Implementation of an Event-triggered Controller in a Helicopter Model

Jorge Viramontes Perez and Michael D. Lemmon

***Abstract*—The use of event-triggered controllers in real-time systems has been proposed as a solution to increase the number of software tasks (unrelated to the control task) that can be scheduled by the system without compromising the stability of the plant. This paper investigates the design and real-time implementation of an event-driven controller for a nonlinear mechanical plant. The plant is a 3 degree-of-freedom (DOF) helicopter system. The use of this non-linear mechanical plant contrasts with the linear (mainly non-mechanical) implementations of previous works. The controller is implemented on a Pentium III PC using the real-time S.Ha.R.K. kernel. A detailed analysis of the performance as well as the improvements achieved by the event-triggered controller are presented. These results suggest that event-driven control of such mechanical systems exhibit performance levels comparable to traditional periodic real-time controllers while greatly reducing overall task set utilization.**

***Index Terms*—Event-triggered control systems, Real-time systems.**

I. Introduction

In the design of embedded feedback control systems, control tasks are traditionally implemented in a periodic fashion. Using this approach, the desired control performance can be obtained using a sufficiently fast sampling period. Periodicity, however, may lead to a significant usage of processing resources [1]. Applications in networked control may have tight constraints on both computer resource utilization and overall control system performance. These applications have therefore motivated the study of aperiodic real-time control systems in hopes of better managing the tradeoff between computer utilization and application performance. Two recent paradigms for aperiodic real-time control are *event-triggered control* [2] and *self-triggered control* [3]. In both cases control tasks are invoked when some internal error or gap signal exceeds a specified threshold. This leads to the sporadic invocation of control tasks, thereby leaving more time available for other non-control related tasks to be invoked. This paper examines the use of event-triggered controllers This work focuses on the use of event-driven controllers that appear to reduce computer utilization by executing the control task only when the information in the control loop shows that execution is required.

The idea of an event-driven control system implementation has been explored by many researchers. In [2] an Input-to-State stability approach for an event-triggered controller implementation is presented, the strategy is illustrated merely with the use of a simulated linear plant. In [4], a self-triggered controller is implemented in a linear analog plant using a real-time kernel. This paper is focused entirely on implementation strategies. The performance of the system with respect to the utilization resources and the stability of the plant is not investigated. In [5], the performance of different scheduling protocols for event-triggered controllers on a shared network is investigated, results are exclusively obtained from numerical examples with first-order systems. These previous works have dealt mainly with simulations of the event-triggered controllers and, only in some cases, simple implementations (mainly linear and non-mechanic) have been investigated.

In this paper, a feasible strategy for the design and implementation of an event-driven control for a non-linear mechanical plant is introduced. Special emphasis has been placed on the use of the S.H.a.R.K kernel [8] as the real-time platform for the system's implementation. As a second contribution of this work, a detailed analysis of the performance achieved by the event-triggered control is presented.

The remainder of this work is outlined as follows; Section II provides a general description of the experimental setup and the mathematical modeling of the plant; Section III illustrates the design of both the periodic and event-driven controllers; Section IV follows the experimental implementation with the use of a real-time kernel; Section V provides the experimental results and performance analysis of the system and finally Section VI concludes the paper.

II. Experimental Setup

The experimental system consists of a Quanser© 3DOF helicopter model as the main plant, a MultiQ-3 board for data acquisition, and a pentium III PC running the S.H.a.R.K. real-time kernel. Matlab© and Simulink© are used to simulate the system to verify plant models and controller designs.

The 3DOF helicopter model (shown in Figure 1) has three main components mounted on a table top; a main beam, a twin rotor assembly and a counterweight. The system is actuated by two rotors driven each by an electric DC motor. Encoders for position measurements are mounted in each of the three axis of the system: elevation ($\epsilon(t)$), pitch ($\rho(t)$), and travel ($\tau(t)$). The objective of this experiment is to set the helicopter body to a desired elevation ($\epsilon_c(t)$) and a desired travel rate ($\dot{\tau_c}(t)$).

The system response is described by the following equations [6]:

$$\begin{aligned} J_\epsilon \ddot{\epsilon}(t) &= l_a \cos(\rho(t)) T_{col}(t) - Mgl_\theta \sin(\epsilon(t)) \\ J_\rho \ddot{\rho}(t) &= l_h T_{cyc}(t) - mgl_\phi \sin(\rho(t)) \\ J_\tau \ddot{\tau}(t) &= l_a T_{col}(t) \cos(\epsilon(t)) \sin(\rho(t)) - Drag \end{aligned} \tag{1}$$

J. Viramontes Perez is a graduate student with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 46556 USA e-mail: jviramo1@nd.edu

Manuscript received September 15, 2009.

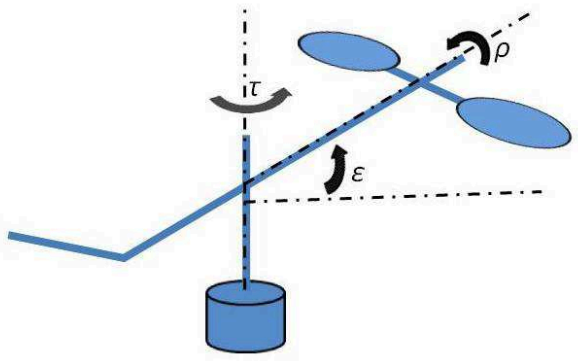

Fig. 1. Schematic of the 3DOF helicopter.

TABLE I
3DOF HELICOPTER PARAMETER VALUES

Parameter	Value	Units
m	1.25	kg
M	2.5	kg
l_a	0.66	m
l_h	0.177	m
J_ϵ	0.9	kgm^2
J_ρ	0.024	kgm^2
J_τ	1.094	kgm^2
g	9.81	m/s^2
l_θ	0.014	m
l_ϕ	0.004	m

Where $J_\epsilon, J_\rho, J_\tau$ denote the moments of inertia, *M* the total mass of the helicopter assembly, *m* the mass of the rotor assembly, l_a the length of the main beam, l_h the distance from the pitch pivot to each of the propellers, l_θ the length of pendulum for the elevation axis, l_ϕ the pendulum for the pitch axis and $Drag$ the aerodynamical drag force on the travel axis. Let T_f and T_b represent the thrust supplied by the forward and backward propellors, respectively. The control inputs in equation 1 therefore become $T_{col}(t)$ and $T_{cyc}(t)$ which represent, respectively, the collective ($T_{col}(t) = T_f(t)+T_b(t)$) and cyclic ($T_{cyc}(t) = T_f(t) - T_b(t)$) thrusts generated by the DC motors.

Neglecting the non-dominant terms and under the assumption that $\sin(\rho(t)) \approx \rho(t)$ and $\sin(\epsilon(t)) \approx \epsilon(t)$ (since $\rho(t)$ and $\epsilon(t)$ are limited to small values) equations (1) can be approximated by the following simplified equations:

$$\begin{aligned} J_\epsilon \ddot{\epsilon}(t) &= -Mgl_\theta \epsilon(t) + l_a \cos(\rho(t)) T_{col}(t) \\ J_\rho \ddot{\rho}(t) &= -mgl_\phi \rho(t) + l_h T_{cyc}(t) \\ J_\tau \ddot{\tau}(t) &= l_a T_{col}(t) \cos(\epsilon(t)) \rho(t) \end{aligned} \tag{2}$$

Values for the plant parameters are shown in Table I.

III. Control Design

This section discusses the controller that was designed for the experimental system. The control objective was to track a specified travel rate and elevation. A feedback linearizing control was designed for the periodically triggered version of the system. An event-triggered implementation of this controller was then designed.

The equations of motion (eqn's 2) show a great amount of nonlinear coupling between the three degrees of freedom. These three equations may be decoupled in the following way. Since the travel dynamics are considerably slower than the pitch dynamics, we can essentially assume that the pitch angle ρ instantaneously tracks the pitch ρ_c required to track a specified travel rate $\dot{\tau}_c$ and elevation ϵ_c. This assumption allows us to decouple the pitch and travel states. The remaining nonlinearities in the elevation and travel equations can then be removed through feedback linearization.

In particular, let the states of the system be denoted as

$$x_\epsilon(t) = \begin{bmatrix} \epsilon(t) - \epsilon_c(t) \\ \dot{\epsilon}(t) \\ \int(\epsilon(t) - \epsilon_c(t))dt \end{bmatrix} \quad x_\rho(t) = \begin{bmatrix} \rho(t) - \rho_c(t) \\ \dot{\rho}(t) \\ \int(\rho(t) - \rho_c(t))ddt \end{bmatrix}$$
$$x_\tau(t) = \begin{bmatrix} \dot{\tau}(t) - \dot{\tau}_c(t) \\ \int(\dot{\tau}(t) - \dot{\tau}_c(t))dt \end{bmatrix}$$

In this system the elevation, $\epsilon(t)$, pitch $\rho(t)$, and travel $\tau(t)$ are measured directly through encoders on the plant. The other state variables must be estimated through observers.

The commanded

In order to control the helicopter body at the desired elevation and travel rate, three control loops are defined:

- *Elevation loop*: with $T_{col}(t)$ selected as control signal.
- *Pitch loop*: with $T_{cyc}(t)$ as the control signal.
- *Travel loop*: with a desired pitch angle ($\rho_c(t)$) introduced as the control signal [7].

Since the travel dynamics are considerably slower than the pitch dynamics of the system, it is assumed for the travel loop that $\rho(t)$ instantaneously and accurately tracks $\rho_c(t)$ ($\rho(t) \approx \rho_c(t)$).

State feedback controllers are then proposed for all three loops in the system. The states of the system are defined as follows:

$$x_\epsilon(t) = \begin{bmatrix} \epsilon(t) - \epsilon_c(t) \\ \dot{\epsilon}(t) \\ \int(\epsilon(t) - \epsilon_c(t)) \end{bmatrix} \quad x_\rho(t) = \begin{bmatrix} \rho(t) - \rho_c(t) \\ \dot{\rho}(t) \\ \int(\rho(t) - \rho_c(t)) \end{bmatrix}$$
$$x_\tau(t) = \begin{bmatrix} \dot{\tau}(t) - \dot{\tau}_c(t) \\ \int(\dot{\tau}(t) - \dot{\tau}_c(t)) \end{bmatrix}$$

where $\epsilon(t)$ and $\rho(t)$ are obtained directly from the system encoders, $\dot{\epsilon}(t)$, $\dot{\rho}(t)$ and $\dot{\tau}(t)$ will be appoximated using state observers and $\int(\epsilon(t) - \epsilon_c(t))$, $\int(\rho(t) - \rho_c(t))$, $\int(\dot{\tau}(t) - \dot{\tau}_c(t))$ approximations will be obtained by an Euler integration algorithm.

Figure 2 shows an schematic of the proposed control system implementation.

A. Feedback Linearization

Given the non-linear expressions for the control loops defined in (2), a feedback linearization approach is used to simplify the model.

Let:

$$\begin{aligned} T_{col}(t) &= \frac{K_\epsilon x_\epsilon(t)}{\cos(\rho(t))} && \forall \cos(\rho(t)) \neq 0, \\ \rho_c(t) &= \frac{K_\tau x_\tau(t)}{T_{col}(t)\cos(\epsilon(t))} && \forall\, T_{col}(t)\cos(\epsilon(t)) \neq 0, \\ T_{cyc}(t) &= K_\rho x_\rho(t) \end{aligned} \tag{3}$$

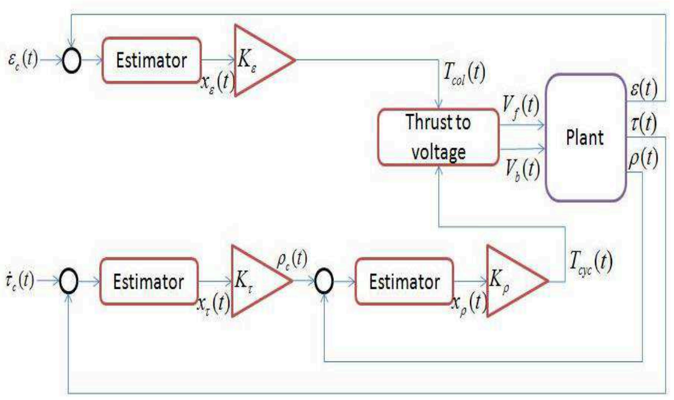

Fig. 2. Diagram of proposed control system.

where $\epsilon(t)$ and $\rho(t)$ are obtained from the encoders on the system and $K_\epsilon x_\epsilon(t)$, $K_\tau x_\tau(t)$, $K_\rho x_\rho(t)$ are the state feedback controllers to be designed for the elevation, travel and pitch loops respectively. Under the assumption that the feedback data is suffeciently accurate the following linearized closed-loop equations are obtained:

$$\begin{aligned} J_\epsilon \ddot{\epsilon}(t) &= -Mgl_\theta \epsilon(t) + l_a K_\epsilon x_\epsilon(t) \\ J_\rho \ddot{\rho}(t) &= -mgl_\phi \rho(t) + l_h K_\rho x_\rho(t) \\ J_\tau \ddot{\tau}(t) &= l_a K_\tau x_\tau(t) \end{aligned} \tag{4}$$

B. State Estimation

Derivative components are approximated with linear state observers of the following form:

$$\begin{aligned} \dot{\hat{x}}_\epsilon(t) &= A_\epsilon \hat{x}_\epsilon(t) + B_\epsilon cos(\rho(t)) T_{col}(t) + L_\epsilon(\epsilon(t) - \hat{\epsilon}(t)) \\ \dot{\hat{x}}_\rho(t) &= A_\rho \hat{x}_\rho(t) + B_\rho T_{cyc}(t) + L_\rho(\rho(t) - \hat{\rho}(t)) \\ \dot{\hat{x}}_\tau(t) &= A_\tau \hat{x}_\tau(t) + B_\tau T_{col}(t) cos(\epsilon(t)) \rho_c(t) + L_\tau(\tau(t) - \hat{\tau}(t)) \end{aligned}$$

Where $\dot{\hat{x}}_\epsilon$, $\dot{\hat{x}}_\rho$ and $\dot{\hat{x}}_\rho$ represent the estimated states and the matrices A and B are obtained from the loop equations (4). State observer gains (L_ϵ,L_ρ,L_τ) are obtained empirically with the use of the simulation platform. A discrete-time version of the linear observers with a fixed sample time of 0.005 seconds is implemented.

The Integral components ($\int(\epsilon(t) - \epsilon_c(t))$, $\int(\rho(t) - \rho_c(t))$, $\int(\dot{\tau}(t) - \dot{\tau}_c(t))$) are obtained using a simple Euler integration algorithm. In this approach the integral approximation is updated by adding a rectangular area equal to the latest measurement multiplied by the sampling period between measurements (the width of the rectangle).

C. Periodic Controller

Controller gains for state feedback controllers are defined, with the use of an LQR approach, as follows:

$$\begin{aligned} K_\epsilon &= [-44.00 \quad -7.00 \quad -68.00] \\ K_\rho &= [-30.65 \quad -3.54 \quad -11.54] \\ K_\tau &= [-22.60 \quad -14.03] \end{aligned}$$

Since the controllers have been designed in continuous time, a 'sufficiently small' period needs to be defined for the execution of a periodic controller. A test case is simulated with different control task periods ranging from 0.001 to 0.1 seconds. A period of 0.01 seconds is identified as the greatest possible time period that has no significant impact in the control performance.

D. Event-driven Controller

A linear system with $x(t) \in \mathtt{R}^n$, $u(t) \in \mathtt{R}^m$ and a state feedback controller rendering the closed-loop system asymptotically stable, can be described by $\dot{x}(t) = Ax(t) + BKx(r_j)$, where r_j represents the last sampling time when the controller was executed.
Introducing the following expression:

$$e(t) = x(r_j) - x(t) \tag{5}$$

which represents the difference ('gap') between the state value at the time of last update of the actuators (r_j) and the current state value. Then, the system can be rewritten as follows:

$$\dot{x}(t) = (A + BK)x(t) + BKe(t) \tag{6}$$

It was shown in [2] that the system in (6) is input-to-state (ISS) stable with respect to the gap measurement $e(t) \in \mathtt{R}^n$ if there exists a ISS Lyapunov function $V : \mathtt{R}^n \to \mathtt{R}_0^+$ satisfying

$$\begin{aligned} \underline{a}|x(t)|^2 \leq V(x(t)) \leq \bar{a}|x(t)|^2 \\ \dot{V}(x(t)) \leq -a|x(t)|^2 + b|e(t)||x(t)| \end{aligned} \tag{7}$$

with $\underline{a}, \bar{a}, a, b \in \mathtt{R}^+$. It was also shown that the inequality holds for a $|e(t)| \leq \sigma|x(t)|$ with some σ satisfying $-a + b\sigma < -a'$ with $a' > 0$.
The equation $|e(t)| = \sigma|x(t)|$ can therefore be adopted as the event-triggering condition for the execution of the control task while guaranteeing the stability of the system.

In the experimental system, the closed loop equations resulting from combining the system equations (2) with the feedback linearization (3) and state feedback controller are:

$$\begin{aligned} \dot{x_\epsilon}(t) &= A_\epsilon x_\epsilon(t) + \frac{cos(\rho(t))}{cos(\rho(r_j))} B_\epsilon K_\epsilon x_\epsilon(r_j), \\ \dot{x_\rho}(t) &= A_\rho x_\rho(t) + B_\rho K_\rho x_\rho(r_j), \\ \dot{x_\tau}(t) &= A_\tau x_\tau(t) + \frac{T_{col}(t) cos(\epsilon(t))}{T_{col}(r_j) cos(\epsilon(r_j))} B_\tau K_\tau x_\tau(r_j). \end{aligned}$$

It is then noted that an inexact cancellation of parameters can occur if the sampled feedback data used for linearization is different than the current values of the states included in the non-linearities of the equations. This error induced by feedback linearization can be viewed as a contribution to the total error generated by the discrete implementation of the controller. It can then be included in the 'gap' expression (5) in the following manner:

$$\begin{aligned} e_\epsilon(t) &= \frac{cos(\rho(t))}{cos(\rho(r_j))} x_\epsilon(r_j) - x_\epsilon(t), \\ e_\rho(t) &= x_\rho(r_j) - x_\rho(t), \\ e_\tau(t) &= \frac{T_{col}(t) cos(\epsilon(t))}{T_{col}(r_j) cos(\epsilon(r_j))} x_\tau(r_j) - x_\tau(t). \end{aligned}$$

An the resulting closed loop expressions can be descibed by:

$$\begin{aligned} \dot{x_\epsilon}(t) &= (A_\epsilon + B_\epsilon K_\epsilon) x_\epsilon(t) + B_\epsilon K_\epsilon e_\epsilon(t), \\ \dot{x_\rho}(t) &= (A_\rho + B_\rho K_\rho) x_\rho(t) + B_\rho K_\rho e_\rho(t), \\ \dot{x_\tau}(t) &= (A_\tau + B_\tau K_\tau) x_\tau(t) + B_\tau K_\tau e_\tau(t). \end{aligned}$$

The triggering conditions for the three loops in the system can then be calculated as defined for equation (6). A quadratic Lyapunov function of the following form is selected:

$$V(x(t)) = x^T(t)Px(t)$$

where P is a real symmetric matrix that holds in:

$$(A + BK)^T P + P(A + BK) = -Ia$$

for the corresponding closed loop term $(A + BK)$ and an arbitrarily chosen constant $a \in \mathtt{R}^+$. It then follows that:

$$\dot{V}(x(t)) \leq -a|x(t)|^2 + |K^T B^T P + PBK||e(t)||x(t)|$$

and the triggering conditions can be calculated following the procedure described for equation (7). The following thresholds were calculated for the loops in the system;

$$\begin{aligned} |e_\epsilon(t)| &= 0.04|x_\epsilon(t)|, \\ |e_\rho(t)| &= 0.09|x_\rho(t)|, \\ |e_\tau(t)| &= 0.28|x_\tau(t)|. \end{aligned} \tag{8}$$

E. DC-Motors Voltage Input

Since the 3DOF Helicopter is controlled by two DC motors it is required to express $T_{col}(t)$ and $T_{cyc}(t)$ in terms of a DC voltage applied to each motor. Ignoring the dynamics of the DC motors (due to much faster response compared to the rest of the dynamics [6]), input voltages are defined by:

$$\begin{aligned} V_f(t) &= \tfrac{1}{2K_f}(T_{col}(t) + T_{cyc}(t)) \\ V_b(t) &= \tfrac{1}{2K_f}(T_{col}(t) - T_{cyc}(t)) \end{aligned}$$

Where $V_f(t)$ and $V_b(t)$ are the voltage applied to the front and back motors respectively and K_f is the motor volt-to-thrust relationship constant [7]. A saturation region is added to maintain the applied voltage within the operating region (± 5 volts).

IV. Implementation

The real-time S.Ha.R.K. [8] kernel (Soft Hard Real-Time Kernel) is used for implementation. The kernel provides a simple enviroment for the development of real-time applications with specific scheduling algorithms. The kernel produces an executable application that runs in a FreeDOS system.

The 3DOF Helicopter is considered a real-time system since it has been established that has restrictive timing constraints that must be met to achieve the desired behavior.
As in most of the real-time systems, the tasks in the 3DOF Helicopter system can be divided in two classes:

- *Hard Tasks* in which completion after a deadline can cause catastrophic consequences on the system.
- *Soft Tasks* in which a missing deadline results in a decrease in the performance of the system but does not jeopardize its correct behavior.

TABLE II
Task set for the Periodic System

Hard Tasks

Task	Period (ms)
Elevation Controller	10
Pitch Controller	10
Travel Controller	10
Data Acquisition and Estimation	5
Dummy Task (Hard)	5

Soft Tasks

Task	Period (ms)
Data Collection	50
System Load Estimation	100
Control Tasks Load Estimation	50
Dummy Task (Soft)	10

TABLE III
Task set for the Event-Driven System

Hard Tasks

Task	Period (ms)
Event Triggering	10
Elevation Controller	-
Pitch Controller	-
Travel Controller	-
Data Acquisition and Estimation	5
Dummy Task (Hard)	5

Soft Tasks

Task	Period (ms)
Data Collection	50
System Load Estimation	100
Control Tasks Load Estimation	50
Dummy Task (Soft)	10

A. Periodic Controller

Control tasks are implemented as three independent hard tasks (one for every loop in the system). Task are activated periodically with a period of 0.01 seconds The developed applications also include: a periodic hard task for data acquisition and estimation of parameters (0.005s period), a periodic soft task for data collection, a periodic soft task for system computation load estimation and a periodic soft task for calculating the load generated by the controller tasks. User-activated dummy hard and soft tasks to be used for testing purposes have also been included. The complete set of tasks implemented for the event-triggered system are presented in table II.

B. Event-driven Controller

The controller is implemented using three aperiodic hard tasks for control purposes (one for every control loop), these tasks are activated by a periodic event-triggering hard task with a period of 0.01 seconds that monitors the event-triggering conditions presented in (8). The data aquisition, load estimation and dummy tasks designed for the periodic controller are also included in the event-triggered system. The complete set of tasks implemented for the event-triggered system are presented in table III.

C. Scheduling Algorithm

An Earliest Deadline First (EDF) algorithm is used as the task scheduling policy for both periodic and event-driven

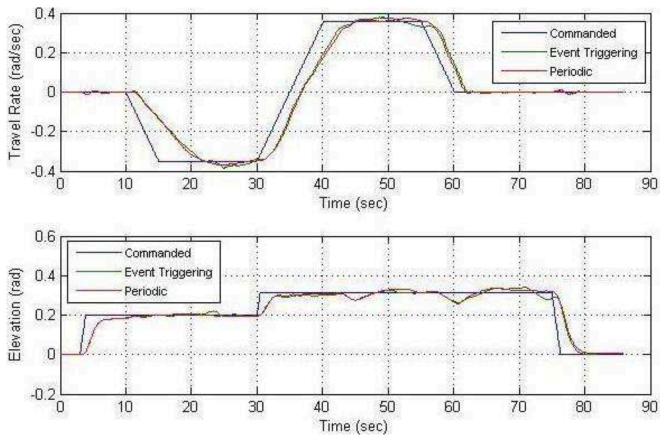

Fig. 3. System Travel Rate and Elevation for a 90s test case.

TABLE IV
NUMBER OF TASK ACTIVATIONS AND CPU TIME UTILIZED BY TASKS.

	Periodic Triggering		Event-Triggering	
Task	Activations	CPU Time	Activations	CPU Time
Pitch	9000	111.256 ms	4088	52.538 ms
Travel	9000	103.663 ms	2884	34.791 ms
Elevation	9000	40.305 ms	3975	20.109 ms
TOTAL	**27000**	**255.224 ms**	**10947**	**107.438 ms**

systems.
In this algorithm, the task with the earliest absolute deadline is executed and is optimal in the sense of feasibility (completeness of every task according to time constraints) [9].

V. RESULTS

Systems are compared using a 90 seconds test case designed to experiment with different elevation and travel rate configurations. Figure 3 shows the travel rate and elevation response of the systems. It is shown that the event-driven system tracks the reference signals achieving the same performance level shown by the periodic controller. Event-triggered system improvements on resource utilization are presented in Figure 4 where the CPU time utilized by the controllers is compared. The mean CPU utilization [9] presented is obtained by averaging a 1 second window of the processor time spent in a specific task and then dividing it by the sampling period (0.05 seconds).
Improvements are specially evident in the travel rate loop except for the initial and final part of the simulation where the the travel state is close to zero. In these cases, the controller gets triggered at every opportunity. The fact that this behavior is not present in the elevation and pitch loops suggests that the travel rate controller activations might be caused by noise induced in the linear state observer. Further investigation is required to confirm this theory and to design a possible solution to the problem. The improvements shown in Figure 4 are quantified in Table IV with a comparison of the number of executions and total CPU time utilized by the control tasks in the systems.

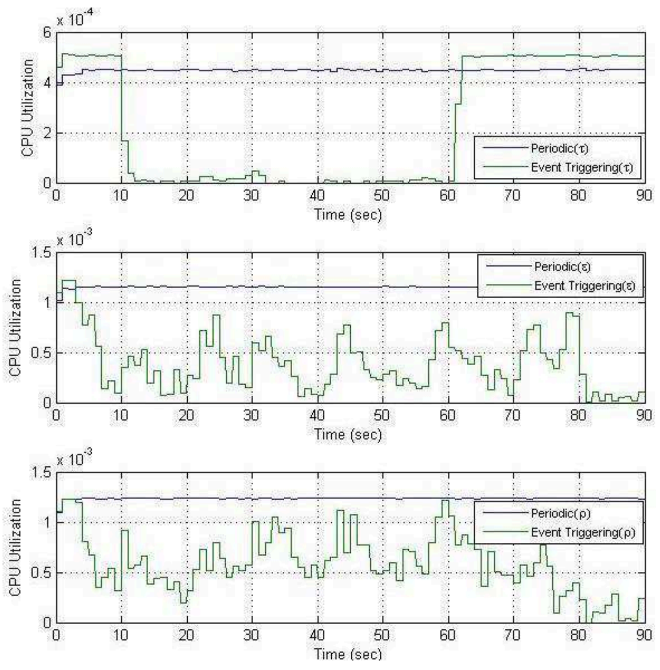

Fig. 4. Mean CPU Utilization of Travel rate controller task (upper), Elevation controller task (middle) and Pitch controller task (lower).

VI. CONCLUSIONS

This paper has discussed the implementation of an event-triggered controller in a non-linear mechanical system. A periodic version of the controller was also implemented for performance comparison. A modification to the ‘gap’ expression introduced in previous works for event-driven controllers, is presented. With the proposed expression, the system is able to respond to errors introduced by an inexact cancellation in the feedback linearization. Systems implementation included the use of a real-time kernel that provides a simple modular architecture in which controllers and additional functions of the system were implemented as hard or soft tasks.

Results have shown that the use of event-driven controllers lead to a significant improvement in the total CPU load generated by the control tasks while maintaining the control performance.
The obtained results have shown that event-driven implementations might be a feasible solution to applications with rigoroug real-time constraints even in highly non-linear and “safety critical” applications such as a helicopter.

Future work still needs to investigate the robustness and performance of the system under different scenarios such as aggressive maneuvers and high CPU loads. In particular, the response of the system when a plant state is near zero should be investigated. Future work will also study the performance of the system under different task configurations and different scheduling algorithms.

REFERENCES

[1] A. Anta, P. Tabuada, *To sample or not to sample: Self-triggered control for nonlinear systems*: To appear in the IEEE Transactions on Automatic Control, 2009.

[2] P. Tabuada, *Event-Triggered Real-Time Scheduling Stabilizing Control Tasks*: IEEE Transactions on Automatic Control. Volume 52, number 9, pp 1680-1685, 2007.

[3] X. Wang, M. Lemmon, *Self-triggered Feedback Control Systems with Finite-Gain L2 Stability*: IEEE Transactions on Automatic Control, Volume 54, Number 3, pages 452-467, 2009.

[4] A. Camacho, P. Marti, M. Velasco, E. Bini, *Implementation of Self-triggered Controllers*: Demo Session of 15th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS09), San Francisco, CA, USA, April 2009.

[5] A. Cervin, T. Henningsson, *Scheduling of Event-Triggered Controllers on a Shared Network*: 47th IEEE Conference on Decision and Control, 2008.

[6] S. Bayraktar, *Aggressive Landing Maneuvers for Unmanned Aerial Vehicles*: MSc Thesis, Massachusetts Institute of Technology, 2006.

[7] *Quanser© 3D Helicopter System manual*.

[8] P. Gai, L. Abeni, M. Giorgi, and G. Buttazzo, *A New Kernel Approach for Modular Real-time Sysetms Development*, Proceedings of the 13th IEEE Euromicro Conference on Real-time Systems, June 2001.

[9] Giorgio Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Second Edition, Springer, 2005