

Dynamic ETM Overview



Dynamic_Triggering_Mechanisms_for_Event-Triggered_Control.pdf
PDF

Explain to me in full detail this paper



The paper "**Dynamic Triggering Mechanisms for Event-Triggered Control**" by Antoine Girard introduces a new class of event-triggering mechanisms (ETM) for event-triggered control systems, characterized by the inclusion of an internal dynamic variable. This novel approach is termed the **dynamic event-triggering mechanism (dynamic ETM)**.

Key Concepts:

1. Event-Triggered Control (ETC):

- In traditional control systems, controllers execute periodically, which may lead to unnecessary computational and communication resource usage. In contrast, ETC systems only update inputs when specific events occur.
- The goal of event-triggered mechanisms (ETMs) is to determine when to trigger these updates. Static ETMs depend only on the system's current state, while dynamic ETMs include internal dynamic variables.

2. Dynamic Event-Triggered Mechanism (ETM):

- The dynamic ETM introduced in this paper incorporates an internal variable that changes over time and is non-monotonic.
- This approach relaxes the requirement that the system's state always meets a strict trigger condition, instead ensuring that the condition holds on average over time, leading to fewer control updates.

3. Stability and Performance:

- The paper demonstrates the stability of the closed-loop system using a dynamic ETM and shows that it guarantees a lower bound on the inter-execution time, which is at least as long as the inter-execution time with static ETMs.
- The decay rate of the Lyapunov function (which indicates system stability) can be tuned by adjusting design parameters. For linear systems, the paper provides formulas for the minimum inter-execution time and analyzes how parameters like the internal dynamic variable and filter coefficients affect system performance.

4. Simulation Results:

- Simulations illustrate that using a dynamic ETM increases the average time between control updates compared to a static ETM. This reduces the system's workload while maintaining stability and performance.

Structure:

- **Section I (Introduction):** Overview of ETC and motivation for introducing dynamic ETMs.

- **Section II (Nonlinear Systems):** Presents the framework for dynamic ETMs in nonlinear systems and discusses the stability of the closed-loop system.
- **Section III (Linear Systems):** Applies the results to linear systems, providing a detailed analysis of how dynamic ETMs influence the system's execution time and performance index.
- **Section IV (Simulation Results):** Illustrates the practical benefits of dynamic ETMs through simulations.

Overall, the paper provides a thorough analysis of dynamic ETMs, showing their potential to reduce control effort in ETC systems while maintaining stability and performance.



Data-driven_event-triggering.pdf
PDF

Explain to me in full detail this paper



The paper titled "**Data-Driven Event-Triggering Mechanism for Linear Systems Subject to Input Saturation**" focuses on the development of an event-triggering mechanism aimed at reducing the number of control updates while maintaining the stability of linear systems that experience input saturation. Below is a detailed breakdown of the paper:

1. Introduction:

- **Objective:** The paper addresses the challenge of controlling linear systems with saturated inputs using a data-driven approach. Traditional methods depend on precise system models, which can be difficult to obtain. This work aims to convert model-based control designs into data-driven solutions using Linear Matrix Inequality (LMI) conditions.
- **Problem:** The paper tackles the issue of regional stabilization for linear systems subject to input saturation and aims to design both the event-triggering mechanism and the control law in a unified way.

2. Key Concepts:

- **Event-Triggered Control (ETC):** Unlike traditional control systems, which update periodically, ETC updates the control input only when a specific event occurs, potentially reducing computational and communication costs.
- **Saturated Inputs:** In many real-world systems, the control input cannot exceed certain limits, leading to input saturation. This paper addresses the challenge of ensuring system stability under these conditions.
- **Data-Driven Control:** Instead of relying on a precise model of the system, this paper proposes using experimental data to design the controller and event-triggering rule.

3. Main Contributions:

- **Co-Design of Event-Triggering Mechanism and Controller:** The paper proposes a method to co-design the event-triggering mechanism and the state feedback gain in the presence of input saturation.

- **LMI Formulation:** The authors derive LMI conditions that can be solved to design the control law and event-triggering rule. These conditions ensure the stability of the system while reducing the frequency of control updates.
- **Data-Driven Approach:** The model-based conditions are transformed into a data-driven context, allowing the design of controllers even when the system model is not fully known.

4. Methodology:

- **Dynamic Event-Triggering Rule:** The triggering mechanism is dynamic, meaning it depends on both the system's state and an internal dynamic variable, which evolves over time. This is inspired by earlier work on dynamic event-triggering mechanisms.
- **LMI-Based Design:** The paper formulates the event-triggering mechanism design as an LMI problem. This allows the co-design of the event-triggering rule and the feedback controller to ensure regional stability of the closed-loop system.
- **Spacecraft Rendezvous Example:** The methodology is applied to a spacecraft rendezvous problem, which involves controlling a chaser spacecraft as it approaches a target spacecraft in orbit. The spacecraft model includes input saturation, making it a relevant case for the proposed method.

5. Results:

- **Simulations:** The proposed method is evaluated through simulations of a spacecraft rendezvous problem. The results show that the dynamic event-triggering mechanism reduces the number of control updates compared to static approaches while maintaining system stability despite input saturation.
- **Stability and Performance:** The co-designed control law and event-triggering mechanism ensure that the system remains stable, and the number of control updates is minimized. The method guarantees a lower bound on the inter-execution time of control actions.

6. Conclusion:

- **Future Directions:** The paper concludes with suggestions for future work, including exploring decentralized event-triggering mechanisms and improving the scalability of the proposed method.

In summary, this paper introduces a data-driven approach to event-triggered control for linear systems with input saturation, providing a framework for co-designing control laws and event-triggering rules that ensure stability while reducing control updates.



In paper "Dynamic Triggering Mechanisms for Event-Triggered Control" how is equation 17 derived?



Equation (17) in the paper "Dynamic Triggering Mechanisms for Event-Triggered Control" is typically derived from the **Lyapunov stability analysis** combined with conditions ensuring the event-triggering mechanism avoids **Zeno behavior** (infinitely frequent triggering events) and guarantees stability for the system.

General Form of Equation (17):

$$\Psi_1 + \Psi_2 + \Psi_2^\top + \Psi_3 \preceq 0$$

This type of inequality typically emerges from the **stability conditions** imposed on the system through a **Lyapunov function** and the associated **event-triggering condition**.

Steps Leading to the Derivation:

1. **Lyapunov Function:** The control design often starts with a **Lyapunov function** $V(x)$ for the system's state x , ensuring that $V(x)$ decreases over time, which implies system stability. A typical quadratic Lyapunov function is used:

$$V(x) = x^\top P x$$

where P is a positive definite matrix.

2. **Lyapunov Decrease Condition:** To ensure stability, the derivative of the Lyapunov function $\dot{V}(x)$ should be negative. For the sampled-data system (like in event-triggered control), the decrease condition might look like:

$$V(x_{k+1}) - V(x_k) \leq -\alpha \|x_k\|^2$$

where $\alpha > 0$ is a design parameter.

3. **Event-Triggering Condition:** The event-triggering rule is designed to trigger an event (control update) when a certain condition is met. This condition often takes the form of a quadratic function:

$$\psi(x, \hat{x}) = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}^\top \begin{bmatrix} \Psi_1 & \Psi_2 \\ \Psi_2^\top & \Psi_3 \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

Here, \hat{x} is the state at the last triggering time, and Ψ_1, Ψ_2, Ψ_3 are matrices defining the event-triggering condition. This quadratic form compares the current state x with the previously triggered state \hat{x} , and when the condition exceeds a certain threshold, an event is triggered.

4. **Ensuring Non-Zeno Behavior:** To prevent Zeno behavior (infinitely fast triggering), a condition must be imposed on the triggering function $\psi(x, \hat{x})$ to ensure it does not trigger too frequently. This is achieved by requiring the matrix inequality:

$$\Psi_1 + \Psi_2 + \Psi_2^\top + \Psi_3 \preceq 0$$

This condition ensures that the quadratic triggering function does not grow too quickly and that events are not triggered unnecessarily.

Conclusion:

Equation (17) is derived by ensuring that the event-triggering condition is compatible with both the Lyapunov-based stability requirement and the need to avoid Zeno behavior. The inequality ensures that the event-triggering mechanism is well-posed, meaning it guarantees both stability and a lower bound on the inter-event times. The matrices Ψ_1, Ψ_2 , and Ψ_3 are chosen to satisfy these conditions based on the system's dynamics and the desired stability properties.



What about eq 17 of the other paper?



In the "**Data-Driven Event-Triggering Mechanism for Linear Systems Subject to Input Saturation**" paper, **Equation (17)** refers to a condition for the **quadratic triggering function** $\psi(x, y)$, which is part of the event-triggering mechanism. It ensures that the event-triggering rule avoids unnecessary triggering immediately after an update and guarantees stability under input saturation.

The equation is:

$$\begin{bmatrix} I & I \end{bmatrix}$$



To sum up explain all the advantages of a dynamic ETM with respect to a static one as explained in the paper "Dynamic Triggering Mechanisms for Event-Triggered Control"



In the paper "**Dynamic Triggering Mechanisms for Event-Triggered Control**", the author introduces the concept of a **dynamic event-triggering mechanism (dynamic ETM)** as an improvement over the traditional **static event-triggering mechanism (static ETM)**. Below are the key advantages of using a dynamic ETM compared to a static ETM:

1. Fewer Control Updates (Increased Average Inter-Event Time):

- **Dynamic ETM:** Incorporates an internal dynamic variable that accumulates over time, allowing the system to tolerate small deviations from the event-triggering condition. This results in fewer control updates, leading to longer average inter-event times.
- **Static ETM:** Triggers an update whenever the error condition is violated, leading to more frequent control updates.

2. Improved Resource Efficiency:

- **Dynamic ETM:** Reduces communication, computation, and actuation load by minimizing the number of control updates. This is especially advantageous in networked control systems where communication resources or power are limited.
- **Static ETM:** Updates more frequently, potentially consuming more communication bandwidth and computational resources.

3. Relaxed Triggering Conditions:

- **Dynamic ETM:** Uses an internal dynamic variable that evolves over time and can temporarily "delay" the triggering condition from being satisfied, allowing the system more flexibility in deciding when to trigger updates.
- **Static ETM:** Has stricter update conditions since the triggering rule only depends on the current system state. This can lead to unnecessary updates when small fluctuations trigger events prematurely.

4. Tunable Performance via Design Parameters:

- **Dynamic ETM:** Allows for tuning of the triggering rule by adjusting parameters related to the internal dynamic variable (such as its rate of change or decay). This gives system designers control over the trade-off between stability and communication efficiency.
- **Static ETM:** Does not offer the same flexibility in tuning, as it is based solely on the instantaneous state error without any internal dynamics.

5. Improved Stability with Fewer Updates:

- **Dynamic ETM:** Despite reducing the number of control updates, it ensures system stability by carefully selecting the parameters governing the internal dynamic variable. The decay rate of the Lyapunov function can be maintained over time, ensuring stability.
- **Static ETM:** Stability is ensured but often at the cost of more frequent updates, which may not be necessary for the system's long-term behavior.

6. Robustness Against Small Disturbances:

- **Dynamic ETM:** By integrating a dynamic variable, the system is more robust to small disturbances or noise in the state. The system can handle these small deviations without triggering an event immediately.
- **Static ETM:** More sensitive to small state variations, which can lead to unnecessary updates.

7. Applicable to Broader System Types:

- **Dynamic ETM:** The dynamic nature makes it applicable to a broader range of systems, particularly those where inputs and outputs are subject to varying conditions over time.
- **Static ETM:** Limited in its ability to adapt to time-varying dynamics or systems with more complex operational conditions.

Summary of Advantages:

- **Reduced number of updates**, increasing efficiency.
- **Resource savings** in communication, computation, and actuation.
- **Greater flexibility** through tunable parameters.
- **Improved stability and robustness** against small disturbances.
- **Wider applicability** to systems with varying dynamics.

The dynamic ETM offers a more efficient and flexible solution for event-triggered control, particularly in resource-constrained environments like wireless networks or systems with computational limitations.



How can you adapt a dynamic ETM in the triggering of a RNN controller?



Adapting a **dynamic Event-Triggered Mechanism (ETM)** to trigger a **Recurrent Neural Network (RNN) controller** requires considering both the characteristics of the RNN and the advantages provided by the dynamic ETM. Here's how you can approach this adaptation:

1. Understand the Role of the RNN Controller:

- **RNNs** are often used for control tasks involving time-series data or systems with memory, due to their ability to capture temporal dependencies. In a control system, the RNN can process the current and past states of the system to generate the control action.
- The RNN is typically updated at regular intervals (periodic updates), but event-triggered control seeks to reduce unnecessary updates by triggering updates only when needed for stability and performance.

2. Incorporate a Dynamic Triggering Condition:

- In a dynamic ETM, a **dynamic variable** evolves over time, influencing when a control update should occur. This dynamic variable typically accumulates the error between the current system state and the previously triggered state.

For an RNN controller, you can define the **triggering function** $\psi(x, \hat{x})$ to depend on:

- The current system state x ,
- The last control input or state \hat{x} when the RNN was last updated,
- A dynamic variable η that evolves over time.

The triggering condition could take the form:

$$\psi(x, \hat{x}) = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}^\top \begin{bmatrix} \Psi_1 & \Psi_2 \\ \Psi_2^\top & \Psi_3 \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

and you would trigger an update if:

$$\psi(x, \hat{x}) \geq \rho\eta$$

where η is the dynamic variable that evolves as:

$$\eta_{k+1} = (\lambda + \rho)\eta_k - \psi(x_k, \hat{x}_k)$$

with $\lambda \in [0, 1)$, and $\rho \geq 0$, ensuring that updates happen less frequently over time unless necessary for system stability.

3. Adaptation to RNNs: Dynamic State Memory:

- **RNNs** process information sequentially, and their internal state (hidden layer) reflects the temporal dependencies of the system. The event-triggering condition could be based on the **difference between the current system state and the hidden state** of the RNN.
- Let h_k be the hidden state of the RNN at time step k , and u_k be the control input generated by the RNN. You can define the dynamic ETM as:

$$\psi(x_k, h_k) = \|x_k - h_k\|^2$$

This measures the deviation between the current system state x_k and the RNN's internal representation h_k . An update to the RNN (i.e., feeding new data into the RNN) occurs when this deviation exceeds the threshold governed by the dynamic variable η .

4. Tuning the Dynamic Variable:

- The dynamic variable η evolves according to the system's performance and stability needs. By tuning λ and ρ , you can control how aggressively the system reacts to deviations in the state.
- For example, setting a low λ results in more conservative updates (i.e., the system waits longer to update), while a higher λ results in quicker updates.

5. Ensuring Stability in the RNN-Based System:

- To ensure the stability of the system under this event-triggering mechanism, you would need to analyze the **Lyapunov stability** of the closed-loop system with the RNN controller. This involves ensuring that the triggering condition prevents the system from deviating too far from its desired trajectory.
- The dynamic ETM can be designed to ensure that the **Lyapunov function** $V(x)$ of the system decreases over time, even with fewer control updates from the RNN.

6. Advantages of Dynamic ETM in RNN Controllers:

- **Reduced Computational Load:** By incorporating a dynamic ETM, the RNN controller will only be triggered when necessary, reducing computational demands compared to periodic updates.
- **Improved Efficiency:** This approach leads to fewer unnecessary RNN activations, optimizing resource usage, especially in resource-constrained systems like embedded controllers or real-time systems.
- **Flexible Performance Tuning:** Dynamic ETMs provide flexibility in adjusting the trade-off between control performance and update frequency by tuning parameters like λ and ρ .
- **Handling Complex Temporal Dependencies:** Since RNNs inherently capture temporal dynamics, using a dynamic ETM helps balance the need for real-time responsiveness with longer-term stability goals.

7. Implementation Steps:

1. **Design the RNN:** Implement an RNN-based controller that takes system states as input and generates control actions.
2. **Define the Triggering Function:** Create a triggering function $\psi(x, h)$ based on the difference between the system state and the RNN's hidden state.
3. **Incorporate the Dynamic Variable:** Add a dynamic variable η that evolves over time and influences when the RNN should update its control input.
4. **Ensure Stability:** Use a Lyapunov-based stability analysis to ensure that the dynamic ETM maintains system stability despite reduced control updates.
5. **Tune the Parameters:** Experiment with λ and ρ to find the optimal balance between control accuracy and efficiency.

In summary, adapting a **dynamic ETM** to trigger an **RNN controller** allows for efficient control updates, leveraging the dynamic evolution of the event-triggering rule to ensure that the RNN processes inputs and produces control actions only when necessary, thus optimizing performance and reducing resource usage.



In this answer you linked the update event to a distance in norm between the hidden RNN state and the system state. What could be other criteria that may be treated as events?



In addition to using the norm-based distance between the hidden state of the **Recurrent Neural Network (RNN)** and the system state as a criterion for triggering updates in a **dynamic Event-Triggered Mechanism (ETM)**, several other criteria can be used to define events that trigger an update in the control input. These criteria could be based on different aspects of system dynamics, control performance, and learning objectives. Here are some alternative criteria for defining events in the context of RNN-based controllers:

1. Error in Control Performance (Tracking Error)

- **Criterion:** The triggering event could be based on the **control error**—the difference between the desired system state (reference trajectory) and the actual system state.
- **Event-Triggering Condition:**

$$\psi(x, x_{\text{ref}}) = \|x_k - x_{\text{ref},k}\|^2$$

where x_k is the current state of the system and $x_{\text{ref},k}$ is the desired reference state. An event is triggered when the tracking error exceeds a certain threshold, indicating the system has deviated too far from its desired state.

- **Advantages:** This method ensures that updates occur only when the system's performance starts to degrade, making it particularly useful for systems that need to follow a reference trajectory or desired behavior.

2. Control Effort or Energy Consumption

- **Criterion:** Events can be triggered based on the magnitude of the **control input** itself, ensuring that updates are only made when the control input is changing significantly.
- **Event-Triggering Condition:**

$$\psi(u_k, u_{k-1}) = \|u_k - u_{k-1}\|^2$$

where u_k is the control input at time k , and u_{k-1} is the previous control input. The update is triggered if the change in control input exceeds a threshold, which can minimize control effort or energy consumption.

- **Advantages:** Reduces unnecessary updates when the control action does not change significantly, leading to energy savings in actuators and reduced control effort.

3. Prediction Error of the RNN (Model Error)

- **Criterion:** In model-based control scenarios, the RNN might be used to predict the system's next state. The event can be triggered when the **prediction error**—the difference between the predicted state and the actual state—exceeds a threshold.
- **Event-Triggering Condition:**

$$\psi(x_k, \hat{x}_{k+1}) = \|x_{k+1} - \hat{x}_{k+1}\|^2$$

where \hat{x}_{k+1} is the state predicted by the RNN, and x_{k+1} is the actual system state at time $k + 1$. This triggers an update when the RNN's predictions diverge from the true system dynamics.

- **Advantages:** This approach focuses on ensuring the accuracy of the RNN model, only triggering an update when the model becomes inaccurate. It's useful for learning-based control strategies where the RNN continuously learns and adapts to the system dynamics.

4. Lyapunov Function-Based Triggering

- **Criterion:** A **Lyapunov function** $V(x)$ is often used to guarantee the stability of a system. An event can be triggered when the value of the Lyapunov function exceeds a certain threshold, indicating the system is moving away from stability.
- **Event-Triggering Condition:**

$$\psi(x) = V(x_k) - V(x_{k-1})$$

where $V(x)$ is a Lyapunov function (e.g., $V(x) = x^T P x$ for some positive definite matrix P). If the Lyapunov function increases significantly, indicating potential instability, a control update is triggered.

- **Advantages:** This method ensures that the system remains stable by triggering updates based on stability conditions. It's particularly useful for nonlinear systems where stability is a concern.

5. Rate of Change of the System State

- **Criterion:** Trigger events based on the **rate of change** of the system state. If the system's state changes rapidly, indicating a potential deviation or disturbance, an update is triggered.
- **Event-Triggering Condition:**

$$\psi(x_k, x_{k-1}) = \|x_k - x_{k-1}\|^2$$

This criterion measures how quickly the system state is changing. An event is triggered if the state change between consecutive time steps exceeds a threshold, indicating that the controller might need to react to rapid variations in the state.

- **Advantages:** Suitable for dynamic systems where sudden changes in state indicate the need for control action, such as when external disturbances or nonlinearities affect the system.

6. State Uncertainty or Confidence in Predictions

- **Criterion:** If the RNN is integrated with a probabilistic model (e.g., a Bayesian RNN), you can trigger updates based on the **uncertainty** in the state estimate or prediction. This is useful in scenarios where the system operates under noisy or uncertain conditions.
- **Event-Triggering Condition:**

$$\psi(\Sigma_k) = \text{trace}(\Sigma_k)$$

where Σ_k represents the uncertainty covariance of the state or the prediction from the RNN. When the uncertainty grows beyond a certain threshold, an event is triggered to update the controller.

- **Advantages:** This criterion is useful in **uncertain environments** or noisy systems where the accuracy of the state estimation or prediction plays a crucial role in control performance.

7. Accumulation of Errors over Time (Integral of Error)

- **Criterion:** Instead of using instantaneous errors, the event-triggering mechanism could rely on the **integral of the error** over time. This allows for more flexibility by only triggering updates when errors accumulate to a significant level.
- **Event-Triggering Condition:**

$$\psi = \int_0^t \|x(\tau) - \hat{x}(\tau)\| d\tau$$

This approach measures the accumulated difference between the current state and a reference or predicted state over time, and triggers an update when the integral exceeds a threshold.

- **Advantages:** This prevents excessive triggering due to short-term fluctuations, focusing on long-term deviations that require corrective action.

8. Control Saturation Detection

- **Criterion:** Events can be triggered when the control input reaches a **saturation limit**, indicating that the controller may no longer be able to affect the system adequately.
- **Event-Triggering Condition:**

$$\psi(u_k) = \max(\|u_k\|) - u_{\max}$$

where u_{\max} is the maximum allowable control input. When the control input reaches or exceeds this saturation level, an update is triggered to re-evaluate the control action.

- **Advantages:** This is particularly useful for systems with **input constraints**, ensuring that updates happen when the control input is no longer able to sufficiently influence the system.

9. Time-Based Trigger with Adaptive Thresholds

- **Criterion:** Use a **time-dependent threshold** that changes dynamically with the system state. While the system does not have to update periodically, the threshold for triggering an event can evolve based on system dynamics.
- **Event-Triggering Condition:**

$$\psi(x, t) = \|x_k - \hat{x}_k\|^2 \quad \text{with} \quad \rho(t) = \rho_0 e^{-\alpha t}$$

where $\rho(t)$ is a time-varying threshold that decreases over time. An update occurs when the state deviation exceeds this evolving threshold.

- **Advantages:** This method combines time-varying dynamics with state-based events, providing flexibility in how updates are triggered over time.

Conclusion:

Each of these alternative event-triggering criteria can be integrated with a **dynamic ETM** in an RNN-based control system, depending on the specific objectives and system requirements. Whether based on error, control effort, stability, uncertainty, or other factors, the goal remains the same: to reduce unnecessary control updates while ensuring that the system operates efficiently and remains stable.

ChatGPT can make mistakes. Check important info.