Review and Analysis of Event Triggered Neuro-Adaptive Controller Design for Uncertain Affine Nonlinear Systems

Nathan Lutes

***Abstract*—The literature reviewed in this work details a neural network (NN) adaptive controller that differs from a traditional controller in that the updates to the control and NN occur only when a certain condition is met. This type of controller is commonly referred to as an "event triggered" controller and the goal is to minimize data transmission while still maintaining desired performance. In this case, the event condition is based on the error between current states and the last state values used to update the controller and NN weights. Furthermore, it is assumed that only a certain number of states could be available for measurement so the controller design is adapted to include a Modified State Observer (MSO). An MSO is a NN augmented observer with the ability of learning the system uncertainty online. The benefit of training the NN using an observer is that it separates the training loop of the NN from the control loop, allowing for faster training and a smooth application of control. This work will also cover the Lyapunov stability analysis included in the literature which proves the stability of the controller and explains the update rule for the NN. Finally, the paper concludes by discussing the numerical simulations in the literature performed on a benchmark 2-link robotic manipulator. The paper being referenced is [1].**

I. Introduction

For a standard, periodic controller the system's state is measured continuously and the controller is updated with each measurement. However, for an event triggered controller, the state measurement used by the controller is only updated when a certain condition is met, typically when some measure of error gets above a certain threshold. The goal of this design is to mitigate the communication cost between the measurement mechanism and the controller. For modern, large-scale, complex systems, this can be very beneficial. Furthermore, it is often the case that some system states are unavailable for measurement and that the system dynamics are very complicated. The controller formulation in the reviewed literature tackles this problem by implementing a NN augmented observer (MSO). A neural network is a machine learning algorithm capable of estimating unknown functions to a certain degree using measured data. The observer uses the known system dynamics information along with the NN's estimation of the uncertainty to estimate the unknown states. As more data is collected, the uncertainty estimation increases in accuracy causing the system model to converge to the true model and thus causing the state estimation to converge to the true state values. The controller then uses the state estimation in its computation of the control effort. By separating the learning mechanism from the control input by including it in the estimation loop using MSO, the neural network is able to learn faster and decrease the adverse affects on the control input.

The hallmark of the event triggered design is using only the state information transmitted during the previous event. In the reviewed literature, both the controller and the NN follow this scheme. Let the current measurement of the state be denoted as $X(t)$ and then the state measurement from event t_i (where t_i denotes the time instant for the i^{th} event) is denoted as $X(t_i)$. Until the next event occurs, the previous sampled state is held using a zero-order hold or simply the value does not change. The error used to define when the next update should occur is formulated as:

$$e_{evt} = X(t) - X(t_i) \tag{1}$$

Note that the error is reset every time an event sample occurs.

II. Event Triggered Controller Design

Consider the dynamics of a nonlinear system with nominal control input that it is desired for the plant system to follow:

$$\dot{X}_d = f^*(X_d(t)) + Bu_n(t_i) = F^*(X_d, u_n) \tag{2}$$

where X_d is the desired state of the system and f^* and B are known. Furthermore, consider the actual dynamics of the plant system defined as:

$$\dot{X} = f(X(t)) + Bu(t_i) + d(X(t)) \tag{3}$$

where f and B are known and $d(X(t))$ is the uncertainty in the system. These dynamics can be rewritten in event sampled form as:

$$\dot{X} = f(X(t_i)) + Bu(t_i) + \check{f}(X(t), X(t_i)) + d(X(t_i)) \tag{4}$$

where $\check{f}(X(t), X(t_i)) = f(X(t)) - f(X(t_i))$ is the error between the nominal system dynamics and the event sampled dynamics.

To estimate any unavailable states and facilitate a framework with which to train the neural network, the MSO is defined as:

$$\dot{\hat{X}} = f(\hat{X}(t)) + Bu(t_i) + \hat{d}(X(t_i)) - K_2(X(t_i) - \hat{X}(t)) \tag{5}$$

where the uncertainty estimation $\hat{d}(X(t_i))$ is an estimate of the true uncertainty: $d(X(t_i))$. Assuming a linear parametrization, $d(X(t_i))$ is expressed as:

$$d(X(t_i)) = W^T \Phi(X(t_i)) + \check{\epsilon} \tag{6}$$

where W is the matrix of optimal weights, $\Phi(X(t_i))$ is a vector of basis functions utilizing the last updated state measurement and $\check{\epsilon}$ is the residual error of the approximation. Since the true weights are unknown, they are replaced with a matrix of estimates which are updated strategically such that they converge to the optimal weights. The weight estimates multiplied by the vector of basis functions make up the uncertainty estimate. More precisely:

$$\hat{d}(X(t_i)) = \hat{W}^T \Phi(X(t_i)) \tag{7}$$

The weight update law used in the reviewed paper is:

$$\dot{\hat{W}} = \gamma Proj_m(\hat{W}, \Phi(X(t_i))\check{e}_r P) \tag{8}$$

where $\gamma \in [0,1]$ is the learning rate, $Proj_m$ denotes the smooth projection operator, $\check{e}_r = X(t_i) - X_d(t)$ and P is the solution to:

$$-Q = K_2^T P + P K_2 \tag{9}$$

where $Q > 0, P > 0$.

Now the controller can be appropriately defined. Due to space constraints much of the derivation will be omitted, however the final equations will be presented and discussed. The equation for the event-triggered controller is given as:

$$u(t_i) = B^{-1}\Big(F^*\big(X_d, u_n(t_i)\big) - K\big(\hat{X}(t) - X_d(t)\big) + K_2\big(X(t_i) - \hat{X}(t)\big) - \hat{d}(X(t_i)) - f(\hat{X}(t))\Big) \tag{10}$$

where $F^*\big(X_d, u_n(t_i)\big)$ denotes the dynamics of the desired trajectory where u_n is the nominal control, K and K_2 are user-defined gain matrices, $\hat{d}(X(t_i))$ is the neural network estimation and $f(\hat{X}(t))$ is the nonlinear system dynamics using the state estimate. Note that this equation requires B to be invertible. If B is not invertible, two slack variables can be introduced, B_s and u_s, such that $\overline{B} = [B, B_s]$ is invertible. Then the controller equation becomes:

$$\begin{bmatrix} u(t_i) \\ u_s \end{bmatrix} = \overline{B}^{-1}\Big(F^*\big(X_d, u_n(t_i)\big) - K\big(\hat{X}(t) - X_d(t)\big) + K_2\big(X(t_i) - \hat{X}(t)\big) + B_s u_s - \hat{d}(X(t_i)) - f(\hat{X}(t))\Big) \tag{11}$$

Finally, the last part of the controller is defining the event trigger condition. The reference literature defines this as:

$$\|e_{evt}\|^2 \leq \alpha\beta\|e_r\|^2 \tag{12}$$

where $\alpha \in (0,1)$ is a user defined parameter and β is defined as:

$$\beta = \frac{\sigma}{1 + (\|P\|(\|L_\Phi\| + \|L_f\|))^2} \tag{13}$$

and $\sigma \in \mathbb{R}$. α can be adjusted to cause more or fewer events to occur by determining how large $\|e_{evt}\|$ must be before the condition is triggered. Simply, α can be used to trade stability for reduced communication cost.

III. Stability Analysis

In this section, the Lyapunov stability proof is presented and shows that the tracking error is uniformly ultimately bounded (UUB). Due to the nature of the controller, there exists two different cases that must be analyzed separately. These cases are: at the sampling instant and during the inter-time event between the last sampling instant and the next. The cases must be regarded individually because updates only happen at the sampling instant. Unfortunately, due to space constraints, some derivations, such as the error dynamics, must be omitted but are included in the proof in certain locations.

Consider the first case at the sampling instant. In this case, $e_{evt} = 0$ and $e_r = \check{e}_r$ where $\check{e}_r = X(t_i) - X_d$. Consider the Lyapunov candidate function:

$$L(e_r, \tilde{W}) = e_r^T P e_r + tr(\tilde{W}^T \gamma^{-1} \tilde{W}) \tag{14}$$

Then its derivative is:

$$\dot{L}(e_r, \tilde{W}) = 2e_r^T P \dot{e}_r + 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}}) \tag{15}$$

$$\begin{aligned}
&= 2e_r^T P(K_2 e_{evt} + K_2 e_r - K_1 \hat{e}_r + \tilde{W}^T \Phi(X(t_i)) + \check{\epsilon} + \tilde{f}(X(t), \hat{X}(t))) + 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}}) \\
&\text{Consider that } \tilde{W} = W - \hat{W} \text{ then } \dot{\tilde{W}} = -\dot{\hat{W}}. \text{ Further } e_{evt} = 0 \text{ in this case, therefore:} \\
\dot{L} &= 2e_r^T P(K_2 e_r + \tilde{W}^T \Phi(X(t_i)) + \check{\epsilon} - K_1 \hat{e}_r + \tilde{f}(X(t), \hat{X}(t))) - 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}}) \\
&= 2e_r^T P K_2 e_r + 2e_r^T P \tilde{W}^T \Phi(X(t_i) + 2e_r^T P \check{\epsilon} - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P \tilde{f}(X(t), \hat{X}(t)) - 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}}) \\
&= e_r^T K_2^T P e_r + e_r^T P K_2 e_r + 2\check{e}_r^T P \tilde{W}^T \Phi(X(t_i)) + 2e_r^T P \check{\epsilon} \\
&\quad - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P \tilde{f}(X(t), \hat{X}(t)) - 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}}) \\
&= e_r^T (K_2^T P + P K_2) + 2\check{e}_r^T P \tilde{W}^T \Phi(X(t_i)) + 2e_r^T P \check{\epsilon} \\
&\quad - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P \tilde{f}(X(t), \hat{X}(t)) - 2tr(\tilde{W}^T \gamma^{-1} \dot{\tilde{W}})
\end{aligned} \tag{16}$$

Using the projection operator for the weight update law: $\dot{\hat{W}} = \gamma Proj_m(\hat{W}, -\Phi(x(t_i))\check{e}_r^T P)$ and $K_2^T P + P K_2 = -Q$:

$$\begin{aligned}
\dot{L} &= -e_r^T Q e_r + 2\check{e}_r^T P \tilde{W}^T \Phi(X(t_i)) + 2e_r^T P \check{\epsilon} \\
&\quad - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P \tilde{f}(X(t), \hat{X}(t)) \\
&\quad - 2tr(\tilde{W}^T \gamma^{-1} \gamma Proj_m(\hat{W}, \Phi(x(t_i))\check{e}_r^T P)) \\
&= -e_r^T Q e_r + 2\check{e}_r^T P \tilde{W}^T \Phi(X(t_i)) + 2e_r^T P \check{\epsilon} \\
&\quad - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P \tilde{f}(X(t), \hat{X}(t)) \\
&\quad + 2tr(-\tilde{W}^T Proj_m(\hat{W}, \Phi(x(t_i))\check{e}_r^T P))
\end{aligned}$$

$$= -e_r^T Q e_r + 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t)) + 2tr(-\tilde{W}^T(Proj_m(\hat{W},\Phi(X(t_i))\check{e}_r^T P) + \Phi(X(t_i))\check{e}_r^T P\tilde{W}^T - \Phi(X(t_i))\check{e}_r^T P\tilde{W}^T)) \quad (17)$$

Consider the trace properties: $tr(A+B) = tr(A) + tr(B)$, $tr(A) = tr(A^T)$ and $tr(AB) = tr(BA)$ if $A \in \mathbb{R}^{nxm}$ and $B \in \mathbb{R}^{nxm}$ for any $n, m \in \mathbb{N}_+$. Then, (17) can be rewritten as:

$$\dot{L} = -e_r^T Q e_r + 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t)) + 2tr(-\tilde{W}^T Proj_m(\hat{W},\Phi(X(t_i))\check{e}_r^T P) - \Phi(X(t_i))\check{e}_r^T P) - 2tr(\Phi(X(t_i))\check{e}_r^T P\tilde{W}^T)$$

$$= -e_r^T Q e_r + 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t)) + 2tr((\hat{W}-W)^T Proj_m(\hat{W},\Phi(X(t_i))\check{e}_r^T P) - \Phi(X(t_i))\check{e}_r^T P) - 2tr(\Phi(X(t_i))\check{e}_r^T P\tilde{W}^T) \quad (18)$$

Note that

$$2tr((\hat{W}-W)^T Proj_m(\hat{W},\Phi(X(t_i))\check{e}_r^T P) - \Phi(X(t_i))\check{e}_r^T P) \leq 0$$

So then (18) becomes:

$$\dot{L} \leq -e_r^T Q e_r + 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t)) - 2tr(\Phi(X(t_i))\check{e}_r^T P\tilde{W}^T) \quad (19)$$

Since $2tr(\Phi(X(t_i))\check{e}_r^T P\tilde{W}^T) = 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i))$, $\check{\epsilon} = W^T(\Phi(X(t)) - \Phi(X(t_i)) + \epsilon$ and $\tilde{f}(X(t),\hat{X}(t)) = f(X(t)) - f(\hat{X}(t))$ then (19) becomes:

$$\dot{L} \leq -e_r^T Q e_r + 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P(W^T(\Phi(X(t)) - \Phi(X(t_i)) + \epsilon) - 2e_r^T PK_1\hat{e}_r + 2e_r^T P(f(X(t)) - f(\hat{X}(t))) - 2\check{e}_r^T P\tilde{W}^T\Phi(X(t_i))$$

$$= -e_r^T Q e_r + 2e_r^T P(W^T(\Phi(X(t)) - \Phi(X(t_i))) + 2e_r^T P\epsilon - 2e_r^T PK_1\hat{e}_r + 2e_r^T P(f(X(t)) - f(\hat{X}(t)))$$

Applying Norm properties:

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + 2\|P\|\|e_r\|\|W\|\|\Phi(X((t)) - \Phi(X(t_i))\| + 2\|P\|\|e_r\|\|\epsilon\| + 2\|PK_1\|\|e_r\|\|\hat{e}_r(t)\| + 2\|P\|\|e_r\|\|f(X(t)) - f(\hat{X}(t))\| \quad (20)$$

Now using $\|\Phi(X(t)) - \Phi(X(t_i))\| \leq L_\Phi\|X(t) - X(t_i)\| = L_\Phi\|e_{evt}\|$ and $\|f(X(t)) - f(\hat{X}(t))\| \leq L_f\|X(t) - X(t_i)\| = L_f\|e_{evt}\|$, $\|\epsilon\| \leq \epsilon^*$, $\|W\| \leq W^*$, $\|\hat{e}_r(t)\| \leq \eta$, (20) becomes:

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + 2\|P\|\|e_r\|W^* L_\Phi\|e_{evt}\| + 2\|P\|\|e_r\|\epsilon^* + 2\eta\|PK_1\|\|e_r\| + 2\|P\|\|e_r\|L_f\|e_{evt}\|$$

In this case, $e_{evt} = 0$ so then:

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + 2(\|PB\|\epsilon^* + \eta\|PK_1\|)\|e_r\| = -(\lambda_{min}(Q)\|e_r\|^2 - 2(\|PB\|\epsilon^* + \eta\|PK_1\|)\|e_r\|)$$

$$= -\left(\sqrt{\lambda_{min}(Q)}\|e_r\| - \frac{2(\|PB\|\epsilon^* + \eta\|PK_1\|}{\sqrt{\lambda_{min}(Q)}}\right)^2 + \frac{(2(\|PB\|\epsilon^* + \eta\|PK_1\|)^2}{\lambda_{min}(Q)}$$

Define $b_b = \frac{2(\|PB\|\epsilon^* + \eta\|PK_1\|)}{\sqrt{\lambda_{min}(Q)}}$ then

$$\dot{L} \leq -\big(\sqrt{\lambda_{min}(Q)}\|e_r\| - b_b\big)^2 + b_b^2$$

$$= -(1-\zeta)\big(\sqrt{\lambda_{min}(Q)}\|e_r\| - b_b\big)^2 - \zeta\big(\sqrt{\lambda_{min}(Q)}\|e_r\| - b_b\big)^2 + b_b^2$$

$$0 < \zeta < 1$$

If $(\sqrt{\lambda_{min}(Q)}\|e_r\| - b_b) \geq \sqrt{b_b^2/\zeta}$, then

$$\dot{L} \leq -(1-\zeta)\big(\sqrt{\lambda_{min}(Q)}\|e_r\| - b_b\big)^2 \quad (21)$$

and the form of upper bound in (21) can be used to conclude that the algorithm is UUB at the sampling instant.

Consider the second case, the inter-event time, where $e_{evt} \neq 0$ and $\dot{\tilde{W}} = 0$. Then if the same Lyapunov function is chosen, its time derivative can be expressed as:

$$\dot{L}(e_r,\tilde{W}) = 2e_r^T P\dot{e}_r + 2tr(\tilde{W}^T\gamma^{-1}\tilde{W}) \quad (22)$$

$$= 2e_r^T P(K_2 e_{evt} + K_2 e_r - K_1\hat{e}_r + \tilde{W}^T\Phi(X(t_i)) + \check{\epsilon} + \tilde{f}(X(t),\hat{X}(t))) + 0$$

$$= 2e_r^T PK_2 e_{evt} + 2e_r^T PK_2 e_r - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} + 2e_r^T P\tilde{f}(X(t),\hat{X}(t))$$

$$= e_r^T K_2^T P e_r + e_r^T PK_2 e_r + 2e_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} + 2e_r^T PK_2 e_{evt} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t))$$

$$= e_r^T(K_2^T P + PK_2)e_r + 2e_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P\check{\epsilon} + 2e_r^T PK_2 e_{evt} - 2e_r^T PK_1\hat{e}_r + 2e_r^T P\tilde{f}(X(t),\hat{X}(t)) \quad (23)$$

Now using $K_2^T P + PK_2 = -Q, \check{\epsilon} = W^T[\Phi(X(t)) - \Phi(X(t_i))] + \epsilon$ and $\tilde{f}(X(t),\hat{X}(t)) = f(X(t)) - f(\hat{X}(t))$ in (23):

$$\dot{L} = -e_r^T Q e_r + 2e_r^T PK_2 e_{evt} + 2e_r^T P\tilde{W}^T\Phi(X(t_i)) + 2e_r^T P(W^T[\Phi(X(t)) - \Phi(X(t_i))] + \epsilon) - 2e_r^T PK_1\hat{e}_r + 2e_r^T P(f(X(t)) - f(\hat{X}(t)))$$

$$= -e_r^T Q e_r + 2e_r^T P K_2 e_{evt} + 2e_r^T P \tilde{W}^T \Phi(X(t_i)) + 2e_r^T P W^T (\Phi(X(t)) - \Phi(X(t_i))) + 2e_r^T P \epsilon - 2e_r^T P K_1 \hat{e}_r + 2e_r^T P (f(X(t)) - f(\hat{X}(t))) \quad (24)$$

Now applying vector and induced matrix norm properties on (24):

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + 2\|PK_2\|\|e_r\|\|e_{evt}\| + 2\|P\|\|\tilde{W}\|\|\Phi(X(t_i))\|\|e_r\| + 2\|P\|\|e_r\|\|W\|\|\Phi(X(t)) - \Phi(X(t_i))\| + 2\|P\|\|e_r\|\|\epsilon\| + 2\|PK_1\|\|e_r\|\|\hat{e}_r(t)\| + 2\|P\|\|e_r\|\|f(X(t)) - f(\hat{X}(t))\| \quad (25)$$

Using several inequalities: $\|\Phi(X(t)) - \Phi(X(t_i))\| \leq L_\Phi\|X(t) - X(t_i)\| = L_\Phi\|e_{evt}\|$, $\|f(X(t)) - f(\hat{X}(t))\| \leq L_f\|X(t) - X(t_i)\| = L_f\|e_{evt}\|$, $\|\epsilon\| \leq \epsilon^*$, $\|W\| \leq W^*$, $\|\hat{e}_r(t)\| \leq \eta$ and $\|\Phi(X(t_i))\| \leq \Phi^*$, (25) becomes:

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + 2\|PK_2\|\|e_r\|\|e_{evt}\| + 2\|P\|\tilde{W}^*\Phi^*\|e_r\| + 2L_\Phi W^*\|P\|\|e_r\|\|e_{evt}\| + 2\epsilon^*\|P\|\|e_r\| + 2\eta\|PK_1\|\|e_r\| + 2L_f\|P\|\|e_r\|\|e_{evt}\|$$

Then using Young's inequality:

$$\dot{L} \leq -\lambda_{min}(Q)\|e_r\|^2 + \sigma\|PK_2\|^2\|e_r\|^2 + (1/\sigma)\|e_{evt}\|^2 + \sigma(W^*)^2\|e_r\|^2 + ((L_\Phi\|P\|)^2/\sigma)\|e_{evt}\|^2 + \sigma\|e_r\|^2 + \big((L_f\|P\|)^2/\sigma\big)\|e_{evt}\|^2 + 2\big(\|P\|(\tilde{W}^*\Phi^* + \epsilon^*) + \|PK_1\|\eta\big)\|e_r\|$$

$$\dot{L} \leq (-\lambda_{min}(Q) + \sigma\|PK_2\|^2 + \sigma(W^*)^2 + \sigma)\|e_r\|^2 + 2\|P\|\big(\tilde{W}^*\Phi^* + \epsilon^* + \|K_1\|\eta\big)\|e_r\| + ((1/\sigma) + (L_\Phi\|P\|)^2/\sigma + (L_f\|P\|)^2/\sigma)\|e_{evt}\|^2$$

$$= (-\lambda_{min}(Q) + \sigma\|PK_2\|^2 + \sigma(W^*)^2 + \sigma)\|e_r\|^2 + 2\|P\|\big(\tilde{W}^*\Phi^* + \epsilon^* + \|K_1\|\eta\big)\|e_r\| + ((1/ + (L_\Phi\|P\|)^2/ + (L_f\|P\|)^2)/\sigma)\|e_{evt}\|^2$$

If $\|e_{evt}\|^2 \leq \beta\|e_r\|^2$, then

$$= (-\lambda_{min}(Q) + \sigma\|PK_2\|^2 + \sigma(W^*)^2 + \sigma)\|e_r\|^2 + 2\|P\|\big(\tilde{W}^*\Phi^* + \epsilon^* + \|K_1\|\eta\big)\|e_r\| + ((1/ + (L_\Phi\|P\|)^2/ + (L_f\|P\|)^2)/\sigma)\beta\|e_r\|^2$$

Then, choosing β as $\beta = \sigma/(1 + (L_\Phi\|P\|)^2 + (L_f\|P\|)^2)$

$$\dot{L} \leq (-\lambda_{min}(Q) + \sigma\|PK_2\|^2 + \sigma(W^*)^2 + \sigma + 1)\|e_r\|^2 + 2\|P\|\big(\tilde{W}^*\Phi^* + \epsilon^* + \|K_1\|\eta\big)\|e_r\| \quad (26)$$

Define: $a_1 = \lambda_{min}(Q) - \sigma\|PK_2\|^2 - \sigma(W^*)^2 - \sigma - 1$, $b_1 = 2\|P\|\big(\tilde{W}^*\Phi^* + \epsilon^* + \|K_1\|\eta\big)$. b_1 is clearly positive and a_1 can be made positive for appropriate choices of K_2. Then, (26) becomes:

$$\dot{L} \leq -a_1\|e_r\|^2 + b_1\|e_r\| = -(a_1\|e_r\|^2 - b_1\|e_r\|) = -\left(\sqrt{a_1}\|e_r\| - \frac{b_1}{2\sqrt{a_1}}\right)^2 + \frac{b_1^2}{4a_1}$$

$$= -(1-\zeta)\left(\sqrt{a_1}\|e_r\| - \frac{b_1}{2\sqrt{a_1}}\right)^2 - \zeta\left(\sqrt{a_1}\|e_r\| - \frac{b_1}{2\sqrt{a_1}}\right)^2 + \frac{b_1^2}{4a_1}$$

If $\left(\sqrt{a_1}\|e_r\| - \frac{b_1}{2\sqrt{a_1}}\right) \geq \sqrt{\frac{b_1}{2\sqrt{a_1}}/\zeta}$, then

$$\dot{L} \leq -(1-\zeta)\left(\sqrt{a_1}\|e_r\| - \frac{b_1}{2\sqrt{a_1}}\right)^2 \quad (27)$$

And the form of (27) can now be used to conclude that the error is UUB at inter-event times which then proves that e_r is always UUB.

Furthermore, let e_a denote the approximation error: $e_a = X(t) - \hat{X}(t)$ then

$$e_r(t) = e_a(t) + \hat{e}_r(t) \quad (28)$$
$$e_a(t) = e_r(t) - \hat{e}_r(t)$$
$$\|e_a(t)\| = \|e_r(t)\| + \|\hat{e}_r(t)\|$$
$$\|e_a(t)\| = \|e_r(t)\| + \eta$$

so then the observer approximation error e_a is also UUB.

IV. Results and Discussion

A simulation of a 2-link planar robot manipulator was done to test the efficacy of the controller. The dynamics of the manipulator are given as:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau(t) \quad (29)$$

where $q \in \mathbb{R}^2$ are the joint angles, $M(q)$ is the inertia matrix, $V_m(q,\dot{q})$ is the Coriolis/centripetal matrix, $G(q)$ is the gravity vector, $F(\dot{q})$ is the friction matrix, τ_d is an unknown disturbance and τ_t is the control torque. Manipulating (29) into nonlinear state space form obtains:

$$\begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix} = \begin{bmatrix}\dot{q}\\ -M^{-1}(q)(V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}))\end{bmatrix} + \begin{bmatrix}0\\ -M^{-1}(q)\end{bmatrix}\tau_d + \begin{bmatrix}0\\ M^{-1}(q)\end{bmatrix}\tau_t \quad (30)$$

The simulation parameters used were: $m_1 = 1kg$, $m_2 = 2.3kg$, $l_1 = 1m$, $l_2 = 1m$, $g = 9.8$, $K_1 = [20I(2), 0(2); 0(2), 10I(2)$, $L_\Phi = 1$, $L_f = 1$, $\sigma = 1$, $\gamma = .4$, $Q = I$, $\alpha = 0.01$, $X(0) = \hat{X}(0)$ and the neural network used was a 1 layer random functional link (RFLNN) network with 10 hidden neurons. The hidden layer activation function was logistic sigmoid. The desired joint trajectories were $sin(0.5t)$ for joint 1 and $cos(0.5t)$ for joint 2 and the simulation time was 20 seconds over a 0.01 second time step. Unfortunately, due to space constraints, some of the results must be omitted, however a good comparison between a periodic controller and the event-triggered controller can be made by the results supplied.

The first three plots are the desired trajectories (Fig. (1)), trajectory error (Fig. (2)) and torque histories (control inputs) (Fig. (3)) from the standard periodic control. Measurements were taken and the control was updated at each time step

as is evident in Fig. (3). From the figures, it is clear that the controller performance is very good as convergence to near zero error is very fast and is consistent throughout the simulation. The control input is also very smooth and continuous and overall does not require a large amount of torque. However, what is absent from the plots is the communication requirements for this kind of controller. This will be discussed in more depth during the section covering the event triggered controller.

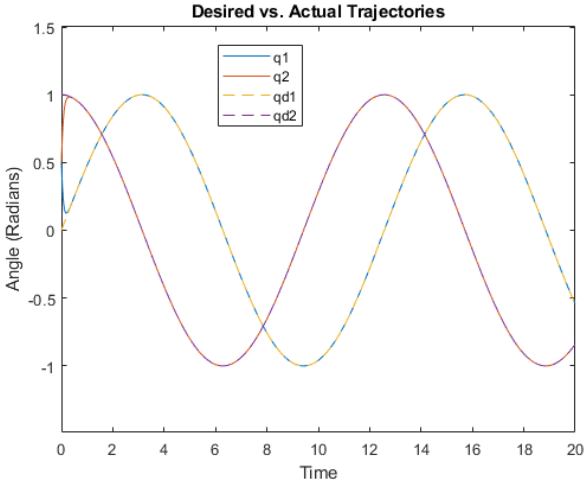

Fig. 1: Periodic Control Trajectories

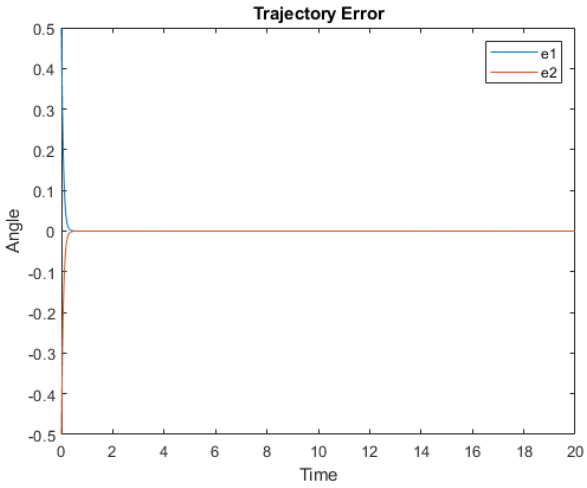

Fig. 2: Periodic Control Errors

Fig. (4)-(7) showcase the event triggered controller results, namely the desired vs. actual trajectories, the trajectory error, the torque histories and the event count. From Fig. (4) and (5), it is evident that the controller performs very well and even with the reduced amount of information the tracking performance on the desired trajectory is nearly perfect. Fig. (5) shows small spikes in error that is the result in the delay between updates, however these are nearly negligible and can be reduced by altering the event condition. The torque inputs in Fig. (6) are reminiscent of the periodic controller's torques with somewhat reduced smoothness and minutely increased

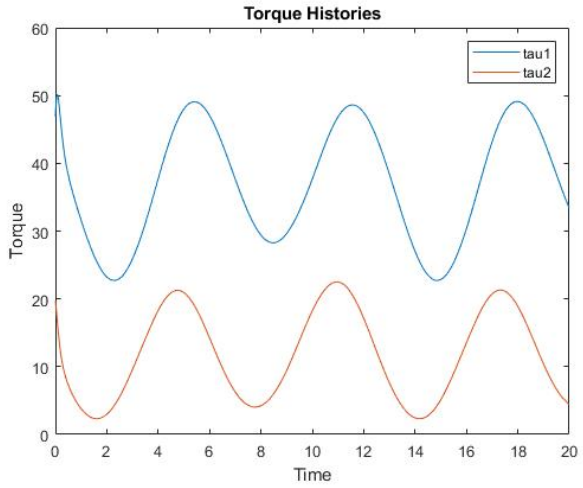

Fig. 3: Periodic Control Torques

magnitude in some locations. However, the control signal is reasonable and is absent of very large control inputs that would be characteristic of trying to correct an unstable system. Finally, Fig. (7) shows the cumulative number of updates throughout the simulation and illustrates a very strong benefit for the event-triggered controller. Given the time step of the simulation, there were 2000 update steps. However, from the figure, the ET controller updated less than 1000 times which means that the communication cost is less than half that of the periodic controller. Considering the almost negligible cost of performance, the appeal of this controller in communication-sensitive applications is self-evident.

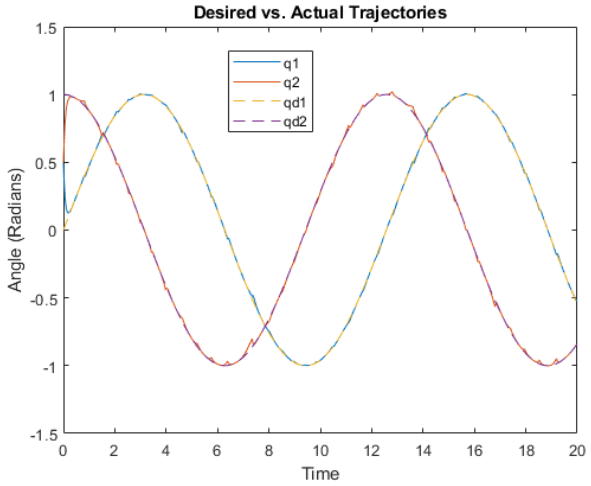

Fig. 4: ET Trajectories

V. Conclusion

This paper presents the highlights and main ideas of the the referenced paper [1]. An event triggered MSO based neuro-adaptive controller was presented and a 2-link planar robot manipulator simulation was conducted to showcase the performance when compared to a standard periodic update controller. The UUB stability of the controller is explored

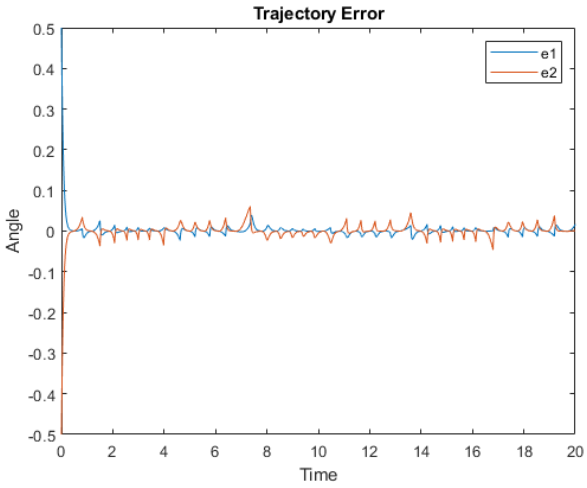

Fig. 5: ET Error

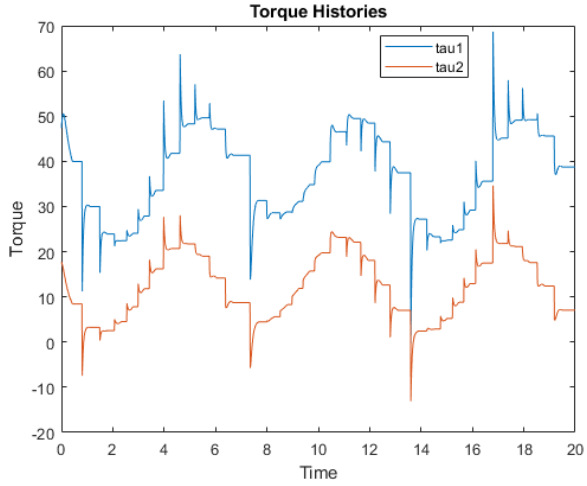

Fig. 6: ET Torque

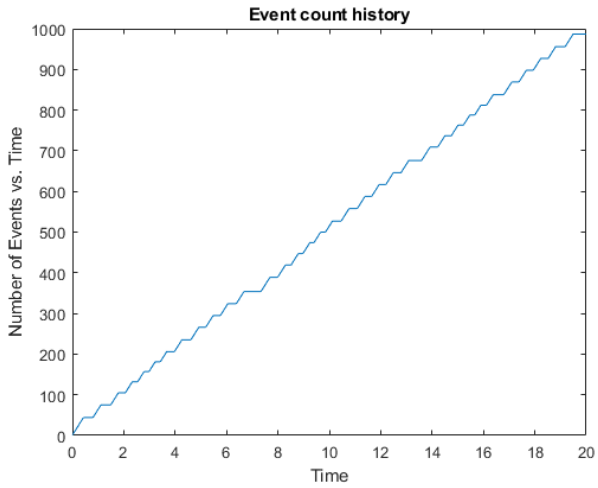

Fig. 7: ET Event Count

and the convergence is illustrated in the simulation results. Overall, the performance of the controller is very good, even with less than half of the state updates as compared to the periodic controller. Additional results are shown in [1] which were omitted for space constraints. This controller represents an important step in control theory and controller development and could be a powerful tool for industries that typically have high communication traffic problems.

This project allowed me to receive a more in-depth look into the domain of event triggered controllers and neural network adaptive systems. I was able to get a firmer grasp on the event triggered concept and where it is most applicable. Also, by studying the mathematical formulation of the controller, I received a much better understanding of how it was derived. Furthermore, by coding the simulation for this project, I was able to get first hand experience in creating, debugging and seeing the inner workings of an event triggered controller. The addition of the neural network component also strengthened my understanding of machine learning applications and uncertainty estimation in control. This experience allowed me to practice implementing what I have learned in a simulated real-world environment, allowing me to get a much better understanding and working knowledge as opposed to simply reading the literature. Finally, studying the stability proof was instrumental in helping me understand the principles behind why the controller works and also gives me more reinforcement in formal proof methodologies. By recreating the proof, I was able to see how certain ideas were implemented and how they lead to the desired conclusion. Overall, I think this project was very important in building my fundamental understanding in event triggered control and in neural network augmented control systems.

References

[1] *Event Triggered Neuroadaptive Controller (ETNAC) Design for Uncertain Affine Nonlinear Systems*. Vol. Volume 1: Advances in Control Design Methods; Advances in Nonlinear Control; Advances in Robotics; Assistive and Rehabilitation Robotics; Automotive Dynamics and Emerging Powertrain Technologies; Automotive Systems; Bio Engineering Applications; Bio-Mechatronics and Physical Human Robot Interaction; Biomedical and Neural Systems; Biomedical and Neural Systems Modeling, Diagnostics, and Healthcare. Dynamic Systems and Control Conference. V001T03A003. Sept. 2018. DOI: 10 . 1115 / DSCC2018 - 9103. eprint: https : / / asmedigitalcollection . asme . org / DSCC / proceedings - pdf / DSCC2018 / 51890 / V001T03A003 / 2376694 / v001t03a003-dscc2018-9103.pdf. URL: https://doi.org/10.1115/DSCC2018-9103.

Appendix: Matlab Code

```
%Nathan Lutes
%Project Simulation code
clear; clc; close all
global xti tau tauhist cnt evtcnt

%constants and initial conditions
states = 4; in = states; out = 2; L
= 10;
cnt = 0;
evtcnt = [];
tauhist = [];

V = 2*rand([L*in,1])-1;
W = 2*rand([L*out,1])-1;
x0 = [0.5; 0.5; 0; 0];
xhat0 = [0.5,0.5,0,0]';
xti = x0;

%simulation
dt = 0.01;
t0 = 0;
tf = 20;
t1 = t0:dt:tf;
x1 = [x0;xhat0;V;W];
%storage
x1hist =
zeros(length(x1),length(t0:dt:tf));
for i = 1:length(t1)
    x1hist(:,i) = x1;
    xdot = ETNACv2(t1(i),x1);
    x1 = x1 + dt*xdot;
end

%calculate desired trajectory
w = 0.5;
xd = [sin(w*t1); cos(w*t1)];

%plot desired and actual
trajectories
figure
plot(t1,x1hist(1,:))
hold on
plot(t1,x1hist(2,:))
plot(t1,xd(1,:),'--')
plot(t1,xd(2,:),'--')
hold off
legend('q1','q2','qd1','qd2')
xlabel('Time')
ylabel('Angle (Radians)')
title('Desired vs. Actual
Trajectories')

%plot error
figure
e = x1hist(1:2,:) - xd;
plot(t1,e(1,:));
hold on
plot(t1,e(2,:));
hold off
legend('e1','e2')
xlabel('Time')
ylabel('Angle')
title('Trajectory Error')

%plot tau history
figure
plot(t1,tauhist(1,:))
hold on
plot(t1,tauhist(2,:))
hold off
legend('tau1','tau2')
xlabel('Time')
ylabel('Torque')
title('Torque Histories')

%plot event count
figure
plot(t1,evtcnt)
xlabel('Time')
ylabel('Number of Events vs. Time')
title('Event count history')

function [qdot] = ETNACv2(t,q)
%This function implements an event
triggered neural adaptive
controller for
%use in NN project

%define event triggered state
global xti tau tauhist cnt evtcnt

%constants
states = 4; est = states;
in = states; L = 10; out = 2; Lphi
= 1; Lf = 1;
K2 =
(0.5)*[40*eye(2),0*eye(2);zeros(2),
20*eye(2)];
Vsize = L*in; Wsize = L*out; sigma
= 1; Q = eye(states);
P=0.0031*Q;
beta =
sigma/(1+(norm(P)*(norm(Lphi) +
norm(Lf)))^2);

% beta =
sigma/(1+(norm(P)*(norm(Lphi) +
norm(Lf)))^2);
```

```
a1 = 1; a2 = 1; m1 = 1; m2 = 2.3; g
= 9.8;
gam = .4; %gam = 30;
Fv=0.05*eye(2);
dist=0.05*[0.01;0.01];
alpha1 = .01; %case 1
alpha2 = 0.005; %case 2
alpha3 = 0.1; %case 3
alpha = alpha1;

%indices
indexInput = 1:states;
indexEst = indexInput(end) +
1:indexInput(end) + est;
indexV = indexEst(end) +
1:indexEst(end) + Vsize;
indexW = indexV(end) +
1:indexV(end) + Wsize;

%get input measurements
x = q(indexInput); xhat =
q(indexEst); V =
reshape(q(indexV),[in,L]);
W = reshape(q(indexW),[L,out]);

%calculate desired trajectory
w = 0.5;
xd = [sin(w*t); cos(w*t)];
xddot = [w*cos(w*t); -w*sin(w*t)];
F = [0 0 1 0; 0 0 0 1; -w^2 0 0 0;
0 -w^2 0 0]*[xd; xddot];

%Robot system
M = [(m1+m2)*a1^2 + m2*a2^2 +
2*m2*a1*a2*cos(x(2))...
    m2*a2^2 +
m2*a1*a2*cos(x(2));...
    m2*a2^2 + m2*a1*a2*cos(x(2))
m2*a2^2];
N = [-m2*a1*a2*(2*x(3)*x(4) +
x(4)^2)*sin(x(2)) +
(m1+m2)*g*a1*cos(x(1)) +
m2*g*a2*cos(x(1) + x(2));...
    m2*a1*a2*(x(3)^2)*sin(x(2)) +
m2*g*a2*cos(x(1) + x(2))];
Mxti = [(m1+m2)*a1^2 + m2*a2^2 +
2*m2*a1*a2*cos(xti(2))...
    m2*a2^2 +
m2*a1*a2*cos(xti(2));...
    m2*a2^2 + m2*a1*a2*cos(xti(2))
m2*a2^2];
Nxti = [-m2*a1*a2*(2*xti(3)*xti(4)
+ xti(4)^2)*sin(xti(2)) +
(m1+m2)*g*a1*cos(xti(1)) +
m2*g*a2*cos(xti(1) + xti(2));...
    m2*a1*a2*(xti(3)^2)*sin(xti(2))
+ m2*g*a2*cos(xti(1) + xti(2))];
fxti = [xti(3:4);-Mxti\Nxti];
B = [zeros(2);inv(M)];
Bs = [-inv(M);inv(M)];
Us = [1;1];
Baug = [B,Bs];

%calculate neural network
phi = 1./(1 + exp(-(V'*x)));
dhat = W'*phi;
d = Fv*xd+dist;

%event triggered
%decide if event is triggered and
xti should be updated
beta=3000/(1+(2*norm(P))^2);
er = x - [xd; xddot];
eevt = x - xti;
if norm(eevt(1:2))^2 <=
alpha*beta*norm(er(1:2))^2
    xti = x;
    %calculate control input
    K = 0.5*[30*eye(2),
0*eye(2);0*eye(2),15*eye(2)];
    tau = Baug\(F- K*(xhat-
[xd;xddot]) - K2*(xti-xhat) -
B*dhat - [x(3); x(4); -M\N] +
Bs*Us);
    cnt = cnt + 1;
    %update weights
    eDC = xti - [xd;xddot];
    Wdot = gam*(phi*eDC'*P)*B;
else
    Wdot = zeros(size(W));
end

%periodic
% xti = x;
%calculate control input
% K = 0.5*[30*eye(2),
0*eye(2);0*eye(2),15*eye(2)];
% tau = Baug\(F- K*(xhat-
[xd;xddot]) - K2*(xti-xhat) -
B*dhat - [x(3); x(4); -M\N] +
Bs*Us);
% %update weights
% eDC = xti - [xd;xddot];
% Wdot = gam*(phi*eDC'*P)*B;

%record control input
tauhist = [tauhist tau];
%record event history
evtcnt = [evtcnt cnt];
```

```
%calculate state and weight update
laws
xdot = [x(3); x(4); -M\N] +
Baug*tau - Bs*Us + B*d;
xhatdot = fxti + Baug*tau - Bs*Us
+B*dhat + K2*(xti-xhat);
Vdot = zeros(size(V));

qdot = [xdot; xhatdot;
reshape(Vdot,[L*in,1]);
reshape(Wdot,[L*out,1])];

end
```