Proceeding Paper

Feedback Linearization Control of Nonlinear System †

Ivan Sergeevich Trenev *‡ and Daniil Dmitrievich Devyatkin ‡

V.A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, 65 Prosfoyuznaya Street, 117997 Moscow, Russia
* Correspondence: trenev.99@mail.ru
† Presented at the 15th International Conference “Intelligent Systems” (INTELS’22), Moscow, Russia, 14–16 December 2022.
‡ These authors contributed equally to this work.

Abstract: In this work, a neural network controller is developed for a wide class of nonlinear systems including dynamic systems in the Brunovsky canonical form and those with skew-symmetry properties and bounded nonlinearities. An example of the applicability of this controller to the control of the position of a magnet over an electromagnet is considered. The modeling has been provided through the Simulink environment.

Keywords: neural network; control; Matlab; Simulink

1. Introduction

Since linearization is based on accurate knowledge of the nonlinearities of the system, the applicability of its approaches to the control of real systems is severely limited. To guarantee the existence of a closed system solution, the nonlinearities must satisfy certain conditions, and for linearization methods for feedback control, system controllability is an important factor [1]. In what follows, several adaptive schemes that allow for linear parameter uncertainties are presented, which will help loosen constraints on model matching. Due to the fact that neural networks are acceptable approximators of nonlinear functions, we can assume that the linearity conditions will not be imposed on the system under consideration [2].

The feedback linearization algorithm is centered around geometric methods. However, due to the fact that this algorithm is based on exact knowledge of the nonlinearities of the system, the applicability of these approaches to the control of real systems is limited. To loosen the limitations of the exact model-matching restrictions, several adaptive schemes have been introduced that allow for linear parametric uncertainties [3]. Due to the properties of the universal approximation, NNs are used to calculate nonlinearities, which means that system parameters are not required [4].

For the sake of simplicity, all components of the state vector are assumed to be measurable. If only some components of the state vector are measurable, which corresponds to the case of an output feedback control; then, an additional dynamic neural network is required to evaluate non-measurable states.

2. Brunovsky Canonical Form

To begin, some definitions are given that are necessary for further reasoning. Let \( f, g : \mathbb{R}^n \to \mathbb{R} \) be the unknown smooth functions, \( x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n \) be the state vector, \( u \) be the control, and \( y \) be the output. The canonical Brunovsky form defines a special form of nonlinear dynamics of continuous time.
\[
\begin{align*}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= x_3, \\
&\quad \ldots \ldots \ldots , \\
\dot{x}_n &= f(x) + g(x)u, \\
y &= h(x).
\end{align*}
\] (1)

As shown in Figure 1, each integrator stores information. Each of them requires an initial condition [5].

**Figure 1.** Continuous-time SISO (one input, one output) Brunovsky form.

Consider a SISO linearizable state feedback system with an unknown disturbance. This system has a representation in the state space in Brunovsky’s canonical form (1). Then, the following assumption is true [6].

**Proposition 1.** (Bounds on disturbance and function \(g(x)\))

1. There exists a known upper bound \(b_d: \|\tilde{d}(t)\|_2 \leq b_d, \forall t \in \mathbb{R}^+;\)
2. Function \(g(x)\) satisfies the condition: \(|g(x)| \geq g_l > 0, \forall x \in \mathbb{R}^n.\)

3. **Tracking Controller and Error Dynamics**

   To begin with, let us form a tracking goal: for a given output \(y_d(t)\), it is necessary to find the control \(u(t) : y(t) = x_1(t)\) (i.e., \(h(x) = x_1\)). This condition implies a desired trajectory with acceptable accuracy, with restrictions on the state and control vector components. Feedback linearization will be used to track the output here. Let us introduce some assumptions to design the tracking controller. The desired trajectory vector will look like this [7]:

\[
x_d(t) = \begin{pmatrix} y_{d,1} & y_{d,2} & \ldots & y_{d,n}\end{pmatrix}^T.
\] (2)

For the state vector approximation error \(e = x - x_d\), define the filtered tracking error with its derivative as follows:

\[
r = \Lambda^T e, \quad \dot{r} = f(x) + g(x)u + d + Y_d, \quad (3)
\]

where \(Y_d = -x_d^{(n)} + \sum_{i=1}^{n-1} \lambda_i e_{i+1} = -x_d^{(n)} + \bar{\Lambda}^T \bar{e}\) is a known signal and \(\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{n-1})^T\) is a well-chosen vector of coefficients, \(\bar{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_{n-1})^T.\)

For both cases, when \(g(x)\) is known or unknown, define the control action in the following form:

\[
u_{\text{exact}} = \frac{1}{g(x)} (-f(x) - K_v r - Y_d),
\]

\[
u_c = \frac{1}{\hat{g}(\hat{\Theta}_g, x)} (-\hat{f}(\hat{\Theta}_f, x) + v), \quad v = -K_v r - Y_d.
\] (4)
where the estimates \( \hat{g}(\hat{\Theta}_g, x) \) and \( \hat{f}(\hat{\Theta}_f, x) \) are to be designed using two neural networks with appropriate weights matrices \( \Theta_g \) and \( \Theta_f \).

4. Neural Networks for Approximating Functions

Let neural networks be used to approximate unknown continuous functions. Then, the compact set \( \mathbb{R}^n \) has ideal target weights \( W_i \) and \( V_i \) \( (i = f \) for the case of a known \( g(x) \) and \( i \in \{ f, g \} \) for the case of an unknown \( g(x) \)):

\[
i(x) = W_i^T \sigma(V_i^T x) + \epsilon_i
\]

where the estimation errors are bounded \( \|\epsilon_i\| < \varepsilon_i \). Let the ideal weights be unknown and possibly not unique but satisfying the following assumption [8].

**Proposition 2.** (Restriction on the neural network target weights) Ideal neural network weights for functions \( f(x) \) and \( g(x) \) are bounded on any compact subset of \( \mathbb{R}^n \):

\[
\|\Theta_i\| \leq \Theta_{im}, \quad \text{where } \Theta_i = \begin{pmatrix} V_i & 0 \\ 0 & W_i \end{pmatrix}
\]

are the weight matrices for \( i, i \in \{ f, g \} \).

Then, using two neural networks, nonlinear functions \( f(x) \) and \( g(x) \), evaluations are obtained in the following form:

\[
\hat{i}(x) = \hat{W}_i^T \sigma(\hat{V}_i^T x), \quad \text{where } \hat{\Theta}_i = \begin{pmatrix} \hat{V}_i & 0 \\ 0 & \hat{W}_i \end{pmatrix}
\]

are the actual value of the weight matrix.

Using the expansion in a Taylor series, the functional errors of the estimate are as follows:

\[
\tilde{i}(x) = i(x) - \hat{i}(x) = \hat{W}_i^T (\hat{\Theta}_i - \Theta_i) \hat{V}_i^T x + \hat{W}_i^T \hat{\Theta}_i' \hat{V}_i^T x + w_i,
\]

where, for some \( C_j \), it is true that \( \|w_i(t)\| \leq C_0 + C_1 \|\hat{\Theta}_i\|_F + C_2 \|r\| \cdot \|\hat{\Theta}_i\|_F \). Estimates based on the boundedness of the standard activation functions together with their derivatives have the following form [7].

**Lemma 1.** (Bounds on state vector and approximated function)

a. The inequality restricting the state space vector is valid for the computed constants \( d_0 \) and \( d_1 \):

\[
\|x\| \leq d_0 + d_1 \|r\|;
\]

b. On any compact set, there are constants \( C_3 \) and \( C_4 \) that restrict the Euclidean norms of the approximated functions:

\[
\|\hat{i}(x)\| = \|W_i^T \sigma(V_i^T x) + \epsilon_i\| \leq C_3 + C_4 \|r\|.
\]

5. Controller Structure

Here, a tracking controller is designed for both cases, when \( g(x) \) is known or unknown. The main goal of this section is to construct a neural network that approximates the unknown function \( f(x) \) (and function \( g(x) \)) [9].

5.1. System with a Known Function \( g(x) \)

Figure 2 shows the resulting controller, described in Table 1.
The resulting controller with known function $g(x)$.

<table>
<thead>
<tr>
<th>Neural network controller:</th>
<th>$u = \frac{1}{g(x)}(-\dot{W}^T\sigma(\dot{x}^T x) - K_x r - Y_d + u_c)$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustifying Term:</td>
<td>$u_c = -K_c(|\hat{\Theta}| + \Theta_m)$.</td>
</tr>
<tr>
<td>Neural network weight update law:</td>
<td>$\dot{W} = M(\ddot{r} - \dot{\hat{V}}^T x) r - \alpha</td>
</tr>
</tbody>
</table>

**Parameters:**
- $\Lambda > 0$ — coefficient vector of the Hurwitz function;
- $M, N$ — positive definite symmetric matrices;
- $\Theta_m$ — bound on the unknown target weight norms;
- $K_c, K_x > 0, \alpha > 0$.

**Signals:**
- $e(t) = x(t) - x_d(t)$ — tracking error;
- $Y_d = -x_d^{(n)} + \Lambda^T e$ — desired trajectory feedforward signal;
- $r(t) = \Lambda^T e(t)$ — filtered tracking error.

Figure 2. Neural network controller with known function $g(x)$.

5.2. System with an Unknown Function $g(x)$

Figure 3 shows the resulting controller, described in Table 2.

Table 2. The resulting controller with unknown function $g(x)$.

| Neural network controller: | $u = \begin{cases} \frac{u_t + u_c - u_e}{2} e^{-g(|u_c| - s)} & \text{if } g \geq g_t \text{ and } |u_c| \leq s, \\ u_t - \frac{u_c}{2} e^{-g(|u_c| - s)} & \text{otherwise.} \end{cases}$ |
|---------------------------|-----------------------------------------------------------------|
| Robustifying Term:        | $u_c = \frac{1}{W^T f \sigma(e^T)}(-\dot{W}^T \sigma(\dot{V}^T x) + v)$. |
| Neural network weight update laws: | $\dot{W}_f = M_f(\ddot{r} - \dot{\hat{V}}^T x) r - \alpha |r| M_f \dot{W}_f$, $\dot{\hat{V}} = N_f x \dot{W}^T \theta' - \alpha |r| N_f \dot{V}$. |

**Signals:**
- $e(t) = x(t) - x_d(t)$ — tracking error;
- $Y_d = -x_d^{(n)} + \Lambda^T e$ — desired trajectory feedforward signal;
- $r(t) = \Lambda^T e(t)$ — filtered tracking error.

**Parameters:**
- $K_N > 0, K_2 > \max\left\{ \frac{C_2}{s \gamma}, \frac{C_l}{s \gamma \Theta_m} \right\}, \alpha > 0, s > 0, \gamma < \ln \frac{2}{s}, \mu \geq 2$;
- $\Theta_m$ — bound on the unknown target weight norms;
- $\Lambda > 0$ — coefficient vector of the Hurwitz function;
- $M_f, N_f$ — positive definite symmetric matrices.
6. Modelling

Consider the system shown in Figure 4 and a magnet suspended above an electromagnet. The magnet can only move in the vertical direction. The purpose of the control is to control the position of the magnet. The equation for motion in the described system is

\[
\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M} i(t) - \frac{\beta}{M} \frac{dy(t)}{dt},
\]

where \(g\) is the standard gravitational acceleration, \(i(t)\) is the current flowing in the electromagnet, \(y(t)\) is the distance above the electromagnet, and \(M\) is the mass of the magnet. The parameter \(\beta\) is the coefficient of viscous friction, and \(\alpha\) is the field strength constant.

Figure 5 shows the results of modeling this system in the presence of two neural networks. (Red graph—desired trajectory; blue graph—approximation using neural networks.) The simulation was carried out in the Matlab/Simulink environment using additional neural network packages [10].

Figure 4. Electromagnet system.

Figure 5. Simulation result.
7. Conclusions

In this paper, we showed how to design neural network controllers for systems in the Brunovsky form for the case of a known input influence function and the case of an unknown input influence function. For the case with an unknown input influence function, it was necessary to make certain efforts to ensure that control remained bounded. Based on the simulation results presented in the last section, we can conclude that neural networks are a universal approximator and can be used to simulate the behavior of a real object.

Author Contributions: These authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.