Period Assignment for Real-Time Cascade Control Tasks Under Stability and Schedulability Constraints

Ismail Hawila
Kopernic, Inria, Paris, France
StatInf, Gentilly, France

Liliana Cucu-Grosjean
Kopernic, Inria, Paris, France

Slim Ben Amor
StatInf, Gentilly, France

Abstract

Existing results for cyber-physical systems have been proposed to merge the requirements associated to the stability of the physical components and the schedulability of the cyber components. Nevertheless, none of the existing results has studied these requirements for multiple real-time cascade control tasks where their periods choice are dependent and affect stability. In this paper, we propose a methodology to evaluate the periods of the real-time cascade control tasks that ensures stability of the physical components, then we present a co-design problem for the period choice that guarantees good performance of the physical components and schedulability of the cyber components under fixed-priority scheduling. We then evaluate this methodology on a real use-case of a drone system. Results show the importance of studying these requirements together as their relation has an impact on stable periods range.

2012 ACM Subject Classification Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Real-time systems; Computing methodologies → Model development and analysis

Keywords and phrases Real-time Systems, Cascade Control, Physical Stability, Control Performance

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2025.7

Funding This work is, partially, funded by FR ANR France Relance 2030 Mach 2 project, FR ANRT CIFRE StatInf-INRIA grant, FR INRIA ADT KDBench project, FR PSPC STARTEC project and FR INRIA Kepler Associated Team.

1 Introduction

Our every day life is improved by the presence of “a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities” [4]. These systems are called cyber-physical systems (CPSs). A defibrillator, a mobile phone, an autonomous car or an aircraft, are all examples of CPSs. Beside constraints like power consumption, security, correctness, size and weight, CPSs may have cyber components required to fulfill their program computations within a limited amount of time, the program is then called real-time task. A real-time task (or simply task) is a recurring program, meaning that consecutive releases of the same program are separated by a predefined time interval. Each release is then referred to as a job, and outputs of the jobs may differ depending on the input received at the start of each job instance. Real-time tasks must meet stringent timing requirements, often imposed by the physical environment, which are also known as real-time constraints. In this paper, a real-time constraint is considered as met by a task if its worst-case response time (WCRT) is always smaller than the minimal inter-arrival time between two jobs. The worst-case response time is the largest time elapsed from the beginning of the job's execution until its end (including preemptions, suspensions, etc).

Concretely, the CPS design implies the implementation of controller programs as periodic tasks that have real-time constraints to meet. More precisely we understand by *controller*, a cyber component algorithm deciding what actions must be taken based on sensor readings or other input data. The output of the controller is the input to the physical component of the system, usually called *plant*, which forces the output of the plant to follow a desired setpoint [6]. This implies that a *control task* implements the controller's algorithm, such that computing the output of the controller program is done within this task. Moreover, by stability we mean that the output of the CPS is under control, following the desired setpoint in the presence of external perturbations.

Usually, the controller programs are modeled as real-time tasks and the associated CPS task model [10] is considered more general than the classical real-time task model as defined by Liu and Layland in [13]. Recent results [5] have underlined that there is a gap between the CPS *stability* as defined within a control theory context and the *schedulability* as defined within the real-time scheduling community and the authors have proposed a new model to fill this gap. Nevertheless, *the problem of having multiple control dependent tasks has not been studied from a control scheduling co-design perspective.* Often, the real-time community has studied the impact of the period variation on the stability of one controller. To the best of our knowledge, no result has been presented for the stability of a system with multiple real-time cascade control tasks. Real-time cascade control tasks represent multiple control tasks concatenated together, where the output of one control task is the input of the other.

Our contribution: In this paper, we investigate the stability of the cascade control system in order to derive an interval of periods upon which the system is stable. Then we propose an optimization problem to choose the proper values of periods from the stable region based on the control performance and system utilization. We then evaluate our methodology for drone dynamics inspired by a real use-case.

Organisation of the paper: The paper is structured as follows. In Section 2 we present existing results that combine control and scheduling requirements. In Section 3, we present the real-time and control system descriptions along with the proposed co-design problem formulation. In Section 4, we describe the methodology to analyse the stability of the system and derive stable regions of periods. In Section 5, we present a bound on the number of allowable deadline misses for a control task based on the stability region. In Section 6, we define the use-case used in our study, and then evaluate our methodology in Section 7. In Section 8 we present interesting observations noted during the evaluation phase and we conclude in Section 9.

2 Control-Scheduling State of the Art

2.1 Scheduling and Control Co-Design

Many studies have been dedicated to the control-scheduling co-design problem. An early result [21] considers the optimization of global system performance using control performance index and subject to computing resources constraints. In [18], authors present an approach to optimize end-to-end timing constraints subject to performance and schedulability constraints of a real-time control system. Here periods of control tasks are modified to satisfy the objective of minimizing systems utilization while respecting the performance requirements. In [20], authors follow the same approach while considering the real-time control system subject to communication delays. [24, 25] targeted the control scheduling co-design, where the assignment of periods and priorities of multiple control applications running on the same processor is done considering some control performance metrics. Authors of [2] consider the effect of delays and jitter on the system stability and provided an approach for period assignment and scheduling scheme to guarantee stability. In [1], authors consider the problem of synthesizing timing contracts that guarantee stability and schedulability. All mentioned results considered hard-deadlines, meaning each task is required to meet all deadlines. In reality, control tasks could allow some relaxation of this constraint by allowing some deadline misses, while periods are chosen, often, as multiples of sensor periods leading to harmonic rates [22].

2.2 Fault Tolerant Control System

The relaxation of hard deadline requirements for real-time control systems is, also, considered in the literature. It is shown that the stability of control systems is not only dependent on the system dynamics and choice of periods, but also the stability depends on the number of consecutive deadline misses that the control task encounters. In [23], authors analyze stability and performance of control systems subject to burst of deadline misses, and show that stability analysis alone is not sufficient to properly quantify robustness regarding deadline misses. In [16], authors propose an adaptive control design to handle the sporadic overloads to counteract the effect on stability and performance. In [10], authors propose a new periodic fault-tolerant CPS task model, which generalize the existing task models. But also the model captures the stability and efficiency of the CPS by capturing tolerable number of control update misses. In this scheduling scheme tasks periods are considered as given.
Authors in [5] propose a first method to combine both previously mentioned working paths. This is done by proposing a model to jointly determine tasks periods and manage deadline misses. In their paper, a stable region of the physical system is presented, each point in this region corresponds to a pair of period and number of allowed consecutive deadline misses. This means in their proposed model to know that a control task can tolerate a certain number of consecutive deadline misses, its period could be chosen between a minimum and a maximum value based on the stability region.
To the best of our knowledge, no other result has been proposed to the co-design problem when the system is composed of multiple dependent control tasks in a cascade structure. So, there is no result on the effect of the dependence between control tasks on the region of stable periods and how do we quantify the maximum number of allowable deadline misses based on the region of stability.

3 Problem Definition

3.1 Real-time Model

We consider a task system τ of n synchronous implicit deadline periodic tasks $\tau_i, \forall 1 \leq i \leq n$ scheduled according to a preemptive fixed-priority scheduling policy on one processor π. The tasks are ordered according to their priority. A task τ_i has a higher priority than a task τ_j when $1 \leq i < j \leq n$. The priority assignment is given and finding such assignment is beyond the purpose of this paper. A task τ_i is defined by C_i, its worst-case execution time, T_i its minimal inter-arrival time or period and D_i, its deadline with $D_i = T_i$. We denote by $U_i = \frac{C_i}{T_i}$ its utilization and the total utilization of the task system τ is denoted by $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$. We assume that all tasks are released for the first time synchronously at $t = 0$.

3.2 Cascade Control Model

A control system regulates or controls the behaviour of other devices or systems using control loops [7, 3, 11, 15]. There are, mainly, two types of control systems, open-loop and closed-loop control. Closed-loop control systems (also known as feedback control systems) have been the first examples of hard real-time systems [23]. In this paper, we deal with closed-loop control systems.

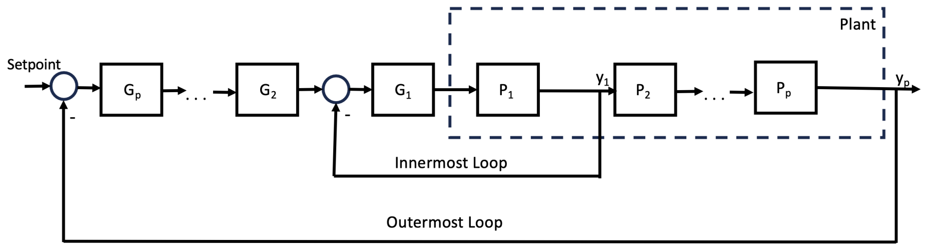


Figure 1 Cascade control system closed-loop.

Cascade closed-loop control is the use of multiple control loops to control the behaviour of a system. In Figure 1, we illustrate p cascade loops each containing a controller G_j and a plant $P_j, \forall 1 \leq j \leq p$. In a cascade control system, the output of the outer loop controller G_2 is the setpoint to the inner loop controller G_1. In addition, G_1 is faster than G_2 allowing the inner loop to react rapidly to fast changing dynamics which reduce the error of the outer loop.

Stability

A control system is said stable if its output is bounded for a given bounded input. For instance, in Figure 1 for a bounded input of controller G_1, a bounded output y_1 is expected. The stability of a cascade control is studied in order to derive a stability region as a function of the sampling period of each controller. That is for a controller G_i we derive a range of periods for T_i in an interval $[T_i^{min}, T_i^{max}]$, such that any period value in this interval ensures stability of the system. Several methods exist to identify such interval and in this paper, we use the Jury's stability [17] (see more details in Section 4).

Control Performance

When it comes to control tasks, we evaluate the performance of the closed-loop system since the stability itself does not guaranty the optimal behaviour of the system. Usually, the control performance is defined in terms of a cost function that is evaluated during the run-time of the controller. In this paper, we define the performance as the cost defined by the integral of time absolute error (ITAE) [8]:

$$J = \int_0^\infty t\,|e(t)|\,dt \tag{1}$$

Where $e(t)$ is the system error between the desired output (setpoint) and the measured output. This is not the unique way to define the control cost, other formulations exist such as the Integral of squared error (ISE) or integral of absolute error (IAE). As stated in [19], ITAE has the advantage of producing smaller overshoots and oscillations than ISE and IAE. In addition, ITAE has the selectivity needed for a good metric meaning a minimum value could be captured when system parameters are varied. A large value of the cost J indicates a low control performance, meaning that there is a large deviation from the desired setpoint. The cost J depends also on the sampling period, the more frequent sampling of control tasks is, the smaller is the error leading to a lower (better) cost.

3.3 Co-design Problem

The co-design problem is defined as the problem where the stability and the schedulability meet with the objective of finding a compromise solution for each of them. From a stability and performance point of view, one wants the control tasks to have the smallest possible periods, while this might overload the system from a scheduling point of view. So, the co-design problem requires a solution optimizing the control performance represented by Equation 1, while respecting stability and schedulability guarantees. Then, we define the optimization problem as follows:

$$\min_{T_i} J \tag{2}$$

$$\begin{aligned} &s.t. \\ &\quad \sum_{i=1}^{n} U_i \leq n(2^{\frac{1}{n}} - 1), \\ &\quad T_i^{min} \leq T_i \leq T_i^{max},\ \forall\ 1 \leq i \leq n \\ &\quad T_i \leq T_j,\ \text{ where } \tau_i \text{ is an inner loop task wrt. } \tau_j \end{aligned}$$

For the control performance we choose sampling periods that decrease the control cost J as much as possible. These periods should be chosen within the interval $[T_i^{min}, T_i^{max}]$ which ensures the stability of the physical system. Our solution to compute this interval is described in Section 4. Also, the period of the inner loop should be smaller than that of the outer loop to reduce overall system error, where in the third condition i corresponds to the inner loop and j to the outer loop. Moreover, from scheduling point of view we chose periods that guarantee there is no violation of the sufficient condition of schedulability according to a fixed priority scheduling algorithm with the total utilization lower than $n(2^{\frac{1}{n}} - 1)$.
This bound is introduced in [13] as follows: *For a set of n tasks scheduled with fixed priority, and the restriction that the ratio between any two request periods is less than 2, the least upper bound to the processor utilization factor is $U = n(2^{\frac{1}{n}} - 1)$.*
For evaluation purposes and when $n \to \infty$, $n(2^{\frac{1}{n}} - 1) \to 0.69$, since $n(2^{\frac{1}{n}} - 1)$ is a decreasing function with respect to the number of tasks, then for any number of tasks 0.69 is a sufficient bound [13].

3.4 Example

To illustrate our definitions and results, we consider an example of tasks set τ_{ex} with 9 tasks ordered according to their priorities associated to the use-case described in Section 6. A task τ_i has a higher priority than a task τ_j when $1 \leq i < j \leq 9$. Tasks are scheduled on one processor according to a preemptive fixed priority scheduling algorithm. Their parameters are described in Table 1. The worst-case execution time C is obtained as the largest observed execution time during 5 minutes of functioning for the associated use-case. The task system contains 9 tasks, 3 of which are control tasks. The relation between control tasks is described by cascade control loops in Figure 2. As for non-control tasks, we do not present them in

Figure 2 because we consider that the output of the plant is complete and no data could be missed. Whereas in real implementation we use sensors to capture the behaviour of the system, in this case data should be estimated, filtered and corrected and this is the difference between real implementation and simulations.

Table 1 The parameters of tasks belonging to τ_{ex}.

Task index	**Task name**	$T_i = D_i$ (ms)	C_i (ms)	**Control task**
τ_1	*Sensors*	10	0.761	Non
τ_2	*EKF2*	10	5.315	Non
τ_3	*Hover trust*	15	0.114	Non
τ_4	*Navigator*	25	1.365	Non
τ_5	*Position control*	15	0.236	Yes
τ_6	*Flight manager*	15	0.511	Non
τ_7	*Attitude control*	15	0.138	Yes
τ_8	*Commander*	50	0.266	Non
τ_9	*Rate control*	15	0.17	Yes

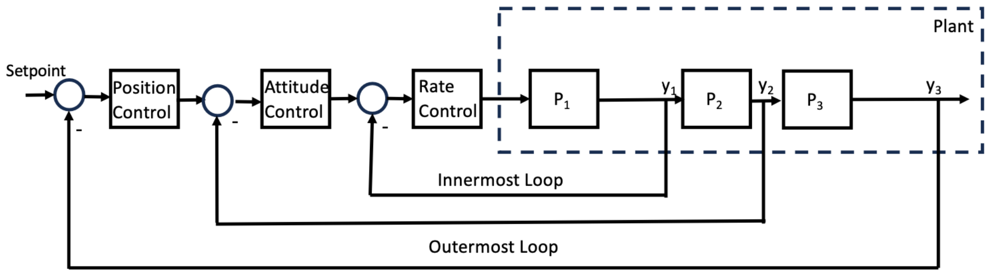


Figure 2 Cascade control system closed-loop for our example.

4 Stability Analysis

When designing and analysing such system the inner-most loop should be treated first, then the loop that generates the setpoint of the inner-most loop until we reach the outer-most loop. To study the stability region of the system with respect to the sampling periods of the cascade controller, we represent the dynamics of the system using a linear transfer function representation. Afterwards, we discretize this transfer function with respect to the sampling period T_i.

4.1 Closed-loop

Plants of real applications are non-linear, but for control design purposes usually the plant dynamics are linearized around an operation point. For our general stability analysis we consider two plants P_1 and P_2 described by transfer functions of the general form:

$$\begin{cases} P_1(s) = \dfrac{a_1}{b_1 s + c_1} \\ P_2(s) = \dfrac{a_2}{b_2 s + c_2} \end{cases} \tag{3}$$

where a_i, b_i and c_i are constants derived from the dynamics of the physical plant as described in Section 6.1 (see Equation 14).
The controllers G_1 and G_2 are assumed to be **PID** controller and its general transfer function is as follows:

$$G_{1,2}(s) = K_{p_{1,2}} + \frac{K_{i_{1,2}}}{s} + K_{d_{1,2}} s = \frac{K_{d_{1,2}} s^2 + K_{p_{1,2}} s + K_{i_{1,2}}}{s} \tag{4}$$

Forward discretization

To introduce the sampling period we discretize the equations of the plants and the controllers to transfer them from continuous-time to discrete-time. We do this by replacing the Laplace variable s with an approximation using the forward Euler method:

$$s \approx \frac{z-1}{T}$$

where T is the sampling period and z is the forward time shift.
Combining the dynamics and the controller descriptions from equations 3 and 4 is used to study of the stability of the closed-loop as follows:

$$\begin{cases} CL_1(z, T_1) = \dfrac{G_1(z)P_1(z)}{1 + G_1(z)P_1(z)} \\ CL_2(z, T_1, T_2) = \dfrac{G_2(z)P_2(z)CL_1(z, T_1)}{1 + G_2(z)P_2(z)CL_1(z, T_1)} \end{cases} \tag{5}$$

where T_1 and T_2 are the sampling periods of the inner and outer loops, respectively. It is to be noted that the outer loop depends on the parameters of the inner loop, so that the closed-loop transfer function of the P_2 would depend on its period T_2 and T_1 the period of P_1. If several loops exist, then this procedure is generalized to obtain the closed-loop transfer functions.

4.2 Jury's stability

Jury's stability is a criterion for discrete-time systems that is applied to the characteristic equations that are a function of z to check there stability. Characteristic equation is the denominator of the closed-loop transfer function. Assuming a general characteristic equation of the form:

$$P(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 \tag{6}$$

Jury's stability provides necessary and sufficient conditions to ensure that $P(z)$ has no roots outside the unit circle.

Necessary condition for stability.

$$\begin{aligned} P(1) &> 0 \\ (-1)^n P(-1) &> 0 \end{aligned}$$

For the sufficient conditions we construct the Jury table first, then check the conditions.

Row	z^0	z^1	z^2	$\cdots$	z^{n-1}	z^n
1	a_0	a_1	a_2	$\cdots$	a_{n-1}	a_n
2	a_n	a_{n-1}	a_{n-2}	$\cdots$	a_1	a_0
3	b_0	b_1	b_2	$\cdots$	b_{n-1}	
4	(Row 3 reversed)					
5	c_0	c_1	c_2	$\cdots$	c_{n-2}	
6	(Row 5 reversed)					
7	d_0	d_1	d_2	$\cdots$	d_{n-3}	
$\vdots$	$\vdots$	$\vdots$	$\vdots$			

where each row is build based on the previous one as follows:

$$b_k = a_0 a_k - a_n a_{n-k}, \quad k = 0, 1, \ldots, n-1$$
$$c_k = b_0 b_k - b_{n-1} b_{n-1-k}, \quad k = 0, 1, \ldots, n-2$$

Sufficient condition for stability. To ensure stability, the following conditions must be satisfied:

$$\begin{cases} |a_0| < |a_n| \\ |b_{n-1}| < |b_0| \\ |c_{n-2}| < |c_0| \\ |d_{n-3}| < |d_0| \\ \quad\vdots \\ |m_2| < |m_0| \end{cases} \tag{7}$$

If any of these conditions fail, the system is unstable.
From these conditions, we solve the characteristic equation of the closed-loop transfer functions in Equation 5 to determine the ranges of T_1 and T_2 which keeping the system stable. Assuming all parameters other than the periods are given, the coefficients of the characteristic equation 6 ($a_n \ldots a_0$) are dependent on the periods only. This means, conditions of stability in Equation 7 could be solved to bound the values of T_i ensuring that the system is stable, leading to the stability interval.

We present in the flow chart of Figure 3 the procedure of deriving all stable periods of a system with 2 real-time cascade control tasks (for the sake of the simplicity, we present it for 2 loops but it is generalizable to m loops) . First, we fix the physical system parameters and controller parameters. We assume that the system at hand has at least a set of parameters where a value of T_1 and T_2 exists. Then we solve for the inner-most loop the Jury's conditions seen before to derive the interval of periods T_1 that are stable. If we did not succeed to find a solution then we go back and change the controller parameters. When a solution is found we then proceed to analyse the next loop. Now we know that the transfer function of the outer loop will depend on both T_1 and T_2, so to solve for T_2 we fix a value of T_1 we do so by looping over all values in the stable interval we derived in the previous step. After fixing T_1 the only variable now is the period T_2 and we solve again the jury's conditions to obtain the stable values of T_2. After looping over all values of T_1 in the stable interval, we obtain the combinations of T_1 and T_2 that make the overall system stable.

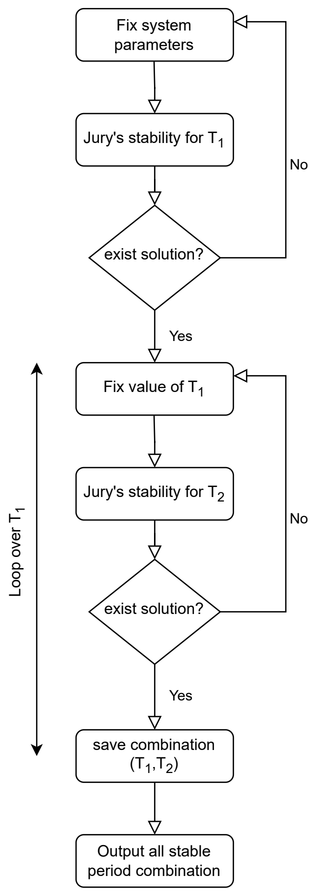


Figure 3 Flow chart describing the procedure followed to extract stable periods using Jury's stability.

5 Allowable deadline miss

A control system remains stable in the presence of a maximum number m of consecutive deadline misses, where the value of m depends on the dynamics of the controlled system (see Section 2.2). We understand by missing a deadline that each time a job does not complete its execution, the system uses data from the last previously completed job. Even if the system is stable for m consecutive deadline misses this may impact the performance of the system. Since the best performance is achieved when updating the control as frequent as possible, the presence of deadline miss reduce the control performance. If a job of a control task misses m consecutive deadlines, this is equivalent to say that we execute with a period $(m+1) \times T_i$. This reasoning allows us to propose the following result.

▶ **Theorem 1.** *In a task system τ, a control task τ_i with a period T_i within a region of stable periods $[T_i^{min}, T_i^{max}]$ keeps the system stable if it is subject up to a maximum number of consecutive deadline misses m is equal to:*

$$m = \left\lfloor \frac{T_i^{max}}{T_i} \right\rfloor - 1 \tag{8}$$

while all other tasks meet their deadlines.

Proof. From the stability point of view $(m+1) \times T_i = \left\lfloor \frac{T_i^{max}}{T_i} \right\rfloor \times T_i \leq T_i^{max}$, and we know that the system is stable for any $T_i \in [T_i^{min}, T_i^{max}]$ then the system is stable for $(m+1) \times T_i$. ◀

▶ **Example 2.** Looking at the example of Figure 4, Case 1 depicts a control task running with a period of $4ms$ we have one control update miss followed by a correct execution and control update. Thus, we get a new data every $8ms$, which is equivalent to Case 2 where we see a control task running with a period of $8ms$ and successfully completing its execution every time. Knowing that the system is stable for Case 2 with a control update each $8ms$, we can then say for a control period of $4ms$ the system remains stable if we miss a single control update.

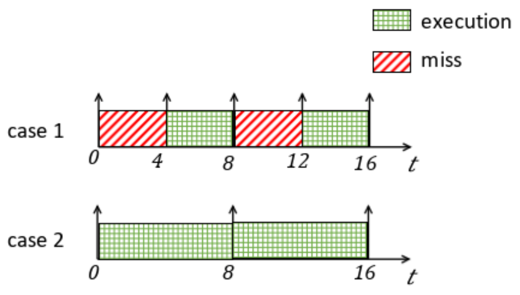


Figure 4 Control tasks deadline miss and update period.

6 Drone Autopilot Use-Case

We provide a use-case using the set of tasks associated to an existing autopilot, PX4 [1] modified to integrate periodic arrivals. The tasks or programs are executed on a real execution platform Pixhawk composed of a microcontroller (a single core ARM Cortex-M4), sensors and actuators. From those executions, we consider for each program a 4-tuple (A, p, $\mathcal{M}$, $c(A)$) composed by a program A, a microcontroller p, a measurement protocol $\mathcal{M}$ and an ordered sequence of execution times $c(A)$ of program A on the microcontroller p. For the program A, one provides the source code as well as the binary code. A measurement protocol $\mathcal{M}$ is defined by the variation of the input variables (associated to sensors) of these benchmarks. In our case, the variation of the input variables is obtained by collecting them during a simulated flight of a drone within a Gazebo simulation environment. The fourth element is an ordered set of execution times, $c(A)$, proposed to overcome the difficulty of the reproducibility of results while comparing execution times measured for the same program on

[1] `https://px4.io/software/software-overview/`

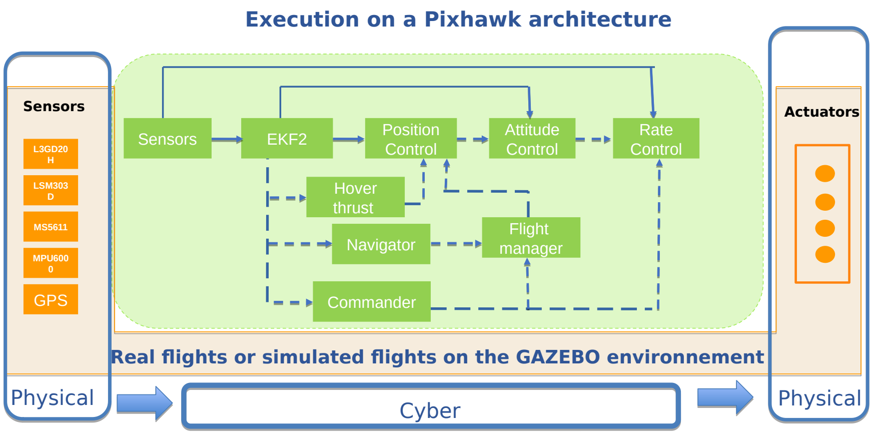


Figure 5 A representation of PX4 with the cyber and physical components.

slightly different microcontroller configurations. Moreover, we use execution times measured at the scheduler level in order to decrease the intrusion of the measurement protocol. Such measurements are said to be obtained in a hardware-in-the-loop manner. We understand by hardware-in-the-loop that the execution of the benchmarks is done on a microcontroller while sensors and actuators of a real-time system, as well as its physical environment, are simulated.
Among the 9 programs of the PX4 Figure 5, three are the control tasks *Position Control*, *Attitude Control* and *Rate Control*. These control tasks form three cascaded loops controlling the overall position of the drone (see more details in Section 3.4).

6.1 Drone dynamics

A first step to control design or stability analysis of any control system is to derive its dynamics representation. As seen in the use-case the targeted system is a drone, so we start with a general drone representation. The dynamics of the drone consists of two parts translational dynamics and rotational dynamics. The dynamics could be derived from the physical laws and has been exhaustively studied in control theory.
It follows from [12, 26], that the dynamics of the drone should be studied with respect to two coordinates, earth coordinate which is the general one and drone body coordinate which is fixed to the drone body and moves with it. The translational position of the drone is studied with respect to the body frame, then transformed into the earth frame using a transformation matrix. Then we get the translational acceleration of the drone as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \sin\phi \sin\psi + \cos\phi \sin\theta \cos\psi \\ -\sin\phi \cos\psi + \cos\phi \sin\theta \sin\psi \\ \cos\phi \cos\theta \end{bmatrix} \cdot u_1 \tag{9}$$

Similarly the rotational position is also first derived in the body frame of the drone and then transformed to the earth frame. Here we present $p, q, r, \dot{p}, \dot{q}, \dot{r}$ the angular velocity and acceleration of drone around x, y, z axis under body frame. Where as the roll, pitch and yaw: $\phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}$ are angular velocity and angular acceleration of drone in the earth frame:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{1}{I_x}\\ \frac{1}{I_y}\\ \frac{1}{I_z}\end{bmatrix} \cdot \begin{bmatrix}lu_2 - q\cdot r\cdot(I_z - I_y)\\ lu_3 - p\cdot r\cdot(I_z - I_x)\\ u_4 - p\cdot q\cdot(I_y - I_x)\end{bmatrix} \tag{10}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \tan\theta\sin\phi & \tan\theta\cos\phi\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix} \cdot \begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{11}$$

Generally the movement of the drone is controlled by varying the speed of its propellers. Here we represent the control outputs u_i in terms of K_T and K_Q the thrust and drag coefficients, and Ω the propeller speed. The u_i would then be computed by the control tasks, and the propeller speeds would have to be calculated from them for a real implementation.

$$\begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix} = \begin{bmatrix}K_T\cdot(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)\\ K_T\cdot(\Omega_1^2 + \Omega_2^4 - \Omega_2^2 - \Omega_3^2)\\ K_T\cdot(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2)\\ K_Q\cdot(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2)\end{bmatrix} \tag{12}$$

As could be noticed from the model the system is a nonlinear one. So for the design of the controller and stability analysis, we linearize the equations presented above. From [26] the linearized equations are as follows:

$$\begin{cases}\ddot{x} = \dfrac{u_1\cdot\theta}{m}\\ \ddot{y} = \dfrac{u_1\cdot\phi}{m}\\ \ddot{z} = \dfrac{u_1}{m}\\ \ddot{\phi} = \dfrac{l\cdot u_2}{I_x}\\ \ddot{\theta} = \dfrac{l\cdot u_3}{I_y}\\ \ddot{\psi} = \dfrac{l\cdot u_4}{I_z}\end{cases} \tag{13}$$

In order to reproduce a similar behavior to the PX4 autopilot, we aim to analyse three controllers:

- *Position Control* which controls both position and velocity of the translational motion x, y, z and $\dot{x}, \dot{y}, \dot{z}$ respectively.
- *Attitude Control* which controls the angular velocity of the drone ϕ, θ, ψ.
- *Rate Control* which controls the angular acceleration $\dot{\phi}, \dot{\theta}, \dot{\psi}$.

Starting from the inner-most loop, which is the *Rate Control*, this controller takes as input setpoint the targeted angular acceleration values provided by the *Attitude Control.* The inner-most loop transfer functions from the linearized equations 13 also called the open-loop transfer functions are:

$$\begin{cases}P_{\dot{\phi}}(s) = \dfrac{l}{I_x s}\\ P_{\dot{\theta}}(s) = \dfrac{l}{I_y s}\\ P_{\dot{\psi}}(s) = \dfrac{l}{I_z s}\end{cases} \tag{14}$$

These three angular rates have similar dynamics, so we will do the representation for one of them. So, for one of the dynamics of Equation 14, the closed-loop equation becomes:

$$P_{\dot{\theta},\mathrm{cl}}(s) = \frac{K_d l s^2 + K_p l s + K_i l}{I_y s^2 + K_d l s^2 + K_p l s + K_i l}, \tag{15}$$

Now discretizing this equation as seen in Section 4.1 we obtain the following discrete closed-loop transfer function:

$$P_{\dot{\theta},\mathrm{cl}}(z) = \frac{\frac{K_d l}{T_1^2} z^2 + \left(\frac{K_p l}{T_1} - \frac{2K_d l}{T_1^2}\right) z + \left(\frac{K_d l}{T_1^2} - \frac{K_p l}{T_1} + K_i l\right)}{\frac{(I_y + K_d l)}{T_1^2} z^2 + \left(\frac{K_p l}{T_1} - \frac{2(I_y + K_d l)}{T_1^2}\right) z + \left(\frac{(I_y + K_d l)}{T_1^2} - \frac{K_p l}{T_1} + K_i l\right)} \tag{16}$$

As a reminder, we have seen that the outer loop depends on the parameters of the inner loop, so that the closed-loop transfer function of the *Attitude Control* would depend on its period T_2 and the period of the *Rate Control* T_1 and so on. Then the closed-loop transfer functions of *Attitude Control* and *Position Control* could be represented as follows:

$$\begin{cases} P_{\theta,\mathrm{cl}}(z) = f(P_{\dot{\theta},\mathrm{cl}}(z, T_1), T_2) \\ P_{x,\mathrm{cl}}(z) = f(P_{\dot{\theta},\mathrm{cl}}(z, T_1), P_{\theta,\mathrm{cl}}(z, T_2), T_3) \end{cases} \tag{17}$$

7 Evaluation

For our evaluation we consider the PX4 dynamics as explained Section 6. We start by assuming the system parameters to be as depicted in Tables 2 and 3.

Table 2 System Parameters.

Parameter	**Symbol**	**Value**	**Unit**
Moment of inertia around y-axis	I_y	0.03	$Kg.m^2$
Distance from propeller to drone center	l	0.25	m
Mass of the drone	m	1.5	Kg

Table 3 PID controller parameters.

Controller	K_p	K_i	K_d
$\dot{\theta}$ rate controller	0.15	0.2	0.003
θ angle controller	1	0	0
Velocity controller	0.2	0.02	0.02
Position controller	0.1	0	0

7.1 Stability Analysis

We implement the dynamics represented in equations 16 and 17, and the procedure of extracting the stability regions for the control tasks periods depicted in Figure 3 by using Maple [2]. We present the results of this analysis in Figure 6, where the blue regions are the period combinations that make the physical system stable. We see that the periods T_1 and T_2 of the *Rate Control* and *Attitude Control* reaches a maximum of around $80ms$, whereas T_3 of the *Position Control* could tolerate a period of $120ms$ for some period combinations. This could be due to the fact that rotational dynamics should be acted upon more frequently than the translational dynamics.

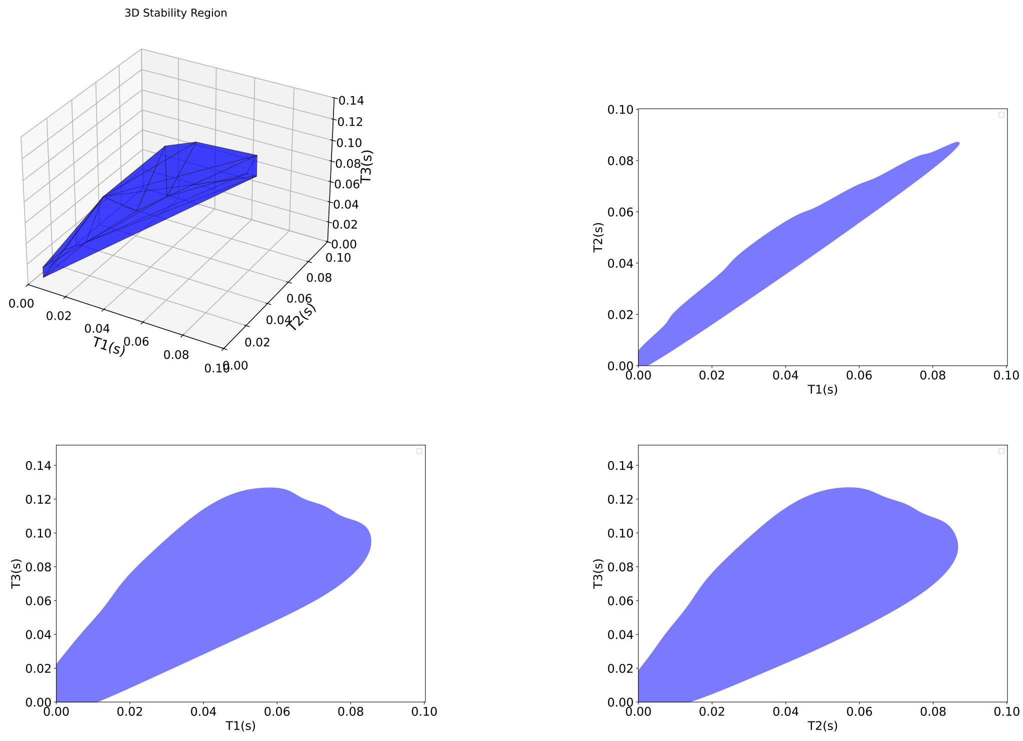


Figure 6 3D Region of periods where the physical system is stable with the projection on each axis.

7.2 Performance Evaluation

Evaluating the performance of the control system represented by Equation 1 should be done during the execution of the system. For this we use Simulink [9] to monitor the step response of the system and measure its performance. The performance is done over all the stable combinations of T_1, T_2 and T_3. For the ease of representation we fix the value of T_1 and plot the variation of the cost J with respect to the periods T_2 and T_3, the results are depicted in Figure 7. We can notice from the plots that the cost exhibits a general increasing trend as periods increase, the presence of some oscillations could be due to the discretization effect

[2] Maple [14] is a powerful computer algebra system used for symbolic computation and mathematical modeling.

where some periods align better with the system dynamics resulting in lower cost. For this reason we consider the optimization since it is not always guaranteed that the lowest stable periods will result in the best performance.

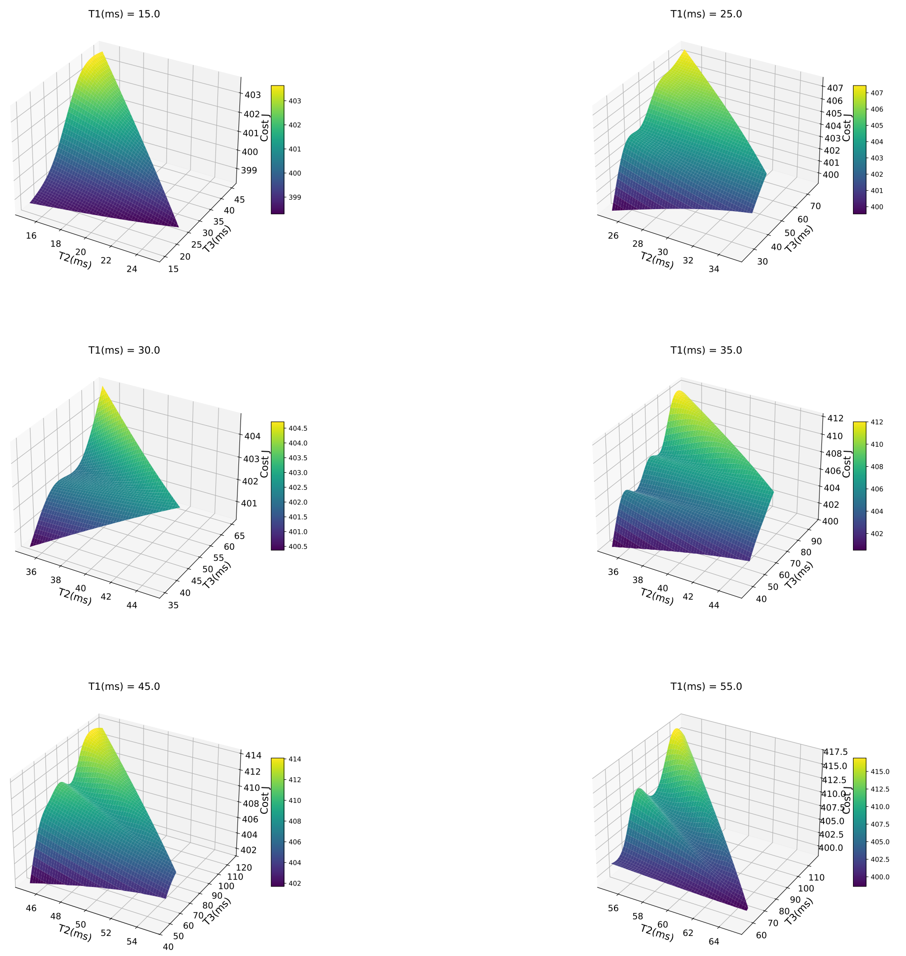


Figure 7 Performance variation as a function of the different combinations of stable periods.

7.3 Co-design Evaluation

After obtaining the stable periods and their corresponding performances it is time to evaluate the co-design problem presented in Section 2.1. From Table 1 we can see that the total utilization of the PX4 tasks excluding the three control tasks is $U_{nc} = 0.59$ (non-control utilization), the output of the co-design problem is depicted in Table 4 Case 1 which shows the values of the three periods that keeps the total utilization U_{total} less than 0.69 while having a low control cost. In an attempt to see what is the effect of adding more load to the system, we increase the value of U_{nc} to 0.68 (Case 2) and we re-evaluate the co-design problem. Increasing the utilization is equivalent to assuming a multi-objective optimization where we add U in the minimization and give it higher weight than the control cost J, this triggers period relaxation while allowing some performance degradation. We see that the periods are more relaxed compared to Case 1 to account for the increased utilization, while we notice that the increase in the cost is not critical this is due to the good choice of controller parameters[3].

Table 4 The optimization output for two cases with different initial utilization due to non-control tasks (U_{nc}).

	U_{nc}	U_{total}	T_1	T_2	T_3	J
Case 1	0.59	0.613	20	25	25	396.3
Case 2	0.68	0.6874	70	75	75	397.93

7.4 Single Control Task

If we implement the 3 controllers into a single program (task), this is equivalent to say that we run the 3 controllers with the same period. To validate the behavior of such case, we check the step response of the system and performance when the period is set to be:

(a) equal to the shortest possible period for the 3 controllers to be stable,

(b) equal to the longest possible period for the 3 controllers to be stable.

We show in Table 5 the performance and total utilization of the two cases. For Case a we see that it provides a slightly better performance than the optimal assignment seen in Case 1 of Table 4. But this case wasn't considered because it violates the condition on the allowable total utilization (U_{total}). Case b shows good total utilization but a slightly higher cost J. It could be noticed from the step response of the two cases in Figure 8, that the two responses coincide which ensures that the performance is not changing by much thanks to a good choice of system parameters.

Table 5 The performance evaluation of the single control task in the case of the shortest possible period (Case a) and in the case of the longest possible period (Case b).

	U_{nc}	U_{total}	T_1	T_2	T_3	J
Case a	0.59	0.6988	5	5	5	394.5
Case b	0.59	0.5968	80	80	80	400.13

8 Important Observations

8.1 Impact of Cascade

A question that could be asked, why don't we assign periods to each controller separately as if they were independent from one another? To answer this question we do stability analysis and check if cascaded loop sampling periods impact each other. We start to solve for the period of the inner-most loop, i.e. the *Rate Control* in our case, by solving the Jury's conditions as discussed in Section 4. We obtain that the inner-most loop by itself is stable for the values of period T_1 in the range $1ms$ to $750ms$. When we add the second loop, i.e. *Attitude Control*, we see that these values of T_1 are stable for small values of T_2, while for

[3] More discussion on this in Section 8.2.

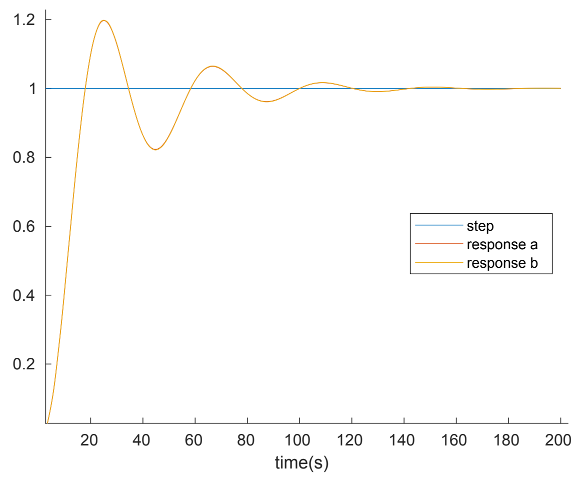
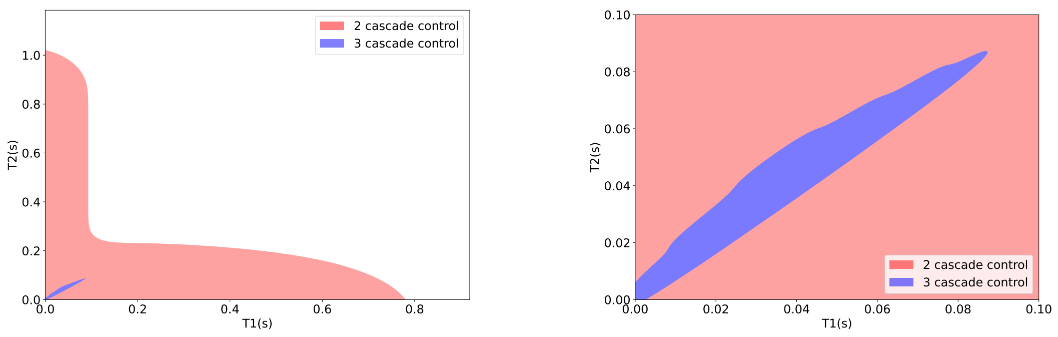


Figure 8 Step response of drone system for Case a where periods are set to be the shortest possible and Case b where periods are set to be the longest possible while maintaining stability.



(a) Stable regions. **(b)** Zoomed view of the 3 cascaded loops.

Figure 9 Region of stable periods for *Rate Control* T_1 and *Attitude Control* T_2 before and after adding the third control loop.

small values of T_1 the period of the second loop T_2 could reach a value of more than $1s$, see Figure 9a (red region). After adding the last loop, i.e. *Position Control*, we notice that the region of stability becomes much more constrained as in Figure 9a (blue region). If we zoom on this region, then we obtain Figure 9b. So, yes there is an important impact of the cascade structure on the sampling period, we've seen a drop from $750ms$ to around $80ms$ for T_1 and a drop from around $1s$ to $80ms$ for T_2.

8.2 Impact of Controller Parameters

In this section, we discuss the importance of the controller parameters choice during the first step of the analysis. For this we attempt to change the integral gain of the velocity controller $K_i = 0.4$ and the proportional gain of the position controller $K_p = 0.01$, and repeat the whole process of evaluation we have seen in Section 7. If we compare Case $1'$ of Table 6 with Case 1 of Table 4, we see that although here the set of optimal periods is lower than that of

Case 1 the cost of the control is much higher. This could be seen on the plot of the step response of the two cases Figure 10, where the response $1'$ converges slowly to the required setpoint whereas response 1 is much faster.
Another observation is that if the controller parameters are badly chosen as in this case, relaxing the periods will lead to much higher cost where we see for Case $2'$ an increase of 45% in the cost with respect to Case $1'$. Whereas when we had good parameters in the previous evaluation relaxing the control tasks periods did not show this drastic impact (Table 4).

Table 6 The optimization output in the case where the controller parameters are modified.

	U_{nc}	U_{total}	T_1	T_2	T_3	J
Case $1'$	0.59	0.626	15	15	15	10276
Case $2'$	0.68	0.689	50	65	65	14878.16

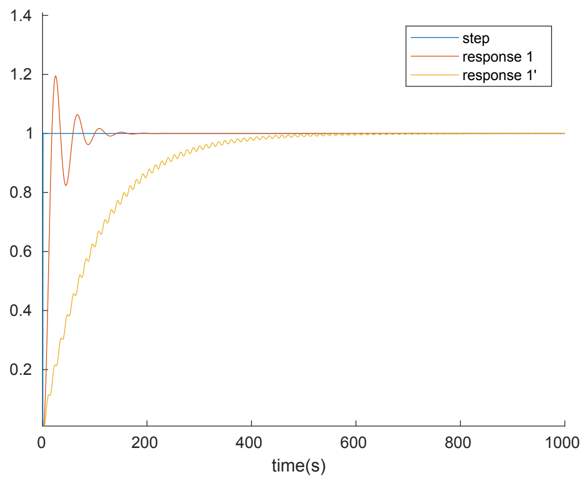


Figure 10 Step response of drone system for Cases 1 and $1'$, where the difference between the two cases comes from modifying some controller parameters.

Similarly if we replicate what is done in Section 7.4 for the current parameters we obtain the results in Table 7. Since as we said, the parameters are not chosen correctly in this section the choice of periods impact a lot the performance of the system. This could be seen from Figure 11, more precisely the "Zoomed In" part where we see that the response of Case a' will eventually converge faster than that of Case b'.

Table 7 The performance evaluation of the single control task in the case where the controller parameters are modified.

	U_{nc}	U_{total}	T_1	T_2	T_3	J
Case a'	0.59	0.6988	5	5	5	10029.7
Case b'	0.59	0.6008	50	50	50	10579.6

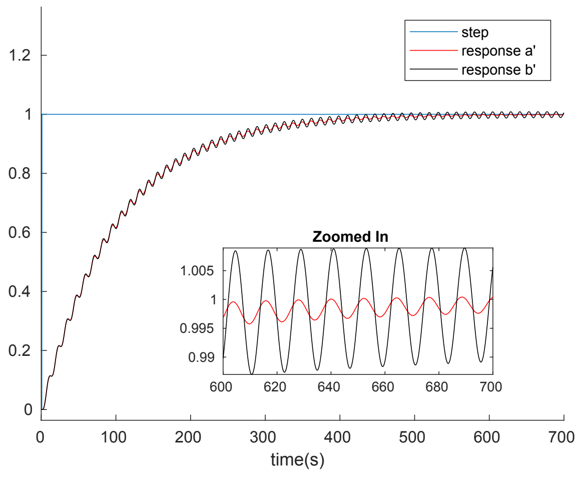


Figure 11 Step response of drone system with modified controller parameters for Case a' where periods are set to be the shortest possible and Case b' where periods are set to be the longest possible while maintaining stability.

9 Conclusion

In this paper we present a methodology to analyse the stability of cascade control system with respect to the sampling periods using Jury's stability. Then we use these periods to solve a co-design problem resulting in a set of periods for the real-time cascade control tasks, that guarantees a good control performance and a schedulable task set under fixed-priority preemptive scheduling. Our evaluation shows the importance of studying the real-time cascade control tasks together as they have an impact on each other when it comes to period choices and stability. An extension of this work could be the integration of the non-control tasks in the co-design problem, because currently we only consider their contribution to the over all utilization of the system where in fact their periods could also be modified and this might have an impact on the system performance. Another possible extension could be the relaxation of the utilization bound used, which allows a better trade-off between the control performance and the schedulability.

References

1 Mohammad Al Khatib, Antoine Girard, and Thao Dang. Scheduling of embedded controllers under timing contracts. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 131–140, 2017. `doi:10.1145/3049797.3049816`.

2 Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. Designing high-quality embedded control systems with guaranteed stability. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium, RTSS 2012*, pages 283–292. IEEE Computer Society, 2012. `doi:10.1109/RTSS.2012.79`.

3 Kagan Koray Ayten, Ahmet Dumlu, Sadrettin Golcugezli, Emre Tusik, and Gurkan Kalınay. Real-time implementation of cascade nonlinear ff-pi retarted controller design for industrial process systems. *Preprints*, August 2024. `doi:10.20944/preprints202408.0288.v1`.

4 Radhakisan Baheti and Helen Gill. *Cyber-physical systems*. IEEE, 2011.

5 Hoon Sung Chwa, Kang G. Shin, and Jinkyu Lee. Closing the gap between stability and schedulability: A new task model for cyber-physical systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, pages 327–337, 2018. doi:10.1109/RTAS.2018.00040.

6 John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory.* Courier Corporation, 2013.

7 RG Franks and CW Worley. Quantitative analysis of cascade control. *Industrial & Engineering Chemistry*, 48(6):1074–1079, 1956.

8 Dunstan Graham and R. C. Lathrop. The synthesis of "optimum" transient response: Criteria and standard forms. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, 72(5):273–288, 1953. doi:10.1109/TAI.1953.6371346.

9 The MathWorks Inc. *MATLAB version: 9.11.0 (R2021b).* Natick, Massachusetts, United States, 2021. URL: https://www.mathworks.com.

10 Jinkyu Lee and Kang G. Shin. Development and use of a new task model for cyber-physical systems: A real-time scheduling perspective. *Journal of Systems and Software*, 126:45–56, 2017. doi:10.1016/j.jss.2017.01.004.

11 Alberto Leva and Federico Terraneo. Teaching task scheduling as multivariable cascade control. *IFAC-PapersOnLine*, 49(6):280–285, 2016. 11th IFAC Symposium on Advances in Control Education ACE 2016. doi:10.1016/j.ifacol.2016.07.190.

12 Jun Li and Yuntang Li. Dynamic analysis and pid control for a quadrotor. In *2011 IEEE International Conference on Mechatronics and Automation*, pages 573–578, 2011. doi:10.1109/ICMA.2011.5985724.

13 Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973. doi:10.1145/321738.321743.

14 Maplesoft, a division of Waterloo Maple Inc. *Maple User Manual*, 1996-2023. Version range: 1996–2023.

15 N. A. Patil and G.V. Lakhekar. Design of pid controller for cascade control process using genetic algorithm. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1089–1095, 2017. doi:10.1109/ICCONS.2017.8250634.

16 Paolo Pazzaglia, Arne Hamann, Dirk Ziegenbein, and Martina Maggio. Adaptive design of real-time control systems subject to sporadic overruns. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1887–1892. IEEE, 2021. doi:10.23919/DATE51398.2021.9473913.

17 Karl J. Åström and Björn Wittenmark. *Computer-controlled systems (3rd ed.).* Prentice-Hall, Inc., USA, 1997.

18 Minsoo Ryu, Seongsoo Hong, and Manas Saksena. Streamlinig real-time controller design: From performance specifications to end-to-end timing constraints. In *Proceedings Third IEEE Real-Time Technology and Applications Symposium*, pages 91–99, July 1997. doi:10.1109/RTTAS.1997.601347.

19 W. C. Schultz and V. C. Rideout. Control system performance measures: Past, present, and future. *IRE Transactions on Automatic Control*, AC-6(1):22–35, 1961. doi:10.1109/TAC.1961.6429306.

20 O. Sename, D. Simon, and D. Robert. Feedback scheduling for real-time control of systems with communication delays. In *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696)*, volume 2, pages 454–461 vol.2, 2003. doi:10.1109/ETFA.2003.1248734.

21 D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin. On task schedulability in real-time control systems. In *17th IEEE Real-Time Systems Symposium*, pages 13–21, 1996. doi:10.1109/REAL.1996.563693.

22 Marion Sudvarg, Ao Li, Daisy Wang, Sanjoy K. Baruah, Jeremy Buhler, Chris Gill, Ning Zhang, and Pontus Ekberg. Elastic scheduling for harmonic task systems. In *30th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, pages 334–347. IEEE, 2024. doi:10.1109/RTAS61025.2024.00034.

23 Nils Vreman, Anton Cervin, and Martina Maggio. Stability and Performance Analysis of Control Systems Subject to Bursts of Deadline Misses. In *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, volume 196, pages 15:1–15:23, 2021. doi:10.4230/LIPICS.ECRTS.2021.15.

24 Yang Xu, Anton Cervin, and Karl-Erik Årzén. Jitter-robust lqg control and real-time scheduling co-design. In *2018 annual American control conference (ACC)*, pages 3189–3196. IEEE, 2018. doi:10.23919/ACC.2018.8430953.

25 Yang Xu, Karl-Erik Årzén, Anton Cervin, Enrico Bini, and Bogdan Tanasa. Exploiting job response-time information in the co-design of real-time control systems. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 247–256, 2015. doi:10.1109/RTCSA.2015.23.

26 Runjiang Zhao. Inner/outer-loop control of a drone under disturbances and uncertainties: a sliding mode control approach. Master's thesis, Northeastern University, 2020. doi:10.17760/D20382857.