE CERTIFICATION COURSE
19

So, this property we will use here. So, they will get canceled automatically, many terms will get cancelled finally, we will get we are left with only terms like this. The derivative  $\dot{L}$  is nothing, but minus  $e^T \alpha e$  like this and then minus  $r^T K r$ . And, after canceling out all the terms using this particular property we get remaining terms to be only this much, these are the few terms and we are assuming that  $v$  is selected like this.

And, then if you cross multiply with these terms further it is canceled norm of  $r$  square into  $p_1$  square is canceled with  $r^T$   $r$  is nothing, but norm of  $r$  square. So, and multiplied by  $p_1$  square so, that is canceled here and we finally, get the terms to be sorry like this.  $\dot{L}$  we will get only this much, it is less than or equal to minus  $e^T \alpha e$  minus  $r^T K r$ .

(Refer Slide Time: 35:12)

## Proof

Simplifying further, we get:

$$\dot{L} \leq -e^T \alpha e - r^T K r$$

Using Rayleigh-Ritz theorem, we get:

$$\dot{L} \leq -\alpha_{\min} \|r\|^2 - \beta_{\min} \|e\|^2$$


where  $\alpha_{\min}$  and  $\beta_{\min}$  are the minimum eigenvalues of matrices  $\alpha$  and  $K$ , respectively.


Since  $L > 0$  and  $\dot{L} \leq 0$ , this shows stability in the sense of Lyapunov so that  $L, r, \hat{V}, \hat{W}$  and  $\hat{\phi}$  (hence  $\hat{V}, \hat{W}$  and  $\hat{\phi}$ ) are all bounded. Boundedness of  $r$  guarantees the boundedness of  $e$  and  $\dot{e}$  whence the boundedness of the desired trajectory shows  $x, \dot{x}, y$  are all bounded. Define a function

$$\Delta(t) = \alpha_{\min} + \beta_{\min} \|e\|^2 \leq -\dot{L}$$

Therefore,  $\dot{\Delta}(t)$  is bounded and hence  $\Delta(t)$  is uniformly continuous. From Barbalat's Lemma, we conclude that  $\Delta(t)$  goes to zero as  $t$  goes to infinity and hence  $\Delta(t)e$  and  $r$  converge to zero asymptotically. So, the sub-task tracking error  $e_v$  also goes to zero asymptotically.

$-(e^T \alpha e + r^T K r)$   
 $L > 0$   
 $\dot{L} < 0$




NPTEL ONLINE  
CERTIFICATION COURSE

20

So, this is nothing, but minus of  $e^T \alpha e$  plus  $r^T K r$ . So, this is a positive term because  $\alpha$  and  $K$  both are positive definite matrices. So, these two are positive value with a minus sign. So, we have that  $L$  is positive definite,  $\dot{L}$  is negative definite strictly less than 0.

So, this implies the system is asymptotically stable using the Lyapunov theory. So, what we get here is because of the Lyapunov theory we get the system which we have considered originally. The control system  $M \ddot{r} = -V \dot{r} + h(y) - \tau + T$  suffixed with this disturbance and the control is  $\tau$  is controlled. And, it tracks the desired trajectory using the control defined by this expression as given here and the neural network weights.

Because, the control involves a neural network expression and the weights are updated using this updated updating algorithm. And, we have proved that the system is asymptotically stable if we apply that particular control and hence it proves that the system tracks the desired trajectory successfully. As  $T$  progresses the trajectory is tracked. So, this is a lengthy procedure, even if we take a very simple example of a two link manipulator this may be a very complicated procedure.

And, this can be demonstrated only using a computer program suitably written for a neural network updating algorithm and assuming a particular function for the disturbance and then proving that the disturbances is avoided using the neural network etcetera.

So, with this I complete the lecture on the neural network based controller design.

Thank you.