Adaptive Control for Haptics with Time-Delay

D. Richert, C.J.B. Macnab, and J.K. Pieper

Abstract—This paper presents an adaptive haptic control for a one degree-of-freedom surgical device. The control addresses the problem of hitting a solid object too hard in the presence of time delay. The proposed control runs in the inner-loop, with no time delay, and follows commanded forces from the outer loop. A Lyapunov-stable backstepping-with-tuning-functions design provides a way to ensure smooth forces are applied that guarantee stability in the presence of unmodeled environmental stiffness. The method naturally becomes a velocity-tracking system when no forces are measured, without need for a switching control law. Experiments using a Phantom hand controller interacting with simulated environment show that collision forces are substantially reduced. The overshoot during a puncture, when moving from a stiff environment to free space, is not worse than with other designs.

I. INTRODUCTION

Developing haptic control methods for teleoperation interests the research community [1], [2], [3], and telerobotic surgery is an important new application. Typical implementation allows the surgeon to command position by moving a hand-controller, and this same hand-device provides force-feedback to the surgeon based on a slave-side force measurement. Automatic control on the slave side tracks commanded position, but it is up to the surgeon to achieve a desired force.

Significant communication delays can exist even when master devices are located in close proximity to the surgical robot, due to practical factors like force-sensor filtering and slow digital communication rates. Automatic position-tracking will try to achieve a commanded position even after the robot has hit a hard surface. It is left to the operator to feel the surface and pull back on the control, which may occur too late to stop a hard collision in the presence of time delay. Damage can result, especially to sensitive and expensive force sensors. This scenario typically occurs during tool-exchange, although could occur when contacting bone as well.

Consider that a commanded force would be less than measured force when the slave hits a hard surface. Thus, a force-tracking control would automatically stop the motion. For this reason we propose a force-force arrangement (Figure 1). A carefully considered control design allows the system to provide control in all other situations as well, including in tissue-puncturing and free-space scenarios.

Although passivity-based approaches are popular for haptic control since they do not require a model of the environment [4], [5], [6], [7], [8], controller design choices are restricted by the difficulties in proving stability. Here we propose a neural-adaptive control that can adapt to unknown and changing environments. A Lyapunov-backstepping design of the control force derivative ensures smooth applied forces. The tuning-function design for weight updates results in fast transient response, so that on-line adaptation to new environments without repetitive training is possible. The control uses an auxiliary error of both force-error and velocity in the feedback control. This naturally reduces overshoot during a puncture and allows one to control the velocity in free space when no forces are measured. There is no need to switch to a different control law in these situations, although we realize the surgeon may indeed want the ability to switch to position-based control when no collisions are imminent (depending on the feel of the resulting control).

An experiment performed with a Phantom hand controller, interacting with a time-delayed virtual environment, verifies that the proposed method improves the performance when hitting a hard surface, yet can still handle punctures and moving in free space. Tests shows the proposed method outperforms both output feedback control and an $\mathcal{H}_2$ control.

II. BACKGROUND

A. Simplified Model for Control Design

We consider a one-degree-of-freedom (DOF) problem. This could represent an endoscopic tool, but multi-DOF surgical robots also typically have an “axis-lock” function where the user can restrict the motion to a desired linear direction. The model of the device pushing through tissue (Figure 2) is

$$M \ddot{x} = -D_r \dot{x} - D_m \dot{x} - K(x)x + F_c,$$  (1)

that includes translational position $x(t)$, robot mass $M$, robot damping coefficient $D_r$, applied actuator force $F_c(t)$, environmental stiffness $K(x)$, environmental damping coefficient $D_m$. Real tissue will display viscoelastic (time-varying) stiffness and damping properties [9]. This gives us further incentive to design an adaptive control law, which could account for time-dependent quantities if adaptation is fast enough.

A force sensor at the end of the tool measures

$$F_m = D_m \dot{x} + K(x)x.$$  (2)

Since the backstepping technique will treat the derivative of force as the control signal, filtering the force-sensor output may be unnecessary. Moreover, allowing the operator to feel high frequencies could improve the quality the haptics [10].

Authors are with the Schulich School of Engineering, University of Calgary, Calgary, Alberta, Canada
Corresponding author Chris Macnab: cmacnab@ucalgary.ca
III. PROPOSED CONTROL

The reference force

\[ F_d(t) = K_s F_h(t - T), \]  

(3)

comes from the force \( F_h \) applied by the human operator, but scaled by \( K_s \) and time-delayed by \( T \) seconds. The force error becomes

\[ \epsilon = F_m - F_d, \]  

(4)

The control uses auxiliary an error that also includes the velocity

\[ s = \Lambda \epsilon + \dot{x}, \]  

(5)

where \( \Lambda \) is a positive constant. Adding the velocity term both slows the device just after a sudden puncture and allows control in free space. Note that if a control can achieve \( s \equiv 0 \) then the auxiliary error becomes a sliding-mode

\[ \dot{x} = -\Lambda \epsilon. \]  

(6)

When in contact with material, \( F_m \neq 0 \), the sliding mode becomes

\[ \dot{x} = -\Lambda (F_m - F_d), \]  

(7)

\[ = -\Lambda K(x) \dot{x} - \Lambda D_m \dot{x} + \Lambda F_d, \]  

(8)

\[ = \frac{-\Lambda K(x)}{1 + \Lambda D_m} \dot{x} + \frac{\Lambda}{1 + \Lambda D_m} F_d, \]  

(9)

which is a controllable system with input \( F_d \). In the free moving case where \( F_m = 0 \) equation (6) becomes

\[ \dot{x} = -\Lambda F_d. \]  

(10)

Then velocity is directly proportional to commanded force input, and we assume the surgeon could learn to achieve a desired velocity.

A. Radial Basis Function Networks

Gaussian radial basis function (RBF) networks can approximate an unknown nonlinear function of \( n \) inputs, \( f(q) \) with \( q \in \mathbb{R}^n \), such that

\[ f(q) = \phi(q) w + d(q), \]  

(11)

Row vector \( \phi(q) \) contains \( m \) Gaussian RBFs and column vector \( w \) has \( m \) ideal weights resulting in approximation error \( d(q) \) bounded by positive constant \( d_{\text{max}} \) in a region \( \mathcal{D} \subset \mathbb{R} \)

\[ |d(q)| < d_{\text{max}} \forall q \in \mathcal{D}. \]  

(12)

The proposed method will not require accurate approximation of nonlinear functions. Instead, performance will follow from fast adaptation based on output error. The output from the RBF network is

\[ \hat{o} = \phi(q) \hat{w}, \]  

(13)

where weights \( \hat{w} \) are estimates of ideal weights \( w \), resulting in weight error

\[ \hat{w} = w - \hat{w}. \]  

(14)

In general this paper uses similar notation, denoting errors \( \hat{\epsilon} = (\cdot) - (\cdot) \).

B. Adaptive Backstepping Control Design

A Lyapunov backstepping technique provides the control design, detailed in the Appendix. We outline the approach in this section. An RBF network approximates terms

\[ \Lambda(K(x) + K\dot{x}) - M^{-1}(\Lambda D_m + 1)\dot{x} = \phi_c(x, F_d, s)w_c + d_c, \]  

(15)

Note that the providing the network with \((x, F_d, s)\) is equivalent to providing \( x \) and \( \dot{x} \). The virtual control design, for desired force, is

\[ \alpha = F_m + \hat{p}^{-1}(-\phi_c \hat{w}_c + \Lambda \tilde{F}_d - G_1 s), \]  

(16)

with \( G_1 \) a positive gain. We do not actually apply this force, but rather will design derivative of force as a control in order to achieve a smoother signal. The virtual control error is

\[ z = F_c - \alpha, \]  

(17)

An additional adaptive parameter \( p \) approximates

\[ M^{-1}(\Lambda D_m + 1) = p + d_p. \]  

(18)

An additional RBFN final neural network models

\[ \tilde{F}_m(x, \dot{x}, F_c) = \phi_c(x, \dot{x}, F_c)w_c + d_c. \]  

(19)

The derivative of applied force is the control signal

\[ u = \tilde{F}_c = \hat{\dot{x}} - s \hat{p} - G_2 z, \]  

(20)

so that the actual force applied on the robot is

\[ F_c(t) = \int_0^t \tilde{F}_c(r) dr, \]  

(21)

Differentiating analytically gives

\[ \dot{\alpha} = \phi^{\alpha} (\hat{w}_c) - \hat{p}^{-2} (\phi^{\alpha} \hat{w}_c + \Lambda \tilde{F}_d - G_1 s) \]  

\[ + \hat{p}^{-1} \left[-\phi^{\alpha} \hat{w}_c - \left(\frac{\partial \phi^{\alpha}}{\partial \beta} \frac{\partial \beta}{\partial \dot{F}_d} \right)\hat{w}_c \right. \]  

\[ + \Lambda \tilde{F}_d - \left(\frac{\partial \phi^{\alpha}}{\partial s} \hat{w}_c + G_1 \right) \hat{s}], \]  

(22)

3651
where

$$\dot{s} = -\Lambda \hat{F}_d + \phi_x \hat{w}_x + \hat{p}(-F_m + F_c).$$

(23)

Robust weight update laws are:

$$\dot{\hat{w}}_x = \beta_x \left[ \phi_x^T \left( \frac{\partial \phi_x}{\partial s} \hat{w}_x + G_1 \right) \hat{p}^{-1} z + \tau_x - \nu_x \hat{w}_x \right],$$

(24)

$$\dot{\hat{w}}_a = \beta_a \left( -\phi_a^T z - \nu_a \hat{w}_a \right).$$

(25)

where positive constants $\beta_x, \beta_a$ provides adaptation rates and $\tau_x$ is the tuning function

$$\tau_x = \phi_x^T s.$$  

(26)

The leakage terms containing positive constants $\nu_x, \nu_a$ and ensure the boundedness of the weights. A similar parameter update

$$\dot{\hat{p}} = \beta_p \text{Proj} \left[ \left(-F_m + F_c\right) \left( \frac{\partial \phi}{\partial s} \hat{w}_x + G_1 \right) \hat{p}^{-1} z + \zeta \left( \hat{p} - \bar{p} \right) + \tau_p \right]$$

(27)

where $\tau_p = s$, uses projection operator

$$\text{Proj}[\cdot] = \begin{cases} 0 & \text{if } |p| < |p|_{\text{min}} \text{ and } \cdot < 0 \\ \cdot & \text{otherwise}, \end{cases}$$

(28)

Projection ensures invertibility of $\hat{p}$, and $|p|_{\text{min}}$ indicates a predetermined desired minimum value. The term $\zeta \left( \hat{p} - \bar{p} \right)$ serves as a supervisory learning term, improving performance of neural network weight updates that include control multiplied by state error [11]. The value of $\bar{p}$ remains constant, our best estimate expected or average value of $p$.

It is important to note that these update laws occur online without any expectation of pre-training. Thus, the effectiveness of the neural networks in the controller follows from adaptation speed rather than their accurate modeling of the system. Although performance will improve with repetition, it is not a requirement for to achieve suitable performance.

IV. EXPERIMENT

The experimental setup consists of a combination of simulation and real hardware. A human operator supplies the force trajectory through a master haptic hand controller. However, the slave device and environment exist in simulation.

A PHANTOM Omni haptic device manufactured by Sens- Able (Figure 3) supplies the master device. The Omni provides force feedback in 3 dimensional space X, Y, Z.

Positional sensing is provided in 6 degrees (X, T, Z and roll, pitch, yaw) from digital encoders with a positional resolution of 0.055 mm. Dimensions of 160 x 120 x 70 mm define the workspace. An IEEE-1394 FireWire port connects the device to a PC, allowing for fast communications between the PC and the device. A stylus located at the end effector has an apparent mass of 45 g. The motors have a backdrive friction of 0.26 N. The device exhibits a maximum stiffness of 2310 N/m.

Communication to the device occurs through Quanser’s QuaRC control software solution, using a PHANTOM Omni blockset in MATLAB’s Simulink environment. QuaRC fully supports Simulink’s external mode, including scopes, online parameter tuning, and data logging directly to the MATLAB workspace. The Omni Simulink block sends force commands to the Omni’s motors and receives x, y, z encoder positions as well as roll, pitch, yaw positions. QuaRC allows communication with the Omni at sample frequencies up to 1000 Hz, as used for this work.

Our design of the proposed controller ensures a straightforward extension to a multiple degree-of-freedom haptic setup, however this paper only tests the one degree of freedom case. As such, throughout the experiment two proportional controllers lock the haptic device onto the (x, 0, 0) line in the usable workspace.

The Omni end effector does not have a force sensor, yet we want the human to be able to provide a desired force. We design a virtual force sensor in software, based on the actual stylus position $x_{\text{haptic}}$ compared to a nominal (constant) position $x_0$

$$F_{\text{haptic}} = K_{\text{haptic}}(x_{\text{haptic}} - x_0),$$

(29)

where chosen parameter $K_{\text{haptic}}$ determines the amount the stylus will move. Since the human feels the measured force supplied through the haptic hand controller, the commanded force must be the addition of these terms

$$F_d(t) = F_m(t - T) + K_{\text{haptic}}(x_{\text{haptic}}(t) - x_0).$$

(30)

The commanded force remains constant when the human keeps the position at $x_0$. We emphasize the single-direction time-delay $T$ seconds, since the force error measured by the inner control loop at time $t$ is

$$e(t) = F_d(t - T) - F_m(t),$$

$$= F_m(t - 2T) + K_{\text{haptic}}(x_{\text{haptic}}(t - T) - x_0) - F_m(t).$$

(31)

(32)
Thus, if there was no communications delay \((T = 0)\) the force error would simply be a scaled position error.

Slave dynamics consist of a mass-damper system in one-dimension with driving force \(F_c\) and opposing force \(F_m\). Table I shows the nominal parameters used in the experiments. Communication delays are added to the system so that any information transported from the master to the slave (or vice-versa) undergoes a time delay of 0.1 seconds, for a total time delay of \(T = 0.2\) seconds. Two scenarios test the proposed method. In the first the user pushes through a medium with constant force before hitting a solid surface: a stiff contact test. In the second, the user pushes through a medium with constant force and suddenly punctures through into free space: a loss of contact test.

Two alternative controllers provide a comparison: an output feedback controller (OFC) and an \(H_2\) controller (H2C). Under usual operation, the OFC contains no inner feedback control. However, if the system becomes active a feedback control aims at stabilizing the system. The applied force comes from a sum of the human command and an 

Passivity-based control from [12]. For the H2C, the system states become force errors and a simple gain scheduling routine updates the control gains with increasing measured force. Only the proposed method and the OFC appear in the loss of contact test because the controller gains become infinite for the H2C when free motion is encountered.

During the experiment, the human operator moves the tool from left (starting at \(x = 0\)) to right in the force profile (Figure 4). The human tries to move the tool to \(x_t\) (wall or puncture point) within 3 or 4 seconds. Because a human provides the desired force trajectory, there are inherent differences between tests. However, repeated experimentation concludes that the controller responses do not vary significantly, as long as the environment remains constant.

V. RESULTS

A. Collision: Stiff contact test

In the first experiment, the tool moves through a medium with \(K_e = 20\) N/m and then hits a solid surface at modeled as a highly stiff environment with \(K_e = 3000\) N/m (Figure 4). The human operator attempts to keep the force constant during this test. Simply changing the stiffness models an elastic collision, an appropriate test as it offers a worst-case scenario for system stability. Colliding with a long thin tool during tool-exchange offers a realistic example of an elastic collision with a solid object.

The proposed controller stops immediately after contact compared to 0.15 s for the OFC and nearly 2 s for the H2C (Figure 5, top two rows of graphs). Moreover, the proposed controller hits the surface with a maximum force of only 4.1 N, which is 21 N less than the OFC and 118 N less than the H2C (Figure 5, bottom three rows of graphs). We conclude that the proposed method avoids large collision forces that could cause damage.

In addition to hitting with less force, the proposed method avoids the high frequency oscillations seen with the other two methods. Thus, the proposed method offers better stability properties, avoiding excitation of higher order dynamics in the stiff environment.

B. Puncture: Loss of contact

In the second experiment, the tool moves through a medium with \(K_e = 20\) N/m and then encounters no force measurement after position \(x_t\). The human operator attempts to keep the force constant in the medium and attempts to stop the robot altogether in free space. The new method succeeds in slowing down the robot after the loss of contact (Figure 7, top graph), and we conclude that adding a velocity term in the auxiliary error was appropriate. One can see the velocity actually goes negative immediately after the puncture using the new method, unlike OFC (Figure 7, second row of graphs). This results in only 1 cm overshoot immediately after the puncture for the proposed method, compared to 5 cm overshoot for the OFC. (The slowly-moving position offset in the top graph of Figure 7 after the 4 second mark is due to human variance in tracking during the free-space stage, well after the puncture has occurred.)
C. Weight Convergence

One common problem found in neuro-adaptive control systems is the bursting phenomenon, where the state errors suddenly jump to large magnitudes after a period of apparent convergence [13]. This may be an acceptable risk in some applications, but not with surgical robots. Bursting is due to weights drifting to large magnitudes, both positive and negative, which tend to cancel out each other at first but eventually cause a disruptive control signal. The leakage terms, multiplied by constants \( \nu_l \), \( \nu_n \) in (24) and (25), are well known to prevent bursting if the constants are chosen large enough [14]. Additionally, our supervisory term multiplied by constant \( \zeta \) in (28), a term closely related to the well known leakage term, works to prevent drift of the adaptive parameter \( p \). To verify the weights are prevented from drifting to large values over time, we apply a preset desired force of

\[
F_h = F_m + 0.2N
\]

(33)
to the control in the stiff-contact scenario, repeated for 200 trials. Results confirms that convergence of the neural network weights is achieved (Figure 8).

VI. CONCLUSIONS

An adaptive backstepping approach with tuning functions provides an inner-loop for a force-force bilateral control designed for haptic systems with time-delay. Using backstepping allows a design of force-derivative as the control signal, resulting in a smooth applied force. The feedback is based on an auxiliary error including force error and velocity, which provides a way to control the robot in free-space as well as contributing to automatic slowing in a puncture situation. Utilizing a neural network allows the method to adapt to unmodeled environments. The method was tested with a one degree-of-freedom scenario utilizing a hand-controller experiment interacting with a simulated environment. The proposed design achieves tracking of a desired force in the presence of significant time delay, yet can still slow the mechanism down automatically when puncturing or colliding with a solid surface. Moreover, the collision occurs with so little force that damage would be unlikely, unlike with the resulting forceful collisions found when using \( H_2 \) or a passivity-based output feedback control. Future work will include experimental verification with a real slave robot.

APPENDIX

A positive-definite Lyapunov candidate function for the first step of backstepping is

\[
V_1 = \frac{1}{2} s^2 + \frac{1}{2\beta_p } \tilde{p} + \frac{1}{2\beta_k } \tilde{w}_c^T \tilde{w}_c.
\]

(34)
The time derivative is

\[
\dot{V}_1 = s \dot{s} - \frac{1}{\beta_p } \ddot{p} \tilde{p} - \frac{1}{\beta_k } \tilde{w}_c^T \dot{\tilde{w}}_c.
\]

(35)
Expanding \( \dot{s} \) gives

\[
\dot{s} = \Lambda \dot{c} + \ddot{x},
\]

(36)
\[
= \Lambda(\dot{F}_m - \dot{F}_d) + M^{-1}( -D_c \dot{x} - F_m + F_c ),
\]

(37)
\[
= \Lambda(\dot{K} x + K \dot{x} - \dot{F}_d) + (\Lambda D_m + M^{-1})( -D_c \dot{x} - F_m + F_c ).
\]
Subbing in the neural network model (15) and adaptive parameter (18) gives

\[
\dot{s} = -\Lambda \dot{F}_d + \phi_c \tilde{w}_c + d_c + (p + d_p)( -F_m + F_c ),
\]

(38)
and the Lyapunov time derivative becomes

\[
\dot{V}_1 = s \left[ -\Lambda \dot{F}_d + \phi_c \tilde{w}_c + d_c + (p + d_p)( -F_m + F_c ) \right]
\]

\[- \ddot{p} / \beta_p - \tilde{w}_c^T \dot{\tilde{w}}_c / \beta_k. \]

(39)
Introducing the virtual control \( \alpha \) and the virtual control error \( \tilde{z} \),

\[
\dot{V}_1 = s \left[ -\Lambda \dot{F}_d + \phi_c \tilde{w}_c + \delta_1 + \tilde{p}( -F_m + z + \alpha ) \right]
\]

\[+ \ddot{p}_c( -F_m + F_c ) - \dot{\tilde{p}} / \beta_p + \tilde{w}_c^T( \phi_c^T s - \tilde{w}_c / \beta_k), \]

(40)
where uncertainties are contained in

\[
\delta_1 = d_c + d_p( -F_m + F_c ). 
\]

(41)
Using the designed virtual control (16), tuning function definitions and weight error results in
\[
\dot{V}_1 = s \dot{p} z - G_1 s^2 + s \dot{\delta}_1 + \dot{p}(\tau_p - \dot{\delta}_p) + \hat{w}_T^T (\tau_c - \hat{\dot{w}}_{kc}/\beta_k).
\]  
(42)

An over-parameterized design would use these functions as the weight/parameter updates immediately. When using tuning functions, choice of weight update occurs at the next step. The second backstepping stage requires a final Lyapunov function
\[
V_2 = V_1 + \frac{1}{2} z^2 + \frac{1}{\beta_c} \hat{w}_T \hat{w}_o.
\]  
(43)

Taking the time derivative gives
\[
\dot{V}_2 = s \dot{p} z - G_1 s^2 + s \dot{\delta}_1 + \dot{p}(\tau_p - \dot{\delta}_p) + \hat{w}_T^T (\tau_c - \hat{\dot{w}}_{c}/\beta_c) + z (\dot{F}_e - \hat{\dot{\alpha}}) - \hat{\dot{w}}_T \hat{w}_o/\beta_o.
\]  
(44)

Substituting in the designed \(F_c\),
\[
\dot{V}_2 = -G_1 s^2 + s \dot{\delta}_1 + \dot{p}(\tau_p - \dot{\delta}_p) + \hat{w}_T^T (\tau_c - \hat{\dot{w}}_{c}/\beta_c) - z \hat{\dot{\alpha}} - \hat{\dot{w}}_T \hat{w}_o/\beta_o.
\]  
(45)

Determining \(\hat{\dot{\alpha}}\) is largely an algebraic process and is omitted from the stability proof. The reader may easily verify the proof by understanding that \(\hat{\dot{\alpha}}\) contains the error between unknown terms in \(\hat{\dot{\alpha}}\) and the estimates of these respective terms in \(\hat{\dot{\alpha}}\) (the expression (23) is also required). Since many of the terms in \(\hat{\dot{\alpha}}\) are known there are a lot of terms that drop out of \(\hat{\dot{\alpha}}\). The final result is that
\[
\dot{V}_2 = -G_1 s^2 - G_2 z^2 + s \dot{\delta}_2 + z \hat{\dot{w}}_o + \hat{w}_o \left( \frac{\partial \dot{w}_o}{\partial \dot{s}} z - \hat{\dot{w}}_{o}/\beta_o \right)
\]
\[
+ \hat{\dot{p}} \left( (-F_M + F_e) (\dot{\phi}_1^T \frac{\partial \phi}{\partial s} \hat{w}_c + G_1) \hat{p}^{-1} z + \tau_c - \hat{\dot{w}}_{c}/\beta_c \right)
\]
\[
+ \hat{w}_T^T \left( \frac{\partial \phi}{\partial s} \hat{w}_c + G_1 \right) \hat{p}^{-1} z + \tau_c - \hat{\dot{w}}_{c}/\beta_c,
\]  
(46)

where uncertainties have been included in
\[
\delta_2 = d_o + \left( \frac{\partial \phi}{\partial s} \hat{w}_c + G_1 \right) \hat{p}^{-1} (-F_M + F_e) d_p + d_o.
\]  
(47)

A final substitution of the weight (or parameter) update laws results in the complete Lyapunov function
\[
\dot{V}_2 = -G_1 s^2 - G_2 z^2 + s \dot{\delta}_2 + z \hat{\dot{w}}_o + \nu_\alpha \hat{\dot{w}}_o \hat{w}_o.
\]  
(48)

If one can establish limits on the forces the uncertainties become bounded by |\(\delta_k\)| \(\leq \delta_{\text{max}}\), where \(\delta = [\delta_1 \ \delta_2]^T\) and \(\delta_{\text{max}}\) is a positive constant. Then defining \(G = \min(G_1, G_2), \ e = [s \ \dot{z}]^T\) and \(\mathbf{w} = [p \ \mathbf{w}_T \ \mathbf{w}_o]^T\) results in
\[
\dot{V}_2 \leq -G\|e\|^2 + \delta_{\text{max}}\|e\| + \nu\|\mathbf{w}\|\|\dot{\mathbf{w}}\| - \nu\|\mathbf{w}\|^2,
\]  
(49)
implying \(\dot{V}_2 < 0\) outside a compact set on the \((\|e\|, \|\dot{\mathbf{w}}\|)\) plane. This result guarantees uniformly ultimately bounded (UB) signals.

If one cannot establish bounds on the forces, utilizing nonlinear damping terms in the (virtual) control terms results in bounded signals [15]. A practical way to obtain bounds on the forces is to saturate them in the system. The tuning-function method, however, achieves a large degree of inherent robustness because the sign of the weight update is no longer directly related to the sign of the output error (as in over-parameterized designs). This tends to limit weight drift in practical application.

**References**


