INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 13: Neural Network based Feedback Linearization

Good morning, everyone. Today, we are going to see neural network-based feedback linearization control algorithms. The organization of this lecture will be as follows. First, we will see the gist of feedback linearization. What is feedback linearization?

We will first see. Then, we will see neural network-based feedback linearization, where there is a theorem to justify this control algorithm and the verification of this theorem. Then, we will see the neural network-based feedback linearization for f(x) known, which is basically a non-linear function associated with the controller being, sorry, this is being unknown, and g(x) being known. So, we are going to take a case where the non-linear function f(x) is unknown, and g(x) is known, with the approximation, the error approximation term being zero and being non-zero.

Under these cases, we are going to define, derive, or design the neural network-based feedback linearization approach. Let us start. First of all, let us talk about the feedback linearization approach. Let us consider a family of single-link or single-input, single-output affine systems Expressed by the given equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= f(x) + g(x)u \\ y &= x_1 \end{aligned}$$

Let this equation be number 1.

Which defines a family of single-input systems. Single-output affine system. Here, x equals x_1, x_2 up to x_n, which belongs to the n-dimensional Euclidean space, and y belongs to the one-dimensional Euclidean space, and u also belongs to the one-dimensional Euclidean space. So, the objective here For this feedback linearization-based controller approach, is that to find a control law such that X follows the desired given trajectory X desired. That is the objective of this. Controller. Now let us define a variable called filtered tracking error, which is represented by r, that is given by

$$r = e^{(n-1)} + \lambda_1 e^{(n-2)} + \cdots \lambda_{n-1} e$$

Here, as I mentioned, $\overset{(n-1)}{e} \Rightarrow (n-1)^{th}$ derivative of e and E is nothing but the output tracking error.

The output tracking error is given by Now, $e = y_{des} - y$ by choosing the control algorithm, choosing the control you can say precisely the control law, which is given

$$u = \frac{1}{g(x)}\left[-f(x) + k_v r + \lambda_1 \overset{(n-1)}{e} + \cdots + \lambda_{n-1} \overset{(1)}{e} + \dot{x}_{nd}\right]$$

That is the derivative of the desired state nth state. So, by choosing this control law, the closed-loop error dynamics become $\dot{r} = \underline{-k_v r}$ which implies that it is Stable and. Linear okay by choosing this control law. We have the closed loop. Aerodynamics. Becoming linear. As well as stable. That is given by this expression $\dot{r} = \underline{-k_v r}$

Where KV and the lambdas are chosen positive. Design parameters so that your r dot is negative, all right? And such an approach is called the feedback linearization control approach. Okay, so this is the fundamentals of neuro this. Feedback linearization. Now, let us talk about the condition when f(x) is unknown and g(x) is known. So, there is something with the feedback linearization algorithm.

Phase approach or technique. There is a problem associated when either the non-linear function f(x) is unknown, or g(x) is unknown, or both g(x) and f(x) are unknown. So, we go for the function approximators to estimate the unknown functions f hat or G hat. By neural networks approach or fuzzy systems. Okay again, I repeat. For the feedback linearization. There is a general problem. Which is associated when. Any of the nonlinear functions f(x), or g(x), or both are unknown. So, in that case.

We have to go for the function approximators. Okay. Such as neural networks or fuzzy systems in order to estimate the unknown function, and the estimated function is termed as f hat or g hat. Then the controller parameters can be updated in such a way that the closed loop Error converges to 0.

That is what it is. Again, I repeat, the controller parameters can be updated in such a way that the closed loop system error becomes 0 or converges to 0. So here, in this concept, let us assume that only f of x is Unknown, but g of x is known. So, the output of the neural network, if you consider a neural network in order to approximate the function which is unknown, then the output of the network is considered as the estimated value of f of x. So, the output of the network could be represented as f hat of x, where x is the input.

Here we approximate f of x by using a radial basis function network, which is a neural network. We consider that okay because it is simpler and also essential to estimate the unknown function for our particular motor. Okay, let us consider an n input and 1 output RBFN. Let us consider an n input and 1 output radial basis function neural network shown by which is x1 of k, xn of k, hidden neurons output neuron. The output is f hat of x(k), which is the approximation of the function f of x of k. Okay, so by this neural network, which is an n input and one output RBFN, we are approximating the unknown function f of x. So let us term this hidden layer by phi$_1$ phi$_2$ up to phi$_L$. So, the weights, the network weights will constitute a vector $\hat{W} \in \mathbb{R}^{L\times 1}$ Okay.

So the weight vector is associated with this. You can see. Okay. So this is the thing, and we all know that the output is basically the linear representation of the weights associated with the output of the hidden layers.

Okay. So now in this case we have the strong notion that the f of x is unknown, but g of x is known for the system that is expressed in the very first slide, which is system number one, given by the expression equation one, which is the input of a system. So let f(x) be approximated as $\hat{f}(x) = \underline{\hat{W}^T \phi(x)}$

using a radial basis function network. Because the output of the radial basis function network is given by the product of

Phi of X, which is the output of the hidden neurons multiplied by the weights connecting the hidden layer to the output layer. Okay. So, those are the estimated weights, which is why it is W hat. Okay. So, now the control law is given by

$$u = \frac{1}{g(x)}\left[-\hat{f}(x) + k_v r + \lambda e^{(n-1)} + \lambda_2 e^{(n-2)} + \cdots + \lambda_{n-1} e^{(1)} + \dot{x}_{nd}\right]$$

This control law will stabilize the system 1, given by equation 1, in the sense of Lyapunov stability. Stability analysis. Provided. W hat.

Is updated. Using. $\boxed{\dot{\hat{W}} = -\Gamma \phi r}$

Where the gamma is a positive stability definite matrix, okay? So we have to have the w hat to be updated at every instant, and the w hat dot is the update law here, okay? So that is the situation here in order to approximate the function f of x so that the network output is f hat of x, okay? So now let us see.

That a theorem where this control proof has been briefly proposed by this theorem, say it is Theorem 1, which is stating this control logic that is feedback linearization by an RBFN network. Theorem 1 states as let. The output tracking $e = y_{des} - y = x_{1d} - x_1$

Desired output of the system. Okay. First of all, in the theorem, let us define the output tracking error E, which is nothing but the discrepancy between the desired output minus actual output of the system, where the desired output is 1 x_{1d} and the actual output is x_1, y equal to x_1. Now, let us talk about the filtering error, the. Filtered tracking error is given by $r = e^{(n-1)} + \lambda_1 e^{(n-2)} + \cdots + \lambda_{n-1} e$ equation 2represents a stable dynamics.

Okay. Such that equation 2 represents stable dynamics by choosing

$\lambda_1, \lambda_2 \ldots \lambda_{n-1}$

And we know that n minus e power n minus 1 is given by the n minus 1th derivative of the tracking error e. Now, the objective here is to achieve satisfactory tracking performance as well as maintaining the boundedness of the weight vector of the neural network $\hat{W}$ & e

So, this is the objective for this control design, which is to achieve satisfactory tracking results, and we need to maintain the boundedness of the neural network weight vector W hat as well as the tracking error.

So, now let us continue this. So, in this case, the control law is defined as the control law is defined as

$$u = \frac{1}{g(x)}\left[-\hat{f}(x) + k_v r + \lambda_1 e^{(n-1)} + \lambda_2 e^{(n-2)} + \cdots + \lambda_{n-1} e^{(1)} + \dot{x}_{nd}\right]$$

See, this be equation number 3. Here, f hat of x is given by we know that $\hat{f}(x) = \hat{W}^T \phi(x)$

which is nothing but the response of the neural network. Neural network response or the output of the neural network. Let us assume that there exists an ideal weight vector w such that the original function f of x can be represented as $f(x) = W^T \phi(x)$

where w is the ideal weight vector corresponding to the original function.

Now, as per the properties, as per the approximation of the universal approximation property of neural networks, the approximation property of neural networks. For any smooth function represented by $f()$

there exist weights such that $f(x) = W^T \phi(x) + \varepsilon,$

where epsilon is called the estimation error. Okay. So, now it is assumed that the estimation error is 0.

It is assumed, as of now, that the estimation error is 0 for our current case. We are going to consider, at the end of this lecture, that the estimation error is non-zero. What will happen to the control law? Now, this assumption will hold for comparatively less complex non-linear functions. Non-linear functions which are less complex, comparatively.

So, if you take a sufficiently large number of adjustable weights, if you take a large number of adjustable weights. So, this assumption, where epsilon equal to estimation error is 0, holds true for non-linear functions which are less complex and there are sufficiently large number of adjustable weights associated with the network. Okay, now let us continue this with the approach that, now substituting the control law given by equation number 3 into the system, which is the original input affine system given by equation 1. We get

$$\dot{x}_n = f(x) + g(x)\cdot \frac{1}{g(x)}\left[-\hat{f}(x) + k_v r + \lambda_1 e^{(n-1)} + \lambda_2 e^{(n-2)} + \dots + \lambda_{n-1} e^{(1)} + \dot{x}_{nd}\right]$$

which is given further by, we know

$$W^T \phi(x) - \hat{W}^T \phi(x) + k_v r + \lambda_1 e^{(n-1)} + \lambda_2 e^{(n-2)} + \dots + \lambda_{n-1} e^{(1)} + \dot{x}_{nd} \rightarrow (4)$$

Okay guys, now let us continue this. We know that W tilde is given by the discrepancy between the ideal weight vector and the estimated weight vector.

So, again I go back W tilde which is nothing but the weight vector error is given $\tilde{W} = W^T - \hat{W}^T$

Therefore, We can write

$$\dot{x}_n = \tilde{W}^T \phi + k_v r + \lambda_1 e^{(n-1)} + \lambda_2 e^{(n-2)} + \dots + \lambda_{n-1} e^{(1)} + \dot{x}_{nd}$$

So, this equation can also be written in this way that $\dot{x}_{nd} - \dot{x}_n = e^{(n)} = -\tilde{W}^T\phi - k_v r - \lambda_1 e^{(n-1)} - \lambda_2 e^{(n-2)} \dots - \lambda_{n-1} e^{(1)} \rightarrow \textcircled{5}$

So, this be equation number 5. Now, by differentiating equation 2

Differentiating, eqn. $\textcircled{2}$:

$$\dot{r} = e^{n} + \lambda_1 e^{(n-1)} + \lambda_2 e^{(n-2)} + \dots + \lambda_{n-1} e^{(1)} \rightarrow \textcircled{6}$$

So, this be equation number 6. Now, substitute let us substitute

The nth derivative of the tracking error from the above equation number 6 into equation number 5. That

$$\dot{r} - \lambda_1 e^{(n-1)} - \dots - \lambda_{n-1} e^{(1)}$$
$$= -\tilde{W}^T\phi - k_v r - \lambda_1 e^{(n-1)} - \lambda_2 e^{(n-2)} \dots - \lambda_{n-1} e^{(1)}$$
$$(\text{or})\quad \dot{r} = -k_v r - \tilde{W}^T\phi \rightarrow \textcircled{7}$$

is equation number 7.

Now consider a Lyapunov function candidate. That is given by

$$V = \frac{1}{2} r^2 + \frac{1}{2}\tilde{W}^T\Gamma^{-1}\tilde{W} \rightarrow \textcircled{8}$$

Say this equation 8.

So here we know that gamma is a positive definite matrix. So, now let us continue this. Let us differentiate equation 8, that is, the Lyapunov function candidate is derived in order to give

$$\dot{V} = r\dot{r} + \tilde{W}^T\Gamma^{-1}\dot{\tilde{W}} \rightarrow \textcircled{9}$$

Now, substitute R dot from equation 7 into equation 9 into the equation 9.

We get $\dot{V} = r(-k_v r - \tilde{W}^T\phi) + \tilde{W}^T\Gamma^{-1}\dot{\tilde{W}} \rightarrow \textcircled{10}$ That is equation number 10 now. Okay.

Since W is a constant, $\dot{\tilde{W}} = -\dot{\hat{W}}$

Now,

$$\dot{V} = -k_v r^2 - \tilde{W}^T\phi r - \tilde{W}^T\Gamma^{-1}\dot{\hat{W}}$$
$$= -k_v r^2 - \tilde{W}^T\underline{\left(\phi r + \Gamma^{-1}\dot{\hat{W}}\right)} \rightarrow \textcircled{11}$$

That is equation 11.

Now, we can have the second term of equation 11 being equated to 0. This equation equated to 0, we have

$$\phi r + \Gamma^{-1}\dot{\hat{w}} = 0$$
$$\Rightarrow \dot{\hat{w}} = -\Gamma \phi r \rightarrow \boxed{12}$$

That is equation number 12.

This is the update law. So now, using this update law, which is given by equation 12, equation 11 becomes, by substituting this back, we have $\dot{V} = -k_v r^2 \rightarrow \boxed{13}$ that is equation number 13.

Okay. So, we can talk about this stability further in such a way that since $V > 0$ & $\dot{V} \leq 0$, it proves that

Stability of the system in the sense of Lyapunov ensures that R and W tilde are bounded. Thus, the verification of the theorem is complete. This is the proof of the theorem. Now, let us see furthermore. $\int_0^\infty -\dot{V}\, dt < \infty$

And v double dot is equal to, from v dot, $\ddot{V} = -2 r k_v \dot{r}$

And all the signals on the right-hand side of equation 7 verify that. The boundedness of R dot as well as V double dot.

Therefore, V dot is uniformly continuous. Thus, we can say V dot is uniformly continuous. Okay. So, according to Barbalat's theorem, V dot tends to 0 as T tends to infinity.

That is according to Barbalat's lemma, you can say. Since $\dot{V} = -k_v \underline{\underline{r^2}}$

in our case, V dot will be 0 and Only when r is 0. Okay. So, this is the situation.

Thus, we can say that r converges to 0 with factor time. Since equation 2 represents stable dynamics. A stable dynamics. Okay. The output tracks precisely the output tracking error e of t, which also vanishes with time t. Okay.

As we know that it is the filtered error of the tracking error, we can say that r converges to 0 as time tends to infinity. Equation 2 Represents stable dynamics as well. So, the output tracking error E also tends to 0 when time tends to infinity. Now, let us consider the last portion of this topic, which is nothing but inclusion. That means $\varepsilon \neq 0$

That means the RBFN approximation error is not equal to 0. When this is the case, what will happen? Let us now assume a non-zero. So, assume a non-zero RBFN approximation error epsilon, which is bounded by a small number, a small number.

which is a positive number epsilon n. Therefore, we can say that the original $f(x) = W^T \phi(x) + \varepsilon$

where epsilon is bounded by strictly bounded by epsilon n, which is bounded by the boundary or the positive number epsilon n. Okay, so with this modification, so with this modification the time derivative of the Lyapunov function candidate $\dot{V} = r(-k_v r - \tilde{W}^T \phi - \varepsilon) - \tilde{W}^T \Gamma^{-1} \dot{\hat{W}}$
Okay. So, now considering the same update law.

So, considering the same update law that is $\dot{\hat{W}} = -\Gamma \phi r$

we get $\dot{V} = -k_v r^2 - \varepsilon r$ Now V dot can be further expressed as $\dot{V} \leq -k_v r^2 + \| \varepsilon r \|$

or

$$\dot{V} \leq -k_v r^2 + \varepsilon_N \|r\| = -\|r\| (k_v \|r\| - \varepsilon_N)$$

Okay. So this is the case here. Now if $k_v \|r\| - \varepsilon_N$ is $-ve$,

The above expression will be. Negative as well. Will be. So, in other words, I can say K_v into the norm of r minus epsilon N is greater than 0 or the norm of R is greater than epsilon n by K_v. Okay. So, this implies that V dot is negative definite. As long as r norm. Greater than r norm. Greater than. Epsilon n by K_v. Okay. So that is the case. Which we can say that r is inside a ball of radius. Which is given by. Epsilon n by k_v. Thus by choosing k_v to be. Larger or can be made smaller, facilitating the tracking task. With this, I come to the conclusion that in this class, we have seen feedback linearization and then RBFN-based feedback linearization with f(x) unknown but g(x) known, with RBFN approximation error equal to 0 and not equal to 0. Thus, I can say that I am going to conclude now this lecture in such a way that we have seen The introduction to feedback linearization control algorithm, and then we have moved to neural network-based feedback linearization, where the neural network considered is a simple one-hidden-layer-based RBFN, where we are approximating the unknown function f(x) by the output response of the neural network. With the consideration of two cases. One is with the RBFN approximation error being 0, and the other case with the error being non-zero. Thank you so much for your time.