INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 15: NN based Hybrid Force/Position Control of Robot Manipulator

Good morning, everyone. Today, we are going to see the lecture on Neural Network-Based Hybrid Force Position Control of Robotic Manipulators. The outline of this lecture will be as follows. First, we have the reduced order dynamic model of a robotic manipulator when it is in interaction with the environment. Then, we take this reduced order model represented in terms of filtered error and the non-linear function f of x. This non-linear function has to be estimated by a neural network, and hence, this is called neural network-based

force position controller. So, about that, we will be seeing after the representation of the reduced order dynamic model of a robotic arm with the filtered error R and with the unknown function, which has all the potential unknown parameters of a robotic arm. Then, we move on to the precise neural network-based position force hybrid controller of a robot manipulator, the design of that controller. Then, we have the stability analysis of this proposed neural network-based hybrid position force controller. Okay, so in the stability analysis, we will be seeing the assumptions pertaining to the boundary or the boundedness of the system parameters.

Boundedness. Okay. So, the boundedness or the assumptions pertaining to the boundedness of the system parameters, we will be seeing in the end. So, this is it. Now, we will start the lecture.

Now, let us write the dynamic model of a robotic system when it is in interaction with the environment, given by

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau + J^T(q)\rho$$

This is, say, equation number 1, which represents the dynamic model of a robotic arm when it is in interaction with the environment. Here, q(t) belongs to the n-dimensional Euclidean space, and J(q) is the Jacobian matrix. That is associated with the contact surface geometry, and rho is basically the contact force vector, you can say, exerted normal to the surface.

By the robot arm, and it is described in the coordinates relative to the surface. It is rho, a force vector which is represented as a vector of contact forces exerted normal to the surface, and it is described in the coordinate system relative to the surface. And tau d of t, as it is indicated with d subscript. So, it is representing disturbances. And we have

$$\tau_f = \underline{J^T(q)\rho}$$

So, this is basically the force in joint coordinates corresponding to rho(t). So, tau f is basically given by the expression J transpose into rho, which is basically the forces in the joints of the robotic system corresponding to the reaction force that is obtained from the environment or the contact surface. Now, let us see the force-constrained motion. And error dynamics, okay, error dynamics, constrained error dynamics pertaining to the constrained motion on a surface. Force-constrained motion and error dynamics, constrained motion on a surface. Let $x = f(q)$

be the forward kinematic relationship that relates the motion of the robot end effector in the Cartesian space by a given joint angular vector q. So, the prescribed surface can be described by the geometric holonomic.

Holonomic constraint equation that is given by $\phi(x) = 0$ where the phi function is defined in such a way that the equilibrium space is now reduced to the space of dimension m from n. The n belongs to the joint coordinate q and M is getting reduced. The order is now getting reduced to M by the constraint equation. Okay. That is phi of X. Now, the constrained Jacobian matrix $J(q) = \frac{\partial(\phi(f(q)))}{\partial q}$

And we must note that the normal velocity is given by $J(q)\dot{q} = 0$

due to the prescribed motion of the robot arm. Now, let us see the constraint equation. The constraint equation reduces the number of degrees of freedom to the number of degrees of freedom to $n_1 = n - m$

So, let the motion be in the plane of the surface, be described by the reduced position variable that the motion of the robot arm in the plane of the surface be described by the reduced position variable. That is $q_1(t)$ that belongs to the n_1 dimensional Euclidean space, okay. Then by an implicit function, one may relate q_2 to q_1 by the implicit function gamma that maps q_2 and q_1, which means they are related by the implicit function gamma where q2(t) belongs to the Euclidean space of dimension m, okay. So here, q2 is a dependent variable. $q_2 = \gamma(q_1)$

One must note that now the joint variable is given by $q = [q_1(t)^T \; q_2(t)^T]^T$

okay. Now we know where is q2 and where is q1. q2 is a dependent variable on q1, and the q2 q1 relationship is given by the implicit function gamma, okay. So now let us see

That the robot, constrained for motion along the surface, satisfies reduced-order dynamics in terms of, the variable q1, that belongs to the nth-dimensional space of the Euclidean space. Okay, so now let us define the extended Jacobian matrix. That is given by

$$L(q_1) = \begin{bmatrix} I_{n_1} \\ \frac{\partial \gamma}{\partial q_1} \end{bmatrix}$$

Again, I repeat, the robot is constrained to move in a plane or surface.

And that constrained motion reduces the dynamics of the robotic arm. And the reduced dynamic model is defined in terms of the variable q1, where q1 belongs to the n1-dimensional Euclidean space. Let us derive this reduced-order dynamic model of the robotic system in interaction with the environment.

$$L(q_1) = \begin{bmatrix} I_{n_1} \\ \frac{\partial \gamma}{\partial q_1} \end{bmatrix}$$

Where In1, as you can see, is an identity matrix of size n1 cross n1. Then the relationship between The tangential velocity, which is Q1 dot, to the full joint velocity Q dot is given by $\dot{q} = L(q_1)\dot{q}_1$

So, the extended Jacobian matrix relates the full velocity to the tangential velocity of the robotic system. we can have $\ddot{q} = L(q_1)\ddot{q}_1 + \dot{L}(q_1)\dot{q}_1$

Okay, so we can have these two expressions. Now, substituting all these expressions, substitute all the expressions into L. Equation 1, that is the dynamic model of the robotic system in interaction with the environment, we are getting

Sub. into (1).

$$B(q_1)L(q_1)\ddot{q}_1 + \underline{C(q_1,\dot{q}_1)\dot{q}_1} + F(\dot{q}_1) + G(q_1) + \tau_d = \tau + J^T(q_1)\rho$$

So now this is equation, say 2.

Now we can have this expression be

$$L^T B(q_1) L(q_1) \ddot{q}_1 + L^T C_1 \dot{q}_1 + L^T F + L^T G + L^T \tau_d = L^T \tau + L^T J^T \rho$$

$$J(q_1) L(q_1) = 0$$

This is a fact associated with the extended Jacobian matrix and the actual Jacobian matrix of the robotic system. So, we use this fact that leads to

$$\bar{B} \ddot{q}_1 + \bar{C}_1 \dot{q}_1 + \bar{F} + \bar{G} + \bar{\tau}_d = L^T \tau \quad \rightarrow \textcircled{3}.$$

where we used $J(q_1) L(q_1) = 0$

Now, this equation number 3 is the dynamic model of the robot motion

in the prescribed planar surface. With force information removed. So here, this equation we do not have the force information. That means the contact force information.

Now, let us get into the filtered error concept. Because we are going to derive or design the neural network-based hybrid position force controller using the filtered error-based approximation approach. So, let us talk about the filtered error concept.

The potential aim of the hybrid position-force controllers' problem is to make the robot arm travel a prescribed constrained surface while exerting a desired normal force to the surface and the desired motion trajectory

is in terms of Q_{1d} that belongs to the n_1 Euclidean space

and the prescribed force is given by rho d of t that belongs to the M-dimensional space

The force is normal to the contact surface, and q_{1d} represents the motion trajectory of the robot arm on the constrained surface. Okay, so now let us see. To design this hybrid position-force controller, we need to follow the filtered error approximation. Let us talk about the motion error, which is given by $e_m = q_{1d} - q_1$

where this equation is labeled as number 4. And the

Filtered, let us see what the equation is. Yes, number 4. Let the filtered error, error r, be given by $r = \dot{e}_m + \Lambda e_m$

That is the filtered error. Where λ is a positive diagonal matrix.

Positive diagonal matrix for the design of the controller. And let us define a force error given by $\tilde{\rho} = \rho_d - \rho$ that is given by equation number 6.

So ρ(t) here is the actual contact force that is normal to the contact surface, which is defined in the coordinate system attached to the contact surface. That is your actual force vector, ρ(t). Now, differentiating the filtered tracking error and substituting that back into equation 2, the reduced-order equation will be obtained.

You are ready for the dynamics in terms of R. That is the filtered error. That is given by

$$\bar{B}\dot{r} = -\bar{C}_1 r + L^T f(x) + L^T \tau_d - L^T \tau$$ Say this is equation number 7,

where the robot nonlinear function

$$\begin{aligned} f(x) = {} & B(q_1) L(q_1)\left(\ddot{q}_{1d} + \Lambda \dot{e}_m\right) \\ & + C_1(q_1, \dot{q}_1)\left(\dot{q}_{1d} + \Lambda e_m\right) \\ & + F(\dot{q}_1) + h(q_1) \rightarrow \textcircled{8} \end{aligned}$$

Okay. So, we can say that f(x) is represented by this, and it is basically the F function which has all the potential unknown parameters of the robot arm. Okay. Unknown parameters of the robot. So, here f(x), which means x is defined by as a vector that contains all the time signals to compute F of X. That is given by

$$x = \left[e_m^T \;\; \dot{e}_m^T \;\; q_{1d}^T \;\; \overset{\text{robot}}{\dot{q}_{1d}^T} \;\; \ddot{q}_{1d}^T\right]^T$$

So, this is the input to find the unknown function that contains all the unknown parameters of the robot term. Now, we are going to represent equation 1 in terms of the filtered error and the function f(x), the nonlinear function f(x), that is given by

$$\bar{B} L \dot{r} = -C_1 r + f(x) + \tau_d - \tau - J^T \rho \rightarrow \textcircled{9}$$ Okay. Rho is the contact force.

So, now, this representation of the interaction with the environment in terms of the filtered error r and the unknown function f(x) is done to prove the force error is very small with the control scheme proposed. Okay. Now this is done to prove the force error is very small with the proposed control scheme. Okay, so it is necessary to also work with equation one. Okay, so now this representation to bring the equation number nine is made in order to make the force error much smaller by the proposed control scheme.

Also, we must note that equation number 1 also has the force vector. So we need to work with that equation number 1 also, which contains the force information. So, which contains the force information. So I repeat clearly, equation 9 is obtained to make the force error much smaller by the proposed control scheme. So, we also need to work with equation 1 that has the force information. That is all.

Now, let us move into the neural network-based hybrid force-position controller for the n-dimensional robotic arm. Now, let us consider a two-layered neural network that is given by this schematic W2 transpose So, the weights connecting the input layer to the hidden layer, okay, this is the input layer, okay. Now, let us talk about what the network parameters are here.

The network parameters are W1 transpose and W2 transpose, where W1 transpose is basically the weight vector connecting the input layer to the hidden layer, and W2 transpose is the weight vector connecting the hidden layer, which is layer 1, to layer 2 or the output layer, and the output is represented by y_1, y_2 up to y_m. So, layer 1 is termed as the hidden layer, and layer 2 is basically the output layer for our case. So, let us consider this for the neural network-based position-force hybrid control.

So, here, sigma of X is basically the activation function involved here with the input X, which is nothing but X into W1 transpose. Okay, so the output from the hidden layer is basically the activation function sigma of X and W1 transpose. Okay, now the neural network controller is given

$$\tau = \hat{W}_2^T \sigma(\hat{W}_1^T x) + k_v L(q_1) r - J^T [\rho_d + k_f \tilde{\rho}] - v \quad \text{⑩}$$

by

and here W1 hat and W2 hat are the current weights of the actual neural network obtained by the training algorithm of the network parameters. And having given the control equation for the neural network-based controller, we can also see the neural network weight update law. That is given by

$$\dot{\hat{W}}_2 = F \sigma(\hat{W}_1^T x)(Lr)^T - F \hat{\sigma}' \hat{W}_1^T x (Lr)^T - k F \|(Lr)\| \hat{W}_2$$

and

$$\dot{\hat{W}}_1 = G x (\hat{\sigma}'^T \hat{W}_2 (Lr))^T - k G \|Lr\| \hat{W}_1$$

So, given the equation of the controller of the neural network-based hybrid position force controller and the network weight update law is also given here in these two equations W2 hat dot and W1 hat dot. Now, let us see the design parameters. What are the design parameters for the controllers are F and G. The design parameters here in the controller are F and G, which are nothing but the positive definite matrices. These are nothing but the positive definite matrices and kappa greater than 0 is a small parameter. Okay, it is a small parameter. Now, we have to see the robustifying term in the controller that is

$$v(t) = -k_z\left(\|\hat{Z}\|_F + Z_B\right) r$$

where the vector

$$\overset{\checkmark}{\hat{Z}} = \begin{bmatrix} \hat{W}_2 & 0 \\ 0 & \hat{W}_1 \end{bmatrix}$$

is the matrix which contains the overall weight estimate overall weight estimate that is W1 hat and W2 hat collectively called the overall weight estimate of the network that is represented by the matrix called z hat which is called the overall weight estimate matrix and we also have certain one more matrix that is called z tilde that is given by

$$\overset{\checkmark}{\tilde{Z}} = \begin{bmatrix} \tilde{W}_2 & 0 \\ 0 & \tilde{W}_1 \end{bmatrix}$$

as the first and second rows respectively is called the overall error matrix

So, there are two matrices associated with the robustifying term of the controller. That is, one is Z hat and Z tilde along with ZB, where ZB is a small constant meant for the boundedness of Z hat. Okay. Now, let us talk about the assumptions on the system parameters and desired trajectories. Now, let us see the assumptions on the system parameters, particularly on the boundedness of the system parameters and desired trajectories, so the first case is the desired trajectory boundedness. This is

$$\left\| \begin{matrix} q_{1d}(t) \\ \dot{q}_{1d}(t) \\ \ddot{q}_{1d}(t) \end{matrix} \right\| \leq \underset{\downarrow}{q_B}$$

where q_B is a known scalar bounded Likewise, we can have the matrix Z, which is defined in such a way that it is the matrix of ideal weight vectors of a neural network whose output is exactly equal to the target. So, that Z matrix is given

$$Z = \begin{bmatrix} W_2 & 0 \\ 0 & W_1 \end{bmatrix}$$

as the first and second row, where W1 and W2 are the ideal weight vectors, which are constant weight vectors of the ideal neural network that gives the output which is equal to the exact target. Okay, so here the ideal weights are target or constants and are bounded, so that the

$$\|Z\|_F \leq Z_B$$

So, Z_B, as I mentioned earlier, is a known bound. It is a known bound. Now, let us talk about an important topic, which is closed-loop error dynamics using neural network controllers.

Okay, so we have stated the neural network controller. Substitute the control equation into equation number 7. That means substitute the neural network controller into the reduced-order dynamics that is given by equation 7. We get closed-loop error dynamics. Dynamics. That is given by

$$\bar{B}\dot{r} = -L^T k_v L r - \bar{C} r + L^T \left[W_2^T \sigma(W_1^T x) - \hat{W}_2^T \sigma(\hat{W}_1^T x) \right] + L^T \left[\tau_d + \varepsilon + v \right] \rightarrow \text{(11)}$$

Okay, so this is equation number 11. Now, expanding from equation 11, we can have the expansion of, that is, expanding $\sigma(W_1^T x)$

in a Taylor series about sigma of W1 hat transpose into X and with some other manipulations, we get

$$\bar{B}\dot{r} = -L^T k_v L r - \bar{C} r + L^T \left[\tilde{W}_2^T (\hat{\sigma} - \hat{\sigma}' \hat{W}_1^T x) + \hat{W}_2^T \hat{\sigma}' \tilde{W}_1^T x \right] + L^T \left[w + v \right]$$

Here w(t) is the disturbance term that is given by

w(t) is the

$$w(t) = \tilde{W}_2^T \hat{\sigma}' W_1^T x + W_2^T \underline{O(\tilde{W}_1^T x)^2} + \tau_d + \varepsilon$$

Where $O(z)^2$ denotes the terms of order 2, In z, so which is nothing but O of W1 tilde transpose into X power 2 denotes the terms of order 2 in W1 tilde transpose into X. So here, the neural network weight estimation error These errors are given by

$$\tilde{W}_1 = W_1 - \hat{W}_1$$
$$\& \quad \tilde{W}_2 = W_2 - \hat{W}_2$$

Where W_1 is a constant, and W_1 hat is the estimated one obtained from the actual neural network. And also, we must know what sigma hat is. It is nothing but the activation function of

$$\hat{\sigma} = \underline{\sigma(\hat{w}_1^T x)}$$

$$\hat{\sigma}' = \left.\frac{\partial(\sigma(z))}{\partial z}\right|_{z = \hat{w}_1^T x}$$

Now, to bound W(t), that is the disturbance term, we can have this expression:

$$\text{bound } w(t)$$

$$\|w(t)\| \le C_0 + C_1 \|\tilde{Z}\|_F + C_2 \|\tilde{Z}\|_F \|r\|$$

That is, the Frobenius norm of Z tilde plus C2 multiplied by the Frobenius norm of Z tilde multiplied by the norm of R. With C0, C1, and C2 being computable constants expressed in various bounds. Say, C naught is a computable constant that is expressed in terms of various bounds. That is given by $C_0 = \varepsilon_N + d_B + c_0 Z_B$

where small c naught is a constant, a small constant. Okay, now let us see the final topic of today's lecture, which is the stability proof. Using a Lyapunov function stability analysis. So, let us consider the Lyapunov function stability analysis given by

$$\overline{L} = \frac{1}{2} r^T \bar{B} r + \frac{1}{2} tr\left[\tilde{w}_2^T \Gamma^{-1} \tilde{w}_2\right] + \frac{1}{2} tr\left[\tilde{w}_1^T G^{-1} \tilde{w}_1\right]$$

So, here F and G are symmetric positive definite matrices. And now, taking the time derivative of the Lyapunov function candidate, we get

$$\dot{L} = -r^T L^T k_v L r + tr\left(\tilde{w}^T\left[\Gamma^{-1}\dot{\tilde{w}} + (\hat{\sigma} - \hat{\sigma}'\hat{w}_1^T x)(Lr)^T\right]\right) + tr\left(\tilde{w}_1^T\left[G^{-1}\dot{\tilde{w}}_1 + x(\hat{\sigma}'^T \hat{w}_2 (Lr))^T\right]\right) + r^T L^T [w + v]$$

With assumptions $\dot{w}_1 = 0$, & $\dot{w}_2 = 0$ because W1, W2 are constants.

Now, we get

$$\dot{L} = -r^T L^T k_v L r + k_e \|Lr\| \left(tr\left[\tilde{W}_2^T (W_2 - \tilde{W}_2)\right] + tr\left[\tilde{W}_1^T (W_1 - \tilde{W}_1)\right] \right) + r^T L^T [w + v]$$

Okay, so now using various norm. Inequalities, okay, using various norm inequalities and the bounds of the disturbance term W of T, we get the boundedness of position error as.

LR bounded. Okay. So, $\overset{\checkmark}{\|Lr\|} > B_r$

where $$\tilde{B}_r = \frac{k_e (Z_B + C_1/k_e)^2/4 + C_0}{k_v} \checkmark$$

So, which means that. The position error can be made very small by increasing the gain KV. That is what it means. So, this is regarding the boundedness of the positional error. Now, let us talk about the boundedness of force error. That is rho tilde. So, now...

Substitute the neural network controller, say, given by equation number 10 into the error dynamics, that is, equation number 9, we get.

$$BL\dot{r} = -C_1 r + f + \tau_d - J^T \rho - \hat{W}_2^T \sigma(\hat{W}_1^T x) - k_v L r + J^T(\rho_d + k_f \tilde{\rho}) + v$$

Which $$BL\dot{r} = -k_v L r - C_1 r + \tilde{W}_2^T\left[\hat{\sigma} - \hat{\sigma}' \hat{W}_1^T x\right] + \hat{W}_2^T \hat{\sigma}' \tilde{W}_1^T x + J^T[I + k_f]\tilde{\rho} + (w + v).$$

Now, this equation can be written as $$\underline{J^T[I + k_f]\tilde{\rho}} = BL\dot{r} + k_v L r + C_1 r - \tilde{W}_2^T(\hat{\sigma} - \hat{\sigma}' \hat{W}_1^T x) - \hat{W}_2^T \hat{\sigma}' \tilde{W}_1^T x - (w + v) = B(\dot{r}, x, \tilde{Z})$$

So, now I am equating this to this B. So, now you can say that. From this equation, rho tilde equal to b, okay? So rho tilde is going to be given by b upon j transpose into i plus kf. But before that, we can have the pre-multiplication of this equation too, okay? So let us pre-multiply. The above equation by

$$J J^T [I + K_f] \tilde{\rho} = J B(\dot{r}, x, \tilde{z})$$

Therefore, we can say that $\tilde{\rho} = [I + K_f]^{-1} (J J^T)^{-1} J \, B(\dot{r}, x, \tilde{z})$

So what is implied here?

It is said that now with J J transpose being non-singular. So that the inverse can be possible. The force error is. Okay, it can be reduced much by increasing the force tracking gain, okay, that is Kf. And also we can say that the force error is bounded, okay.

So, with this, I come to the conclusion that We come to the conclusion that we have seen in today's class the dynamic model of the reduced order fashion or manner. Okay, the dynamic model in the reduced order case, the reduced order dynamic model of the robotic system in interaction with the environment, and we have seen the neural network-based hybrid force position controller. And then we have seen the stability analysis of the system or the controller. Now I can conclude more precisely that in this lecture we have seen the dynamic model of the robotic system while interacting with the environment.

Then we come to the reduced order dynamic model of the robotic system in terms of the variable Q1, where Q1 belongs to the dimension n1, where n1 is given by the reduced degrees of freedom n minus m, and through the implicit function Q2 equal to gamma of Q1, we have the reduced order dynamic model expressed in terms of function f of x and the filtered error term, and then we moved on to the neural network-based controller of the hybrid position force approach, where we have the update law for the weight parameters associated with the two-layered network being updated by the stability analysis involving Lyapunov analysis. Thank you so much.