Commande robuste avec relâchement des contraintes temps-réel

Patrick Andrianiaina

UNIVERSITÉ DE GRENOBLE

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité: Automatique Productique

Arrêté ministériel : 7 août 2006

Présentée par

Patrick Jocelyn ANDRIANIAINA

Thèse dirigée par **Daniel SIMON** et

codirigée par **Alexandre SEURET**

préparée au sein de

l'INRIA GRENOBLE RHÔNE-ALPES

dans l'**École Doctorale**:**EEATS**

Commande robuste avec relâchement des contraintes temps-réel

Robust control under slackened real-time constraints

Thèse soutenue publiquement le **26 septembre 2012**,

devant le jury composé de:

Michel DAMBRINE

Professeur, Université de Valenciennes, Rapporteur

Arben CELA

Professeur associé ESIEE Noisy-le-Grand, Rapporteur

Sophie TARBOURIECH

Directeur de Recherche CNRS, LAAS Toulouse, Présidente

David POWELL

Directeur de Recherche CNRS, LAAS Toulouse, Membre

Jean-Claude LAPERCHE

Ingénieur AIRBUS, Toulouse, Membre

Daniel SIMON

Chargé de Recherche INRIA, Grenoble, Membre

Alexandre SEURET

Chargé de Recherche CNRS, LAAS Toulouse, Invité

Guilhem PUYOU

Ingénieur AIRBUS, Toulouse, Invité

Contents

General Introduction

Foreword

The works presented in this thesis are the results of a CIFRE grant (Convention Industrielle de Formation par la REcherche) between Airbus Operations SAS (formerly Airbus France) and INRIA (Institut National de Recherche en Informatique et Automatique) Grenoble Rhône-Alpes. It has been financially supported by the ANRT (Association Nationale de la Recherche et de la Technologie) and Airbus Operations SAS. The thesis works have resulted in the following publications and patents :

- Patrick Jocelyn Andrianiaina, Alexandre Seuret and Daniel Simon. *Robust control under weakened real-time constraints*. In 50th IEEE Conference on Decision and Control and European Control Conference CDC/ECC, Orlando, United States, December 2011 [Andrianiaina 2011a]
- Patrick Jocelyn Andrianiaina, Daniel Simon and Alexandre Seuret. *Commande robuste avec relâchement des contraintes temps-réel*. In Conférence Internationale Francophone d'Automatique CIFA2012, Grenoble, France, Jul 2012 [Andrianiaina 2012]
- Patrick Jocelyn Andrianiaina, Daniel Simon and Alexandre Seuret. *Robust system control method with short execution deadlines*, Sep 2011. (Patent n° EP11306103.0 claimed by Airbus Operations SAS) [Andrianiaina 2011b]
- Patrick Jocelyn Andrianiaina, Daniel Simon, Alexandre Seuret, Jean-Michel Crayssac and Jean-Claude Laperche. *Weakening Real-time Constraints for Embedded Control Systems*. Rapport de recherche RR-7831, INRIA, December 2011 [Andrianiaina 2011c]

Context : Real-time control and embedded systems

Over the last decades, the evolution of computer technologies has made that our society more and more reliant on embedded systems [Moravec 1998]. In transport systems this trend has resulted in the computerization of systems, leading to *X-by-wire systems*, where fly-by-wire and brake-by-wire are well-known instances as far as avionics (and automotive) systems are concerned. In systems that are qualified to be *X-by-wire* the conventional mechanical control components are replaced by computing devices and software functions. In the transport domains these embedded systems are considered to be critical in terms of behavior, safety considerations and fault-tolerance.

A successful design and implementation of a computer-controlled system requires a good understanding and integration of both control theory and real-time computing. From a practical point of view, many control systems are embedded systems where the control

functions are implemented using a set of microprocessors, some kind of real-time operating systems and programming languages. The traditional approach for the design of a computer-controlled system is based on the separation of concerns between control and computing. In particular it is assumed that the computing resources platform are able to provide deterministic, predictable and equidistant sampling, which fit with the well established theory of control of sample-data systems.

On one hand, real-time control is an inter-disciplinary area that include mostly aspects of control engineering and computer science [Årzén 1999, Törngren 1998]. On the other hand, the need for safety-critical systems requires to meet very stringent guarantees on the capacity of the computing resources to process workloads within predefined bounds.

Currently, the demonstration of the capacity of the computing resource to handle workloads within a given deadline is done through the WCET (Worst Case Execution Times) determination. The WCET is defined as the longest time taken for the computer to process a given task, considered at the worst possible conditions. While the first objective is to ensure safety, the correctness and soundness of the WCET is of great importance in classical design methodology because it allows for the sizing of the overall system. However, worst case approaches induce a systematic conservatism due to idle states and bandwidth wasting, which means an over-dimensioning of resources and increased energy consumption, weight and overall costs w.r.t. the average conditions case. As weight is of particular interest in avionics, and energy is a limited resource is most embedded systems, it is worth to search for new methods and find cost-effective solutions w.r.t. the specified safety requirements.

While technologies develop, avionic systems manufacturers such as Airbus need to optimize the cost and performance ratio. It has been shown in recent works [Louise 2002, Thesing 2004], that the complexity of new computing node technologies (processors and multicore computing units) increases the difficulty of WCET evaluations, and that the spreading of control tasks execution times over large intervals even enlarges the conservatism of worst case based scheduling methods. Therefore the traditional separation of concerns between control and computing deserves to be revisited, and a better understanding of the interaction between these two domains is expected to provide cost-effective tradeoffs between control robustness, computing resources efficiency and safety requirements.

This thesis focuses on the joint design and implementation of real-time control systems, with application in avionics. More especially, this thesis examines the timing aspects of computer-controlled systems by revisiting the usual design and implementation based on the WCET. Its result allows for a less conservative choice of the allocated times for the control tasks by taking into account the robustness of feedback control, using some flexibility in the control tasks scheduling schemes.

A simplified view of an avionics control systems is shown in Figure 1. Here, the focus is to study the control performance and robustness of the control application in relation with the temporal behavior of the computer implementation. Conversely with the classical approach based on the separation of concern (see section 1.7), this thesis approaches the problematic from a more holistic point of view. In fact, the goal of the real-time control system consists in achieving an **end-to-end** control objective, i.e. ensuring the aircraft stability and control performance for all conditions (including occasional worst cases conditions).

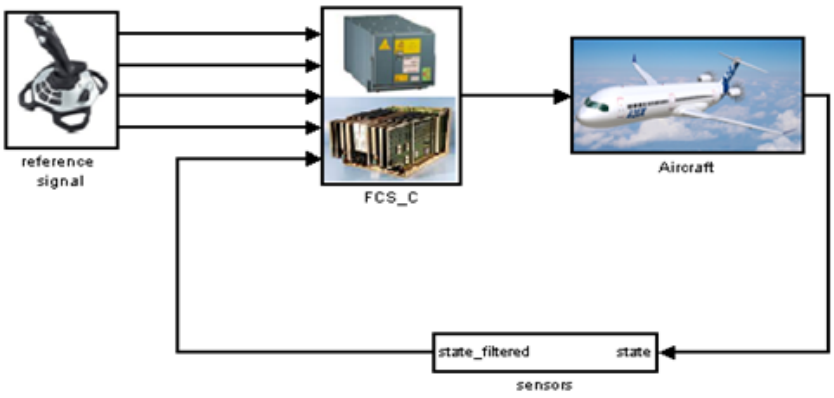

Figure 1: Flight control system schematic

In this scheme, the FCS_C box represents the embedded controller composed of a processing unit and a set of control tasks.

The final objective is to improve the resource utilization while keeping the control performance inside the safety requirements.

Problem description and objectives

Real-time control system design presents two separate aspects. From the control point of view, the main objective is to design control laws that are stable and robust. From the implementation point of view, the goal is to provide physical computers and programs that respond to the safety critical requirements.

The usual way is to specify the system requirements and the corresponding models in the control domain and forward them to the implementation community. This approach has been used for long using assumptions such as pre-determined constant delays and sampling periods.

The step of implementation of control systems, is based, on hardware and software technologies that generate other constraints that combine with those of the control system designers. These constraints include functional aspects and legal aspects based on certification processes and safety guidelines. Functional constraints include different constraints such as the respect of the initial assumptions (constant delays, equidistant sampling) while the legal aspects consist of different constraints based on the methods of implementations, the ability or not to use some components. The main goal is to be able to produce the required system with respect to all of these constraints. Well-known guidelines include the RTCA-Do178B [DO-178B 1993] for safety-critical software development, the DO254 [DO-254 2005] for hardware system development. Among other verifications, timing analysis, segregation of development in order to ensure redundancy are also required by these guidelines.

When all these design and implementation constraints are all blended together, the design and realization of a system becomes a huge challenge. Consider the RTCA DO-

178B standard, which, among others imposes an additional requirement which is to ensure a predictable timing behavior on the software component of the system. Such requirement has led implementation and manufacturers to turn toward worst case analysis.

Currently, the aerospace industries such as Airbus (and likely BOEING) and many automotive industries require their equipment suppliers to provide analysable systems as far as worst case execution time analysis is concerned.

However, approaches based on worst case execution time are likely to meet their limits due to technological evolutions of computing hardware and programming languages. [Mueller 2006] presents these limits and the associated efforts to overcome them. In particular the RTCA DO-178B guideline is not absolutely rigid, in fact the equipment manufacturers have degrees of freedom to follow their specific method to achieve a certification level, as long as they prove to be compliant with the requirements.

During the last few years, the embedded hardware has evolved from dedicated processors and field-bus (enhancing determinism) towards shared components. A considerable amount of effort was undertaken in the area of avionics systems (such as integrated modular avionic (IMA) [Parr 1999] or digital communication based on AFDX switch [Li 2010]) to minimize aircraft wiring and to provide weight and cost reduction, while keeping high operational reliability and maintainability.

The real-time control software side is to consider a flexible timing behavior of control tasks. Feedback control is expected to provide a control performance which is robust against timing deviations such as sampling jitter and occasional samples loss.

The usual hard real-time scheduling approach (necessarily based on worst-case considerations), considers that temporal faults (i.e. deadline misses) are totally forbidden and that the occurrence of such a fault must lead to an emergency behavior (e.g. switching to a spare system). The control robustness based approach considers that a specified set of timing faults can be safely handled (following the "weakly-hard" real time approach as in [Bernat 2002, Bernat 2001]), provided that the real-time scheduler is equipped with the corresponding exception handlers. The consequence is that the computing system can be proportioned according to computing loads below the WCET of the control tasks, inducing gains in processing power and associated costs.

By challenging current industrial applications and traditional methods, we show in this thesis that control feedback loops are robust enough to bear timing deviations. The timing deviations are changes from the ideal timing patterns that were scheduled for control tasks.

We also show that it is possible to enhance resources utilization and to reduce the conservatism of worst-case based implementations. A method is proposed to improve utilization and availability of computing resources, while ensuring system stability and quality of control. Indeed, with traditional design, control systems were as rigid as software hard-deadlines, and a deadline violation of a software inevitably led to the system unavailability.

Specifically, this thesis demonstrates that it possible to slacken the hard real-time approach by tolerating a specified number of deadline overrun from the control tasks while keeping the performance and stability of the system. This leads to a better performance and schedulability, while permitting to prospect on the use more flexible and fault-tolerant systems.

This idea allows to demonstrate that enhancements of the overall utilization of comput-

ing resources can be gained, by accepting a temporary degradation of performance. This process of accepting a temporary degradation of performance is called "Graceful performance degradation". On one hand, the computing loads are alleviated to allow saving for computing power, while the control performance is degraded by deadline misses. On the other hand, the control performance degradation is controlled by an offline analysis in order to keep the system stable and safe.

The work is based on a control and scheduling co-design approach, in which the features and constraints of one domain (e.g. task triggering and deadlines in the real-time domain) are translated into the other domain (e.g. sampling rate and delays in the control domain). A Quality of Control (QoC) metrics is based on the tolerable numbers of consecutive timing faults that can be safely beared by the control system : it is computed analytically, to be further used to implement the timing faults handlers. Hence flexible timing constraints with fault compensation principles are also among the contributions of the thesis.

Thesis outline

The organization of the document is as follows:

- Chapter 1 gives the general context and the motivation of the work. A concise review of the different states of the art of the domains that are covered by this thesis.

  These states of the art range from that of safety-critical systems to computer-controlled systems. In order to introduce the different safety constraints that are imposed by certification guidelines and specification in the development of highly critical systems, a state of art of safety critical system is given in 1.4. A state of art of computer-controlled systems, is also reviewed in section 1.5.

  One of the objectives of this thesis is to show the inefficiency of the current design and implementation methodologies (which are based on worst case execution time considerations), mainly in terms of computing resources utilization.

  The idea is to pull out the problematics linked with the robustness of the implemented system and to show that it is possible to have a better computing- aware methodology.

  The problem statement is presented in section 1.6. An overview of control and scheduling co-design is finally presented in Section 1.7.

- Chapter 2 starts with the presentation of a robust weakened real-time system, in which the motivation for the need of a more flexible framework is proposed.

  In section 2.2 the notion of weakened real-time framework is presented as well as the different possible scheduling scenarii that can be associated with it. As the introduction of this new idea, on weakening the real-time constraints, has risen new problems, the theoretical basis to solve these new problems are discussed in section 2.4.2. In section 2.4.1 and it is shown how to handle these different problems.

  Then, section 2.4.2 shows how to apply the presented tools with systems having sampled-data inputs because such models are used all along this thesis to model

computer-controlled systems. Finally, generalization of the approach are considered and stability conditions in the form of Linear Matrix Inequalities (LMI) are given in section 2.4.3.

- Chapter 3 is divided into two main parts. The first part describe the simulation of model of the case study, the aircraft model, in section 3.2. This section describes the nonlinear aircraft equations, the linearized ones and the decoupled equations such as the longitudinal motion models versus the lateral motion models, and also more fine-grain decoupling such as the short period and the so-called phugoïd. This section is not intended to be a reference on aircraft modeling, but instead a summary of the different sub-models of a 6 degree-of- freedom aircraft system as well as the different control systems that compose it. This description of aircraft model is not exhaustive and only parts and sub-models of interests were considered in this chapter.

  The second part of this chapter discusses the application of the new ideas and solutions in Chapter 2 with an aircraft control system. Most naturally, the pitch-rate control system has been chosen to conduct experimentation and simulation for various reasons and motivations that are explained in section 3.3.2. Finally, comments and guidelines on how to read the results are given in section 3.3.3.

A conclusion is given in chapter 4, where a summary of the thesis and new perspectives of research are presented.

CHAPTER 1

Framework and Problems statements

Contents

1.1 Introduction to Chapter 1

This chapter motivates the thesis and the general context in which it can be applied. This mainly deals with avionics and computer-controlled systems, taken in the very stringent context of highly safety-critical avionics systems. These points will be covered in section 1.3 after section 1.2 which presents a preliminary definitions of some technical terms used all along the thesis.

A concise presentation of the framework of safety-critical avionics systems follows in Section 1.4. This presentation presents generalities on safety-critical systems, which will later permit us to introduce some elements of constraints that make it difficult to develop safety-critical real-time systems such as flight control systems or braking control systems.

Then we develop a section dedicated to real-time control, in which, classical continuous-time control theory will be rapidly revisited with its classical assumptions. This section is followed by a section on computer-controlled systems and its classical problematics, which include timing parameters settings problems.

The problem statement of the thesis is presented in section 1.6, where the problematics of resource utilization and separation of concern are developed. At the end of this section, one would understand that the problematic is directed more toward robustness and performance with respect to timing constraints.

Finally, section 1.7 sets the relation between control and scheduling communities and the mutual impact on these domains. This section is directed toward the demonstration that control system are robust enough to tolerate timing deviations from the ideal hard- deadline pattern.

1.2 Useful definitions

Before going into the contents of this chapter, it is necessary to give some definitions. These definitions will be used throughout this thesis.

- System: A system can be defined as the combination of different coherent structures, whose elements contribute to achieve a goal. In the context of this thesis, a control system is a combination of different elements such as controllers, sensors, ... to achieve a goal which to control and stabilize the plant. Another example: avionics systems can be seen as the combination of different sub-systems which achieve the goal of flying the aircraft safely.

- Error is the execution of a fault (or more precisely a passive fault) which may lead to failure.

- Faults are potential flaws in a system, which can be later activated to produce an error.

- Failure is the fact that a system has an erronous behavior (for instance an incorrect state). that does not meet its intended and desirable objective.

- Critical systems are systems whose failure may lead in serious injury to people, large loss or severe damage to equipments or very severe environmental harms.

- Dependability of a system is its ability to deliver specified services to the end users so that they can justifiably rely on and trust the services provided by the considered system.

- Reliability is associated to a period of time. The reliability of a system considered in for a period $(0, t)$ is the probability that the system is continuously operational (i.e., does not fail) in time interval $(0, t)$ given that it is operational at time 0.

- Robustness is the persistence of a system's characteristic behavior under perturbations or conditions of uncertainty. It means that a robust system is a system that can tolerate deviation from the ideal case for which it has been designed, while providing a reliable service without failure.

- Safety of a system for a period $(0, t)$ is the probability that the system will not incur any catastrophic failures in time interval $(0, t)$.

- Fault-tolerance describes system designed so that, in the event that a component fails, a backup component or procedure can immediately take its place with no loss of service. Fault tolerance can be provided with software, or embedded in hardware, or provided by some combination.

1.3 Context and motivation

1.3.1 System considerations and avionics functions

Aircraft systems and subsystems development is a complex task. There are hundreds and even thousands of subsystems based on various expertises. This complexity is shown by figure 1.1. From this complexity, this thesis will focus on elements such as the Avionics Systems and Equipments.

Avionics systems' main essence is to provide aircraft functions [Cary 2001]. These functions satisfy very specific needs and requirements on the aircraft. Among these functions, we can list for instance, flight control systems, navigation and tracking systems, flight management systems or synthetic vision and display tools. Those functions are coded as ATA (Air Transport Association) chapters by the FAA (Federal Aviation Administration) [Cary 2001]. The unique aspect of the ATA chapter numbers is its relevance for all aircraft. Thus a chapter reference number for an Airbus 380 will be the same for a Boeing 747 or a BAe 125. Examples of this include Oxygen (ATA chapter 35), Electrical Power (ATA Chapter 24) and Doors (ATA Chapter 52). Each function is associated to a level of criticity and dependability. For instance, the subsystem responsible for Light functions (ATA 33) would not be as critical as the one responsible for flight control functions (ATA 27) or the one for the Autopilot (ATA 22) function, as the loss of the light functions would not be as catastrophic as the loss of the flight control functions, in some senses.

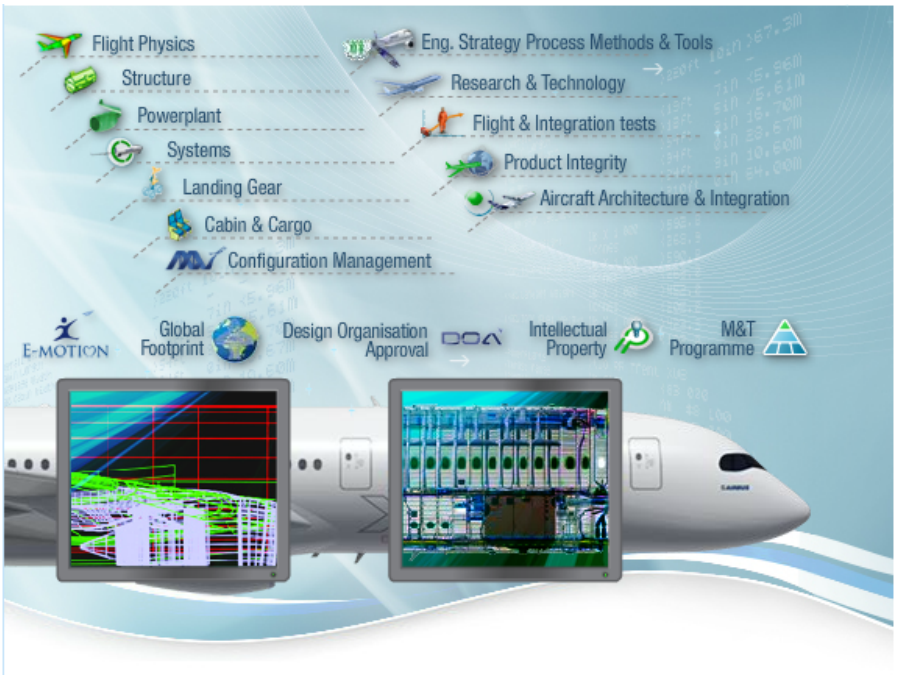

Figure 1.1: Aircraft Systems and subsystems

For such critical applications, the probability of losing the functions or the occurrence of a failure in such functions and leading to its loss, must be less than 10^{-9} per hour of flight. These systems have to meet stringent requirements of dependability but also requirements on performance and usability and so, are submitted to severe regulations. At the higher level of an aircraft for instance, these guarantees are achieved through strategies like redundancy (see [Goupil 2010] and [Bertrand 2009]). However, the above-cited works do not consider the system at a single equipment level, where, things happen quite differently. Indeed, equipment (i.e software and hardware) designers also have stringent requirements to meet in order to guarantee determinism and to achieve certifications of their products. They are constrained by the high level avionics requirements, certification regulations but also by implementation limitations. These constraints lead equipment designers and manufacturers to set determinism and predictability as a priority and not take into account fault tolerance and robustness [Bedin França 2011, Souyris 2005] at some levels. This approach is known to be very conservative because apart from being long and difficult in implementation, designers and manufacturers of equipments must consider taking some spare safety margins, especially in timing analysis, to guarantee required timing objectives is reached.

The case study in this thesis is the flight control systems (ATA 27, ATA22) , although the proposed approach works with any system integrating a control loop, such as Landing Gear (ATA 32) braking.

1.3.2 Flight Control Systems and the computer era

For many decades now, computers have made their way into our daily life. It is nowadays classical to find computer as a controller of a given process by using complex control laws implemented as software tasks. This evolution has increased the possible applications of computer systems and when conjugated with the relatively low cost and the relative reliability of computers components, it is not surprising that almost every system that we see nowadays is computer-controlled.

Then the notion of X-by- wire started to show up (in reference to the computerization meaning that the usual mechanical links are replaced by electrical wires).

To give examples, we can mention the *brake-by-wire* for example to represents the replacement of traditional mechanical component links into electronic sensors and actuators, the *drive by wire* or even the *Fly-by- wire* (FBW), which is commonly called the Electrical Flight Control System (EFCS) which replaces replaces the conventional mechanical links of aircrafts with an electronic interfaces, networked links and digital devices.

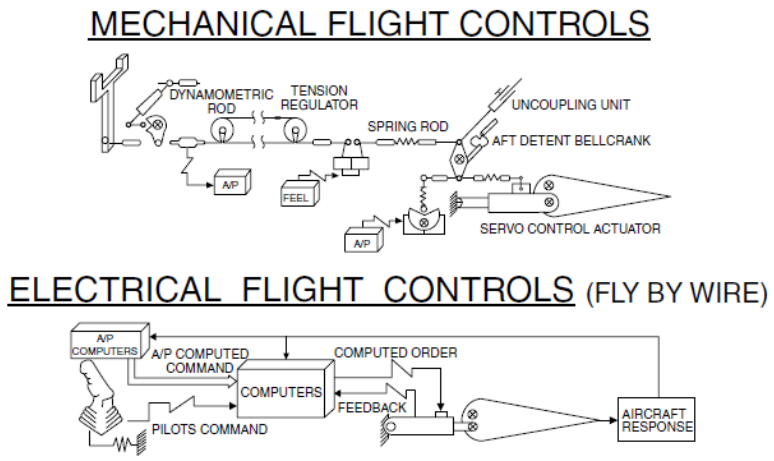

Figure 1.2: Mechanical Flight Control vs Fly-by-wire

Figure 1.2 shows a typical comparison between the classical mechanical flight control systems and a typical computer-based flight control, the fly-by-wire. The Electrical Flight Control System (EFCS) was first developed in the 1960s by Aerospatiale and installed on the Concorde in 1969 as an analogue system. Much later, digital EFCS was introduced first on the A310 around 1982, on the spoilers, slats and flaps control surfaces only, and then was generalized on all control surfaces (the 3-axis aircraft) on the A320 in 1987. This was rapidly followed by installation of EFCS on the A340 aircraft, certified at the end of 1992 [Briére 1993]. The FBW systems started then to constitute an industrial standard not only for commercial aircrafts [Favre 1994] while it has already been implemented on military aircrafts from 1965 on the F-8 Crusader.

The fly-by-wire principles consist in the fact that the pilot commands are sensed electrically and processed by a computer to compute the equivalent actuation of the control surfaces. In earlier times, this system simply provided electrical signals to the control actuators, without any form of enhancement. With the improvements of computer-based

systems (more processing power, more memory, performing chip-sets, ...), the EFCS has also been improved taking benefits of these changes. These changes include the incorporation of more complex control laws in the flight control software and hardware, that provide extensive stability and control augmentation systems and more and more complex control laws [Favre 1994, Philippe 2011]. The positions of the control surfaces are no longer a direct reflection of the pilot's control inputs and conversely, the natural aerodynamic characteristics of the aircraft are not fed back directly to the pilot. Figure 1.3 shows a diagram summary of a simple control law.

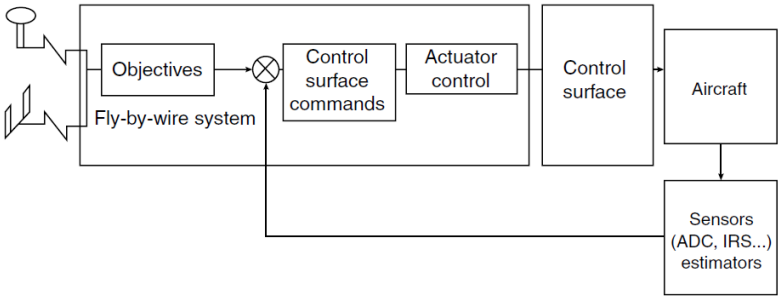

Figure 1.3: Flight Control Laws

1.4 Safety critical systems

1.4.1 Development process life cycle

According to [Falla 1997], a system is classified as critical , if a malfunction of the system leads to:

1. loss of life
2. injury or illness
3. serious environmental damage
4. significant loss of, or damage to, property
5. failure of an important mission
6. major economic loss

Of these classes of systems, the reference gives a classification describing, systems of class 1-5, as " safety-critical" , class 5 as "mission-critical" and systems of class 4- 6 as "business-critical". The boundaries between each description is not sharp, thus experts would argue that a system is safety critical if its malfunction could lead to major economic loss, and conversely, a system which causes loss of life could have a severe economical consequences for involved operators.

The massive introduction of computer-controlled systems by means of *X-By -Wire systems*, have introduced new failure modes, new processes of development and new methods

of validation. A need for standardization has emerged in terms of safety, dependability and development processes of these systems. The way to insure safety and dependability of developed systems, in some way, is to insure that the designers and manufacturers have followed a given and known process when developing the system. This permits and somehow forces the manufacturers to follow some safety recommendations and to do the required verification.

Consequently, industrial systems such as aircraft, automotive and railways industries had to adopt a regulated way of doing their developments processes. As far as avionics systems are concerned, the overall development processes are integrated into a life cycle of a product. It sets clearly and step by step, the places of the different industrial activities such as system design and modeling, design and concept verification, code and hardware implementation and prototyping, verification of the prototypes... until the final product that is intended to be safe.

For avionics products ,this life cycle is shown in figure 1.4 [Philippe 2011] for aircraft development.

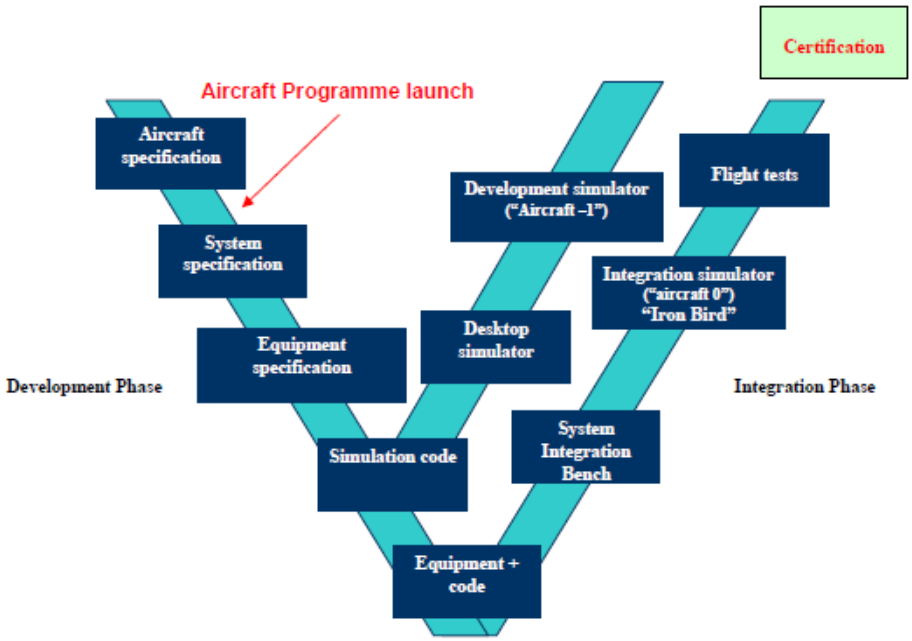

Figure 1.4: V-Life cycle for aircraft development

In many industries coping with embedded systems problematics, the respect of the different activities and the different processes governed by this life cycle is crucial to obtain certification and qualification of the products. Indeed, in highly critical and complex projects, such as, aircraft development, the final stage is to convince external regulating organizations that the system is safe, and that the process of development has been followed. Strictly following this life cycle already achieves and demonstrates a form of fault-tolerance [Philippe 2011] at some levels of the product.

This type of life cycle permits to track the development process of the aircraft from its specification to the validation. The first branch of the V- cycle is the development phase. It starts with the aircraft specification corresponding to the "top-level requirements": the definition of the needs, the choice of concepts, control laws, technologies, etc. The aircraft is decomposed into functional sub-parts, which are specified in the next step and decomposed into "equipments" (e.g. a computer systems, meaning hardware and software), which provide the specified functions.

The other part of the life cycle consists of verification and validation. A severe campaign of validation is undertaken after the integration of the equipments. This validation phases ends with a flight test campaign [Philippe 2011].

1.4.2 Legislation, regulations and certification

Almost all aspects of the design, production, and operation of civil aircraft are subject to extensive regulation by authorities. This section describes the most significant regulatory involvement a developer is likely to encounter in the certification of avionics systems. Certification is a critical element in the safety-conscious culture on which civil aviation is based. The legal purpose of avionics certification is to document a regulatory judgment that a device meets all applicable regulatory requirements and can be manufactured and operate properly.

The regulating organizations are the FAA (Federal Aviation Administration) in the USA or the JAA (Joint Aviation Authorities) and the EASA (European Aviation Safety Agency) from 2003 in Europe. They act primarily through the publications and the enforcement of the Federal Aviation Regulations (the FARs) or the Joint Aviation Regulation (the JARs).

In the scope of this thesis, as foretold the requirements that interest us consist of those which are in relation with computer-controlled systems (hardware, software and control laws) as we consider the EFCS as our case study.

1.4.3 Fly by Wire System Certification Background

From [Cary 2001], it is stated that no airplane can fly unless this airplane has a type certificate issued by the aviation authorities of the airline country (FAA or JAA. See section 1.4.2). Each country has their own regulatory material although the common core content is basically the same.

This common part is basically composed of two main parts [Cary 2001]:

- the requirements: An example of such requirements is "The airplane system must be designed such that the occurrence of any failure condition, which would prevent the continued safe flight and the landing of the airplane, is extremely improbable" (from the Federal and Joint Aviations Requirements 25.1309 or EASA Certification Specifications 25.1309 [EASA 2009]).

  Remark: The Certification Specification (CS and the JAR / FAR) 25 are the specifications for large airplanes. As so, there exist many others Certification Specifications

for other aircraft types. An exhaustive list can be found at *http:// www.easa.eu.int/agency-measures/certification- specifications.php*.

- a set of interpretations and acceptable means of compliance: A part of the regulation (called Advisory Circular) gives the meaning and discusses the terms such as "failure conditions" and "extremely improbable". Moreover, a guidance on how to achieve compliance is given.

These aviation regulations are evolving to be able to consider the scope of new technologies (such as the use of fly-by-wire) [Cary 2001]. This is mostly done by the use of special conditions targeting a specific aircraft and by modification of the general regulatory material in later times. Many new topics were addressed such as the notions of flight envelope, the side-stick-controller, safety assessments, etc etc (for details see [Cary 2001]). The result of this operations was the integration of new regulatory materials in the core regulatory material consisting of numbers of documents, and mainly: the ARP4754 [of Automotive Engineers (SAE) 1996a], the DO178B [RTCA-EUROCEAE 1992, DO-178B 1993], the DO254 [DO-254 2005] or the ARP4761 [of Automotive Engineers (SAE) 1996b]. More details on the specificities of these guidelines are presented in section 1.4.5.

1.4.4 Dependability issues

The EFCS is a critical system, as far as safety and dependability is concerned, in the sense that a failure or a loss of this function would lead to catastrophic consequences. [Philippe 2011] gives some examples of the faults that could lead to catastrophic events such as control surface runaway or jam, loss of control over pitch axis or oscillatory failure at a frequency critical for the aircraft structures. All the aforementioned fault types must be extremely improbable (i.e: with a probability of less than 10^{-9} per hour of flight and considering qualitative requirements (FAR/JAR 25.1309) . It is also required by the FAR/CS25.1309 and FAR/CS 25.671 that, specifically for flight control systems, the occurrence of failure must not be due to a single failure.

Designing and implementing systems are limited at first sight to satisfying requirements and specifications. But then, the performance level of the product varies as a function of noise and external perturbations. A product can be considered robust as long as its intended response is not altered (or barely altered) by non controlled perturbations and modeling uncertainties. An optimized product that only works in very particular known conditions is not robust. The objective of robust design is to optimize product performance along with minimizing sensibility to perturbations and uncertainties. Robustness is (and must be) a general concern that grows with system complexity and its associated uncertainty.

In order to be compliant with the Airworthiness requirements, there are industrial practices based on the guidelines documents that have been presented in section 1.4.2.

1.4.5 Fault tolerance and robustness implementation in Flight Control Systems

Current industrial practices include well-known and well-mastered implementations of robustness and fault-tolerance at the system level.

An overview of industrial practice is given in [Philippe 2011].

Safety System Assessment

It is an analytic approach based on a type of fault-tree to analyze the effects of each functional failure on the system. It is intended to provide an exhaustive study of all the combinations of failures in order to determine the probability of occurrence of an undesirable events. The probabilities of elementary failures are given by equipment manufacturers and confirmed by adequate experiences. This safety analysis contributes to a design of fault-tolerant systems, and is compliant with the safety requirements in the regulations.

The operation includes:

- Safety criteria and a hazard classification procedure.
- Results of safety analyses performed.
- Identified risks still existing and constituting a hazard to operators and maintenance personnel.
- A hazards list and recommendations for their elimination or reduction to an acceptable level.
- An assessment of the system compliance with the safety requirements.

Because safety analysis is present in every step of the life cycle thanks to regulatory processes, this analytic method is valid for any system (final or intermediate systems).

Development processes mapped on regulatory guidelines

The development process of a critical system, and mainly a flight control system is very stringent, in terms of process constraints, quality and safety.

The following paragraph give different examples of guidelines that are mapped to the development processes for the critical avionics systems.

- ARP4754 [of Automotive Engineers (SAE) 1996a]: A guideline document on system design, verification and validation, configuration management, quality insurance. This document is the certification considerations for highly-integrated or complex aircraft systems
- DO178B [RTCA-EUROCEAE 1992]: A guideline document on software design, verification, configuration management, quality assurance. The level of rigor and processes applied to the software is based on the DAL (Design Assurance Level)

level. At the present time, the DO-178B is about to be update with the Do-178C. In the software development process, the dedicated guidelines DO 178B does not involve in the content of the software but instead in the processes of the development to comply with (planning, programming model, verification and tests to be done including timing, quality insurance, ...) in order to ensure compliance and to obtain equipment and aircraft certification.

- DO254 [DO-254 2005]: A guideline document on hardware design, verification, configuration management, quality assurance. The level of rigor and processes applied to the hardware is based on the DAL (Design Assurance Level) level.

- ARP4761 [of Automotive Engineers (SAE) 1996b]: A document giving guidelines and methods for conducting the safety assessment process on civil airborne systems and equipments.

Hardware redundancy

In the context of real systems, it might happen that, in order to ensure the availability of a given function, it is implemented many times inside the global system. It is the case, for instance, in flight control computers, which include many computer units to do the same function. Thus, it is ensured that if a computer fails in completing its role, the function of flight control can be done by one of the other computers that are left.

In the case of EFCS, these computers are in charge of control law computation as a function of the pilot and sensor inputs. There are some specific knowledge from the manufacturer and system designers to implement redundancy in real systems. Therefore, the system incorporates sufficient redundancies to provide the nominal performance in terms of availability of the function and in terms of safety with one (or two) failing computer, while it is still possible to execute the aircraft function safely with one single computer active. The use of multiple fly-by-wire computers (a total of 5 on the Long Range A330/340 aircraft, up to 6 on the A380 or some other, triplex or quadruplex redundancy architecture for other aircrafts such as Boeing) and the use of different power sources for control surfaces actuation are variant of redundant architecture in practical case.

At the level of a equipment architecture, there is still another level of redundancy called dissimilarity. First, the fly-by-wire computers do not all have the same internal architecture in terms of components and software. In fact those computers are divided at least into two categories: the primary (PRIM) computer or the FCPC (Flight Control Primary Computers) and the secondary (SEC) or the FCSC (Flight Control Secondary Computers). Those different computers are developed by different teams to dissimilarity is ensured. Again, this fact illustrates the fact that following the development process and guidelines implement fault tolerance because, in the case where the development process requires that different teams design the redundant computers, it helps very much in avoiding the so-called "common-mode failures" . This type of redundancy is also implemented at the software level where the software function of the redundant computers are developed in other languages allowing to avoid common-mode failures.

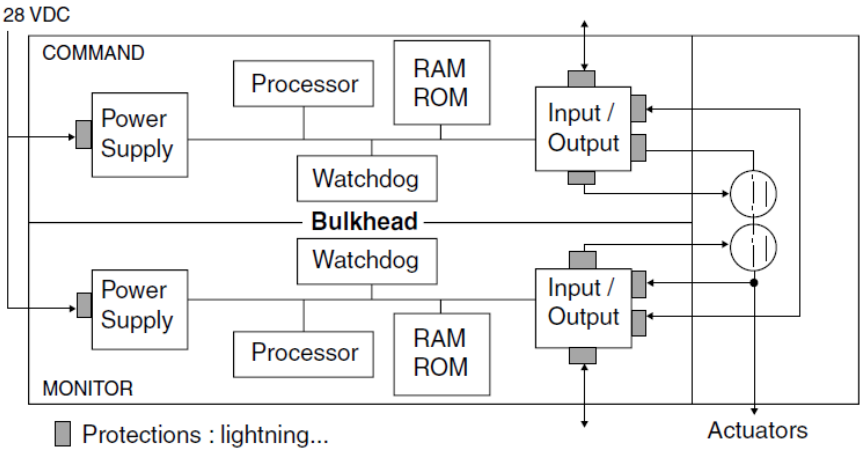

Figure 1.5: Command Monitoring internal architecture

Moreover, in each computer, the internal architecture is still divided into two channels: the command (COM) channel and the monitoring (MON) channel (figure 1.5). Each channel monitors the other but each channel has a specific task. The COM channel provides the main functions allocated to the computer (flight control law computation and sending commands to control surfaces) while the MON channel ensures (mainly) the permanent monitoring of all the components of the flight control system (sensors, actuators, other computers, sensors, probes, etc.) . It is designed to detect failure cases and to trigger reconfiguration by signaling the failure to the COM channel and to the other redundant computers of the same function.

Reconfiguration management

If despite all the measures taken to implement robustness to the system, a fault occurs, then a measure must be taken in order to handle this fault. Among the many ways to handle faults, one of the most practical is a reconfiguration of the system.

Reconfiguration management is defined as the automatic operation following a failure that allows the system to cope up with faults that have occurred. Reconfiguration consists of the operations made in the system to change itself in order to handle perturbation and uncertainties. In flight control systems, there are basically two levels of reconfiguration:

The first level of system reconfiguration consist of handling actuator faults. In transport aircraft, the different control surfaces (rudder, ailerons, slats, flaps, spoilers, elevator) are controlled with redundant actuators. This mechanism is illustrated by figure 1.6 from [Cary 2001, Traverse 2004] (the same reference presents the same architecture for the A320 aircraft).

The flight control computer act on each actuators of each control surface in a redundant manner. In the figure 1.6, the actuators are denoted by colors (Y: Yellow, G=Green, B=Blue). The principle of this type of reconfiguration consist of a robust choice between the computer that is in charge of updating the actuator states. This choice consist of designating off -line the scenarii of the actuators that are designed to act in passive mode and

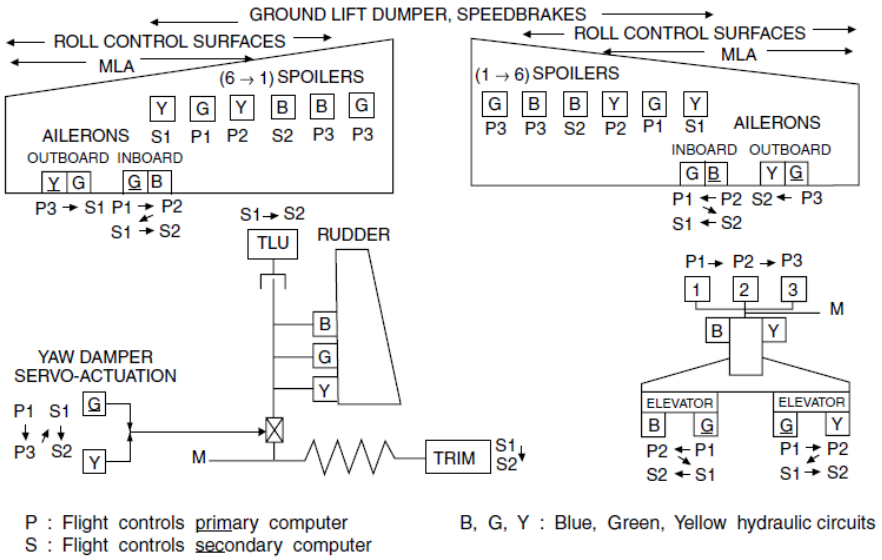

Figure 1.6: A A330/340 flight control architecture

active mode on the different control surfaces.

The second level of reconfiguration is the flight envelope, defined as a system of reconfiguration of the control laws. It implies the existence of numbers of control laws that can be switched from one to another depending on the situation (see Figure 1.7)

In normal conditions, the aircraft is protected against critical forbidden states signaled by events. These events include, for instance, the aircraft protection against stall, excessive altitude (pitch, bank), excessive speed or excessive load factors, etc .... The corresponding flight control law is called the " normal law". It requires a high level of integrity and the redundancy of the computers. Should any protection be lost because of a case of failure (e.g: loss of a control surface, Air Data Reference, Inertial Reference System, loss of the flight control computer FCC), a lower-level control law, called " alternate law " is activated. Flight is still possible but full protection is no longer guaranteed (for instance, there might be no more comfort of the passengers). The last level of control law is called "direct law" which requires manual trim of the aircraft.

1.5 Real-time Control

Real-time control is an interdisciplinary area that include aspects of Control Engineering and Computer Science . In fact, references such as [Årzén 1999, Törngren 1998] show that there is a link between these domains and considering only one domain with a single point of view is a poor approach to real-time control. Thus, this section considers a brief reminder of real- time systems in general, the design and properties which make them critical. It is followed by a brief reminder on continuous-time control systems. It references the different models and analysis tools that is used by the control community. Then a subsection on computer controlled systems looks more closely at these tools and models

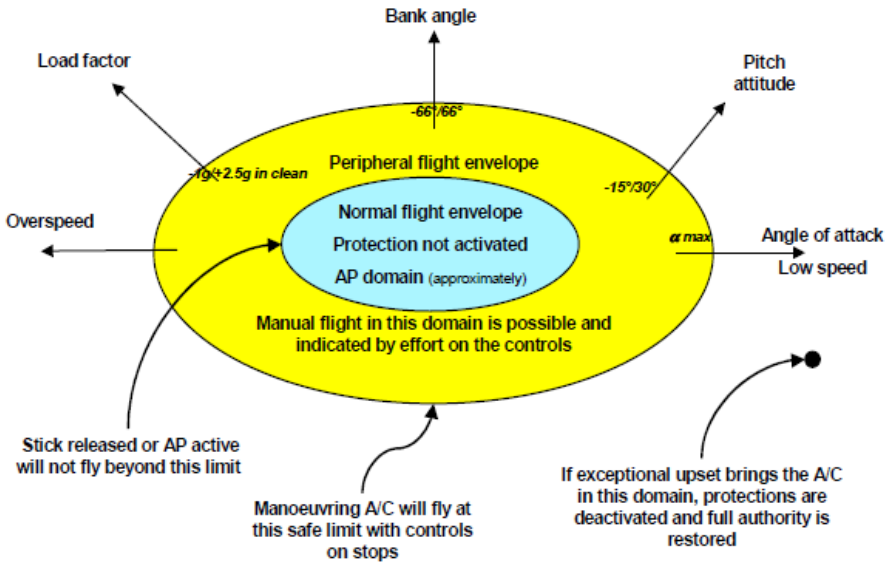

Figure 1.7: Flight Envelope

and makes a link between computers and control systems, and also between computer science and control community. This should allow to convince the reader that the subject is a indeed an interdisciplinary problem.

1.5.1 Real-time systems

Real-time systems architecture

They are systems in which temporal aspects of their behavior are part of their specifications. In fact, the correctness of the system depends not only on the logical correctness of the results of computations but also on the timed arrival or the timed ending of the computation results [Stankovic 1988], implying that time is also considered as a shared resource of the system. Real- time systems are mainly systems that need to have intensive interactions with their external environment which also evolves with time. Avionics systems include many instances of real-time systems, namely flight control systems and braking systems.

The real-time aspects allows to ensure the timing behavior of the control system very precisely.

The simplest architecture of a real-time system is presented as in figure 1.8.

The plant is the controlled system and the controller which interacts with update its states by computing control laws. It is a real-time control system.

In the early age of control, analogue computers were used to work out the control signals, firstly from mechanical devices, and then from electronic amplifiers and integrators : both the plant and the controller remained in the realm of continuous time, while frequency analysis and Laplace transform were the main tools at hand. Nowadays, due to the spreading of computers, most control systems are computer-based systems [Årzén 1999]. The sampling theory and the Z-transform became the standard tools for digital control systems

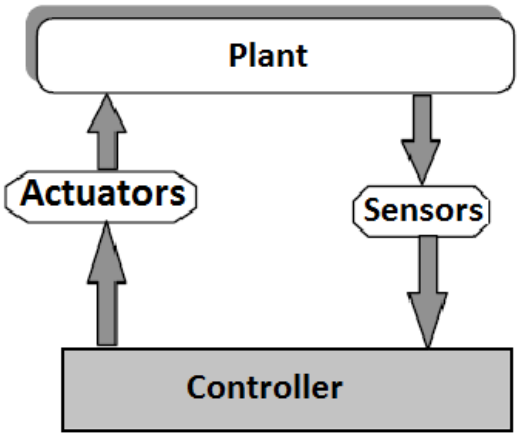

Figure 1.8: Real-time system

analysis and design. A successful design and implementation of a computer-controlled system requires a good understanding of both control theory and real-time scheduling theory.

Real-time control design and implementation

Many real-time control systems are embedded systems. An embedded system is a computer system designed for specific control functions within a larger system, often with a set of constraints such as real-time constraints [Heath 2003]. Those physical constraints arise through interactions of the computational processes with physical world. An classic example of such interactions consist of the stabilization of an aircraft while flying through the airflow where the timed action of the control surfaces permits to keep the aircraft stable.

The traditional design of a computer-controlled system involves mainly two steps. At first step, the control law (or algorithm) is either discretized from its continuous-time equivalent, or directly by using discrete-time control design theory. The second step consists of converting the discrete- time controller into a real-time software tasks, scheduled to execute on computing target platform, where real-time scheduling algorithms based on the worst case execution time is generally used to guarantee the timely behavior. The controller system is then composed of the combination of the control software tasks (constructed from the discrete-time controller), the device drivers and the hardware on which they run, while the physical plant is still in continuous- time.

The design process and verification process of such systems must be compliant and conform to regulatory safety requirements (e.g: the DO178B and the DO254) for safety-critical embedded control systems.

Figure 1.9 shows a schematic of such design and implementation for a flight control system, in which the Reference signal symbolizes the pilot signal and the FCS_C (here meaning the Flight Control System Computer) samples and computes the control value based on a given control law. The sensors capture the states and the outputs of the controlled system.

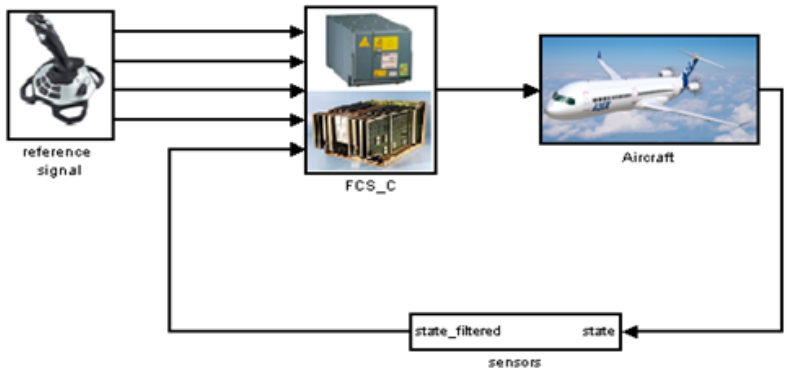

Figure 1.9: Flight Control Systems

Real-time scheduling and task constraints

Real-time scheduling is an important subject in the domain of real-time control systems, as the controller is composed of control software tasks and the hardware processor. It consists of defining the execution orders and the execution timings of each task on the processing units of a given architecture. These parameters are called the scheduling parameters. The main objective is to foresee with precision the temporal behavior of the considered system.

Tasks have parameters that characterizes themselves and that allow scheduling. For a given task τ_i, those parameters are:

- **Arrival time** a_i is the time at which a task becomes ready for execution. It is also referred to as activation time or release time (denoted by r_i).
- **Start time** s_i is the time at which a task starts executing.
- **Finishing time** f_i is the time at which a task finishes its execution.
- **Absolute deadline** d_i is the time before which a task should complete its execution.
- **Worst-case execution time (WCET)** C_i is the maximum time needed for the processor to execute the task without preemption.
- **Relative deadline** D_i is the deadline with respect to the arrival time, that is $D_i = d_i - a_i$.

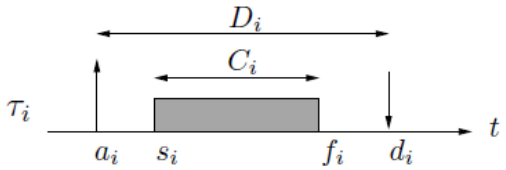

Figure 1.10: A typical real-time task

Figure 1.10 illustrates the characterization of a typical real-time task.

Depending on the system *criticality*, the tasks are designed to be:

- **Hard real-time**, in which the system must operate within the confines of a hard deadlines. Such systems must guarantee that tasks complete on time. A missed deadline in a hard-real-time system is considered catastrophic.

- **Soft real-time**, meaning that only the critical tasks get priority to get completed, over other tasks and retains that priority until it completes. The objective is also to meet deadlines, but the difference resides on the fact that soft-real-time systems allows a few deviations from the expected timing constraints and a few deadlines are accepted, because, in opposition to hard-real-time, it is not considered to be catastrophic. However, it is important to understand that in soft-real-time systems deadlines can be missed (or latency can be added) in a non-predictable way.

- **Weakly-hard**. This is less common that the two strategies above. However, it is known and demonstrated that most hard-real-time systems can miss some deadlines provided that it happens in a known pattern and in a predictable way. In weakly-hard real-time systems, would be similar to the soft-real-time systems, with the difference that deadlines misses happen in a predictable and mastered way. For instance, a deadline miss can happen but it is mastered and its consequences have been awaited for.

WCET-based scheduling

Traditional control design considers constant sampling rates with equidistant samples (e.g. no jitter) and negligible, or fixed and known delays. On the other hand real-time scheduling theory has mainly focused on how to dimension resources for meeting deadlines (or equivalently, on the schedulability analysis for a given resource) [Sha 2004]. Therefore the computer science and real-time scheduling communities do their best to implement control tasks considering fixed periods and hard deadlines, and assuming that the Worst-Case Execution Time (WCET) is precisely known. This assumption has served the separation between control and scheduling designs, but leads to an under utilization of CPU resources, and such approach faces both technical, economical, and industrial challenges.

One of the toughest challenges in the current approach is the determination of the WCET, in order to correctly size the system. The tightness of the result is related to the predictability of the processing unit. The future generations of processors seems to go apart from the predictability and determinism objectives of the execution time. Processing speeds and performances grow up very fast thanks to accelerating but unpredictable mechanisms of new processors. However it becomes difficult to foresee their effects on the execution time considered in the worst case. Nowadays, even if many attempts are proposed to give an upper bound of the WCET (e.g. [Rochange 2007]), both the traditional and current approaches are difficult to be applied to modern processor generations and produce values which are pessimistic [Puschner 2008]. Appendix A provides an overview of methods currently used for the evaluation of the WCET.

Considering the WCET as a reference to implement the control laws, the hard and costly way consists in building a highly deterministic system so that the actual implementation parameters meet the ideal ones. By essence, implementations purely based on WCET and hard deadlines considerations are conservative and lead to a large under-utilization of the computing and networking resources and finally ending to an oversizing of both electrical supplies, cooling systems and aircraft weight.

In summary, we state that either WCET analysis become more and more difficult and more especially for modern processors, in which the probabilistic plot of the execution time is expected to spread larger and larger.

1.5.2 Dynamic systems

The basic assumption in control theory is the causality effect relationship for the components in a system. Therefore, a simple system represented by a block-diagram (as in figure 1.11), accepts inputs and produces outputs.

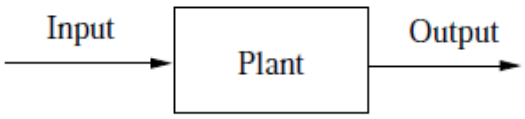

Figure 1.11: A simple block diagram

From the input/output point of view, the system forms the open-loop system. However, in this form, the control system usually fails to achieve the desired purpose if very precise information about the interaction of the plant with the environment is not available (for instance, if there are some noisy disturbances). Another point to take care of, is the fact that the model may not be perfect as well. Among others, uncertain linear plants with the following structure are considered in the sequel :

$$\dot{x}(t) = (A + \Delta_\mu A(t))x(t) + (B + \Delta_\mu B(t))u(t) \tag{1.1}$$

where x is the state vector and u is the control vector. Uncertainties are modeled as :

$$\Delta_\mu A(t) = \mu \sum_{i=1}^{M} \lambda_i(t) A_i \qquad \Delta_\mu B(t) = \mu \sum_{i=1}^{M} \lambda_i(t) B_i \tag{1.2}$$

$$\lambda_i(t) \geq 0 \qquad \sum_{i=1}^{M} \lambda_i(t) = 1$$

The considered controllers are linear state (or output) feedbacks such as :

$$u(t) = Kx(t_{k'} - T_{slot}), \forall t \in [t_{k'},\ t_{k'+1}] \tag{1.3}$$

The major problems that are solvable with control system problems are numerous nowadays. This schema, known as the feedback structure of a control system, is, at the basis of automatic control theory.

It contains all the basic components of a control system and show an ideal abstract model. In practice, there might be some other components in a more realistic system such as, the sensors, the actuators, the different estimation filters and other interfaces.

Systems may be described by many ways but in the thesis, we focus on systems described by the classical state model (see 1.1).

The main objectives of such systems are dealing mainly with the notions of stability, performance.

To conclude this section, a slight reminder on the overall aim of this thesis. It does not consists in developing a new type of controller for dynamical systems, nut instead the objective is to consider and prospect solutions for the dual problems coming from automatic control and their computer implementation. These problems will be described more precisely in the following sections.

1.5.3 Computer-controlled systems

Realistic model: Sampled-Data systems

As the plant is a physical continuous-time system, the system would need to integrate an additional interface between the discrete-time computer and the plant. The system contains:

- An analog (continuous-time) process symbolizing the system to be controlled. It's a dynamical system whose model can be described by Ordinary Differential Equation or Partial Differential Equation.
- The input $u(t)$ and the output $y(t)$ of the process are continuous- time signals.
- A sampler made of an Analog-to-Digital (A-D) converter converts the analog output signal of the process into a discrete- time signal. It paced by an internal sampling clock. Depending on the need of the system, in terms of precision, the sampled can be coded in a finite precision number depending on how many bits the computer needs. This is the *sampling* process.
- A Computer is made of an internal clock and the software tasks implementing the control algorithms. It takes the discretized values y_k from the A-D and computes new values of control signals u_k using the control algorithm. Then the computed control u_k signal is transferred to the DA.
- A Digital-to-Analog (D-A) converter with a hold circuit does the opposite action of the A-D. It takes a finite precision number from the computer controller, which are the newly-computed control signals, and converts them into a continuous-time signal. The hold circuit is in charge of the value of the signal between two sample times. When the hold device is a ZOH (Zero Order Hold), the signal between two sample times is piece-wise constant.

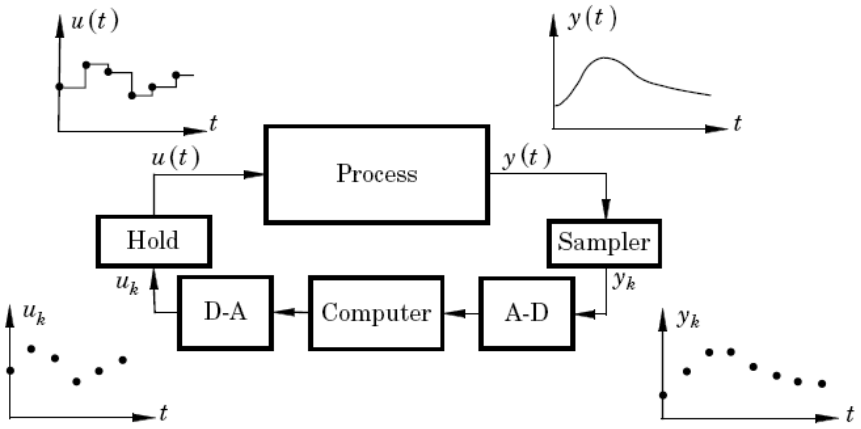

Figure 1.12: Realistic sampled-data model

Such a model is described by figure 1.12. This kind of system is then a mixture of a continuous-time system and a discrete-time system: it is called a sampled-data system.

The sampling process and time-dependence

From a purely analytic point of view, the mixture of a continuous-time signal and a discrete-time signal is problematic. This can be solved by looking at the whole system only at discrete-instants (which is the instants of the discrete-time signal). Theses instants, where the measured variables are converted into discrete sequences, are called the *the sampling instants*.

Also,the fact that computer-controlled systems are clock-driven (by the internal clock of the computer), makes the system to be time-dependent. Assuming that a timing disturbance (for instance response delay), appears at the output of the plant. Time-invariance implies that a shift of the input of the computer (which is the output of the plant), should result in a similar shift at the output of the computer. However, as the internal clock of the computer is not aware of this delay induced by the output of the process, the computer reacts differently when the disturbance is shifted in time. It is stated by [Wittenmark 2003] that the response of the system to and external stimulus depends on how that stimulus is synchronized with the internal clock of the computer. Thus, under synchronization assumptions on the inputs, the outputs *and* the eventual disturbances, the system will somehow stay time-independent.

Zero Order Hold (ZOH) reconstruction

The most commonly used hold model in the signal reconstruction is the ZOH. It consists of taking a sample value in the sampling instants and setting the inter-sample as a piece-wise constant value until the next sample value is obtained.

Assuming a sampling period h, the reconstruction of $x(t)$ from its sampled equivalent $x(s_k)$ (for $k \in \mathbb{N}$) is given by:

$$\forall k \in \mathbb{N},\ x(t) = x(kh)\ \ k.h \leq t < (k+1).h \tag{1.4}$$

Figure 1.13 shows the reconstructed signal.

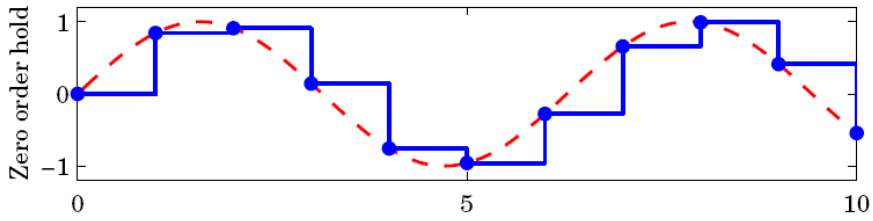

Figure 1.13: ZOH reconstruction

This shows that the sampling mechanisms and the reconstructions introduces a loss of information. This is among the numerous issues that computer-controlled system domain has to deal with.

1.6 Problem Statements

1.6.1 Hard Real-time paradigm

From the real-time scheduling side, modeling and analysis have been initiated in the illustrious seminal paper [Liu 1973]. This first schedulability analysis was based on restrictive assumptions, one of them being the periodicity of all the real-time tasks in the system.

Even if more general assumptions have been progressively introduced to cope with more realistic problems and tasks sets, the periodicity assumption remains very popular. This can be seen in the use of classical scheduling theory in practice which are based on the periodicity of the tasks such as Rate Monotonic, Fixed Priority...). The combination of these popular modeling and analysis methods and tools have likely reinforced the understanding that computer-controlled systems are basically periodic and hard real-time systems.

This description has shaped the computer-controlled theory during more than three decades. It is however questionable if this is true for all control system.

Currently many control systems, e.g. flight control, braking control systems, are considered to be hard real-time that is to say it is most often assumed at design time that control tasks must be executed strictly periodically. Therefore control tasks executions are bounded to fixed time-slots, and deadline misses or jitter are forbidden. It is assumed that any deviation from the ideal timing pattern inevitably leads to a failure of the system. The implementation of such control tasks relies on a safe evaluation of the WCET of each task, which is used to dimension the size of the time slot allocated for the execution of the control tasks.

As for the domain of safety-critical system, integrating fault-tolerance in those systems is difficult without massive redundancy, as it is the case nowadays in flight control systems. In fact, in this kind of application, fault -tolerance is achieved by using many computer in

a redundant way, while at the level of each computer, timing behavior is still defined using the classical hard-real-time assumptions leading to a rigid and non-robust implementation.

Moreover, as program and software get larger and more complex, there is a permanent desire to optimally use the available hardware. According to [Bedin França 2011], it is a challenging problem to try to meet both performance and safety, from the software implementation point of view because these two characteristics may appear as contradictory goals.

In this context, the execution schedule of a single real-time tasks that is based on the respect of the WCET is presented in figure 1.14

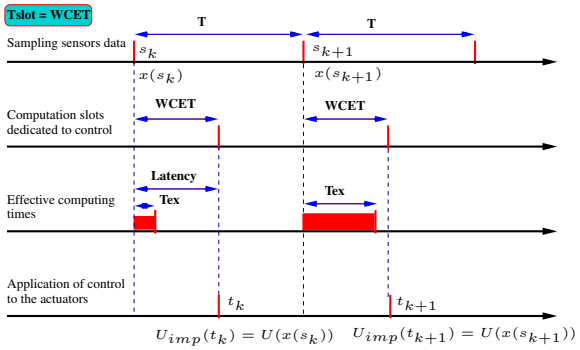

Figure 1.14: WCET based control task execution pattern

A task execution is strictly periodic : a time slot $T_{slot} = WCET$ is allocated to the task execution, it is triggered at a period $s_k - s_{k-1} = T$ by the occurrence of measurements $x(s_k)$ at time s_k. The controller computation takes a time T_{ex} which is, by definition, always smaller than the $WCET$. To avoid output jitter, the control signal $U(x(s_k))$ is applied to the actuators at the end of the slot, i.e. at time $s_k + WCET$:

$$\forall t \in [s_k + WCET, s_{k+1} + WCET[, \quad U = U(x(s_k))$$

. Therefore it is a periodic control system, with constant period T, subject to a constant delay $T_{slot} = WCET$.

This implementation fits well with the hard real- time assumption, and should be applied when the controller tasks' timing constraints are really hard e.g. if it is a Finite State Machine which may fail in an unpredicted state in case of deadline miss and interrupted transition.

1.6.2 Challenging issues

As the time slots are allocated based on the WCET of the control tasks, the computations always finish before the end of the slot : therefore a fraction of the computing power is unused. The *wasted computing power* is all the more important as the WCET is far from the average value of the observed execution time T_{ex}. Indeed, due to an increasing demand on services, new control systems are more and more based on networked architectures and shared off-the- shelf computing devices. However enhancement of computing power are

often based on the usage of multiple levels of cache and pipe-lines, lowering the determinism of the processors and increasing the difficulty of searching for the program's WCET, which are in fact approached by increasingly conservative upper bounds [Souyris 2005].

Indeed the execution times distribution plot (as shown in Figure 1.15 for a dedicated embedded processor [Hansen 2009]) is expected to spread out, so that approaching the worst case execution time is foreseen to become a rare event.

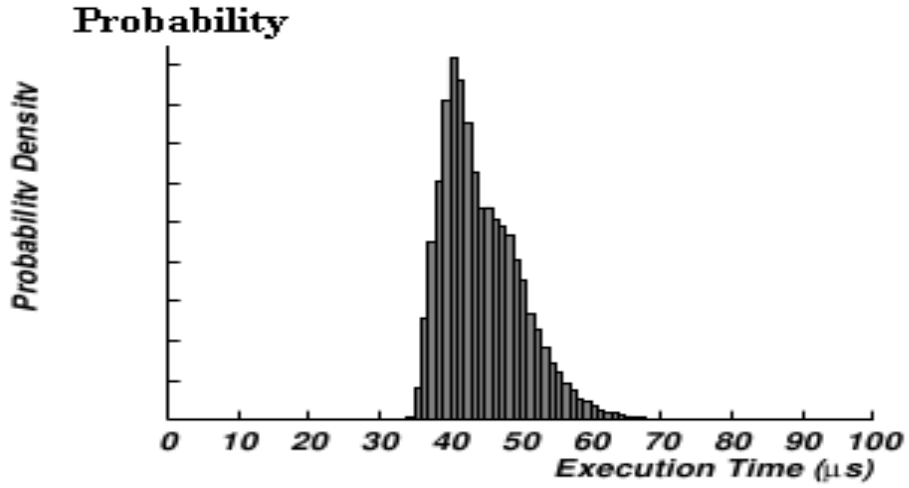

Figure 1.15: Typical execution time distribution

Therefore the amount of *wasted* computing power is expected to increase, leading to the over-sizing of embedded computers, power supplies and cooling systems, among other issues.

Nowadays, even if many attempts are proposed to give an upper bound of the WCET (e.g: [Rochange 2007]), both the traditional and current approaches are difficult to apply to modern processor generations and produce values which are more and more pessimistic.

Lately, a new approach to improve the robustness of the task scheduling has been proposed in [Midonnet 2010]) where on a temporal fault occurrence, the slack time (the difference between the allocated time and the real execution time) can be dynamically determined and assigned to the faulty task in order to complete its treatment. As the authors mean " deadline miss" when they talk about "faults", this approach implies that the computing resource has enough spare resource to add more time slack. This approach has two drawbacks in practice. First, the dynamic addition of the time slack will cause indeterminacy and worsen the predictability of the system. Second, from a control system point of view, it is known that delays decrease the control performance and moreover, as the time slack is dynamically added, it will result into a control system with time-varying and non bounded delay, leading to more difficulties in the analysis. On the other hand, the method proposed in this thesis allows for the enhancement of predictability and determinism of the system while enhancing control performance. This is achieved by " weakening real-time constraints".

Hence it is worth to discuss and revisit the "hard real-time" assumption and examine how it can be weakened in the particular case of *feedback* control systems.

Choice of sampling period

An adequate selection of sampling period is very important in computer controlled systems. Two configurations can be considered:

- Too large, the sampling period would alleviate the computation loads of the processors, but would also present problems, related to the reconstruction of the signal, according to the schema we have mentioned previously in this section.

- Too short, it will cause an overloading in the computer in terms of computing load because shortening the sampling period would increase the number of samples within a considered frame of time. It would lead to an increased probability of deadline violations, which would inevitably lead to certification problems. but on the other side, shortening the sampling period, would also enhance control performance. Indeed, while the sampling period is shorter, it is also closer to the continuous time systems in terms of control performance.

Rules of thumb are proposed to choose approximate bounds for the sampling interval h as in [Wittenmark 2003]. For example practitioners often begin with h values such that the plant is sampled between 3 and 10 times the smallest time constant of the system ,τ, such that: $\frac{\tau}{10} \leq h \leq \frac{\tau}{3}$.

The essence of this section is obviously the considerations of sampling interval choices. Regarding the previous statements, it is not the most relevant approach to consider too small values or too large values of the sampling period. Indeed, considering these two extreme values would be considered as privileging only one point of view: either privileging the control performance point of view (for short sampling period) or privileging the implementation side (the computing performance) at the expenses of control performance. In this thesis, we argue that the best approach is a joint consideration of both subject (control and scheduling domains) and a trade-off between them is the objective of this thesis.

1.7 Control and scheduling co-design

The overall objective of control and scheduling co-design is to achieve better control performance with better resource utilization. As it is implied from previous sections, control performance is a vision of the considered system (mainly feedback systems) from the control theoretical view. On the other side, better resource utilization can be achieved from the scheduling theory point of view. In order to clearly present these aspects, it is important to understand the impact of the scheduling on the control performance and vice versa. It is also needed to introduce the different constraints and different metrics.

1.7.1 Control Systems hard deadline

Control systems are often cited as examples of so-called "Hard real-time systems" where, any deviations from the ideal pattern and trajectory that has been designed off-line are strictly unauthorized. From safety considerations, these terms means that a non respected deadline in a single task, can have catastrophic consequences, due to the eventual loss of the considered function. This means that the criteria used to establish these hard deadlines are safety- based.

As presented in Section 1.5, computer-controlled systems based on the WCET paradigm are hard real-time. Indeed, if a computer is part of a hard-real-time system, the control tasks running on it are subject to its hard constraints. Thus, all software components have to be tuned to satisfy those deadlines.

Section 1.4.5 has also presented that integrating fault- tolerance techniques in real-time safety-critical control systems is difficult without a massive redundancy, because any kind of upset can lead to a deadline being missed. Thus, the approaches based on hard deadlines is justified in terms of safety but not in terms of performance and quality of control.

A historical reflection can be found with the notion of "Allowed State Space" and "hard deadlines for control systems" presented by [Shin 1992] and [Shin 1985]. The idea behind the allowed state space consist of determining a space in which the value of the state space representation of the system stays stable.

This reflection, combined with those proposed by [Shin 1992] and [Shin 1985] permits us to derive a definition for a hard deadline as far as control systems are concerned.

Recently, works such as [Aubrun 2010], [Silva 1993] and [Andrianiaina 2011c] show that control systems are not meant to work close to a "hard deadline" as it would be understood from the perspective of task scheduling. It means that the *hard deadline* of the control software is not the *hard deadline* of the control system seen from end-to end. More specifically, we mean that it is not right to assume that the deadline of a software task would be the timing limits of a control system, as the software task is only a part of the system.

Instead, it is argued in those papers that control systems are supposed to work to a more relaxed "performance deadlines" and that the source of constraints are more from the computing system. It is also argued in those papers that control systems can have a so-called "grace-time", which is the difference between the control system performance deadline and its *hard deadline*.

A *hard deadline for a control system* is defined, for a computer- controlled system (integrating feedback control loops), as the critical value of the computation-time jitter beyond which the system leaves it allowed state space and becomes unstable.

This notion of hard deadline and performance deadline (and grace time) can be found at the basis of linear control system theory with similar notions as phase margins, delay margins and module margins, which permit to quantify the system robustness. It appears that a phase margin (when it exists) implies a delay margin (i.e. the maximum unmodeled constant extra delay that can be suffered before instability) and certainly a jitter margin, which is more difficult to quantify ([Cervin 2004]) but which can be experimentally shown ([Cervin 2000]). The interesting point is that a feedback control system which is robust with respect to the plants parameters uncertainties is also robust, to some extent, with

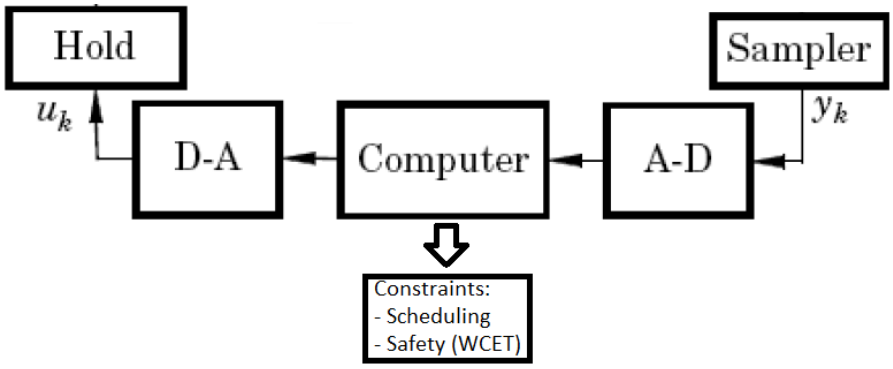

Figure 1.16: Classical control design from computer point of view

respect to timing deviations. Hence a feedback control system is not as hard as it is often considered in the literature, but should be better considered as "weakly hard", that is to say, it is able to tolerate a predefined amount of timing deviations before leaving their specified allowed performance set ([Bernat 2001]).

1.7.2 Control and scheduling integration

Interaction between scheduling and control

Timing parameters is of a capital importance when discussing about real-time control. There are the timing requirements (as introduced in the previous subsection) that are derived from control design. This subsection's objective is to show that these timing requirements imposed by discrete-time design theory on the software implementation are not realistic with respect to the behavior of the implemented system.

However, it is clear that while control theory allows to define the closed- loop requirements, the implementations approaches purely based on WCET and hard deadlines are also by essence very conservative, leading to the definition of more stringent constraints on the computing and networking resources in order to satisfy the corresponding schedulability analysis and leading to oversized systems [Andrianiaina 2011c].

Moreover, works such as [Shin 1999,Shin 1995,Shin 1992] show with different notions such as "‘ Hard Deadlines for Control Systems" and "‘Allowed State Space"’, that control systems are supposed to be more flexible and less constrained.

From the control point of view, the operation schemes can be divided into three parts:

- sampling
- control computation
- actuation

Figure 1.17 illustrates such division. A detailed description about the different components of this schema has been proposed in section 1.5.3. The sampling is classically known to be performed at equidistant instants with a given sampling period denoted h or

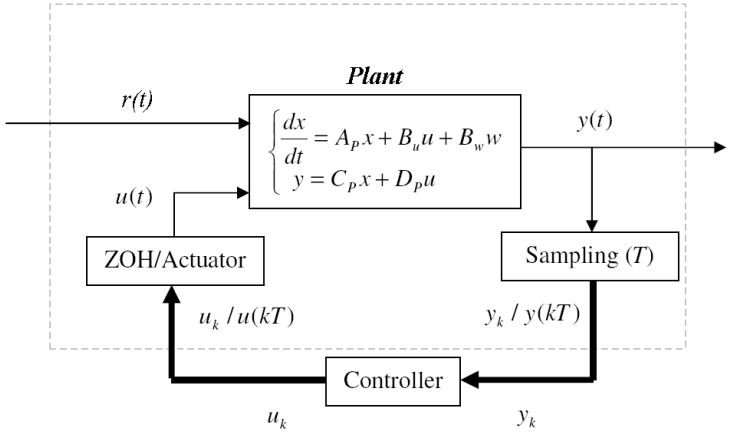

Figure 1.17: Sampled-Data System

T. The time between the sample captures and the actuation operation is also considered to be constant.

The control software computes the input values *as soon as* the samples are available. These events are synchronized by the real-time clock of the computer.

Remark: Recalling the idea of using digital computers as components of a control system beginning around the 1950s [Wittenmark 2003].

The notion of zero-execution time, used mainly in synchronous approaches of real-time programming, has led to the assumption that computing resources are perfectly deterministic. However, in practice, it is almost never the case. This is especially the case for COTS (Commercial Off The Shelf) material and software, which are optimized to achieve good average-case performance rather than worst-case performance. For this reason it is impossible to use them in highly safety-critical systems such as flight control systems without very specific considerations and studies.

As the computation inevitably takes some physical time, the zero-execution time assumption is incomplete and more realistic assumptions have been introduced. From the point of view of real-time computing community, the Worst Case Execution Time approach has been used . From the point of view of the control community, time-delay systems (see 2.4.1) have been deeply studied to provide more formal models and less conservative approaches.

The general idea is to consider the computing time as equal to a quantity τ instead of zero, where τ is set to be equal to the WCET of that software task. Time-delay systems theory considers many approaches (for instance by assuming that τ is constant or time-varying).

A combination of these different parameters have been used in practice for long. Until recently, the most common and the most realistic is the combination of constant sampling with constant delay.

These different assumptions have provided a separation of concern between control community and scheduling community. This fact has allowed the control community to be independent from any implementation problems and then to concentrate on control prob-

lems and conversely, it has freed the scheduling community from the need to understand the impact of their scheduling on the system.

One of the means to guarantee this achievement is to avoid timing variability and deviations. The source of the variability of the delay is mainly the computation time of the control laws. To avoid variability, it becomes necessary to bound this value with an upper bound, the WCET (Worst Case Execution Time), in order to keep it constant.

This approach is not realistic because it generates many constraints on the equipments and on the methods of development (e.g: to restrain on the choice processor and the real-time software programming methods in order to improve analyzability at worst case).

In the previous section, we have seen that applying standard timing requirements obtained from classical discrete-time control theory to express the timing parameters of the tasks leads to hard-to-meet constraints and poor schedulability (only decidable for non multi-tasks systems if an acceptable control performance is to be ensured [Martí 2002]). This is caused by the fact that tasks are assigned time slot equal to their WCET and this implies that each time a control tasks is launched, it has to run its execution immediately. One way to improve and to permit multi-task scheduling is to relax the strict deadline assignment of control tasks and allow to use some kind of scheduling strategies. However, it has been shown that enhancing performance by using dynamic scheduling generates jitter (time-varying delays) which are hard to cope with, when considering a joint real-time and control system analysis.

Therefore, real-time scheduling theory has mainly focused on how to dimension resources to meet deadlines, or equivalently, on the schedulability analysis for a given resource. New paradigms were introduced from real-time community in order to consider control tasks with constant sampling periods, hard deadlines and scheduling based on worst case execution times. Well-known scheduling policies (EDF, RM, ...) have been proposed to enhance scheduling performance. The main idea is to avoid resource waste and to improve processor utilization.

These policies assign priorities to tasks according to timing parameters such as their periodicity or their deadlines.

They optimize resources in terms of number of treated task jobs (for instance: maximizing the number of schedulable tasks jobs)

Unfortunately, they hardly take into account control constraints such as precedence and synchronization. In particular, precedence and synchronization between parts of the control algorithm can be used to minimize the I/O latency on some stability-critical paths of the control loop [Simon 2009]. These features are not handled by traditional real-time scheduling policies, which consequently do not comply well with feedback control purpose.

1.7.3 Co-design approaches: methods and tools

In the previous section, we have discussed about the separation of concerns between control and scheduling communities. Some sources of problems from one community to another have also been discussed. It has been shown that a new approach based on unification of

concern could be interesting in order to solve the problems of performance and complexity based on separation of concerns. Generally speaking, after looking at the interaction between control and scheduling, there are two ways of achieving an efficient control system design and implementation. The two new notions are respectively *Control-aware computing* and *Computing aware control*. These two notions respectively consider taking into account computing and scheduling implementation constraints from the perspective of control theory and conversely the consideration of control performance while implementing systems. These notions are generally called co- design approaches. They encompass control loop applications and their implementation. From centralized control system to distributed networked systems point of view [Aubrun 2010] , these notions aims, for instance, to consider data-loss and implementation- induced delays (or jitters) by using robust approaches.

Control-aware computing

In industrial problematics, this idea is almost unknown. In fact, in many industrial application, the position of control design in placed before the implementation in order to reduce development costs, and this without taking into account eventual implementation constraints. However, it is known that in real -time control system development, control and real-time computing implementation have been associated activities for long. The idea consist of the integration of control performance awareness in the scheduling parameters' assignment. Briefly speaking, from the control perspective, the objective to be reached is performance and stability (as would imply figure 1.18).

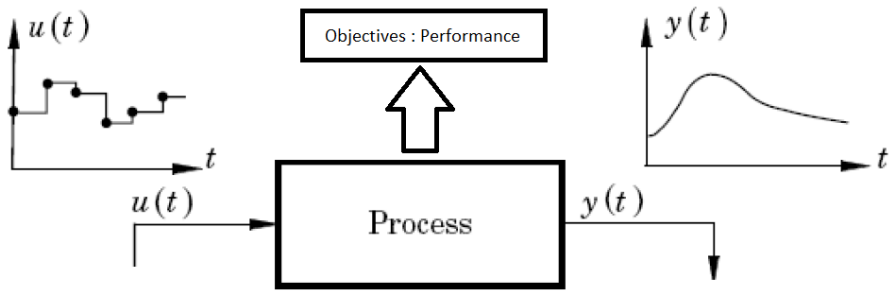

Figure 1.18: Performance of a computer-controlled system.

Many approaches based on existing works can be listed, and namely off-line approaches and QoS-based approaches.

Off-line approaches

Off-line approaches consist of methods that permit to set the scheduling parameters of the associated control tasks during control design. These parameters are supposed to ideally maximize control performance under scheduling constraints. Computational tasks that realize the control algorithms are usually scheduled by treating their execution times and periods as unchangeable parameters. Most of the proposed methods are based on the

selection of the sampling period while taking into account the scheduling constraints. An adequate choice of the sampling period is known to be one of the most important control parameter as explained in [Simon 2009] . Moreover, as the controller design primarily depends on the continuous dynamics, the resulting sets of tasks may not be schedulable with the limited resources available in embedded systems. In brief, too small the sampling period, the computer will be overloaded with too many workloads generated by too numerous but not necessarily useful computations, and too large, the control system will have poor performance or even go unstable.

Among the first works on off-line sampling period choice is found in [Seto 1996]. Even if a set of tasks if schedulable , the performance from control point of view, might not be optimal, in the sense that the resources are not fully and efficiently used (see section 1.6. In this references works, the tasks periods are allowed to vary within a given range in order to optimize resources use, while the main objective is to guarantee that critical control functions such as stability is not affected by this change. The problem of optimizing the performance index was stated as an optimization of a cost function with respect to some given constraints. Tasks scheduling (using algorithms such as Earliest Deadline First or Rate Monotonic) with respect to optimized processor utilization were also assessed.

Reflection on joint optimization of control and off-line scheduling has already been brought long ago. [Krishna 1983] has proposed the notion of performance measurement in control systems based on multiprocessor computer. The idea also considered the use of a performance index as in [Seto 1996]. However, their performance index was computed based on the controller response time. The main goal was to compare the performance of two rival computers by having metrics that permits to have an objective measure. One of the important ideas in the reference was the consideration of different classes of tasks: critical ones and non-critical ones.

This has been confirmed in [Aubrun 2010], where the authors stated that current industrial practice mainly implements controller as a single real- time task that is executed sequentially in a single loop, while, it appears that all the components of the algorithm do not require the same timing parameters and the same weight in terms of urgency. In fact, in order to minimize the latency, a control law can be basically implemented as different real-time blocks of software From a purely scheduling theory point of view, a quasi-similar approach was proposed in [Ayav 2006] (TAFT: Time Aware Fault Tolerance), where the task was divide into two parts: the urgent part and the non urgent part. Case studies, having to do with aircraft systems, were reported in these articles.

A beginning of reflection on delays with consideration to practical real-time control systems has also been proposed in [Shin 1993]. The idea in this work was focused on the tolerance of fault derived from SEU (Single Event Upset) or Electromagnetic interference leading to value fault of the computed controls and thus forcing to restart the computing from scratch.

Quality Of Service oriented control and scheduling

The main idea of those approaches consist of giving a quality (or performance) measurement indices based on a function of the sampling periods that can be adaptable at

run-time. The work presented in [Abdelzaher 2000], presents an online adaptation of the Quality of Services of the system based on the use of a middle-ware service on top of a best-effort operating system (or a scheduler). The mechanism is supposed to be an alternative for worst-case based systems, which considers off-line analysis of the reserved resources under the constraint of resource sufficiency.

Other significant analysis methods based on the adaptation of the sampling period and theoretically based on the well-known *Optimal Control Theory* , has been proposed in [Shin 1999]. This work has provided different algorithms for adaptive processor utilization allocation that can be used to make a graceful degradation of the Quality of Service. One algorithm considers independent tasks while the other take into account tasks that are related to each other. However, finding a systematic and non subjetive way of determining the perfomance index is difficult. This difficulty makes it hard to apply such approaches with safety-critical systems. A concise and well-described state of the art of the different methods on the problematics has been furnished in [Aubrun 2010].

Afterwords

Off-line approaches are more oriented in ensuring predictability in real-time systems, while QoS-oriented methods consider graceful degradation ensures the ability to cope with changing load and failure conditions, while maximizing system utility.

Computing-aware control

While control-aware computing consist of the control system's determination of the equivalent scheduling parameters or the graceful adaptation of the QoS with respect to the control requirements, *computing-aware control* consist the converse approach. It deals with the achievement of a good control under comupting system constraints. It then leads to considering the different hardware and software implementation limitations within the control design process. These consideration can include for instance robustness of the control system with respect to variable computing loads, control under time-varying sampling intervals (see section 1.7.2).

Although the assumptions based on constant sampling and constant and pre- fixed delays have been thoroughly studied, taking into account some realistic delay patterns and sampling jitters in practical operations brought new interesting results. The main objectives are to adapt computation constraints while preserving control objectives.

This section will include the notion of hard deadlines for control systems, then we will see the different effects of delays on real-time control systems in which some robustness issues are recalled.

Impact of time-delays on Control

One of the means to understand timing deviations consist of prospecting the sources of delays and jitters. Implementation-induced delays and jitters are inevitable in computer-controlled systems.

In this thesis, we will mainly focus on delays sources for centralized controller, while there exist subtleties for networked and distributed control systems.

In a given control system, latencies have several sources:

- Computation time: In older times, the computation times were quasi-constant and bounding the delays with the constant sum of times taken by the different instructions composing the software was enough to evaluate the value of the delay [Souyris 2005]. However, [Souyris 2005] states that the new challenging hardware render this methodology difficult to apply because of variability.
- Preemption: Most of the times, the CPU is shared between several activities (not all having to do with control tasks). The switching between tasks, the preemption and also the insertion of sporadic tasks generates jitter and asynchronous delays (and even sampling) from the control system point of view.

As we already know, one way to attack the problem consist of the WCET-based approach but the delay is maximized in this case and thus leading to poor control performance and poor schedulability performance (as see in section 1.6).

The most severe form of delay for the performance and stability of the system is time-varying delay.

Different problems happen when it comes to consider delays in a system loop. We will look more closely at time delays in section 2.4.1 with chapter 2.

Diverse methods

More methods have been presented in the context of computing-aware control. Some methods based on adaptive sampling methods consider online- reallocation of resources are proposed in [Seto 1996] , performance indices-oriented solutions (the performance index is computed online and not off-line) such as those proposed by [Cervin 2003b] , [Cervin 2006] , [Cervin 2006] and [Seto 1996], a robust online approach is provided by the H_∞/LPV (Linear Parameter Variant) [Robert 2010], ...

In this last method, he approach considers the sampling interval as a varying parameter of the system to get a H∞ based design, where the requested control performance also varies according to the expected sampling rate. The advantage is that the sampling period may vary at any time and speed inside predefined bounds while preserving the system's stability, and still have a coherent mathematical model to work with. This fact is important because with such a mathematical model, it is possible to demonstrate results theoretically, and then experimentally. According to our inquiries, some enhancements would be provided in terms of fault modeling. In fact, as in the Jitterbug tool, timing faults could be modeled as a Markov chain through probability density associated with a LPV model and robust control synthesis. This kind of fault model could be used to have a pattern of fault as an input of the model (taken from a processor characteristics' book or probabilistic experimentation for instance) and as output the behavior of the control system.

1.8 Conclusion of Chapter 1

In this chapter, safety-critical control system has been revisited in different aspects. The current state of art of system design and the different constraints that are related were also

recalled. The different motivations of this thesis were also recalled with the context of avionics systems and mainly within flight control systems. Then, safety- critical systems state of art has been proposed in order to derive the source of constraints, which are related to certification and safety models. It has been shown that the most stringent constraints derive from the certification side and the need to satisfy the regulations.

With older continuous-time systems, things were quite different. Thus, the effects and consequences of *computerization* was also studied in this chapter. This study has shown that the computer trend has led to new problematics, involving mainly safety. Sampling operations induce variability in the computer task execution and leads to give computers a time-varying property.

The variability that computers induce, has been for long, bounded by an upper bound which is the worst case execution time, leading to the notion of hard deadline. This value in itself is difficult to determine, as shown in Apppendix A, and even if there was a way for an evaluation of this quantity, the occurrence of this value (in terms of probability is so rare), that it becomes conservative to base the design of system upon this assumption.

Thus, a solution is called for to solve the problem of variability induced by the sampling, while using computer systems. Another problem to be solved is also the reduction of conservatism.

An direct method would be to consider jointly computer-science and automatic control, instead of working separately in cascade. This would lead to the methodology called control and scheduling co-design (which include computing-aware control and/or control-aware computing).

This idea takes back some points reviewed in this chapter about the problems of sampling period choice, requiring the dual consideration and trade-off between the control performance obtained by reducing the sampling period or implementation performance by choosing, instead, a larger sampling period.

In the sequel, this new idea will be developed, and a presentation of a weakened implementation which relaxes the hard real-time constraints will be exposed, as well as solutions proposed by real-time control theory to ensure the robustness and stability of the new implementation.

CHAPTER 2

Robust control under slackened real-time constraints

Contents

2.1 Introduction to Chapter 2

It has been exposed in Chapter 1, that computer-controlled systems are more and more ubiquitous by means of *X-by-wire systems*. In terms of applicability, computer-controlled systems and more especially those having real-time aspects cover a large spectrum going from very simple micro- controller-based systems to highly complex systems such as flight control systems.

During the last decades, the increasing demands and complexity have led to the emergence of new ways of approaching complex and safety-critical systems. It concerns analysis and conception methods, new languages, more and more powerful hardware on one side but also new standards and new regulations stregthening safety regulations and implementation constraints. In this Chapter, we will recall how these facts have led to the separation of concern between the involved communities (control and scheduling community), such that those communities became completely independent (as it is usually seen in the certification process). Indeed each community did not have to care about each others' issues. The previous chapter has also shown that the relation between performance and scheduling is difficult to find while using the traditional approaches of design and implementation. Some aspects of fault -tolerance and robustness have also been assessed.

In the present Chapter 2, an innovative solution to the previous problem statement is proposed.

This solution is argued to be robust enough to provide flexibility with respect to real-time control systems based on the current methods of designs and implementation. Another aim of this new methodology is to provide fault-tolerance. In fact, as explained in 1.6.1, designs and implementations of systems based on the paradigm of hard real-time is not robust and tolerant enough to accept timing faults. Such a property is currently non-existant in traditional computer-controlled system implementations based on the WCET.

The main challenge is to prove that the proposed solution provides the same safety and availability requirements with additional operational reliability and added value. From this perspective, the newly-presented "*weakened real-time approach*" encounters some problems, that requires the introduction of new domains of study such as the mathematical model of sampled-data system and time-delay systems.

In fact, the reason is simple: the more we want to reduce the allocated time, the more the system is prone to miss its deadline. But as soon as the system misses its deadline, it is necessary to handle this event in order to implement fault tolerance. Thus, there is need for a formal model to prove that the resulting system is safe and robust enough to continue its mission as required.

This chapter will then be divided into three sections:

- Section 2.2 recalls the motivation for a more flexible framework in the context of real-time systems design and implementation. Then the notion of weakened real-time system is given. As new solutions also lead to new problematics, the resulting new issues are also visited.

- Section 2.3 considers a concise study of the mathematical modelling of computer-controlled systems, while drawing the different problems and limitations that make

that this thesis' problematic is challenging. Classical analysis methods and tools will be exposed as well. The control objectives corresponding to weakened real-time systems are also formalized in this section.

- Section 2.4 is a state of art of time-delays systems. As the proposed approach in this thesis involve time-delay systems, it is interesting to formulate the relation between the models described in section 2.3 and the different delay formulations, as proposed by litterature. This section also uses classical tools such as LMI (Linear Matrix Inequalities) to formalize the proposed stability criteria.

- Section 2.5 gives results of the stability criteria tested on different academic examples in order to show the efficiency of the method so as to provide a comparison basis with existing results provided by the literature.

- Section 2.6 will draw a summary of this chapter.

2.2 Weakened real-time systems

Unlike, traditional implementation with mechanical machinery, computer-controlled systems implementation is supposed to provide more flexibility, more performance and much more efficiency (see section 1.5.3). However, as exposed in section 1.6, concerning problems statements, it has been seen, either from practice or from deep analysis that there are still improvements on this sector.

The hard real-time paradigm, presented in 1.6.1, shows the different limitations of current implementations. A new design and implementation co-design approach is the base of this solution proposal. This approach is based on a flexible, dynamic and interactive combination of computing-aware control and control-aware computing with the following features.

Some features can be listed such as:

- Reduction of the latencies in the processor.

- Relaxation the WCET paradigm.

- Robustness with respect to timing variations and uncertainties of the system parameters.

To improve the efficiency of embedded computers while preserving the control stability and performance, and relying on the robustness of feedback control laws, it is proposed to weaken the usual real-time constraints (commonly known as the worst-case-based assumption which has been already presented in section 1.6.1). The most difficult challenge in this objective is to ensure the same degree of safety and performance as in the initial design approaches. However, experiences and lessons learnt from past projects, have shown that the WCET-based approach is prohibitively rigid and completely conservative. The requirement to meet a worst case objective, that is considered as a rare event, leads to an

overestimation and an oversizing of the system resources (cooling, memory needs, mass, energy consumption...).

The idea is to spread fault tolerance ability through end-to-end component of the functional system. This end-to-end component is seen at the control loop level. Instead of looking at the timing problems at a fine-grain level as it has been the traditional case, for the software timing problems, it is now worth prospecting at a coarser-grain level which is the control loop.

In the classical case, a specification of all the functions embedded in flight control was provided in the form of a graphical formalism (SCADE, SAO, ...) [Goupil 2010]. These specifications are translated into compilable source code by automatic code generation tools to be directly implemented in the flight control computer system. Such tools has as input the functional specification and as output a C or assembly source code that can directly be embedded in the computer. Then this code is compiled to produce executable program which is analyzed in terms of timing in order to ensure that the computing platform has *enough computing resources* (which can be translated as "enough computation time") to handle the different software task. This show that the WCET analysis is done at a fine-grain level of software, tasks and computer-controller.

Truly, all these subsystems are part of a bigger system, which is the control system loop. And in order to achieve an acceptable required safety, performance and quality of service, this bigger system needs resources.

Most of the time, control loops usually run in nominal mode. Neither the execution resource parameters (available computing resources, actual execution time, slack time due to the scheduling algorithm...), nor the uncertainties which the process might be subject to, can be perfectly known or modelled off-line.

The resources required for the control loop cannot be modelled by the WCET for many reasons. For instance, in section 1.7.3 and mainly subsection 1.7.1, it has been shown that the hard deadline of a control system is not the same hard deadline considered in real-time software systems.

Another point to be cited would be the effects of delays in a control loop, presented in section 1.7.3. . In brief, from a control perspective, software deadlines are primarily used to bound the input-output latency of the controllers. Assuming that the sampling and the signal reconstruction take place, respectively, before task start and after task completion, figure 2.1 gives a relation between the response time, the latency and the deadline.

For controlled plants with unstable dynamics (plant that are naturally unstable), bounded latency makes some sense and even is absolutely necessary

in order to guarantee the stability of the closed loop system [Cervin 2001]. Nevertheless, the smaller the latency can be made, the better are the control performance and robustness that can be achieved. Thus, it is not justified to assign deadlines that are much larger or much tighter than what is dictated by stability considerations. It means that it is necessary to choose the deadlines with stability objectives first.

However, more severe for control performance is the jitter (time-varying delays). In [Buttazzo 2007], a comparative assessment of delays and jitters has been done. The objective of the paper was to compare the performance of different schemes. Among evaluated scheme include the reduction of jitter by forcing the execution of input arrival and output

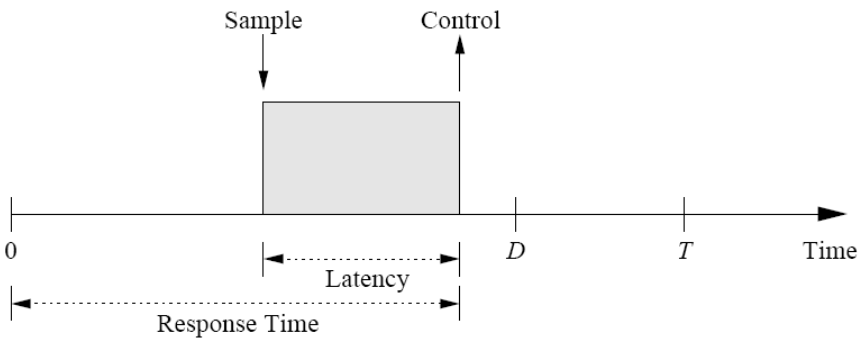

Figure 2.1: Relation between Deadline (D), response-time and input-output latency for control tasks

sending at the boundaries of the sampling period, thus trading jitter for pure constant delay as it is the case in the fashion of worst case execution time approach. Conclusions have been drawn that such approach is considered to be a *"safe"* choice of implementation for systems that are considered non-robust (e.g: with very small phase and delay margins). However, for other kinds of systems, it is considered as the worst approach in terms of control performance issues and this approach would not perform any better unless in systems which are severely under-dimensioned.

We believe that our idea combined with the usual safety approaches (see 1.4.5) as it is clearly explained in [Goupil 2010], will allow to reduce conservatism in the implementation, to weaken the legendary hard real-time constraints, while achieving at least the same dependability, performance and availability objectives.

Following [Shin 1995, Cervin 2000, Simon 2009], what is called for is a procedure that allows to specify and evaluate a controller dependability range and performance enabling systematic application and providing objective results that lend themselves to a formal analysis.

2.2.1 Weakened scheduling scenario proposals

On one hand, as already observed and reported (e.g. [Cervin 2002]), it is likely that a robust feedback control systems can keep stability despite timing disturbances such as jitter and occasional data samples losses. However, a formal approach is necessary and is called for. Indeed, the approaches proposed in [Cervin 2002] is not formal enough to be applied in safety critical systems such as avionics.

On the other hand, the previously presented weakened analysis framework is supposed to provide an analytical process to be applied on the control system in order to justify the choice of the allocated time for control tasks.

Several scheduling patterns are then proposed in the sequel, for control tasks executions under weakened real-time constraints. All these scheduling scenario rely on the allocation of periodic time slots T_{slot}, which is smaller than the WCET of the control tasks (Figure 2.2).

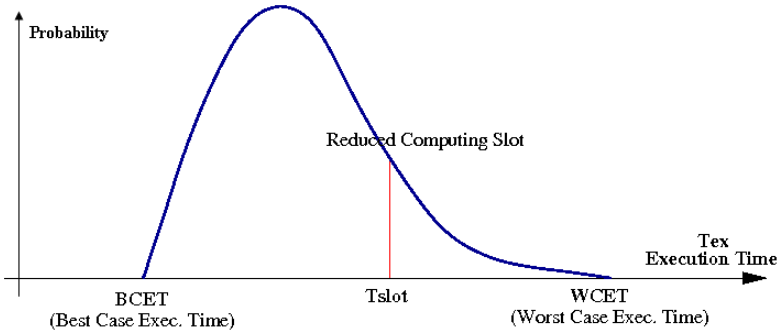

Figure 2.2: Reduced computing slot

To keep the context as close as possible to current practice, the control tasks' triggering is assumed to be done periodically (although asynchronous sampling might be accounted as presented in [Seuret 2011a]) by the occurrence of equidistant sensors inputs. Reducing the time slot to be smaller than the WCET of the task allows to the reduction of the number of wasted CPU cycles at the price of occasional deadlines misses (see 1.6.1). However, it also induces potential reductions of both the control period and of the Input/Output latency, leading to performance improvements.

It is expected that the disturbances introduced within the control loops by the timing faults would be compensated by the gains obtained by reducing the sampling intervals and the overall delay.

However, this affirmation must be considered with caution, more especially, for the case where the system is submitted to timing perturbation and state perturbation. That is why, later in this thesis, uncertainty models have been integrated to the analytical model, so as to permit to consider both uncertainties.

1 Abort The sensors data are still expected to occur at a fixed period T, and their occurrence trigger the control tasks. The time slot allocated to a given task $T_{slot} < WCET$ is now smaller than the worst case. As usual the control signal is sent to the actuators at the end of the slot, i.e. $U(x(s_k))$ is sent at time $s_k + T_{slot}$, and the delay is equal to T_{slot}

$$\forall t \in [s_k + T_{slot}, s_{k+1} + T_{slot}[, \quad U = U(x(s_k)).$$

But now it may happen that occasionally a control task deadline is missed : in that case it is proposed to stop the current computation, hold the current value of the control signal for the next period and start a fresh computation with the next sensor value. In that case, the control signal $U(x(s_k))$ is hold for one extra period, i.e. if the deadline miss occur at time $s_k + T_{slot}$:

$$\forall t \in [s_k + T_{slot}, s_{k+2} + T_{slot}[, \quad U = U(x(s_k))$$

and for N consecutive deadline misses:

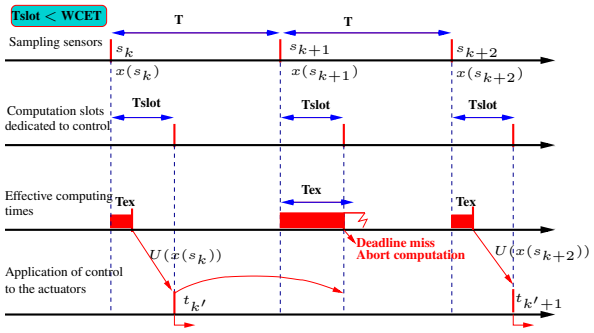

Figure 2.3: Abort scheduling pattern

$$\forall t \in [s_k + T_{slot}, s_{k+N} + T_{slot}[, \quad U = U(x(s_k))$$

In other words a newly computed control signal is sent to the actuators at non - equidistant instants $t_{k'}$ only if the control computation has not been successfully carried out:

$$t_{k'} = s_k + T_{slot} \quad \text{if} \quad T_{ex} \leq T_{slot},$$

where k' is a positive integer representing the total number of sampling intervals between an input capture and a successful actuation.

Then, the control input may be aperiodic since the difference between two sampling instant $t_{k'+1} - t_{k'}$ is time-varying but bounded by T and NT. Hence $t_{k'+1} - t_{k'} = \alpha T$, where the integer $\alpha \in [1, ..., N]$ and the aperiodic sampling is determined by the couple (T, N).

The value of N is computed by the LMIs in 2.4.3. It defines the number of successive admissible deadline violation. If it is exceeded, then a specific overrun management system shall be activated. This approach is definitely less contrained than the worst-case-based approach.

2 Abort' Again, a time slot $T_{slot} < WCET$ is allocated to the control task, but the system's period is now also reduced by the same value, while the time remaining for computing other activities remains T_{others} as in the initial scheme. In that case control improvement is expected from both the latency and sampling period reduction (Figure 2.4) .

where T_init is the initial sampling interval that is used in the WCET-based case.

From the computing point of view, these two schemes might not be as simple to implement, as they seem, at first glance. Indeed, in case of deadline miss it is necessary to instantaneously abort the running instance of the control task and cleanly restart a new instance with the new sensors data, and with cleanly re-initialized internal variables, filters and other related data structures. This scheduling pattern may be difficult to implement,

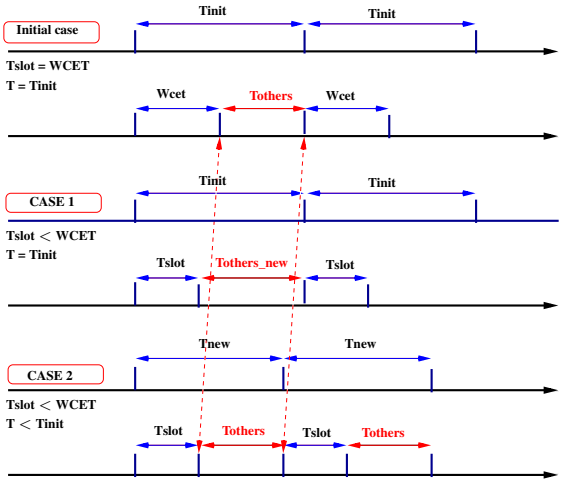

Figure 2.4: Abort and Abort' scheduling cases

e.g. in Posix threads written in C where a thread's function (used to implement a periodic control task) cannot be easily aborted and immediately restarted.

The following scheduling patterns are in such cases easier to carry out.

3 Skip For the nominal case this pattern is similar to Case 1, where the period of the control task is kept equal to the initial case. However, in case of deadline miss, rather than aborting the running instance, the running task continues its computation until reaching its completion. The execution of the task which begun during slot T_{slot_k} continues during the following slot allocated for this task ($T_{slot_{k+1}}$), and possibly on the following allocated slots $T_{slot_{k+n}}$, until completion. The last computed control signal is hold on the actuator input until a new one has been produced and sent to the actuators, as in Figure 2.5 where $U(x(s_k)$ is hold until $s_{k+2} + T_{slot}$ due to the deadline miss during slot $T_{slotk+1}$. Note that, in the case given in figure 2.5 the control signal sent at time $s_{k+2} + T_{slot}$ is $U(x(s_{k+1}))$ computed using the sensor's value s_{k+1} received one period and one slot before.

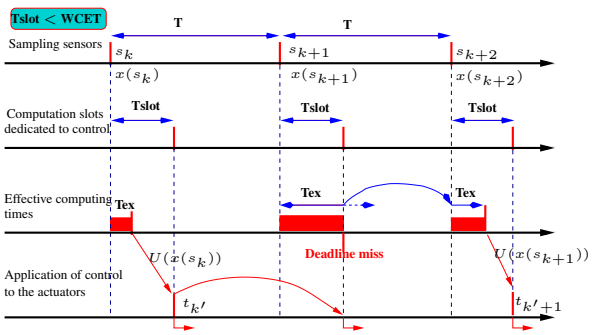

Figure 2.5: Skip scheduling pattern

4 Spare Note that some spare computing time can be reserved inside the T_{others} slots. Usually this spare time is used for housekeeping activities, but it could also be used to handle the extra computation cycles needed to fulfill the computation of an overrunning control task which slides to the spare slots (Figure 2.6). In that case, if the spare slot is long enough to finish the running computation, the control signal can be sent either just at the end of the spare slot (to minimize the I/O latency) or at the end of the period (to minimize the output jitter). The remaining computations may slide until the next available spare slots until completion (or In both cases the nominal scheduling pattern can be recovered at the next received sensor signal.

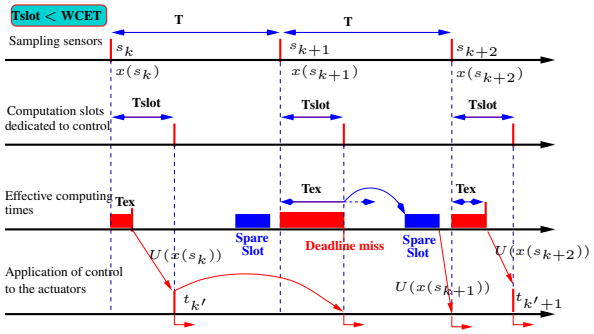

Figure 2.6: Slipping on a spare slot

5 Skip' Combining the previous cases 2 and 3, it is proposed to reduce the control period according to the reduction of T_{slot} (as in the **Abort'** case), and to adopt the **Skip** overrun handler in case of deadline miss.

6-> Just-in-time In all the previous cases, the newly computed control signal is sent to the actuator waiting the end of the allocated T_{slot} to allow for minimal jitter and equidistant sampling at the actuator's input. However, the control signal can also be sent to the actuator immediately after the computation completion. In that case it is expected (e.g. following [Buttazzo 2007]) that the disturbance due to the output jitter can be compensated by the average lowered value of the I/O latency, therefore probably increasing the control performance. This **Just-in-time** control sending strategy can be associated with any of the previously described scheduling patterns and overrun handlers.

...

Indeed the combination of possible scheduling schemes (including fully aperiodic or event-based control laws) with various overrun handlers is potentially very large. The right combination is the result of a trade-off to be evaluated for each case study between control algorithms, controlled plants, execution resources, operating systems capabilities and certification constraints...

2.2.2 Robustness considerations

The design of critical systems must satisfy requirements, specifications and certification levels. Robustness is a general concern that grows with system complexity. For instance, it is known that small task core execution time modifications in systems with complex performance dependencies can have drastic non-intuitive effects on the overall system performance, and might lead to constraint violations. [Hamann 2006] claims that robustness evaluation using simulation is a tedious tasks and practically impossible for the reason that simulation models do not support many of the possible property changes (for instance, increased processor execution times or modified communication volumes). This thesis proposes an interesting formal approach to robustness of embedded real-time systems with definition of robustness metrics. However the present work uses recent developments and moreover is more dedicated to practical industrial problems.

Robustness in control usually deals with the plant's parameters uncertainties, but in the present case the insensitivity or adaptability w.r.t. timing deviations from the theoretical pattern, such as jitter or deadlines misses, is also investigated. A feedback control system can be even robust enough to tolerate missed samples, e.g. in [Felicioni 2010], where selective data dropping is applied to lighten the computing and networking burden while preserving closed -loop stability. The interesting point is that a feedback control system which is robust with respect to the plants parameters uncertainties is also robust, to some extent, w.r.t. timing deviations. Hence a feedback control system is not as hard as it is often considered in the literature, but should be better considered as *weakly-hard*, i.e. able to tolerate specified timing deviations without leaving its requested performance [Bernat 2001].

2.2.3 Challenging implementation problems

Within the presented framework, it is indeed obvious, that things are not so simple. Different problems arise. In the traditional (and most current methods), control laws are designed and sent to computer science community to be implemented into hardware and software. These operations are done completely within a context of separation of concern.

The fact of reducing the task allocated time, from the worst-case execution time to a smaller time T_{slot} will definitely lead to an occasional deadline miss. In fact, according to figure 2.2, while allocating the time slots to the control tasks for their execution, it is seen that timing violation is a rare event. However, when reducing this allocated slot to a muh smaller time as T_{slot}, this event is no more considered as a rare event and in terms of probability, becomes more frequent. From the computer science perspective, a processor time has been gained by reducing the allocated time for the task, while from the perspective of control theory, performance is enhanced due to the reduction of sampling period and the overall latency.

However, in terms of safety of execution, a specific deadline overrun management system shall be considered and the main difficulty is to find ways to cope with the combination of constraints from both the side of computer science and classical control theory.

Thus, advances in automatic control theory are needed. The domain that need to be involved in the problem resolution consist of the sampled-data theory and time-delay the-

ory. These tools will give the necessary tools to model and solve the problems of sample loss while still considering latency within systems. These tools will provide formal ways to implement the framework proposed in Annex B and to prove that the system is safe and robust.

Thus, in the sequel, a presentation of sampled-data models and time-delay system is considered necessary.

2.3 Dynamic models of computer-controlled systems

2.3.1 Systems under consideration

Consider the linearized system representing a plant with a sampled and delayed input.

The following model include aggregated uncertainty modelling for the system parameters:

$$\begin{aligned} \dot{x}(t) &= [A + \Delta_\mu A(t)]\, x(t) + [B + \Delta_\mu B(t)]\, u(t) \\ y(t) &= [C + \Delta_\mu C(t)]\, x(t) \end{aligned} \tag{2.1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ represent the state variable and the input vector. The matrices A and B are constant and of appropriate dimension. The matrices $\Delta_\mu A$ and $\Delta_\mu B$ represent the uncertainties of the model which can be constant or time varying. The (time-varying) uncertainties are given in a polytopic representation:

$$\Delta_\mu A(t) = \mu \textstyle\sum_{i=1}^{M} \nu_i(t) A_i,$$

$$\Delta_\mu B(t) = \mu \textstyle\sum_{i=1}^{M} \nu_i(t) B_i,$$

$$\Delta_\mu C(t) = \mu \textstyle\sum_{i=1}^{M} \nu_i(t) C_i,$$

where M corresponds to the numbers of vertices. The matrices A_i, B_i and C_i are constant and of appropriate dimension. The scalar $\mu \in R$ characterizes the size of the uncertainties. Note that when $\mu = 0$, no parameter uncertainty is disturbing the system. However the greater the μ , the greater the disturbances. The functions $\lambda_i(.)$ are weighted scalar functions which follow a convexity property, ie. for all $i = 1, .., M$ and for all $t \geq 0$:

$$\nu_i(t) \geq 0, \quad \sum_{i=1}^{M} \nu_i(t) = 1$$

The control law is a piecewise-constant static state-feedback of the form:

$$u(t) = Kx(t_{k'} - T_{slot}), \quad t_{k'} \leq t < t_{k'+1},$$

where the gain K in $\mathbb{R}^{n \times m}$ is given.

These instants $t_{k'}$ represent the instants where the control input is updated. According to the previous problem formulation, the time interval between two successive sampling instants is time-varying and characterized by:

$$T_{k'} = t_{k'+1} - t_{k'} = \eta(k')T \tag{2.2}$$

where $\eta(k')$ is a time-varying integer which belongs to the interval $[1\ N]$. The closed loop system is thus rewritten as:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t_{k'} - T_{slot}) \quad \forall t \in [t_{k'},\ t_{k'+1}] \tag{2.3}$$

where $\bar{A}(t) = A + \Delta_\mu A(t)$ and $\bar{B}(t) = B + \Delta_\mu B(t)$.

2.3.2 Identification of the difficulties

Considering the closed-loop systems defined in equation (2.3), the main difficulties for the stability analysis of this class of systems comes from three phenomena listed below

- A constant delay T_{slot}
- An aperiodic sampling characterized by T
- Time varying uncertainties in the systems dynamics ν_i

Each of these difficulties have been intensively studied in the literature separately. Here the novelty of the thesis aims at considering all these phenomena together in a stability analysis.

2.3.3 Control Objectives

Considering a plant, a known distribution of the execution times and a weakened real-time scheduling pattern, the problem to solved can be informally stated as follows:

- Find N, the maximum value of consecutive sample loss before loosing the closed-loop stability;
- Find an adequate value of T_{slot} to fulfill a given trade-off between the control and the computing performances.
- Evaluate the weakly-hard closed-loop robustness with respect to the plant's parameters uncertainties.

In order to analyze the stability of such class of systems, one has to look into the framework of Automatic Control. Indeed existing results on the stability analysis of time-delay systems, sampled-data systems and robust analysis fit perfectly with the problem described above.

Nowadays time-delays and sampled-data systems are very popular in the field of Automatic Control. One of the reason of its popularity comes from the fact that these two phenomena arise in the context of Networked Control Systems (NCS) [Hespanha 2007], [Zampieri 2008] and the references therein.

This area of research is concerned by the stabilization of processes in a situation where the controller and the process are looped over a network and digital hardwares and softwares. The main issues are to take into account the induced dynamics generated by the execution delays (i.e. computation and networking). As it was mentioned earlier in the document, ones of the most considered effect of a network are the time-discretization of the information (related to sampled-data systems) and the induced delay which is due to the transmission process of the time to perform the control computation (related to time-delay systems).

In order to facilitate the understanding of the thesis, we will present some basic considerations on both problems.

2.4 State of the art in networked control systems

Recall the system under consideration in the thesis. As it is noted in the previous section, this problem can be rewritten as a problem of ensuring stability of a networked control systems of the form given by:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t_{k'} - T_{slot}) \quad \forall t \in [t_{k'},\ t_{k'+1}] \tag{2.4}$$

In order to have a better understanding of the difficulties induced by this class of systems, we will consider the following situations

(a) "$T \to 0$": *Systems with constant delay*. In this situation, the systems dynamics (2.4) become:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t - T_{slot}) \tag{2.5}$$

(b) "$T_{slot} \to 0$" : *Systems with aperiodic sampling*,

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t_{k'}) \quad \forall t \in [t_{k'},\ t_{k'+1}] \tag{2.6}$$

(c) *Systems with constant delay and aperiodic sampling*:

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t_{k'} - T_{slot}) \quad \forall t \in [t_{k'},\ t_{k'+1}] \tag{2.7}$$

2.4.1 Systems with delays

Delays generally appear in the modelling of physical plants. Many types of delays can be distinguished, according to figure 2.7, in a simple control loop:

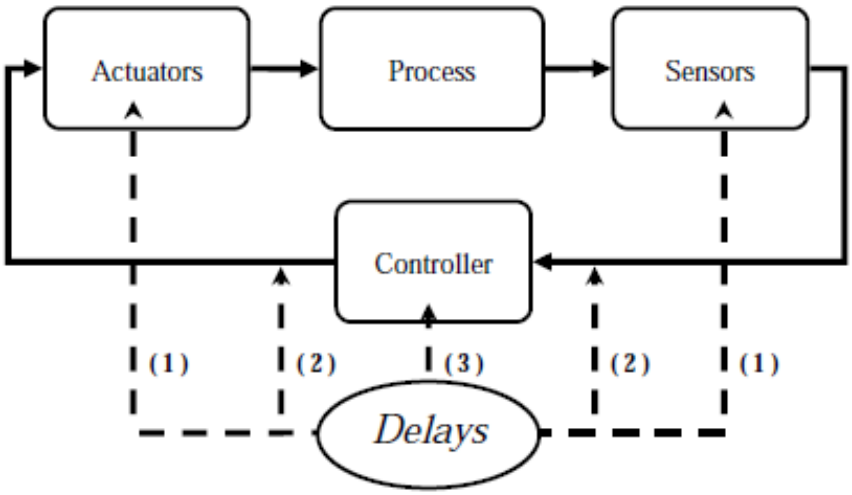

Figure 2.7: Time-delay location in a control loop

Time-delay generally occurs for different reasons and from different sources. One of these sources can be the nature of the system or the way it works. For instance, in a flight control computer, the control laws are being computed and delays can appear due to the task scheduling (preemption for instance) or due to the internal functions of the target processor (cache memories effects, pipelines,...). Another source of delays is the transport delay, for some material to travel through a given system. An example can be the actuation delay, where the time from the actual computation of the control signals and their arrival at the actuator. This can be comprised as another type of delay: communication delays. In fact, time is needed for the signal to travel from point to point as shown in figure 2.7.

The interests of a number of community of researchers [Richard 2003, Niculescu 2001, Gu 2003, Sipahi 2011] comes from the fact that delays modify the structure, the dynamics and the stability properties and the performance of control systems. Here, we are considering this framework since time-delays appear in real-time systems as a critical factor as shown in the previous section.

2.4.1.1 Basic definitions of time-delay systems

From mathematical perspectives, time-delay systems, also called *systems with after-effects* or *dead-time*, *hereditary systems*, *equations with deviating arguments* or *differential-difference equations*, belong to the class of *Functional Differential Equations* (FDE).

In a general way, a time-delay system is a class of dynamical system represented by differential equations, in some unknown functions (see [Niculescu 2001]) and some of its derivatives, evaluated at arguments which are distributed over some intervals in the past.

A general model of a time-delay systems is shown, as presented in [Richard 2003, Kolmanovskii 1999, Gu 2003, Niculescu 2001]:

$$\begin{cases} \dot{x}(t) &= f(t, x_t, u_t), \quad \forall t \geq t_0 \\ y(t) &= g(t, x_t, u_t) \end{cases}$$

where the initial conditions need to be specified as:

$$\begin{cases} x_{t_0} &= \phi(\theta) \quad \text{where} \theta \in [t_0 - h, t_0] \\ u_{t_0} &= \zeta(\theta) \quad \text{where} \theta \in [t_0 - h, t_0] \end{cases} \tag{2.8}$$

and where the function x_t represent the state vector at time t and the function u_t the input vector at time t and are defined by:

$$x_t = \begin{cases} [-h, 0] &\longmapsto \mathbb{R}^n, \\ \theta &\longmapsto x_t(\theta) = x(t + \theta), \end{cases} \tag{2.9}$$

$$u_t = \begin{cases} [-h, 0] &\longmapsto \mathbb{R}^m, \\ \theta &\longmapsto u_t(\theta) = u(t + \theta), \end{cases} \tag{2.10}$$

The initial conditions $\phi(.)$and$\zeta(.)$ must be prescribed as piecewise functions of the initial instant, t_0, and the delay h and defined by:

$$\phi : [t_0 - h, t_0] \longmapsto \mathbb{R}^n$$

and

$$\zeta : [t_0 - h, t_0] \longmapsto \mathbb{R}^m.$$

It has to be noticed that the class of differential equations described above differs from the case of ordinary differential equations in many aspects. First, as it was described earlier, the initial conditions of delayed differential systems is a function defined over an interval of the form $[-h, 0]$. Since the set of functions from $[-h, 0]$ to $\mathbb{R}^n$ is of infinite dimension, it implies that the solutions of this differential equations also are of infinite dimension.

A by product of the previous remark concerns the state of such differential equations. The state of a time-delay system is not only the vector x considered at a given instant t but the function x_t which gathers all the vectors $x(t)$ where t belongs to the interval $[-h, 0]$.

Coming back to the problem exposed in the previous section, we will further consider a time-delay system of the form

$$\dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t - T_{slot}) \tag{2.11}$$

2.4.1.2 General assumptions on the time-delay functions

Constant delays: This corresponds to the simplest and initially studied case. This can be a good model that works fairly well in practice. One way to achieve a constant delay, in practice, is the introduction of timed buffers after each computation (or transfer). By overestimating the time buffer size to a longer value than the estimated delay case, the

delay can be regarded as a constant value. A perfect illustration of this case is proposed in figure 1.14. This case can also be called, the case of overestimated constant delays. This model can be assumed as the existence of $T_{slot} > 0$, such that in equations 2.8, the functionals are valid in the interval $[-T_{slot}, 0]$.

If the systems does not contain uncertainties, relevant and accurate stability analysis can be obtained using a frequential approach. For instance the reader may refer to [Sipahi 2011] to have a good overview of the recent advances in the domain.

However in the situation of uncertain of time-varying parameters, the frequential approach fails ensuring stability. In this situation a continuous-time approach is more convenient as it is be shown in the next section.

Before entering into stability analysis based on a continuous-time approach, let us remark that other delay functions has been considered in the literature.

- **Bounded time-varying delays:** The assumptions on constant delay is too conservative in practical applications. It is seldom verified and not justified to oversize resources (to cite an example, applying this strategy in embedded avionics computers lead to overweight and increased power consumptions). Then the notion of bounded and time-varying delay has been introduced. It consist of allowing the delay to vary from sample to sample but within an upper maximum value not to be exceeded. This type of delay is defined as a function $h(t)$, for which, there exists an upper value $h_2 > 0$ and such that:

$$0 \leq h(t) \leq h_2 \tag{2.12}$$

  Related works on the analysis of systems subject to this type of delay can be found in [Busenberg 1991]. This delay can be known in advance (meaning that each discrete value of the delay can be known, as proposed in the section on *Discrete-Time Delays* in [Busenberg 1991], where the delays are modelled by a time-varying delay ht) but whose values are taken out from any finite number of discrete delays $h_1, \cdots, h_2$ unknown and uncertain as shown in [Kao 2007]).

- **Interval time-varying delays or non-small delays:** This type of delay can be derived from the previous type. In fact, results on the previous type of delays admit that delays can vary from $[0, \tau_2]$. However, this assumption needs to be tightened in order to fit physical reality. When looking at the example of the computation time in a flight control system, allowing the delay to take the 0 value, means considering that the computation has been made instantaneously, which does not model the subtleties of the reality. Thus, tightening the delay function to exclude the zero value is relevant. The conditions on the delay become as follows: There exist two positive scalars τ_1 and τ_2 such that:

$$0 < \tau_1 \leq \tau(t) \leq \tau_2 \tag{2.13}$$

  Ways to tackle with this type delay consist of decomposing the delay $\tau(t)$ into two parts: a nominal part and a residual part, which is considered as a perturbation with

respect to the nominal delay. In [Fridman 2004a], stability conditions of such systems, obtained from a Lyapunov-Krasovskii approach, are given in terms of Linear Matrix Inequalities.

- **First Derivative-constrained delays:** This type of delay allow the designer to take into account the variability of the delay, which is modelled by the speed of variation and thus the conditions require that the delay function can vary arbitrarily. An additional condition can then be inserted in the delay specification. This condition is a condition defined on the derivative of the delays function, $\dot{\tau}(t)$ as:

$$\dot{\tau(t)} \leq d < 1 \tag{2.14}$$

This condition enforces the causality of the function $f(t) = t - \tau(t)$ such that this function always grows slower than the time t.

2.4.1.3 Stability of time-delay systems

This section gives an overview of the theoretical approaches concerning the stability analysis of time-delay systems. The scope of this thesis will include only time-domain approach and the second method of Lyapunov.

The second method of Lyapunov

Consider the following model of time-delay system:

$$\begin{array}{rl} \dot{x}(t) & = f(t, x(t), x_t), \\ x_{t_0}(\lambda) & = \phi(\lambda), \ \ pour \ \ tout \ \ \lambda \in [-\tau, 0] \end{array} \tag{2.15}$$

where x_t denotes a functional as Supposing that equation 2.15, has a unique solution and an equilibrium state $x_t = 0$ (If the equiibrium is non-zero, an equivalent condition can be achieved by coordinate change).

The second method of Lyapunov is based on the existence of a function of the state variable x_t, denoted V, such that V is positive definite along the trajectories of equation 2.15 ans such that $\frac{dV}{dt} < 0$, while $x(t) \neq 0$.

This method, is only valid for some reduced classes of time- delay systems. The reason is that $\frac{dV}{dt}$ depends on the variable x_t. It is then very hard to apply this approach to the stability of the general case of time-delay systems.

Then, two extensions of the second method of Lyapunov have been provided especially for the case of functional differential equations.

In the case of ordinary differential equations (i.e. without delays), a candidate for a Lyapunov function is of the form $V = V(t; x(t))$ which only depends on the current "position" (the current state of the system).

In the retarded case, two approaches are possible: $V = V(t; x(t))$ or $V = V(t; x_t)$. Practically, the first case leads to difficulties related to the fact that the derivatives are

dependant on past instants. The second case report this difficulty in the construction of the functional.

These two approaches have been proposed for the retarded cases described with functional differential equations: the Lyapunov-Razumikhin method and the Lyapunov-Krasovskii method.

In this thesis, we only consider the Lyapunov-Krasovskii method, which is based on a functional of the form $V = V(t, x_t)$, instead of a simple function V.

The Lyapunov-Krasovskii method

This method is an extension of the Lyapunov theorem for functional differential equations. It consist of the research of a *functional* of the form $\mathcal{V}(t, x_t)$, which is a decreasing functional along the trajectories of 2.15. Similarly, $\mathcal{V}$ is called a functional because it depends on the delayed state vector x_t, which is a vector function considered in the interval $[t - \tau, t]$.

Theorem 1 *[Kolmanovskii 1996]*

Let u, v and $w : \mathbb{R}_+ \longmapsto \mathbb{R}_+$, be continuous and increasing functions, such that $\forall\, \lambda > 0$, $u(\lambda) > 0$ and $v(\lambda) > 0$ and $u(0) = v(0) = 0$. Assuming that the vector field f (in 2.15 is bounded for bounded values of its arguments. Then there exists a functional $\mathcal{V} : \mathbb{R} \times C \longmapsto \mathbb{R}_+$ such that:

1. $u(\|\phi(0)\|) \leq \mathcal{V}(t, \phi) \leq v(\|\phi\|)$
2. $\dot{\mathcal{V}}(t, \phi) \leq -w(\|\phi(0)\|)$ *for all trajectories of 2.15*

Then the solution $x_t = 0$ of 2.15 is uniformly stable.
Moreover, if $\forall \lambda > 0, \quad w(\lambda) > 0$, the solution $x_t = 0$ of 2.15 is uniformly aymptotically stable.

If $\mathcal{V}$ satisifes for the following conditions:

1. $u(\|\phi(0)\|) \leq \mathcal{V}(t, \phi) \leq v(\|\phi\|)$
2. $\dot{\mathcal{V}}(t, \phi) \leq -w(\|\phi(0)\|)$, *for* $\forall t \geq t_0$ *and* $w(\lambda) > 0, \quad \forall \lambda > 0$
3. *$\mathcal{V}$ is Lipschitz, with respect to its second argument.*

then, the solution x_t of 2.15 is exponentially stable and the functional $\mathcal{V}$ is called a Lyapunov-Krasovkii functional.

COMMENT: In the previous theorem , the derivative $\dot{\mathcal{V}}(t, \phi)$ refers to the derivative in the sense of Dini, that is to say :

$$\lim_{\varepsilon \to 0^+} sup \frac{\mathcal{V}(t + \varepsilon, x_{t+\varepsilon}) - \mathcal{V}(t, x_t)}{\varepsilon}$$

The main idea of the Lyapunov-Krasovskii approach consist of determining a positive definite functional $\mathcal{V}$, whose derivative functional $\dot{\mathcal{V}}$, considered along the system trajectories is negative- definite. However, a pain point in this approach is the design of such functional $\mathcal{V}$ when it exists. Classical form of such functional $\mathcal{V}$ is given in [Kolmanovskii 1996] (section 2.2.2, page 24):

$$\begin{aligned}\mathcal{V}(t,\phi) =&\phi^T(0)\ P(t)\phi(0) + 2\phi^T(0)\left(\int_{-\tau}^{0} Q(t,\sigma)\ \phi(\sigma)d\sigma\right)\\ &+\int_{-\tau}^{0}\int_{-\tau}^{0}\phi^T\left(\sigma\ R(t,\sigma,\rho)\phi(\rho)d\sigma\ d\rho\right) + \int_{-\tau}^{0}\phi^T(\varsigma)S(\varsigma)\phi(\varsigma)d\varsigma \end{aligned} \tag{2.16}$$

where, P, Q, R and S are quare matrix functions of dimensions $n \times n$. $P(t)$ and $S(\varsigma)$ are symmetric and positive definite matrices, while R satisfies $R(t,\sigma,\rho) = R^T((t,\sigma,\rho)$. It assumed that each matrix element is bounded and that their derivatives are piecewise continuous functions.

In practice, the determination of these functionals set difficult problems due to the time-varying aspects of the matrices. It is preferable to restrain to the use of functionals in which the matrix functions P, Q, R, and S are constant. Under such restriction, the Lyapunov-Krasovskii functional becomes:

$$\begin{aligned}\mathcal{V}(t,\phi) =&\phi^T(0)\ P\phi(0) + \int_{-T_{slot}}^{0}\phi^T(\sigma)\ S\ \phi(\sigma)d\sigma\\ &+ 2\phi^T(0)\int_{-T_{slot}}^{0} Q\ \phi(\sigma)d\sigma \qquad (2.17)\\ &+\int_{-T_{slot}}^{0}\int_{-T_{slot}}^{0}\phi^T(\sigma)\ R\ \phi(\sigma)d\sigma\ d\rho \qquad (2.18)\\ &+\int_{-T_{slot}}^{0}\int_{\theta}^{0}\dot{\phi}^T(\sigma)\ R\ \dot{\phi}(\sigma)d\sigma\ d\theta \qquad (2.19)\end{aligned}$$

This class of Lyappunov-Krasovksii functionals has been widely used in the literature in order to assess stability of the system for the delay T_{slot}.

2.4.2 Asynchronous sampled-data systems

In this section we consider the problem of stability analysis of sampled-data systems. This class of systems is governed by the following dynamics:

$$\dot{x}(t) = A(t)x(t) + B(t)Kx(t_{k'}) \quad \forall t \in [t_{k'},\ t_{k'+1}] \tag{2.20}$$

Sampled-data systems have been extensively studied in the literature [Chen 1995, Fridman 2004b, Fujioka 2009, Zhang 2001a, Zhang 2001b] and the references therein. It is now reasonable to study the guarantee of the robustness of the solutions of the closed-loop system under periodic samplings. However the case of aperiodic samplings still leads to several open problems. This corresponds to the realistic situation where the difference between

two successive sampling instants is time-varying. Several articles drive the problem of time-varying periods based on a discrete-time approach [Suh 2008,Oishi 2009,Hetel 2006].

In this framework the dynamics of the systems are rewritten in term of the discretized version of the system.

Indeed, using the ZOH integration method shown in equation C.1 and C.3, we obtain:

$$x(t_{k'}+\tau) = \tilde{A}(\tau)x(t_{k'}) + \tilde{B}(\tau)Kx(t_{k'}), \tag{2.21}$$

where

$$\tilde{A}(\tau) = e^{\bar{A}\tau} \text{ and } \quad \tilde{B}(\tau) = \int_0^\tau e^{\bar{A}(\tau-\theta)}d\theta\bar{B}$$

Using this framework, a stability analysis can be performed by exploiting the properties of the exponential matrices. The objectives are to create convex hull which depends on the sampling interval $t_{k'+1} - t'_k$ and to ensure stability for all possible combinaison of matrices which belongs to this convex hull. This method leads to very interesting results with a very low degree of conservatism. However extensions to uncertain or time-varying systems is hard because the uncertainties and the time-varying terms appears in the exponential matrices. Thus defining a convex hull which encloses all the possible uncertainties on the systems and on the sampling periods is a really difficult problem.

On the other side, an input delay approach was provided in [Fridman 2004b]. The main interest is to rewrite a sampled-data systems into a continuous-time systems with a particular time-delay function. The admissible discrete control law can be modelled as a delayed continuous control law, as follows:

$$\begin{aligned} \forall t \in \left[{t_k}', t_{k'+1}\right[, u(t) = u_d(t_{k'}) = u_d\left[t-(t-t_{k'})\right] = u_d\left[t-\tau(t)\right], \\ \text{and } \tau(t) = t - t_{k'} \end{aligned} \tag{2.22}$$

in which u_d is the discrete control law and $\tau(t)$ is the piecewise continuous time-varying delays function, whose derivative is $\dot{\tau}(t) = 1$, for $t \neq t_{k'}$. Moreover, it is assumed that the delay $\tau(t)$ is always smaller than the sampling period $t_{k'+1} - t_{k'}$. To illustrate this method of modelling, figure 2.8 shows the sampling of a continuous-time signal at two different rates $T = 0.2s$ and $T = 0.5s$ and the associated sampling delays.

This formulation was already introduced in [Fridman 1988] but the first stability analysis was developed later by [Fridman 2004b]. The main interest of such method remains in the fact that some previously results on the stability analysis of systems subject to time-varying delays can be also applied to systems with a sampled input. More particularly, it allows using the Lyapunov-Krasovskii theorem which can be found in [Kolmanovskii 1999, Gu 2003, Seuret 2006].

Improvements are provided in [Fujioka 2009, Mirkin 2007], using the small gain theorem and in [Naghshtabrizi 2008] based on the analysis of impulsive systems. Recently

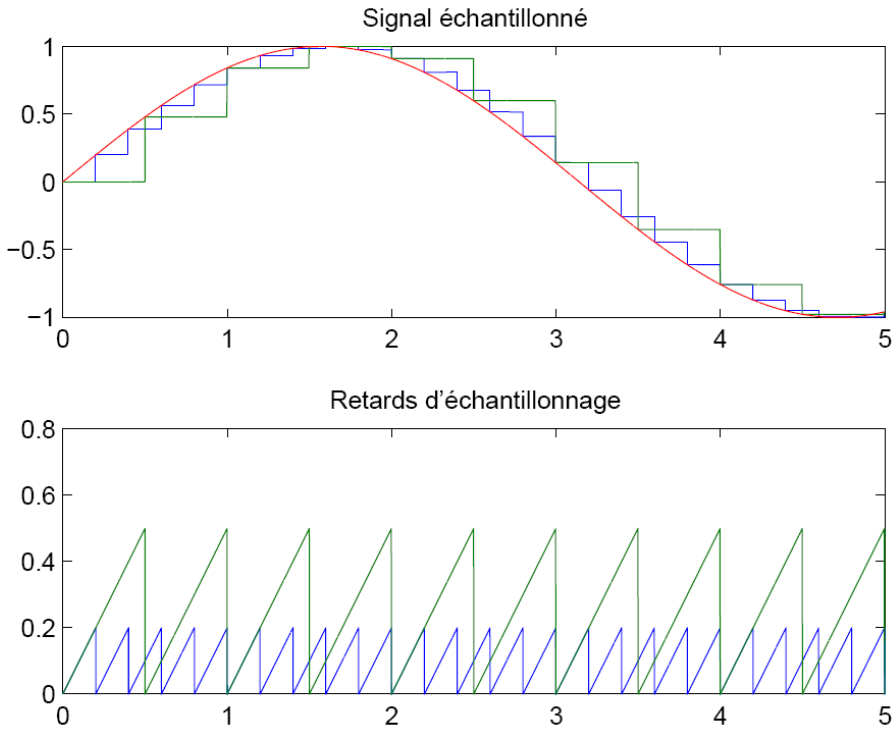

Figure 2.8: Illustration of a sampling delay

[Fridman 2010,Liu 2010,Seuret 2009] refined those approaches and obtained tighter conditions. These approaches are relevant to the problem considered in this thesis, because they cope with time-varying sampling periods as well as with uncertain systems in a simpler manner. Nevertheless, these sufficient conditions are still more conservative than discrete-time approaches.

More recently a novel approach was introduced in [Seuret 2012] . The main interest of this method is the definition of a new class of functionals, depicting the exact behavior of sampled-data systems.

After integration, the dynamics of the system in 2.20, satisfies:

$$\begin{array}{ll} \forall t \in [t_{k'},\ t_{k'+1}], & x(t) = \Gamma(t - t_{k'})x(t_{k'}), \\ \forall \tau \in [0,\ T_{k'}], & \Gamma(\tau) = \left[e^{\bar{A}\tau} + \int_0^\tau e^{\bar{A}(\tau-\theta)}\mathrm{d}\theta \bar{B}K\right]. \end{array} \tag{2.23}$$

This equality leads naturally to the introduction of the following notation.

For any integer $k' \in \mathbb{N}$, define the function $\chi_{k'} \in \mathbb{K}$, such that, for all $\tau \in [0,\ T_{k'}]$:

$$\left\{ \begin{array}{ll} \chi_k(\tau) = x(t_{k'} + \tau) & = \Gamma(\tau)\chi_{k'}(0), \\ \dot{\chi}_{k'}(\tau) = \frac{\mathrm{d}}{\mathrm{d}\tau}\chi_{k'}(\tau) & = A\chi_{k'}(\tau) + BK\chi_{k'}(0). \end{array} \right. \tag{2.24}$$

The definition of $\chi_{k'}$ yields $x(t_{k'+1}) = \chi_k(T_{k'}) = \chi_{k'+1}(0)$.
Using this new notations, the following theorem is derived.

Theorem 2 *Let $0 < \mathcal{T}_1 \leq \mathcal{T}_2$ be two positive scalars and $V : \mathbb{R}^n \to \mathbb{R}^+$ be a differentiable function for which there exist positive scalars $\mu_1 < \mu_2$ and p such that:*

$$\forall x \in \mathbb{R}^n, \quad \mu_1 |x|^p \leq V(x) \leq \mu_2 |x|^p. \tag{2.25}$$

Then the two following statements are equivalent.

(i) *The increment of the Lyapunov function is strictly negative for all $k' \in \mathbb{N}$ and $T_{k'} \in [\mathcal{T}_1,\ \mathcal{T}_2]$, i.e,*

$$\Delta_0 V(k') = V(\chi_{k'}(T_{k'})) - V(\chi_{k'}(0)) < 0;$$

(ii) *There exists a continuous and differentiable functional $\mathcal{V}_0 : [0,\ \mathcal{T}_2] \times \mathbb{K} \to \mathbb{R}$ which satisfies for all $z \in \mathbb{K}$*

$$\forall T \in [\mathcal{T}_1,\ \mathcal{T}_2] \quad \mathcal{V}_0(T, z(\cdot)) = \mathcal{V}_0(0, z(\cdot)), \tag{2.26}$$

and such that, for all $(k', T_{k'}, \tau) \in \mathbb{N} \times [\mathcal{T}_1,\ \mathcal{T}_2] \times [0\ T_{k'}]$,

$$\dot{\mathcal{W}}_0(\tau, \chi_{k'}) = \frac{\mathrm{d}}{\mathrm{d}\tau}[V(\chi_{k'}(\tau)) + \mathcal{V}_0(\tau, \chi_{k'})] < 0. \tag{2.27}$$

Moreover, if one of these two statements is satisfied, then the solutions of the system (2.24) are asymptotically stable.

Proof 1 *Let $k' \in \mathbb{N}$, $T_{k'} \in [\mathcal{T}_1,\ \mathcal{T}_2]$ and $\tau \in [0,\ T_{k'}]$. Assume that (ii) is satisfied. Integrating $\dot{\mathcal{W}}_0$ with respect to τ over $[0,\ T_{k'}]$ and assuming that (2.26) holds, this leads to $\int_0^{T_{k'}} \dot{\mathcal{W}}_0(\tau, \chi_{k'})\mathrm{d}\tau = \Delta_0 V(k')$. Then $\Delta_0 V(k')$ is strictly negative since $\dot{\mathcal{W}}_0$ is negative over $[0,\ T_{k'}]$.*

Assume that (i) is satisfied. Introduce the functional $\mathcal{V}_0(\tau, \chi_{k'}) = -V(\chi_{k'}(\tau)) + \tau/T_{k'}\Delta_0 V(k')$, as in Lemma 2 in [Peet 2009]. By simple computations, it is easy to see that it satisfies (2.26) and $\dot{\mathcal{W}}_0(\tau, \chi_{k'}) = \Delta_0 V(k')/T_{k'}$. This proves the equivalence between (i) and (ii).

The function $\Gamma(\cdot)$ is continuous and consequently bounded over $[0,\ \mathcal{T}_2]$. Then Equation (2.21) proves that $x(t)$ and the continuous Lyapunov function uniformly and asymptotically tend to zero.

Remark: This theorem is the same as in [Seuret 2011a]. In fact, this thesis is an application of the theorem (Even if other approaches such as [Naghshtabrizi 2009, Naghshtabrizi 2008, Fridman 2006] could have been used) in order to provide stability conditions for the considered systems. The main contribution of the thesis lays in the proposition of a weakened implementation scheme for real-time feedback controllers in order to reduce the conservatism due to traditional worst-cases considerations, while preserving the stability and control performance.

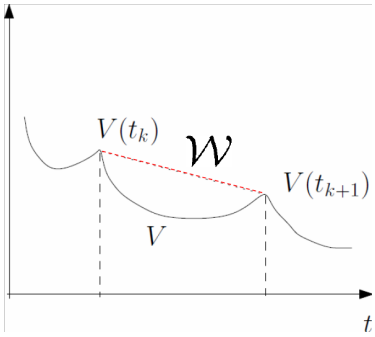

Figure 2.9: Illustration of Theorem 2 with $\mathcal{V}_0(T_{k'}, \chi_{k'}) = \mathcal{V}_0(0, \chi_{k'}) = 0$.

A graphical illustration of the proof of Theorem 2 is shown in Figure 2.9. The main contribution of the theorem is the introduction of a new kind of Lyapunov functionals for sampled-data systems.

The main interest of such approach relies on the fact that the stability conditions are not related to the Lyapunov-Krasovskii Theorem. Indeed, using this theorem, the functional $\mathcal{V}$ is not needed to be definite positive. The only contraint is to satisfy the boundary conditions proposed in conditions (2.26). This is the main contribution of the previous theorem.

This theorem has already been extended to the stability analysis of sampled-data using Sum Of Squares [Seuret 2011b], Impulsive systems [Briat 2012].

In the present thesis we will show how the previous theorem can be adapted to the problem under consideration in the thesis.

2.4.3 Stability criteria

System without uncertainties

In this section, a study of the asymptotic stability of the solutions of sampled-data systems presented in 2.1 with $\mu = 0$ is provided. There exist several results in the literature (e.g: [Fridman 2004b, Hespanha 2007]) to ensure asymptotic stability of linear systems with input delay and sampling.

The contribution of this section is based on the asymptotic stability conditions developed in [Seuret 2011a].

Theorem 3 *Let N be a positive integer and T_{slot} and T, two non-negative scalars. Assume that there exist $Q > 0$, $R_1 > 0$ and $R_2 > 0 \in \mathbb{S}^n$, $P > 0$, $U > 0$ and $S_1 \in \mathbb{S}^{2n}$ and S_2 and $X \in \mathbb{R}^{2n\times 2n}$, $Y \in \mathbb{R}^{5n\times 2n}$ that satisfy for $j = 1, 2$:*

$$\begin{aligned} \Psi_1(A,B) &= \Pi_1(T_{slot}) + T_j\Pi_2 + T_j\Pi_3 < 0, \\ \Psi_2(A,B) &= \begin{bmatrix} \Pi_1(T_{slot}) - T_j\Pi_3 & T_jY \\ T_jY^T & -T_jU \end{bmatrix} < 0, \end{aligned} \tag{2.28}$$

where $T_1 = T$, $T_2 = NT$ *and*

$$\begin{aligned}\Pi_1(T_{slot}) &= N_1^T P N_0 + M_1^T Q M_1 - M_2^T Q M_2 + M_0^T (R_1 + T_{slot}\, R_2) M_0 \\ &\quad -\tfrac{1}{T_{slot}} \left[M_{12}^T R_2 M_{12}\right] - M_5^T R_1 M_5 - N_{12}^T S_1 N_{12} - Y N_{12} - N_2^T S_2 N_{12} \\ \Pi_2 &= N_0^T U N_0 + N_0^T (S_1 N_{12} + S_2^T N_2) \\ \Pi_3 &= N_2^T X N_2 \end{aligned} \tag{2.29}$$

and

$$\begin{array}{ll} M_0 = [A\ 0\ 0\ BK\ 0] & M_1 = [I\ 0\ 0\ 0\ 0] \\ M_2 = [0\ I\ 0\ 0\ 0] & M_3 = [0\ 0\ I\ 0\ 0] \\ M_4 = [0\ 0\ 0\ I\ 0] & M_5 = [0\ 0\ 0\ 0\ I] \\ N_0 = [M_0^T\ M_5^T]^T & N_1 = [M_1^T\ M_2^T]^T \\ N_2 = [M_3^T\ M_4^T]^T & M_{12} = M_1 - M_2 \\ N_{12} = N_1 - N_2. & \end{array}$$

Systems described by equation (2.3) are thus asymptotically stable for any aperiodic sampling period lying in $[T, NT]$ *and the constant delay* T_{slot}.

Remark: The asynchrony in the sampling is considered with a discrete step. The changes must only be a multiple of the initial sampling period. This means that the sampling period can take different value from th set $\{T, 2T, \cdots, NT\}$ where N is a positive integer.

Proof 2 *Consider the functional:*

$$\begin{aligned} V(t, x_t, \dot{x}_t) = y^T(t) P y(t) &+ \int_{t-T_{slot}}^{t} x^T(s) Q x(s) ds \\ &+ \int_{t-T_{slot}}^{t} \dot{x}^T(s)(R_1 + (T_{slot} - t + s) R_2)\dot{x}(s) ds \end{aligned} \tag{2.30}$$

where $y(t) = [x^T(t)\ x^T(t - T_{slot})]^T$. *Note that* V *corresponds to a classical Lyapunov-Krasovskii functional type to cope with the stability of systems with constant time-delay.*

The objective of the proof is to ensure that the variation of the of V *between two successive sampling instants is negative (as Theorem2 implies).*

This means that:

$$\Delta V = V(t_{k'+1}, x_{t_{k'+1}}, \dot{x}_{t_{k'+1}}) - V(t_k, x_{t_{k'}}, \dot{x}_{t_{k'}}) < 0$$

meaning that ΔV *is negative definite for all positive integer* k'. *This relation can be easily demonstrated by using the following conditions.*

For $t = t_{k'}$, $\zeta_0(t_k) = 0$, *and* $\theta = \tau(t) = t - t_{k'}$, *we can say that* $\tau(t_k) = 0$) *and also, we can thus say that* $\int_{t_k}^{t} \dot{y}^T(s) U \dot{y}(s) ds = 0$ *and* $(\Upsilon_{k'} - \theta)\theta y^T(t_k) X y(t_k)$

Also $t = t_{k'} + 1$ *leads to* $(\Upsilon_{k'} - \theta) = 0$, *thus, it leads to satisfy the condition of equation 2.26. Moreover, this allows to show that* $\mathcal{V}$ *is continuous and differentiable over* $[t_{k'}; t_{k'} + 1[$

Now, the next step is of the proof consist of ensuring that condition 2.26 is verified on the additional functional $\mathcal{V}$.
For any integer k', the sampling period is denoted $\Upsilon_{k'} = t_{k'+1} - t_{k'}$. Now, considering the following additional functional:

$$\begin{aligned}\mathcal{V}(t, \chi_k^{T_{slot}}) &= (\Upsilon_{k'} - \theta)\zeta_0^T(t)[S_1\zeta_0(t) + 2S_2 y(t_k)] \\ &+(\Upsilon_{k'} - \theta)\int_{t_k}^{t} \dot{y}^T(s)U\dot{y}(s)ds \\ &+(\Upsilon_{k'} - \theta)\theta y^T(t_k)Xy(t_k)\end{aligned} \tag{2.31}$$

where:

$$\zeta_0(t) = y(t) - y(t_{k'})$$

$$\xi(s) = \left[y^T(s)\ y^T(t_{k'})\ \dot{x}^T(s - T_{slot})\right]^T$$

As suggested in the theorem, no additional constraint is introduced on S_1, S_2, U and X and $\mathcal{V}$ is not necessarily positive definite within two sampling instants. This corresponds to the improvement with respect to the previous approaches exposed in [Liu 2009, Naghshtabrizi 2009]. Note that the positivity of U is not required but we will be introduced in the next steps.

The rest of the proof consists in ensuring $\dot{\mathcal{W}} < 0$ over $[0\ \bar{T}_{k'}[$.

The computation of the derivative of $\dot{\mathcal{W}}$ leads to:

$$\begin{aligned}\dot{\mathcal{W}}(\tau, \chi_{k'}) &= 2\bar{\chi}_{k'}^T(\tau)P\dot{\bar{\chi}}_{k'}(\tau) + \dot{\chi}_{k'}^T(\tau, 0)(R_1 + hR_2)\dot{\chi}_{k'}(\tau, 0) \\ &+\chi_{k'}^T(\tau, 0)Q\chi_{k'}(\tau, 0) - \chi_{k'}^T(\tau, -h)Q\chi_{k'}(\tau, -h) \\ &-\dot{\chi}_{k'}^T(\tau, -h)R_1\dot{\chi}_{k'}(\tau, -h) - \int_{-h}^{0} \dot{\chi}_{k'}^T(\tau, s)R_2\dot{\chi}_{k'}(\tau, s)ds \\ &+(T_{k'} - \tau)\dot{\bar{\chi}}_{k'}^T(\tau)[U\dot{\bar{\chi}}_{k'}(\tau) + 2S_1\zeta_{k'}(\tau) + 2S_2\bar{\chi}_{k'}(0)] \\ &-\zeta_{k'}^T(\tau)[S_1\zeta_{k'}(\tau) + 2S_2\bar{\chi}_k(0)] - \int_0^\tau \dot{\bar{\chi}}_{k'}^T(s)U\dot{\bar{\chi}}_{k'}(s)ds \\ &+(T_{k'} - 2\tau)\bar{\chi}_{k'}^T(0)X\bar{\chi}_{k'}(0).\end{aligned} \tag{2.32}$$

Applying Jensen inequality [Jensen 1906] to the first integral, we have that

$$-\int_{-h}^{0} \dot{\chi}_{k'}^T(\tau, s)R_2\dot{\chi}_{k'}(\tau, s)ds \leq -(\chi_{k'}(\tau, 0) - \chi_{k'}(\tau, -h))^T\frac{R_2}{h}(\chi_{k'}(\tau, 0) - \chi_{k'}(\tau, -h))$$

Consider the augmented vector $\xi_{k'}(\tau) = [\bar{\chi}_{k'}^T(\tau)\bar{\chi}_{k'}^T(0)\ \dot{\chi}_{k'}^T(\tau, -h)]^T$ and a matrix $Y \in \mathbb{R}^{5n\times 2*n}$. The following equality holds

$$2\xi_{k'}^T(\tau)Y\left[\bar{\chi}_{k'}(\tau) - \bar{\chi}_{k'}(0)\right] = \int_0^\tau \left[2\xi_{k'}^T(\tau)Y\dot{\bar{\chi}}_{k'}(s)\right]ds. \tag{2.33}$$

Since U is assumed to be positive definite and thus non singular, a classical bounding ensures that for all $\tau \in [0,\ T_{k'}]$ and for all $s \in [0,\ \tau]$

$$2\xi_{k'}^T(\tau)Y\dot{\bar{\chi}}_{k'}(s) \leq \xi_{k'}^T(\tau)YU^{-1}Y^T\xi_{k'}(\tau) + \dot{\bar{\chi}}_{k'}^T(s)U\dot{\bar{\chi}}_{k'}(s).$$

Integrating the previous inequality over $[0,\ \tau]$, the following inequality is obtained

$$-\int_0^\tau \dot{\bar{\chi}}_{k'}^T(s)U\dot{\bar{\chi}}_{k'}(s)ds \leq -2\xi_{k'}^T(\tau)Y(\bar{\chi}_{k'}(\tau) - \chi_{k'}(0)) + \tau\xi_{k'}^T(\tau)YU^{-1}Y^T\xi_{k'}(\tau). \tag{2.34}$$

By identifying the following relations:

$$\begin{array}{ll}
\dot{\chi}_{k'}(\tau,0) & = A\chi_{k'}(\tau,0) + A_d\chi_{k'}(0,-h) = M_0\xi_{k'}(\tau), \\
\chi_{k'}(\tau,0) & = M_1\xi_{k'}(\tau), \quad \chi_{k'}(\tau,-h) = M_2\xi_{k'}(\tau), \\
\chi_{k'}(\tau,0) - \chi_{k'}(\tau,-h) & = M_{12}\xi_{k'}(\tau), \quad \bar{\chi}_{k'}(\tau) = N_1\xi_{k'}(\tau), \\
\bar{\chi}_{k'}(0) & = N_2\xi_{k'}(\tau), \quad \zeta_{k'}(\tau) \\
\bar{\chi}_{k'}(\tau) - \chi_{k'}(0) & = N_{12}\xi_{k'}(\tau) \\
\dot{\bar{\chi}}_{k'}(\tau) & = [(M_0\xi_{k'}(\tau))^T \quad \dot{\chi}_{k'}^T(\tau,-h)]^T = N_0\xi_(\tau)
\end{array}$$

and substituting (2.34) into (2.32), the following inequality is obtained for all $\tau \in [0,\ T_{k'}[$

$$\dot{\mathcal{W}}(\tau,\chi_{k'}) \quad \leq \xi_{k'}^T(\tau)[\Pi_1(h) + (T_{k'} - \tau)\Pi_2 + (T_{k'} - 2\tau)N_2^TXN_2 + \tau NU^{-1}N^T]\xi_{k'}(\tau). \tag{2.35}$$

Based on a convexity argument on θ, the right hand-side term is negative definite if and only if

$$\Pi_1(h) + T_{k'}(\Pi_2 + N_2^TXN_2) < 0,$$

and

$$\Pi_1(h) + T_{k'}(YU^{-1}Y^T - N_2^TXN_2) < 0.$$

Using the same convexity argument on $T_{k'} \in [\mathcal{T}_1,\ \mathcal{T}_2]$, the LMI's (2.28) and (2.29) are retrieved. By virtue of Theorem 3, the asymptotic stability of the system (2.3) is guaranteed.

Note that the conditions from Theorem 3 include the robust stability properties with respect to the input delay T_{slot}. This means that (2.28) requires the system to be stable at least for the transmission delay T_{slot} and $T = T_i$.

Uncertain systems

In this section we will consider the case of $\mu \neq 0$ for systems of the form 2.1. Then we want to extend the previous theorem to the case of time-varying uncertainties. In the previous stability theorem , the conditions depends almost linearly on the matrices defining the continuous -time model. Then the following corollary presents an extension of the previous theorem to uncertain and time-varying model.

Corollary 1 *Consider an integer N and there non negative scalars T_{slot}, T and μ. Assume that there exist: $Q > 0$, $R_1 > 0$ and $R_2 > 0 \in \mathbb{S}^n$, $P > 0$, $U > 0$ and $S_1 \in \mathbb{S}^{2n}$ and three matrices S_2 and $X_i \in \mathbb{R}^{2n\times 2n}$, $Y \in \mathbb{R}^{5n\times 2n}$ that satisfy, for $i = 1,\ldots,M$ and $j = 1,2$:*

$$\Psi_{1i}(A_i, B_i) = \Pi_{1i}(T_{slot}) + \mathcal{T}_j\Pi_{2i} + \mathcal{T}_j\Pi_{3i} < 0 \tag{2.36}$$

$$\Psi_{2i}(A_i, B_i) = \begin{bmatrix} \Pi_{1i}(T_{slot}) - \mathcal{T}_j\Pi_{3i} & \mathcal{T}_jY_i \\ \mathcal{T}_jY_i^T & -\mathcal{T}_jU \end{bmatrix} < 0, \tag{2.37}$$

where

$$\begin{aligned}
\Pi_{1i}(T_{slot}) &= 2He\{N_1^TPN_{0i}\} + M_1^TQM_1 - M_2^TQM_2 \\
&\quad + M_{0i}^T(R_1 + T_{slot}R_2)M_{0i} - M_{12}^TR_2/T_{slot}M_{12} \\
&\quad - M_5^TR_1M_5 - N_{12}^TS_1N_{12} - 2He\{Y_iN_{12}\} \\
&\quad - 2He\{N_2^TS_2N_{12}\} \\
\Pi_{2i} &= N_{0i}^TUN_{0i} + 2He\{N_{0i}^T(S_1N_{12} + S_2^TN_2)\}, \\
\Pi_{3i} &= N_2^TX_iN_2
\end{aligned}$$

and

$$\begin{aligned}
M_{0i} &= [A_i\ 0\ 0\ B_iK\ 0], & N_{0i} &= [M_{0i}^T\ M_5^T]^T, \\
A_i &= A + \mu A_i & B_i &= B + \mu B_i
\end{aligned}$$

System (2.3) is thus asymptotically stable for for the periodic sampling defined by T and the delay T_{slot}.

Proof 3 *Consider the stability conditions from Theorem 3. By noting that*

$$\begin{aligned}
M_0(t) &= \left[\bar{A}(t)\ 0\ 0\ \bar{B}(t)K\ 0\right] = \textstyle\sum_{i=1}^{M} \lambda_i(t)M_{0i} \\
N_0(t) &= \left[M_0^T(t) M_5^T\right] = \textstyle\sum_{i=1}^{M} \lambda_i(t)N_{0i}
\end{aligned}$$

and by introducing the matrices variables

$$\begin{aligned}
Y(t) &= \textstyle\sum_{i=1}^{M} \lambda_i(t)Y_i \\
X(t) &= \textstyle\sum_{i=1}^{M} \lambda_i(t)X_i
\end{aligned}$$

Remark:

Most of the terms defined in $\Psi_1(\bar{A}(t), \bar{B}(t))$ and $\Psi_2(\bar{A}(t), \bar{B}(t))$ are linear with respect to the time-varying terms. However the terms $M_0^T(t)(R_1 + T_{slot}R_2)M_0(t)$ and $N_0^T(t)UN_0(t)$ are still not linear with respect to the matrices $M_0(t)$ and $N_0(t)$. However the Schur complement allows obtaining expressions which become linear with respect to this two time-varying matrices.

Then this proves that

$$\begin{aligned}
\Psi_1(\bar{A}(t), \bar{B}(t)) &= \textstyle\sum_{i=1}^{M} \nu_i(t)\Psi_{1i}(A_i, B_i), \\
\Psi_2(\bar{A}(t), \bar{B}(t)) &= \textstyle\sum_{i=1}^{M} \nu_i(t)\Psi_{2i}(A_i, B_i)
\end{aligned}$$

Then if all the LMI's $\Psi_{1i}(A_i, B_i)$ and $\Psi_{2i}(A_i, B_i)$ are satisfied for $i = 1, \ldots, M$, where M is the number of vertices of the polytopic model, then the conditions of Theorem 3 are also verified for the time-varying system (2.24).

Ex.1 T_{slot}	10^{-3}	0.2	0.4	0.6
[Naghshtabrizi 2009]	1.111	0.714	0.469	0.269
[Millán 2009]	1.043	0.846	0.650	0.456
[Liu 2009]	1.638	1.063	0.786	0.541
[Liu 2012]	1.36	1.18	1.08	0.82
Th. 2	1.717	1.435	1.149	0.858

Table 2.1: Maximal sampling period at the actuator T for several constant delays using Theorem 2 for example 1.

2.5 Examples

Example 1 [Fridman 2004b, Naghshtabrizi 2008]

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 \\ -0.375 & -1.15 \end{bmatrix} x(s_{k'}).$$

The results are summarized in Table 2.1 for time-varying sampling and constant delay.

When the constant computation delay T_{slot} indicated obtained by solving Theorem 3 using $\bar{\mathcal{T}}_1 = \mathcal{T}_1 = 0$. It can be seen in Table 2.1 that it delivers less conservative results than the existing ones based on a continuous time-approach.

2.6 Conclusion to Chapter 2

In this chapter, a weakened real-time implementation was studied. Different scenarii to relax the implementation of real-time controllers were proposed. Then a stability analysis based on Lyapunov-Krasovskii and looped functionals was developed. They have been tested on academic examples in order to show the efficiency of the method so as to provide a comparison basis with existing results in the literature.

The idea of a weakened real-time system consist of allowing the reduction of the allocated time in order to improve resource utilization.

Moreover, the weakened real-time system would allow some flexibility and implement robustness at the same time.

It has been observed that the robustness of a control system would allow to handle timing and model uncertainties and lead to a better performance and quality of control.

From the control system perspective, the act of reducing the allocated time for running control tasks lead to control robustness and quality of control enhancement due to the reduction of the overall latency and eventually the reduction of sampling period. On the contrary, from the computing resource perspective, reducing this parameter lead to an occasional occurrence of deadline violation, and thus to a degradation of performance due to the occasional sample loss (if the retarded computation sample is aborted for instance).

As far as stability analysis is concerned, several improvements can be viewed. Indeed, potential improvements were proposed in [Liu 2012] by considering the Wirtinger inequal-

ities. The combinaison of the two methods could potentially reduce the conservatism of the stability conditions.

Applying the delay definitions in this chapter, notice that each of the proposed scenarii requires a dedicated stability analysis which were not all studied in the present report.

However they represent a relevant direction for future researches in order to relax contraints related to real-time implementation. However, this thesis is limited to implementations related to avionics systems.

CHAPTER 3

Case Study : Aircraft pitch-rate control

Contents

3.1 Introduction to chapter 3

The previous chapter has focused on the theoretical part of the holistic timing analysis approach. The objective of that chapter is to show how the robustness of a feedback control system w.r.t. timing variabilities and plant uncertainties can be evaluated through a case study. The weakly-hard nature of feedback has been advocated by theoretical approaches based on discrete-time and time-delay theories, applied in the framework of real-time control and scheduling co-design. This chapter is aimed at supporting these theoretical schemes through a case study related to aircraft control design and implementation.

By following the steps shown of Figure B.1, the goal is to find an adequate allocated time that allows for an enhanced resource utilization while keeping the stability of the

considered control system. It is first needed to get a *Mathematical model of the controlled system*, which can be given using a state space representation. In this case study, this model represents the aircraft dynamics, or a significant part of the aircraft. The *control algorithm* is first designed as a function used to compute the control actions from the measurements (states or outputs) of the system. As this thesis is not focused on control design, we use existing controllers that can be obtained from classical control design methods, for instance using pole placement.

Instead, the focus of the thesis is more on providing a complete framework that can be used to analyze the effects of deadline misses and to compute the robustness of a safety critical realtime control system.

The *estimated WCET* is also needed. In fact, the model used for this part is an empiric statistical plot of the execution time probability, as given by figure 1.15. Appendix A.6 provides current methods used to model the probability functions of the execution times of real-time control tasks.

In the remainder, it is described how to drive the analysis process as described in figure B.1 and how to analyze the corresponding results, as well as how to use them.

This chapter is divided into four main parts:

- Section 3.2 describes several models of an aircraft's dynamics, starting from the non-linear equations derived from first principles of aerodynamics. These equations are then linearized for control design purpose, and the subsystem needed for the pitch rate control case study is extracted from the full model.

  An instantiation of the formal model is done for the F-16 aircraft's set of parameters, this is an adequate choice since this particular model is available and well-documented.

- In section 3.3 several weakly-hard scheduling schemes defined in chapter 2 are applied to the implementation of a pitch-rate controller for the F16 model. For each of the proposed implementation scenario, various ranges of parameters, i.e. reduced computing slots and plant's uncertainties, have been evaluated through LMIs solving, until loosing stability conditions. For all cases it is possible to use computing slots smaller than the WCET while keeping the control system's stability.

- Section 3.4.4 describes the validation of the results using a non-linear model based simulation. Such a simulation allows the validation of the analytic result (based on a linearized plant) by using a more realistic full non-linear model of the aircraft augmented with uncertainties on the system's parameters.

3.2 Models for simulation and control

This section gives a concise overview of aircraft control systems.

Building an analytic and simulation model starts with setting up the equations of the six-degree-of-freedom (6-DOF) dynamic aircraft model. These equations are well-known and can be found in many textbooks [Stengel 2004, Stevens 2003, McLean 1990].

A precise knowledge of the control derivatives together with aerodynamic and mechanical parameters is essential to develop a high fidelity simulation model including all the items relevant for flight analysis. On the other hand, simplified models, which only need to capture the essential modes of the aircraft's dynamics, must be derived for control design purpose.

3.2.1 Frames and rotation parameters definition

Before giving the aircraft equations, it is necessary to define the different frames within which these equations are expressed. In fact, various reference frames are used to describe flight dynamics(e.g. [McLean 1990], [Peters 2003]), the four following are used in the sequel :

- Earth-fixed reference frame F_E
- Body-fixed reference frame F_B
- Stability reference frame F_S
- Flight-path or wind reference frame F_W

The earth-fixed reference frame (also called inertial frame) is fixed on a point of the earth surface and is aligned so that, its positive x -axis points to the true North direction, its y-axis points to the East and its z-axis points down and is normal to the surface of the earth (the earth being assumed to be flat). This reference frame is depicted in figure 3.1 (a) where the x, y and z axes are respectively denoted also as x_i, y_i and z_i.

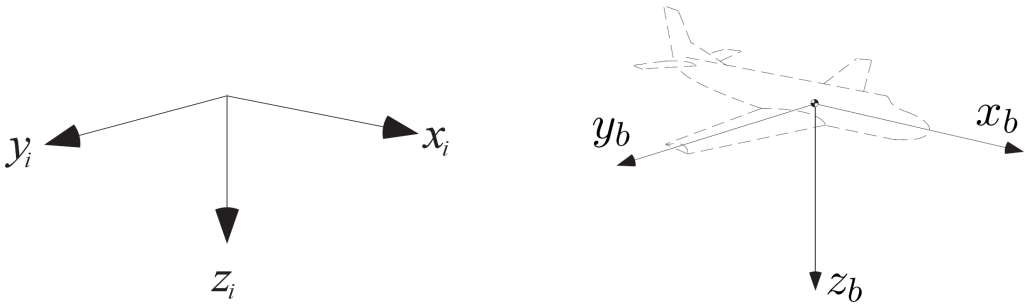

Figure 3.1: Inertial reference frame (a) body-fixed reference frame (b)

The body-fixed reference frame is a frame fixed on the aircraft, whose origin is at the center of gravity. Its x-axis points its positive direction toward the nose of the aircraft and is aligned with the axis of the body, the y-axis direction points out along the right-hand side of the aircraft when looking forward and its z-axis points down to complete a right -handed coordinate system. This reference frame is depicted in figure 3.1 (b) where the x, y and z axes are respectively denoted as x_b, y_b and z_b.

Euler angles The definition of those two reference frames allows to express *the attitude or orientation* of the aircraft in the body- fixed frame with respect to the inertial frame. The relationship between the two reference frames are depicted in figure 3.2.

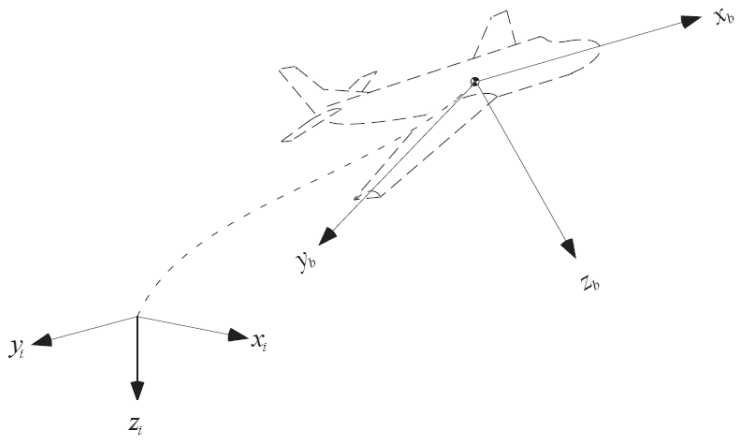

Figure 3.2: Relationship between body and inertial reference frame [Peters 2003]

This attitude information is described by a system of Euler angles. This defines three angles ψ (yaw), θ (pitch) and ϕ (roll) which are respectively the rotation of the aircraft around the inertial x_i, y_i and z_i axis, to an intermediate axis system one step at a time. This means that the first rotation ψ about the z_I axis leads to an expression of the attitude of the aircraft into an intermediate reference frame (here arbitrarily denoted with the "1" subscript), the next rotation, about the y_1 axis of the newly defined intermediate leads to a expression of the attitude into another intermediate reference frame denoted "2" and the last rotation allows about the x_2 axis of the newly defined reference frame.

These rotations are respectively described in figure 3.3.

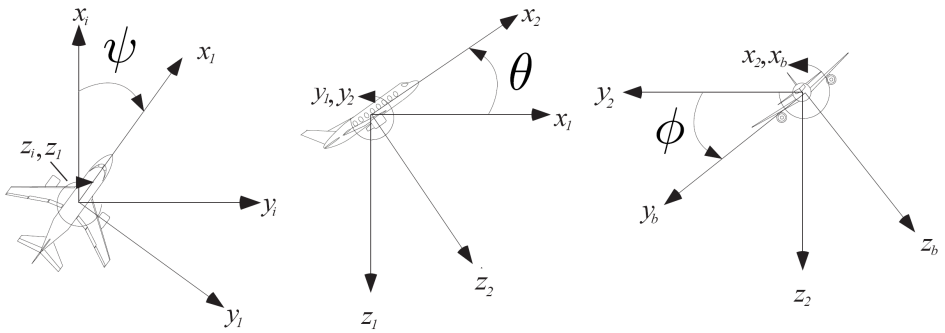

Figure 3.3: The Euler sequence of rotations to express the aircraft attitude

Thus the conversion between the inertial frame and the body frame of the aircraft is accomplished easily using specific direction cosine matrices constructed from such sequence of rotations. Such matrices are given in [Stengel 2004, Peters 2003].

For a reference use, the aircraft forces equation 3.1 and 3.3 were given in the body-axis.

However, new additional references frames are needed to express the relationship between the body-axes forces X_A,Y_A and Z_A and the commonly known aerodynamic forces of lift L (not the rolling moment), drag D and the side force SF. These reference frames are the stability-axes and the wind-axes reference frames. These reference frames characterize the relationship of the aerodynamic angles with the body-axis reference frame. These aerodynamic angles are composed of the angle of attack α and the side-slip angle β, which is the angle formed by the stability frame and the wind frame angle . They are expressed by means of coordinate rotations.

The stability-axis reference frame: It allows to define the stability-axis reference frame, and allows to define the angle of attack, by completing a coordinate rotation around the y_b axis of the body-axis system. Assuming that there is no side-slip angle, the angle of attack is the angle between x_b and the aircraft velocity vector relative to the surrounding air.

The wind-axis reference frame: A coordinate rotation around the z_b -axis allows to define the wind-axis reference frame. The wind- axis reference frame's x_w-axis is aligned with the aircraft velocity vector which is the vector sum of the velocities expressed in the body-axis.

The orientations of the different axes and the new frames with respect to the body-axes system is depicted in figure 3.4.

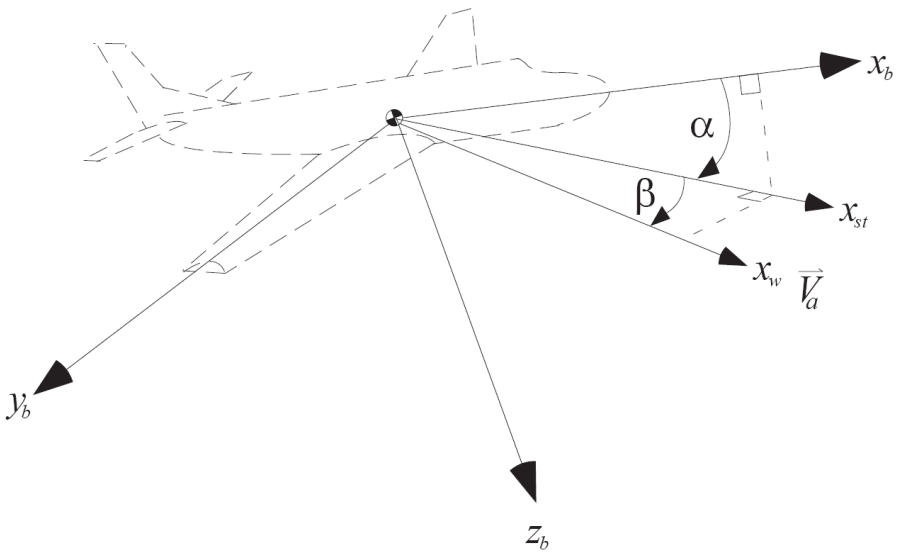

Figure 3.4: Stability and wind axis reference frames

3.2.2 Non-linear dynamics

The standard 6-DOF equations of motions for a conventional aircraft are used for preliminary modeling, control design and first simulations. The aircraft is assumed to be rigid, all

flexible modes are neglected, and the dynamics of the actuators is also simplified (or even neglected).

The body-axes equations are as follows:

Forces equations:

$$\dot{u} = rv - q\,w - g\,sin\theta + \frac{(X_A + X_T)}{m} \tag{3.1}$$

$$\dot{v} = -ru + p\,w + g\,sin\phi\,cos\theta + \frac{(Y_A + Y_T)}{m} \tag{3.2}$$

$$\dot{w} = qu - p\,v + g\,cos\phi\,cos\theta + \frac{(Z_A + Z_T)}{m} \tag{3.3}$$

Kinematic equations:

$$\dot{\phi} = p + tan\theta\,(q\,sin\phi + r\,cos\phi) \tag{3.4}$$

$$\dot{\theta} = q\,cos\phi - r\,sin\phi \tag{3.5}$$

$$\dot{\psi} = \frac{(q\,sin\phi + r\,cos\phi)}{cos\theta} \tag{3.6}$$

Navigation equations:

$$\dot{p}^N = u\,cos\theta\,cos\psi + v\,(-cos\phi\,sin\psi + sin\phi\,sin\theta\,cos\psi) + w\,(sin\phi\,sin\psi + cos\phi\,sin\theta\,cos\psi) \tag{3.7}$$

$$\dot{p}^E = u\,cos\theta\,sin\psi + v\,(cos\phi\,cos\psi + sin\phi\,sin\theta\,sin\psi) + w\,(-sin\phi\,cos\psi + cos\phi\,sin\theta\,sin\psi) \tag{3.8}$$

$$\dot{p}^D = -usin\theta + vsin\phi\,cos\theta + wcos\phi\,cos\theta \tag{3.9}$$

Moment equations:

$$\dot{p} = \frac{J_z\,L + J_{xz}\,N - \left[J_{xz}\,(J_y - J_x - J_z)\;p + \left(J_{xz}^2 + J_z\,\{J_z - J_y\}\right)\;r\right]\;q}{J_x\,J_z - J_{xz}^2} \tag{3.10}$$

$$\dot{q} = \frac{\left[M - (J_x - J_z)\;p\,r - J_{xz}\left(p^2 - r^2\right)\right]}{I_y} \tag{3.11}$$

$$\dot{r} = \frac{J_{xz}\,L + J_x\,N - \left[J_{xz}\,(J_y - J_x - J_z)\;r + \left(J_{xz}^2 + J_x\,\{J_x - J_y\}\right)\;p\right]\;q}{J_x\,J_z - J_{xz}^2} \tag{3.12}$$

where the definition of each state variable, according to [Stevens 2003, Stengel 2004], are defined as:

$$x = [x_1^T\;x_2^T\;x_3^T\;x_4^T]^T$$

where:

$x_1 = [u\;v\;w]^T$

$x_2 = [\phi\;\theta\;\psi]^T$

$x_3 = [p^N\;p^E\;p^D]^T$

$x_4 = [p\;q\;r]^T$

in which x_1, x_2, x_3 and x_4 are respectively the translational velocity, the attitude, the position and the rotational velocity sub-vectors.

The standard notation for describing the motion of the aircraft and the forces and moments acting upon it are described in figure 3.5, where each components of these sub-vectors and some other elements of the nonlinear aircraft are defined as follows:

- u is the translational velocity along the x_b-axis of the body-axes reference frame.
- v is the translational velocity along the y_b-axis of the body-axes reference frame.
- w is the translational velocity along the z_b-axis of the body-axes reference frame, and where X_T, Y_T and Z_T are the thrusting effects components of the engine along the x, y and z axes respectively.
- ϕ is the roll angle.
- θ is the pitch angle.
- ψ is the yaw angle. each angle is described by figure 3.3.
- p^N is the position vector component on the x_i inertial-axis reference frame but expressed in the body-axis.
- p^E is the position vector component on the y_i inertial-axis reference frame but expressed in the body-axis.
- p^D is the position vector component on the z_i inertial-axis reference frame but expressed in the body-axis.
- p is the roll speed.
- q is the pitch speed.
- r is the yaw speed, in which, the variables L, M, and N respectively represent the rolling, the pitching and the yawing moment.

Remark: Although not represented in this figure, the variables ϕ , θ, ψ represent the angular rotations relative to the equilibrium state, around the x, y, and z axes, respectively. Thus, $p = \dot{\phi}$, $q = \dot{\theta}$ and $r = \dot{\psi}$.

In the body-axes frame, forces equations equations are expressed as sums of aerodynamic (X_A, Y_A and Z_A) and thrusting (X_T, Y_T, and Z_T) effects in equations 3.1. It is possible to express these aerodynamic forces in the wind axis system, which result in the expression of the drag force D along the x-axis, the side force SF along the y-axis and the lift L along the z-axis.

$$x^T = \begin{bmatrix} u\ v\ w\ \phi\ \theta\ \psi\ p^N\ p^E\ p^D\ p\ q\ r \end{bmatrix}^T \tag{3.13}$$

and the input vector is

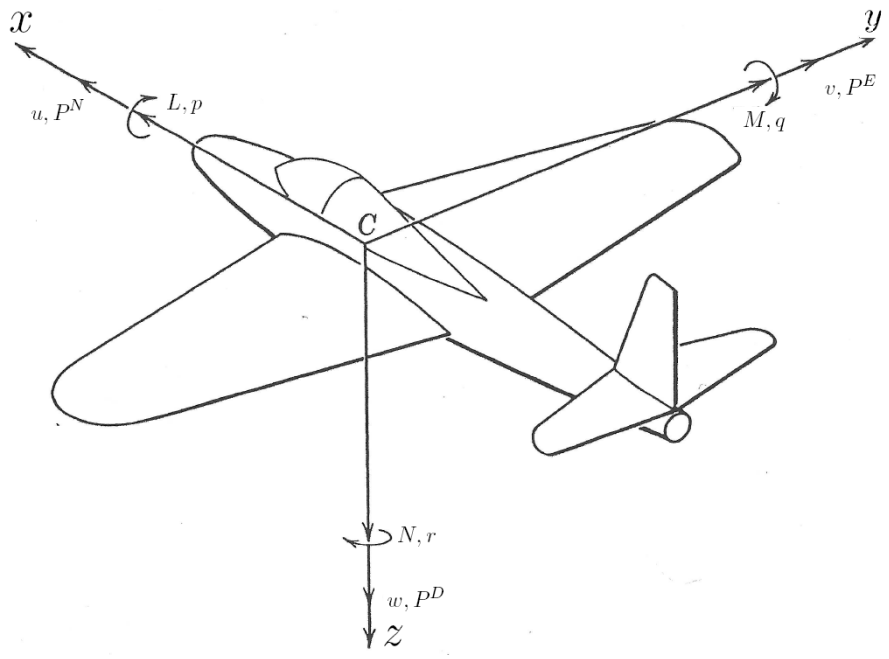

Figure 3.5: Standard notation for aerodynamic forces, moments, linear and rotational velocities in body axis-system; the origin is the center of mass C.

$$u^T = [\delta_t \; \delta_e \; \delta_a \; \delta_r]^T \tag{3.14}$$

where the elements are respectively, the engine throttle setting, the elevator deflection, the aileron deflection and the rudder deflection.

The aerodynamic forces and moments are obtained from the *dimensionless aerodynamics coefficients* at a given flight condition:

$$\begin{array}{ll} X_A & = C_X \, \bar{q} \, S \\ Y_A & = C_Y \, \bar{q} \, S \\ Z_A & = C_Z \, \bar{q} \, S \\ L & = C_l \, \bar{q} \, S \, b \\ M & = C_m \, \bar{q} \, S \, \bar{c} \\ N & = C_n \, \bar{q} \, S \, b \end{array} \tag{3.15}$$

where the quantities in the above expressions are defined as follows:

- C_X, X_Y, C_Z are the aerodynamic force coefficients
- C_l, C_n, C_m are the moment coefficients
- $\bar{q} \stackrel{\text{def}}{=} \frac{\rho(z) \, V_A^2}{2}$ is the dynamic pressure
- $\rho(z)$ is the air density and it is a function of the altitude z
- V_A is the true airspeed which allows to compute the mach number $\mathcal{M} = \frac{V_A}{a(z)}$.

- S is the reference area (which is usually the wing area).
- b is a reference value called the Standard Mean Chord (or wing span).
- c is a the Mean Aerodynamic Chord.
- X_A, Y_A, and Z_A are the aerodynamic forces components along the x, y and z axes, respectively.
- L, M, and N represent the rolling, the pitching and the yawing moment.

These various dimensionless coefficients C_X, X_Y, C_Z,C_l, C_n, C_m are primarily dependent on the aerodynamic angles α and β, and more or less dependent on a number of other variables. [Stevens 2003] also describes the dependence on the rate of changes on the aerodynamic angles and the components p, q, r of the center of gravity angular velocity.

The analysis of the aerodynamic behavior of the aircraft is better understood in the stability axis, or the wind axis system. More importantly, the aerodynamic forces are easily handled in the wind axes system, which is used to express the forces equations in terms of the total airspeed V_T, the angle of attack α and the side slip angle β. These two angles allows to express the orientation of the velocity vector with respect to body axis, which plays an important role in determining the aircraft's aerodynamic forces and moments.

It is possible thus, to express the equation as function of the following new state vector

$$x^T = \left[V_T\ \alpha\ \beta\ \phi\ \theta\ \psi\ p^N\ p^E\ p^D\ p\ q\ r\right]^T \tag{3.16}$$

which allow to express the forces equations as functions of the variables v_T, β, and α as follows:

$$\begin{array}{ll} m\,\dot{V_T} & = F_T\,cos\,(\alpha+\alpha_T)\ cos\beta\ - D + m\,g_1 \\ m\,\dot{\beta}\,V_T & = -F_T\,cos\,(\alpha+\alpha_T)\ sin\beta\ - C_w + m\,g_2 - m\,V_T\,r_s \\ m\,\dot{\alpha}\,V_T\,cos\beta & = -T\,sin\,(\alpha+\alpha_T) - L + m\,g_3 + m\,V_T(q\,cos\beta - p_s\,sin\beta) \end{array} \tag{3.17}$$

and where V_T is the total airspeed, p_s and r_s are the roll and yaw rates projected on the wind axes, q is the pitch rate, m is the mass of the aircraft, g_1 and g_2 are the projections of the gravitational acceleration on the wind axis system.

The aerodynamic forces are the lift L, the drag D, the cross wind force C_w and the engine thrust T. It is clear that aerodynamic axial, lateral and vertical forces components X_A, Y_A and Z_A have been expressed using drag (D), side force (SF) and lift (L). We must beware of the use of variables because here L is not the rolling moment, as expressed in equation 3.15 but instead the lift force. From now on, we will then denote the lift as Li.

The transformation described in 3.17 induces transformations in equation 3.15. It can thus be rewritten as the following:

$$\begin{aligned} -D &= C_D\,\bar{q}\,S \\ SF &= C_{SF}\,\bar{q}\,S \\ Li &= C_L\,\bar{q}\,S \\ L &= C_l\,\bar{q}\,S\,b \\ M &= C_m\,\bar{q}\,S\,\bar{c} \\ N &= C_n\,\bar{q}\,S\,b \end{aligned} \tag{3.18}$$

where C_D, C_{SF} and C_L are respectively the drag, the side force and the lift coefficients.

The expressions of the different dimensionless coefficients in analytic form allows to see clearly the effects of control variables on the aircraft equations because these coefficients are also dependent on control surface deflections, δ_t, δ_e, δ_a, δ_r in order to provide control inputs to the aircraft. The relations can be described as the following.

The side force coefficient can be written as a function of side-slip angle β as:

$$C_{SF} = C_{SF_0} + C_{SF\beta}\beta$$

However, for most aircraft and intended flight conditions, side force coefficients and derivatives are small [Stengel 2004]. Thus, lift and drag coefficients are left to be described. Lift and drag coefficients are typically described as a functions of several motion and control variables as follows:

$$\begin{aligned} C_L &= C_L(\alpha, \mathcal{M}, \delta_f) + C_{L_{\hat{q}}}(\alpha, \mathcal{M}, \delta_f) + C_{L_{\delta_e}}(\alpha, \mathcal{M}, \delta_f)\delta_e \\ C_D &= C_D(\alpha, \mathcal{M}, \delta_f) + C_{D_{\hat{q}}}(\alpha, \mathcal{M}, \delta_f) + C_{D_{\delta_e}}(\alpha, \mathcal{M}, \delta_f)\delta_e \end{aligned} \tag{3.19}$$

where $\hat{q} = \frac{q\,\bar{c}}{2V}$. δ_f is the wing flap setting and δ_e is the elevator angle.

Remark: The wing flap deflection δ_f is only considered for a special flight condition. Here it is given to keep the general form of the formula. In a steady level flight, the flap deflection is zero.

The aerodynamic coefficients are typically modeled as linear functions of the airflow angles, angular rates and control inputs δ_a, δ_e and δ_r.

They are given by:

$$\begin{aligned} C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_{\hat{p}}}\hat{p} + C_{l_{\hat{r}}}\hat{r} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\ C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\hat{q}}}\hat{q} + C_{l_{\delta_e}}\delta_e \\ C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_{\hat{p}}}\hat{p} + C_{n_{\hat{r}}}\hat{r} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \end{aligned} \tag{3.20}$$

where $\hat{p} = \frac{p\,b}{2V}$ and $\hat{r} = \frac{r\,b}{2V}$

For the analytic transformation from the body-axes equations 3.1- 3.12 to wind-axis equations. A detailed derivation of the final wind-axes non-linear equations can be found in [Stevens 2003] as well as a body-axes to wind-axes transformation matrix.

3.2.3 Aircraft Linearized Equation of Motion

For the final use in the analytic process, it is necessary to transform these sets of non-linear equations into linear ones. The process and the different assumptions used to obtain

the equations will be explained briefly in the next section. Details can be found in the references such as [Stevens 2003, Stengel 2004]

The linearization process involve the use of "small-perturbation equations", which are linear equations derived from the nonlinear ones (3.1- 3.12). In these equations, the nonlinear aerodynamic coefficients are replaced by terms involving the quantities called stability derivatives.

The derivation of the linear model involve the examination of the different flight conditions and the concept of equilibrium or singular points.

The coordinate of the singular point is given by the solution $X = X_e$ of the following equation 3.21 (assuming that it is state equation of steady state flight)

$$f(\dot{x}, x, U) = 0 \tag{3.21}$$

On the other hand, steady state flight is defined by [Stevens 2003] as a condition in which all the motion variables (linear and angular velocity components) are constant or zero.

The steady state flight is restrictive and thus induces some simplifying assumptions (e.g the mass of the aircraft remains constant...,). By assuming that the flat-earth assumption is satisfactory for control purposes, the definition allows for a steady state flight.

In this case, the navigation equations of 3.7 -3.9 are not coupled with the overall equation and thus are useless.

Thus, there are only nine (9) equations left and indeed they can be used to define the steady-state conditions .

Remark: The 9 remaining states, considered in the wind-axis frame, are thus the following

$$x = [V_T \ \alpha \ \beta \ \phi \ \theta \ \psi \ p_s \ q \ r_s]^T$$

The steady-state flight condition is defined as follows:

$$\dot{p}, \dot{q}, \dot{r}, \dot{V}_T, \dot{\beta}, \dot{\alpha} \equiv 0, \quad U = constant \tag{3.22}$$

The following additional constraints can be considered according to the flight conditions:

$$\begin{array}{lrl} \text{Steady wings level flight:} & \dot{\phi}, \dot{\phi}, \dot{\theta}, \dot{\psi} \equiv 0, & (\therefore p, q, r) \equiv 0 \\ \text{Steady turning flight:} & \dot{\phi}, \dot{\theta} \equiv 0, & \dot{\psi} \equiv \text{ turn rate} \\ \text{Steady pull-up:} & \phi, \dot{\phi}, \dot{\psi} \equiv 0, & \dot{\theta} \equiv \text{ pull-up rate} \\ \text{Steady roll:} & \dot{\theta}, \dot{\psi} \equiv 0, & \dot{\phi} \equiv rollrate \end{array} \tag{3.23}$$

In our case study, we will look at the steady wings level flight conditions. However, the models are easily generated for other conditions by programs such as those presented in [Stevens 2003].

The implicit nonlinear equation is written as 3.21, while the linear equation can be written implicitly in a state-variable form as follows:

$$E\,\dot{x} = F\,x + G\,u \tag{3.24}$$

Decoupling equations means separating the equations into two sets that are more or less independent. A set describes the longitudinal motion (pitching and translation), while the other set describes the lateral-directional motion (rolling, side slipping and yawing). The decoupled linearized equations are given directly because they are easier to handle for analytic studies.

The longitudinal states and controls are thus:

$$x = [V_T \;\; \alpha \;\; \theta \;\; q]^T \qquad u = [\delta_e \;\; \delta_t]^T \tag{3.25}$$

Using these variables, the longitudinal model of the linearized aircraft is given by 3.24, in which:

$$E = \begin{bmatrix} V_{T_e} - Z_{\dot{\alpha}} & 0 & 0 & 0 \\ -M_{\dot{\alpha}} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} Z_{\delta_e} & -X_{\delta_t}\, sin\,(\alpha_e + \alpha_T) \\ M_{\delta_e} & M_{\delta_t} \\ X_{\delta_e} & X_{\delta_t}\, cos\,(\alpha_e + \alpha_T) \\ 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} Z_\alpha & V_{T_e} + Z_q & Z_V - X_{T_V}\, sin\,(\alpha_e + \alpha_T) & -g_D\, sin\gamma_e \\ M_\alpha + M_{T_\alpha} & M_q & M_V + M_{T_V} & 0 \\ X_\alpha & 0 & X_V + X_{T_V}\, cos\,(\alpha_e + \alpha_T) & -g_D\, cos\gamma_e \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.26}$$

And the lateral/directional states and controls are:

$$x = [\beta \;\; \phi \;\; p_s \;\; r_s]^T \qquad u = [\delta_a \;\; \delta_r]^T \tag{3.27}$$

Using these variables, the lateral model of the linearized aircraft is given by 3.24, where:

$$E = \begin{bmatrix} V_{T_e} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ 0 & 0 \\ L'_{\delta_a} & L_{\delta_r} \\ N'_{\delta_a} & N'_{\delta_r} \end{bmatrix}$$

$$A = \begin{bmatrix} Y_\beta & g_D\, cos\theta_e & Y_p & Y_r - V_{T_e} \\ 0 & 0 & cos\gamma_e / cos\theta_e & sin\gamma_e / cos\theta_e \\ L'_\beta & 0 & L'_p & L'_r \\ N'_\beta & 0 & N'_p & N'_r \end{bmatrix} \tag{3.28}$$

where the primed moment derivatives are defined in [McRuer 1974].

3.2.4 Aircraft control systems

In the previous section, we have introduced aircraft models and we have prospected the different parts of such model. In this section, the objective is to use these models to apply the innovative ideas of this thesis in order to relax the hard-real-time assumptions based on the Worst Case Execution Time.

There are many characteristics in the aircraft that need to be controlled or stabilized, such as the responsiveness of the aircraft to maneuvering commands. Also there are different modes in the aircraft that determine this responsiveness. These modes are the short period, phugoïd, roll, dutch roll and the spiral modes.

The frequencies of these different modes tend to be sufficiently high that the pilot would find it difficult or impossible to control the aircraft if the modes were lightly damped or even unstable. Therefore it is necessary to provide automatic control systems to provide these modes with suitable damping and natural frequencies. Such control systems are called stability augmentation systems (SAS) [Stevens 2003, Stengel 2004]. Moreover, the augmentation systems provide the pilot with a particular type of response to the control inputs, it is called a control augmentation systems (CAS). An example of this would be the pitch rate CAS, in which the pilot's inputs are intended to control the pitch rate variable.

As it is undesirable for a pilot to have to pay a constant attention to the control of all of the modes simultaneously, an automatic control system is needed to provide pilot relief functions. This is implemented special control system that control different variables but also provides with other special functions such as automatic landing for instance.

The table 3.1 shows the different control functions that involve the three types of aircraft automatic control .

SAS	CAS	Autopilots
Roll damper	Roll rate	Pitch attitude hold
Pitch damper	Pitch rate	Altitude hold
Yaw damper	Normal acceleration	Speed / Mach hold
		Automatic landing
		Roll-angle hold
		Turn coordination
		Heading hold / VOR hold

Table 3.1: Aircraft automatic control systems

However, in this thesis, only the pitch rate control system is modelled and experimented.

3.2.5 The aircraft rigid body modes

When the longitudinal and lateral decoupling occurs (as in the previous section), it becomes feasible to manipulate the aircraft transformed state equation algebraically. Both sub-models are still of fourth-order and so the different rigid-body modes are obtained from the roots of the fourth-order dynamics' characteristic polynomials.

It has been shown in [Stevens 2003] that it was possible to extract separately the longitudinal aircraft modes: the short period and phugoïd modes. This implies additional decoupling in the dynamic longitudinal equation. Here, we will not show the mathematical demonstration of this decoupling; instead, the different modes equations are given directly in the following.

Note that, the lateral/directional equation does not separate into factors that clearly define each mode. Thus in practical case, approximations are derived that may describe an individual mode reasonably well and they need to be verified for applicability in any given case.

In the next paragraphs, the models of the different modes from longitudinal models are given.

The short period mode: From 3.26, the algebraic model of the short period is obtained as:

$$\begin{bmatrix} V_T - Z_{\dot{\alpha}} & 0 \\ -M_{\dot{\alpha}} & 1 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} Z_\alpha & V_T + Z_q \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} Z_{\delta_e} \\ M_{\delta_e} \end{bmatrix} \delta_e \tag{3.29}$$

The phugoïd mode: The algebraic model of the phugoïd mode is given by:

$$\begin{bmatrix} \dot{v_T} \\ \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_V + X_{T_V}\, cos(\alpha_e + \alpha_T) - \frac{X_\alpha \left[M_q \left\{ Z_V - X_{T_V}\, sin(\alpha_e + \alpha_T) \right\} - (V_{T_e} + Z_q)\left(M_V + M_{T_V} \right) \right]}{M_q\, Z_\alpha - M_\alpha (V_{T_e} + Z_q)} & -g_D \\ \\ \frac{M_\alpha \left(Z_V - X_{T_V}\, sin(\alpha_e + \alpha_T) \right) - Z_\alpha \left(M_V + M_{T_V} \right)}{M_q\, Z_\alpha - M_\alpha (V_{T_e} + Z_q)} & 0 \end{bmatrix} \begin{bmatrix} v_T \\ \\ \theta \end{bmatrix} \tag{3.30}$$

3.3 Case study: pitch rate control of a F16 aircraft

The considered case study will investigate only some of the aircraft control system shown in 3.2.4. This thesis considers only the pitch rate control augmentation system. The experiments on the other control system components can be applied in a similar way.

3.3.1 The F16 lateral and longitudinal numerical models

The generation of the numerical model of the F-16 can be done from the previous algebraic equations, under very specific conditions. There are many variants of these conditions that allow for instance to investigate the steady-state performance of the aircraft such as the specific fuel consumption rate, the rate of climb, various critical speeds for takeoff and landing, radius of turns and so on...

In the example, we will examine the trimmed level flight conditions over a range of speed.

Different set of trimmed conditions for level flight at sea level ($h = p^D = 0$) are given in table 3.2 for the F-16 aircraft. These values, and mainly the nominal conditions will be used for the simulation examples to illustrate the effectiveness of the analytic approach presented in this thesis.

Variable	Nominal	$x_{cg} = 0.3\bar{c}$	$x_{cg} = 0.38\bar{c}$
V_T (ft/s)	502.0	502.0	502.0
α(rad)	0.03691	0.03936	0.03544
β(rad)	$4, 0.10^{-9}$	$4, 1.10^{-9}$	$3, 1.10^{-9}$
ϕ(rad)	0	0	0
θ(rad)	0.03691	0.03936	0.03544
P (rad/s)	0	0	0
Q (rad/s)	0	0	0
R (rad/s)	0	0	
Thtl (0-1)	0.1385	0.1485	0.1325
El (deg)	-0.7588	-1.931	-0.05590
Ail (deg)	$-1, 27.10^{-7}$	$-7, 0.10^{-8}$	$-5, 1.10^{-7}$
Rdr (deg)	$6, 2.10^{-7}$	$8, 3.10^{-7}$	$4, 3.10^{-6}$

Nominal conditions: $h = 0$, $\bar{q} = 300$ psf, $x_{cg} = 0.35\bar{c}$, $\dot{\phi} \equiv \dot{\theta} \equiv \dot{\psi} \equiv \gamma \equiv 0$

Table 3.2: Trimmed flight conditions for the F-16 simulation model [Stevens 2003]

In order to facilitate the model generation, [Stevens 2003] propose a set of program to compute the different derivatives and generate the Jacobian matrices of the system.

The complete equation in 3.1 to 3.10 can be generated under the nominal condition described by the tabular 3.2. The generated aircraft model can be linearized and divided into two sets of equations which are respectively the longitudinal and the lateral dynamics.

They are described in the following paragraphs.

Description of a longitudinal control system: This gives the complete longitudinal model and allows to write the dynamical model for control design and analysis as a state model,

$$\begin{aligned} \dot{x} &= A\,x + B\,u \\ y &= C\,x \end{aligned} \tag{3.31}$$

and where the matrices A and B and C are given in 3.32. The states and inputs for the longitudinal model are as defined in 3.25.

$$A = \begin{bmatrix} -1,93134E-02 & 4,27761E+01 & -3,16936E+01 & -5,74869E-01 \\ -2,53929E-04 & -1,19825E+00 & 2,16840E-18 & 9,05037E-01 \\ 0,00000E+00 & 0,00000E+00 & 0,00000E+00 & 1,00000E+00 \\ 6,30022E-20 & 2,33148E+00 & 0,00000E+00 & -1,07769E+00 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 1,73762E-01 \\ 0 & -2,15050E-03 \\ 0 & 0,00000E+00 \\ 0 & -1,75598E-01 \end{bmatrix},$$

$$C = \begin{bmatrix} 3,98135E-03 & 1,92185E+01 & 0,00000E+00 & 1,48133E+00 \\ 0,00000E+00 & 5,72958E+01 & 0,00000E+00 & 0,00000E+00 \\ 0,00000E+00 & 0,00000E+00 & 0,00000E+00 & 5,72958E+01 \end{bmatrix}. \tag{3.32}$$

These states x are defined as given in 3.25 as $x = [V_T \ \alpha \ \theta \ q]^T$. The output vector y is composed of $[a_n, \alpha, q]$, where a_n is the normal acceleration, α defines the angle of attack and q the pitch rate.

The entries of the second and third line of the C matrix are only composed of conversion to units of degree for the angle, in order to keep consistency with the inputs. The first line is composed of the normal acceleration component with respect to the longitudinal states.

According to the considered flight conditions (see table 3.2), it is assumed that there is another regulator keeping V_T constant using the throttle input δ_t. Thus, in the model, the input δ_t is not considered and, as it is seen in 3.32 (Matrix B), the only single input is the elevator δ_e.

Description of a lateral control system: In an analog way, the lateral model of the aircraft is given in the same form as 3.31 with the system matrices defined as follows:

$$A = \begin{bmatrix} -3,07126E-01 & 6,30912E-02 & 3,65420E-02 & -9,91683E-01 \\ 0,00000E+00 & 0,00000E+00 & 1,00000E+00 & 3,70820E-02 \\ -2,31417E+01 & 0,00000E+00 & -3,66944E+00 & 6,63794E-01 \\ 8,98989E+00 & 0,00000E+00 & -2,55144E-02 & -4,75260E-01 \end{bmatrix}$$

$$B = \begin{bmatrix} 2,94369E-04 & 8,03674E-04 \\ 0,00000E+00 & 0,00000E+00 \\ -7,31610E-01 & 1,31217E-01 \\ -3,17866E-02 & -6,18718E-02 \end{bmatrix},$$

$$C = \begin{bmatrix} -5,02565E+00 & 0,00000E+00 & -8,12311E-03 & 1,19341E-01 \\ 0,00000E+00 & 0,00000E+00 & 5,72958E+01 & 0,00000E+00 \\ 0,00000E+00 & 0,00000E+00 & 0,00000E+00 & 5,72958E+01 \end{bmatrix}. \tag{3.33}$$

The state vector x and the control vector u are defined in 3.27. The output vector y is composed of $[a_y, p, r]$,where a_y is the lateral acceleration, p and q are respectively the roll rate and the yaw rate.

3.3.2 The pitch rate control system

A pitch control system is composed of three components, the pitch damper, the pitch attitude hold and the pitch rate control augmentation system. In fact, when the aircraft is under manual control (as opposed to auto-flight control or autopilot), the stability augmentation systems are in most case the only flight control systems needed. However, for high performance aircraft, specialized control augmentation systems (CAS) are needed. This is admitted to be of a crucial importance in a phase where maneuverability and "agility" in the aircraft are needed.

Another common mode of operation for a pitch-axis CAS is a pitch-rate control system. When a situation requires precise tracking of a target (missile for instance), it has been found that a deadbeat response to pitch-rate commands suits well to the task. Moreover, control of pitch rate is also the preferred systems for approaches and landing.

Thus, we have chosen among the different control systems presented in table 3.1, the pitch rate CAS.

The block diagram of the pitch rate control system is shown in figure 3.6.

The longitudinal dynamics corresponding to the nominal flight condition in table 3.2 is used. Thus, the numerical longitudinal model is given by 3.32.

By taking the short period approximation as shown in 3.29, we obtain the pitch axis plant. In reality, the longitudinal equations in the considered condition do not exhibit a short period mode, but the α and q equations are only loosely coupled to v_T ad θ (as seen in the longitudinal model 3.32, they are coupled only by value approaching 0.

The complete pitch control system is shown in the figure 3.6, where the actuator input is u, and r is the reference input corresponding to the desired pitch rate. Thus the performance output is z corresponds to the pitch rate q. The measured outputs are q and α. In practice,

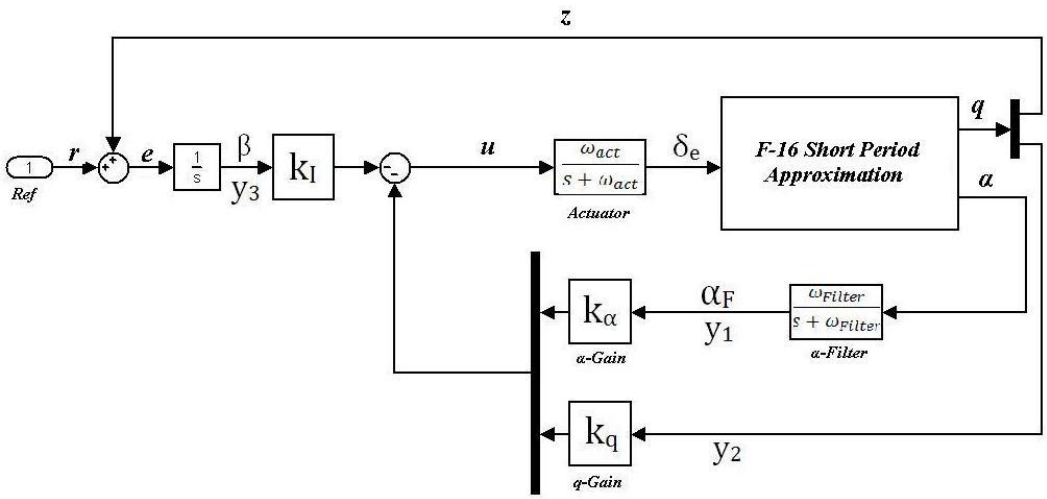

Figure 3.6: Pitch control block-diagram

the measured values of α are quite noisy and the role of the α-filter is to provide filtered measurements α_F of the angle of attack. To ensure zero steady state error, an integrator was added in the feed-forward channel. β is the output of the integrator.

The elevator actuator for the δ_e inputs and α-filter dynamics will be those of the figure 3.6 and their values have been taken from the original F-16 model report [Nguyen 1979] as both simple-lag filters with time constants respectively equal to $\tau_a = \frac{1}{20,2}$ and $\tau_f = \frac{1}{10}$.

The states and outputs of the augmented system (plant plus compensators) are:

$$x = [\alpha\ q\ \delta_e\ \alpha_F\ \beta]^T \qquad y = [\alpha_F\ q\ \beta]^T \tag{3.34}$$

The plant (reduced to the short period approximation for the pitch-rate control system) augmented with the actuator dynamics, the α-filter dynamics can be obtained in the form of 3.31 a classical state space model, where the short period dynamics are extracted from 3.32 by taking the α and q components (or computed directly from 3.29 at the considered flight conditions), and where the system matrices are defined as:

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 & 0 & 0 \\ 0.82225 & -1.07741 & -0.17555 & 0 & 0 \\ 0 & 0 & -20.2 & 0 & 0 \\ 10.0 & 0 & 0 & -10 & 0 \\ 0 & -57.2958 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 20.2 \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 0 & 57.2958 & 0 \\ 0 & 57.2958 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.35}$$

The final closed loop analytic model is a state space model augmented with the additional dynamics of the α-filter, the actuator and the integrator and taking into account the reference signal. The control law is an output-feedback of the form:

$$u = Ky = -k_\alpha \alpha_F - k_q q - k_i \beta$$

The different feedback gains k_α, k_q and k_I are selected to yield a good closed-loop response to a step. They are obtained by classical LQ output feedback techniques as $K = [-0.0807; -0.475; 1.361]$. However, the details of the computation of these controller gains are out of the scope of this thesis, but can be found in [Stevens 2003]. The equivalent digital controller matrix, $K = [-0.04238; -0.4098; 0.8426]$, is obtained through a Tustin transform for a nominal sampling period $T = 0.02$ sec.

3.3.3 Description and results of the case study

In this work, we only present the application of some of the weakened scheduling scenarii proposed in section 2.2.1 on the system above. The parts that we have done, were the *Abort*, *Abort'* and *Skip* cases. However, other scenarii can be done by doing minor modifications in the LMIs proposed for the stability conditions.

To describe correctly the experimentation, let us define $\varepsilon = \frac{T_{slot}}{WCET}$, where T_{slot} is the reduced allocated time and $WCET$ is the allocated time base don the worst case execution time.

It is important to mention that the reduction of the allocated time slot from the $WCET$ value to T_{slot} will inevitably lead to a performance gain (measured by an increased robustness) due to the reduction of the overall latency for case 1 ("Abort") and also due to the reduction of both the sampling period and the overall latency for case 2 ("Abort'") and at last, we will look at the case 3 called ("skip"). However, the counterpart is the fact that, the probability of deadline overrun while executing the software control task is higher, as figure 2.2 shows. Thus, according to the deal-line overrun handler strategies, the objective is to consider a value of the execution time where the resource utilization is enhanced without threatening the stability.

For the three cases, the stability conditions of Theorem 3 are used to find the relationships between the allocated computing time slots for the software control tasks, T_{slot} and the maximum number of consecutive deadline misses before unstability occurs. This is done for different value of the uncertainty factor μ in order to take into account a simplified aggregated uncertainty on the system's parameters.

The results of the computation are plotted in figure 3.7.

The analysis of Case 1 (top of Figure 3.7) show that the tolerance of the feedback controller w.r.t. deadlines misses, measured by N , increases when T_{slot} is decreased (therefore also decreasing the systematic latency). It also shows that increasingly uncertain systems, with growing values of μ, are less tolerant w.r.t. deadlines misses.

In Case 2 the number N of sustainable consecutive deadline misses is even improved, as the reduction of T_{slot} induces a decrease in both the delay and the sampling interval. In case 2, when the delay is reduced of a quantity δ, we also reduce the sampling interval by the same amount. The main difference between case 1 and case 2 is the fact that in case 1,

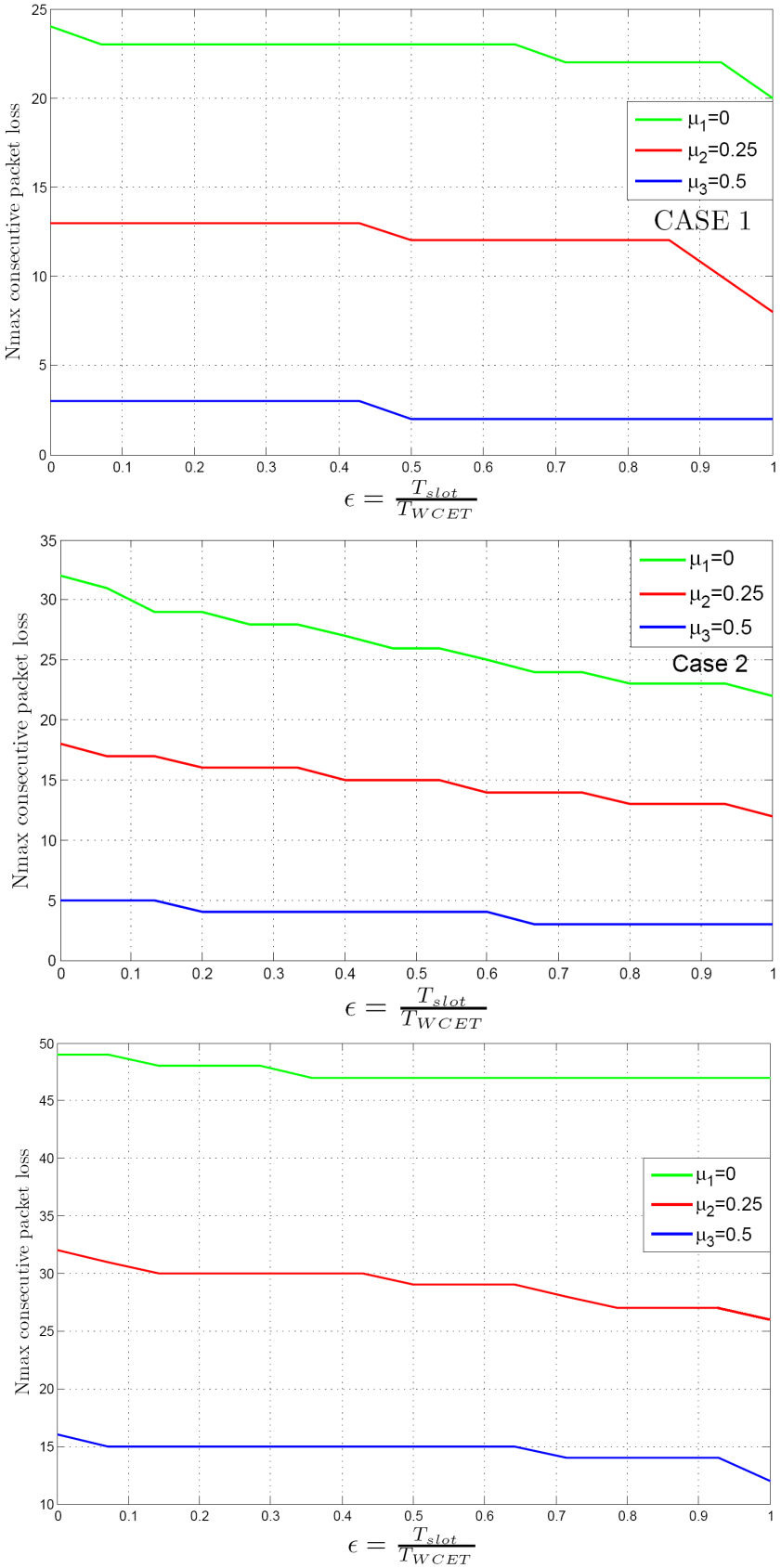

Figure 3.7: Maximum allowed consecutive deadline misses for the three cases

only the delay T_{slot} has been reduced while in case 2, both the delay T_{slot} and the sampling interval T have been reduced. This is expected to enhance the performance of control and the QoC but also provides an improvement of the computing resource utilization.

Now, assume that a distribution of the execution time of the control task is given as in figure 1.15. This distribution is plotted in black in figure 3.8, in which the probability of reaching deadlines decreases from 1 to 0 as the scheduling factor ε decreases from 1 (meaning $T_{slot} = WCET$) to a minimum corresponding to $T_{slot} = BCET$, which is the Best Case Execution Time. By assuming that the execution time of task instances are independent from sample to sample, the probability of reaching the maximum tolerable number of consecutive deadline misses (and thus become unstable), are computed gathering the execution time distribution and the N number of allowed deadlines miss. For example, in case 1, $\varepsilon = 0.5$ leads to a deadline miss probability $p = 0.92$ on the black plot. For $\mu = 0$ and $\varepsilon = 0.5$, the green plot for Case 1 in Figure 3.7 gives $N = 23$. Therefore, assuming that the consecutive execution times are independent, the probability of missing N consecutive deadlines is $0.92^{23} = 0.147$ (on the green plot in Figure 3.8). The results for the full data range are given in the same figure for the different values of the uncertainty factor μ.

Thus, for a given scheduling scheme, a probability function of the execution times and an uncertainty assumption, it is easy to apply this approach to the safety analysis of the execution time. This would involve for instance, the computation of ε, the scheduling factor corresponding to a given failure rate. Thus, it would permit to compute the failure probability for a given value of ε (or T_{slot}) and μ.

In figure 3.9, we see for case 1, that for a value of the uncertainty $\mu = 0.25$, we have a failure probability of 2.3610^{-9}, that is to say, to see the system going unstable, when $\varepsilon = 0.857$, which means that the allocated execution T_{slot} has been reduced to about 85.7% of the worst case execution time.

3.4 Validation of the approach by simulation

In this section, we approach the validation of the proposed approach in two steps:

The first step consist of verifying the numerical results obtained by the LMI of section 2.4.3 by using simulation tools that allow to simulate execution time overrun. The objective is to verify whether the obtained results are the same or otherwise fairly near the numerical results . Thus this first part will be applied on the same linearized pitch control model which was used in the numerical analysis.

The second step consist of a nonlinear simulation which should allow for the validate the approach by verifying the correctness of the obtained results using the same controller and simulation tools as in the first steps but this time using the nonlinear complete aircraft. In fact, the output of the framework is composed of a value of the allocated time slot for the execution of the control tasks. This value has been analytically and numerically determined. However, non-linearities in the system model and the coupling of the different aircraft axis, may induce some changes in the results. Modeling these phenomenon would help provide a more realistic value of the allocated time slot, T_{slot} for the task.

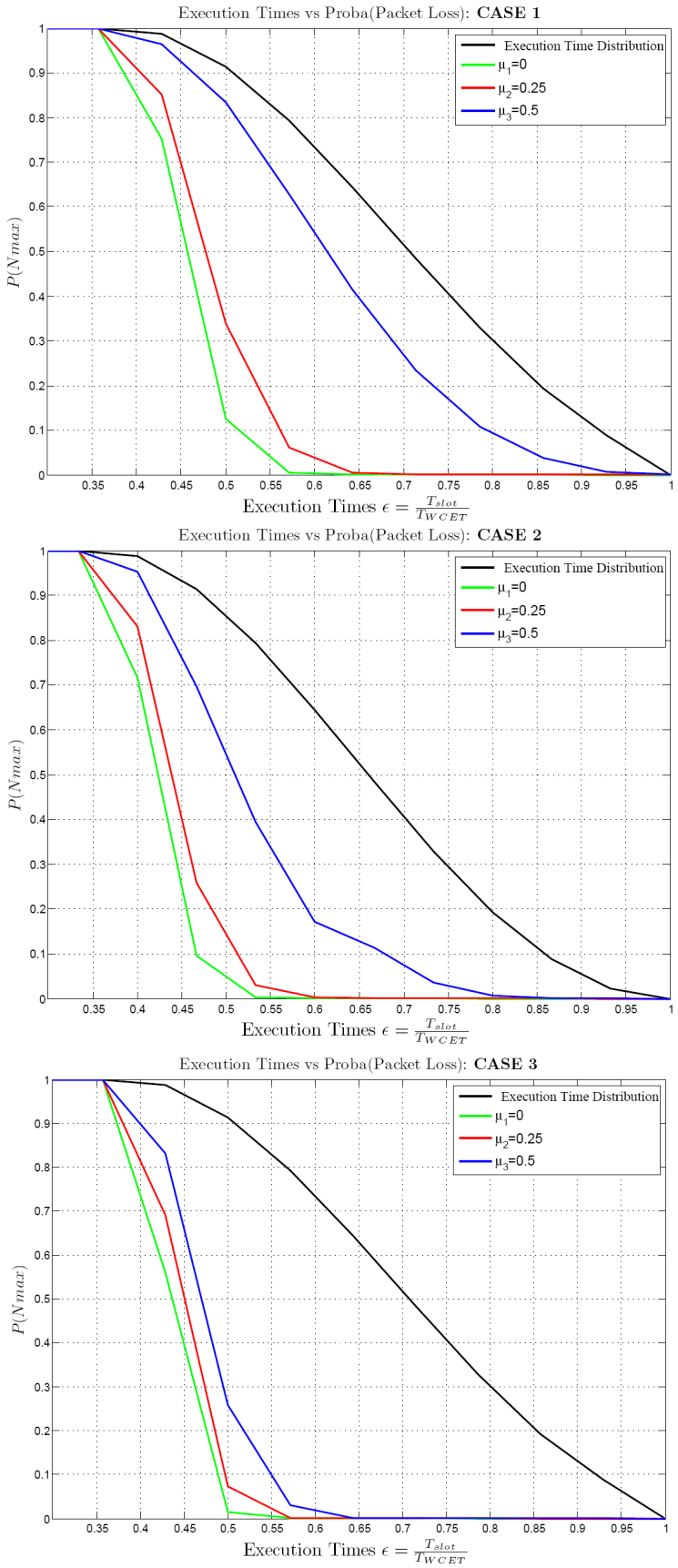

Figure 3.8: Probability of reaching the maximum tolerable consecutive deadline misses

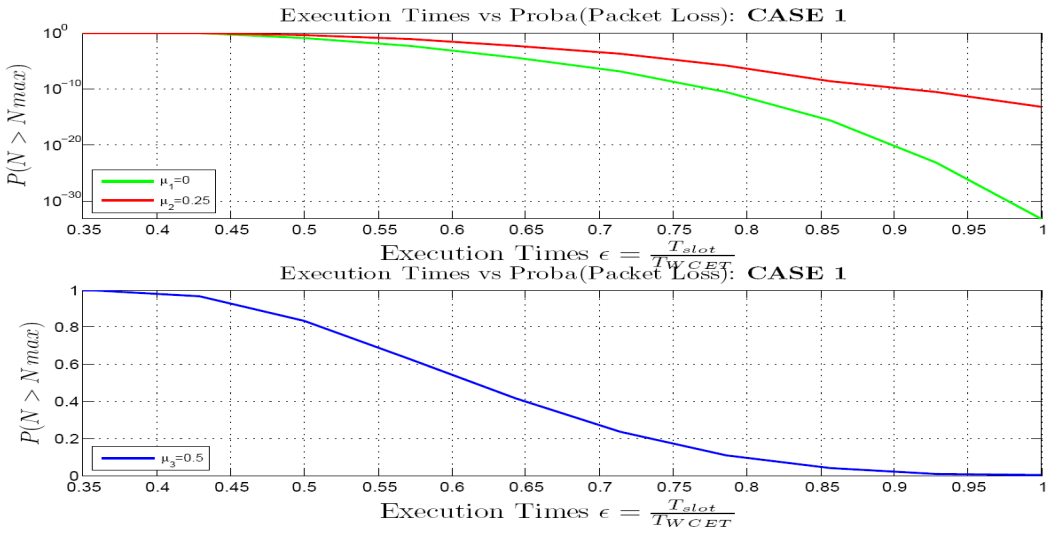

Figure 3.9: Zoomed $P(NMax)$ vs Execution Time: case1

3.4.1 Truetime simulation

Truetime kernel Truetime [Cervin 2009] is a Matlab/Simulink-based simulator for real-time control systems. TrueTime facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. The interests of using Truetime is the fact that the blocks composing its subsystems can be connected with standard Simulink block diagrams in order to form a real-time control system. the main feature of Truetime is also its possibility of co-simulation of the interaction of continuous-time systems (real-world physical models) and computer systems in the form of control software tasks and network communication. The API (Application Programming Interface) provided by Truetime also provides many possibilities that allows to simulate interrupts and event handlers such as deadline misses handlers or WCET overrun handlers, or network and execution time latencies.

Figure 3.10 shows the Truetime simulator, which is composed of different Simulink libraries with complex functionalities which are provided by a collection of C-MEX files.

In this library, the TrueTime Kernel is of interest for our simulation because this allows us to simulate an embedded operating system or a real- time kernel, which allows to implement task execution. Among others, this block offers very interesting functionalities such as A/D and D/A converters, a network interface, external interrupt channels (for external connections) which all should allow us to simulate a complete computer-controlled system.

Internally, the kernel has the ability to execute user-defined tasks, many interrupt handlers and offers many useful data structures that are ready- to- use like time FIFO, timers, monitors, semaphores, mailboxes, etc etc.

Configuring the Truetime kernel blockset involves specifying the control system by specifying the number of inputs and output of the considered controller, the scheduling policy of the control tasks and creating these tasks and their associated interrupt handlers.

The TrueTime kernel allows to create different types of tasks such as periodic tasks

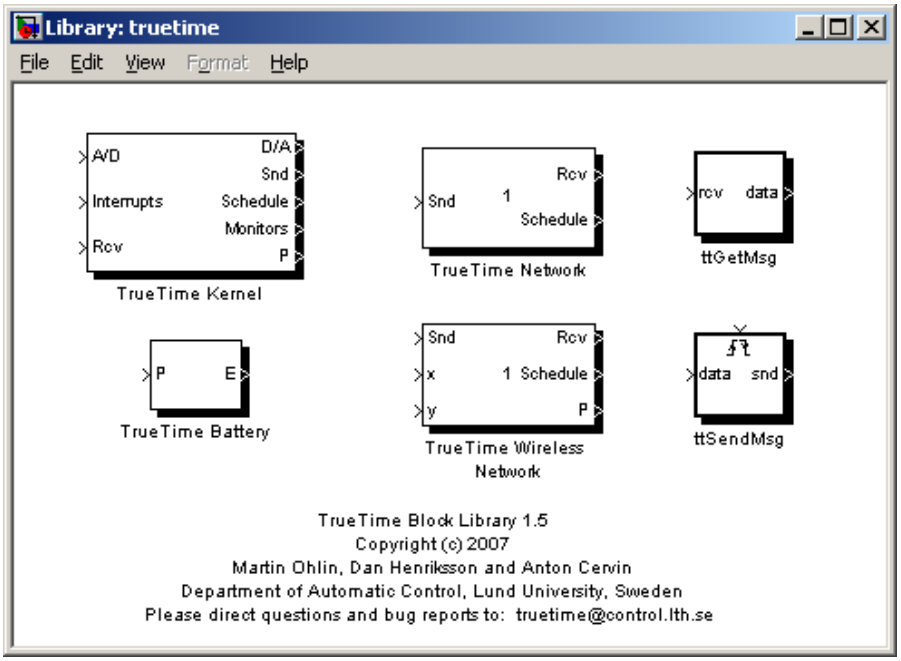

Figure 3.10: The Truetime library

or sporadic tasks. It also provides different real-time primitives that allow to manipulate software tasks such as creating a new sporadic tasks at runtime, killing a running task, waiting for a task to finish using semaphores or monitors or primitive that allow to make tasks non preemptible.

Among many functionalities, the tool provides means to implement or to use pre - defined priority functions for classical scheduling algorithms such as fixed- priority, deadline monotonic or earliest deadline first [Cervin 2006].

In brief, Truetime makes it possile to investigate the impact of delays, jitters lost samples, etc, on control performance.

Details on the packages and their functionalities can be found in [Cervin 2009].

3.4.2 The Simulink model

To test our results, we implement the control system presented in 3.6 with classical Simulink block diagrams as presented in figure 3.11.

The slight difference between the model is the insertion of a transport delay block in our Simulink model. This transport delay will help us simulate computation delay.

Although the model presented in figure 3.11 may respond to our needs, it is extremely difficult to simulate phenomenon such as execution times, input output, control tasks directly in Simulink.

Therefore, in the remainder of this chapter, we have integrated the linearized plant into a Truetime controller. In order to obtain the same basis for the analysis, we have used the same controller as that used for the computation shown in section 3.3.2

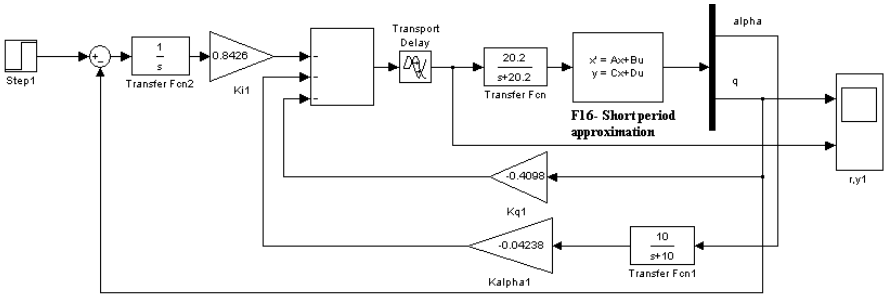

Figure 3.11: Pitch rate Control system implemented in Simulink

Figure 3.12, shows the final simulation model, when integrated with a TrueTime kernel block-set.

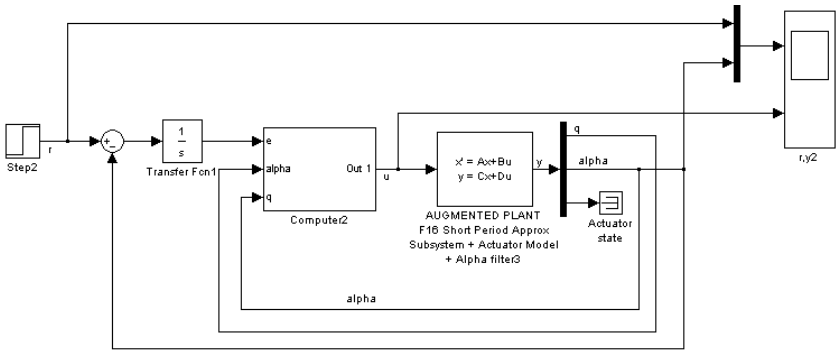

Figure 3.12: Pitch Control System with a TrueTime Controller

The added value of the simulation model as presented in figure 3.11 is that the True-Time system already provides a base framework to apply some simulation parameters such as sampling periods, time-delays and latencies. Truetime also provides some primitives for the management of deadline misses and deadline overruns, the management of computing loads simulations.

3.4.3 Simulation using linear plant

The simulation using the linearized plants are interesting because they should be able to confirm the numerical results obtained by the LMI, as the plant is strictly the same but with the controllers are in the form of software tasks simulated by the TrueTime Kernel. The values of these controllers stay the same as those used in the LMIs computations.

Moreover, it greatly eases the nonlinear simulation because, it can be done by only changing the plant into the nonlinear plant model without changing any controller parameters.

Although many cases have been analyzed in section 3.3.3, we provide here, only the Case 1 results. The other cases, such as case 2 (*Abort*) or the case 3 (*Skip*) can be derived by modifying the simulation parameters of the Truetime controller and the software tasks.

$\mu = 0$ The first case to be presented is the one where the plant parameters are perfectly known without uncertainties, combined with the different values of $NMAX$. The polytopic model introduced in chapter 2 would only have one vertex. Thus, only the number of sample losses $NMAX$ are displayed for this case.

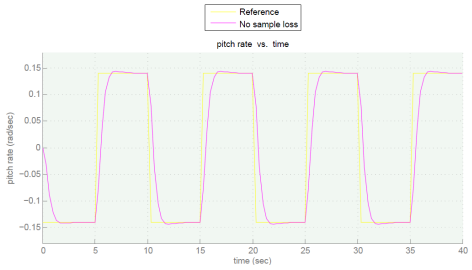

Figure 3.13: No sample loss

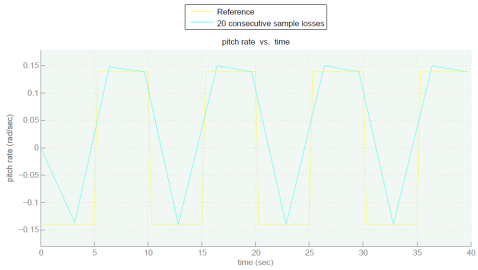

Figure 3.14: 20 consecutive sample losses

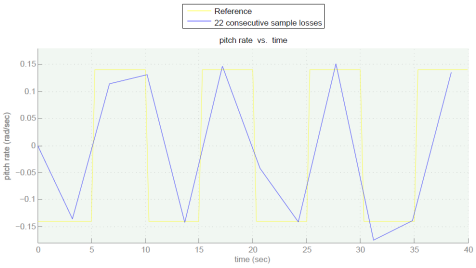

Figure 3.15: 22 consecutive sample losses

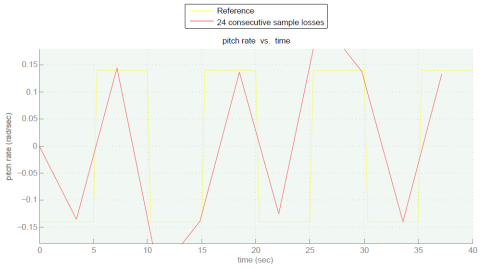

Figure 3.16: 24 consecutive sample losses

Figure 3.17: Simulation results for $\mu = 0$

By analyzing the different plots shown in figure 3.17, it can be seen that the numerical and analytic results obtained in section 3.3.3 are confirmed, as illustrated in figures 3.7 and 3.8. However, although the system seems to be asymptotically stable, it is noticed that the performance and quality of control of the system is very poor for large values of N. Also, note that the pessimism of the analytic results compared with the stability limits found by simulations show the conservatism due to the Lyapunov-based analysis.

$\mu = 0.25$ **and** $\mu = 0.5$ The results associated with the other values of μ are also confirmed. In these results, the simulation was run all over the different models composing the polytopic model with the values of $\mu = 0.25$ and $\mu = 0.5$.

It may be seen in the comparison of these figures also that the effects plant uncertainty μ are not as severe as those of the delays. On one hand, in the case where the system has no sample loss, the performance of the system is good. On the other hand, even if the uncertainty of the plant is zero, the performance degrades very fast when subject to sample losses.

The result of the linear simulation for the case of $\mu = 0.50$ is presented in figure 3.27.

By looking at the result of the linear simulation for a value of $\mu = 0.50$, we see that indeed, the control performance is acceptable for the case where the system is not subject to sample loss, even if the plant uncertainty is quite large.

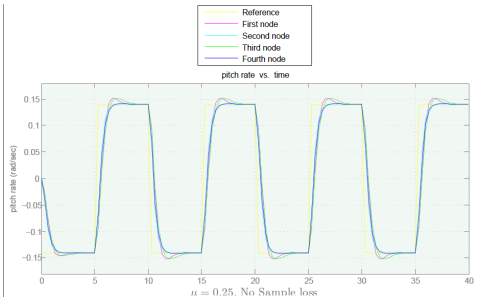

Figure 3.18: No sample loss

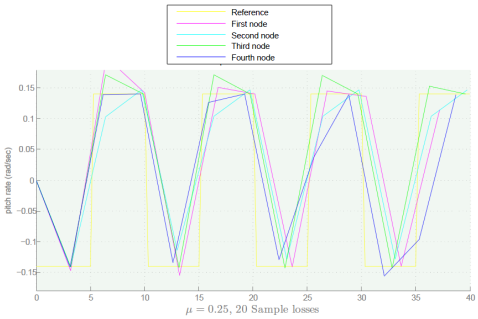

Figure 3.19: 20 consecutive sample losses

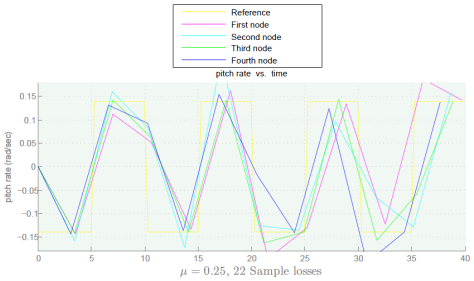

Figure 3.20: 22 consecutive sample losses

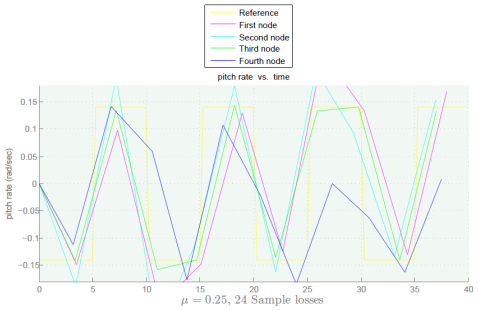

Figure 3.21: 24 consecutive sample losses

Figure 3.22: Simulation results for $\mu = 0.25$

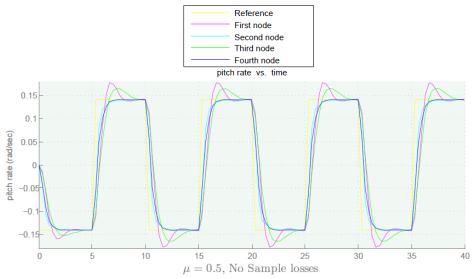

Figure 3.23: No sample loss

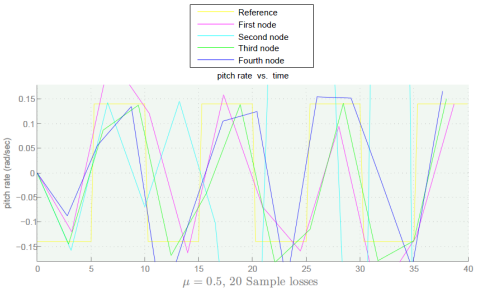

Figure 3.24: 20 consecutive sample losses

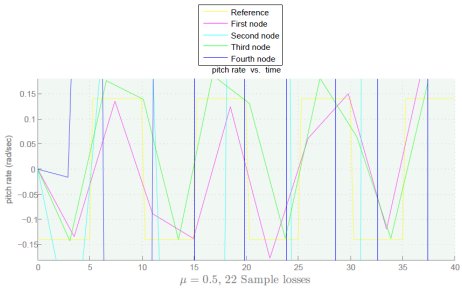

Figure 3.25: 22 consecutive sample losses

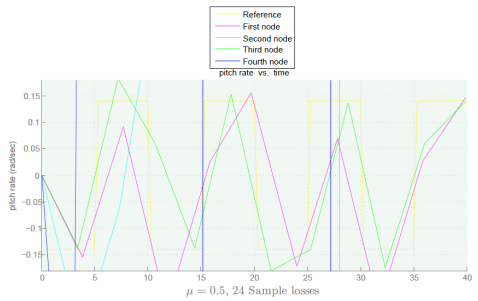

Figure 3.26: 24 consecutive sample losses

Figure 3.27: Simulation results for $\mu = 0.50$

3.4.4 Nonlinear simulation

In this section, the idea is to apply all the analysis done in the previous sections on the nonlinear model of aircraft. To do so, it is necessary to develop a nonlinear aircraft model based on the equations studied in section 3.2.

The aircraft nonlinear model is a C-language program which implements all the motion equations that have been exposed in section 3.2.2.

The nonlinear system is depicted in figure 3.28 and has been directly linked with the same pitch-rate controller which has been used in the previous simulation of section 3.4.1. Moreover, an engine controller has been added to the model. The inputs of the nonlinear aircraft, corresponding to the other aircraft axis and modes (mainly lateral axis and phugoïd), have been set to constant or zero, although, internally, they were set to the trimmed value computed from the flight conditions.

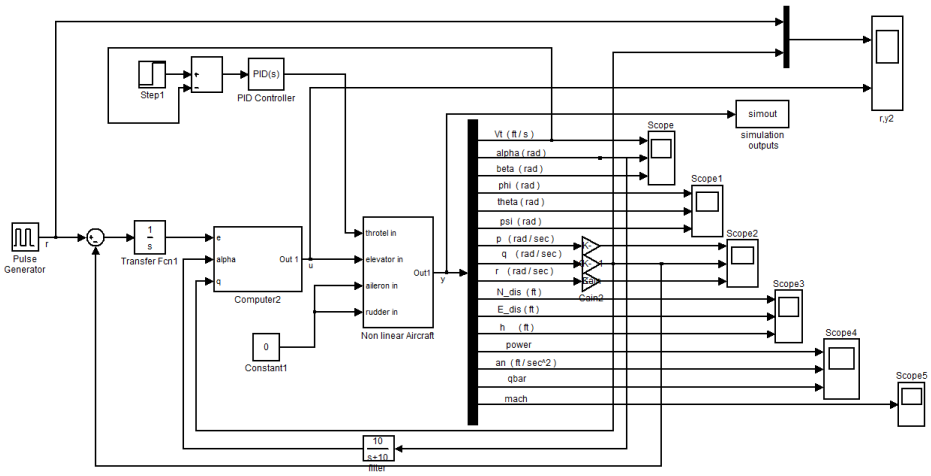

Figure 3.28: Nonlinear simulation of aircraft control systems

In order to respect the assumptions of constant velocity and to delete undesirable coupling effects of the different aircraft axis (pitch, roll and yaw), a throttle controller based on a hand-tuned PID has been integrated to the system. The initial value of the other inputs (apart from the elevator input) has been initialized to 0, although internally, these inputs are initialized at the flight trim condition values (see table 3.2).

This nonlinear simulation is intended to demonstrate that with the same parameters, it is possible to confirm the results of the LMIs, giving the computed values of $NMAX$. Moreover the nonlinear model allows to integrate many aspects of the real model into the simulation model, such as the plant non-linearities, axis coupling, actuators saturations and bandwidth, ...

The associated nonlinear simulation results for the case 1 is presented in figure 3.33.

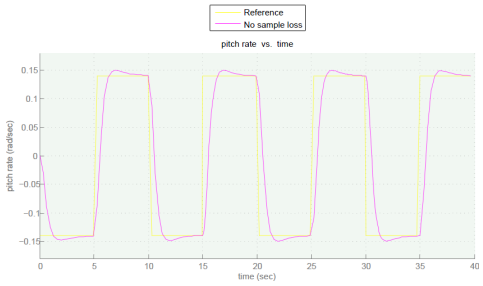

Figure 3.29: No sample loss

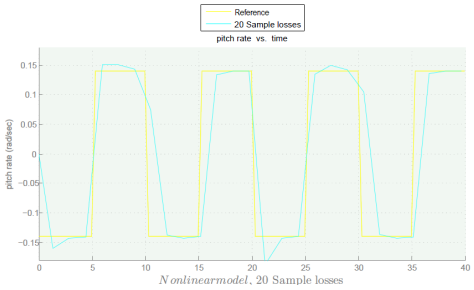

Figure 3.30: 20 consecutive sample losses

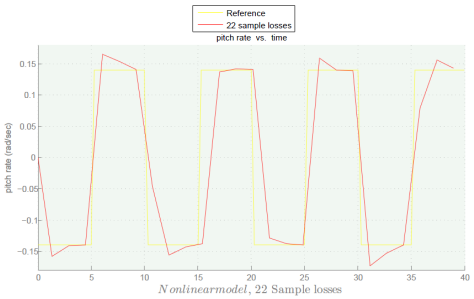

Figure 3.31: 22 consecutive sample losses

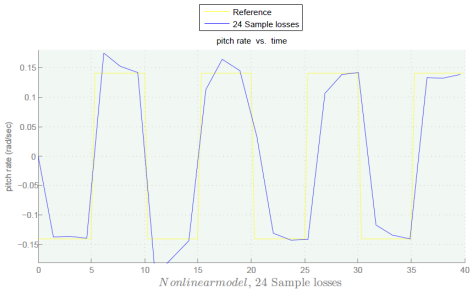

Figure 3.32: 24 consecutive sample losses

Figure 3.33: Nonlinear simulation results

3.5 Conclusion of Chapter3

This chapter was dedicated to the application of the analytic process proposed in chapter 2 with a realistic control system. It has been shown in chapter 2 that the timing problems of computing control tasks and control robustness should be considered jointly. The case study presented in this chapter can be used as a practical basis to support this idea, where both analysis and simulations show the effects of timing and deadline misses on the robustness and performance of control systems under timing uncertainties.

For this objective, the pitch-rate control system included in an aircraft control system have been considered. A detailed description of these control systems have been presented at this beginning of the chapter, while in the second part the analytic method was applied to the particular aircraft control system. The third section of the chapter confirmed these results using simulations based on both the linear and nonlinear models of the aircraf, to be compliant with the framework presented in section B and depicted in figure B.1.

CHAPTER 4

Conclusion

Contents

4.1 Summary

In this thesis, a new approach for the analysis of real-time computer-controlled systems has been proposed. The main problem focuses on the determination of the allocated time for control tasks, a problem arising at the frontier between real-time computing and control design. The proposed framework provides an analytic approach to correctly evaluate the time slots allocated for the execution of periodic control tasks, taking into consideration the real-time capabilities and fault tolerance requirements of the computer-controlled system.

The allocation of time for critical control tasks is of prime interest in domains such as embedded avionics systems because they govern the real-time programs implementation constraints, with further consequences at the overall system design level (weight, electrical supplies, cooling capabilities, obsolescence management, ...). Traditionally, the tasks scheduling is performed according to hard real-time constraints, and the time allocated for the execution of control tasks is chosen equal to the Worst Case Execution Time (WCET) of the executable control code on the target processor. This worst-case based approach induces several drawbacks.

Firstly, the WCET must be evaluated through computation methods which complexity are growing with modern processors architectures. Secondly, even when the WCET is correctly evaluated, the dispersion in execution times makes the worst case a rare case, so that this scheduling scheme inevitably results in a systematic waste of computing power due to idle cycles at the end of the allocated slot.

Taking into account the robustness of feedback control, an analytic framework and methodology is proposed to improve the real-time scheduling scheme of control tasks and to decrease the over-dimensioning of the computing resources. The approach is grounded on recent advances in the domain of networked controlled systems and time-delay systems. The methodology allows to improve the resource utilization rate, by off-line computing the value of the allocated time for control tasks, which ensures the stability of the plant despite computing slots smaller than the WCET.

A special type of model was examined, where control is done on a discrete-time scale while the plant is still considered as a continuous-time system. The control task durations

are considered as a special type of delay in the system, namely the input delay. From an uncertain linear model of the plant, the determination of delays (hence time-slots) compliant with the system stability is carried out using the Lyapunov Krasovskii stability analysis. In this framework, the stability condition allows for computing the maximum number of consecutive deadline misses which can be tolerated by an uncertain system while keeping stability.

An important characteristics of the methodology is that it includes fault-tolerance capability. In fact, its objective is to keep control stability while managing the computing loads and resource utilization at a specified level. Using the knowledge about the distribution of execution times for a given control task, a trade-off can be carried out between the required control performance for the system and the resource utilization.

It is an example showing that feedback control systems are robust with respect to timing uncertainties and deviations from a rigid pre-defined scheduling pattern. In fact the approach gathers in a single framework robustness w.r.t. timing uncertainties (such as occasional samples misses) and robustness w.r.t. the plant's parameters uncertainties. Therefore, relying on feedback control robustness allows for slackening the commonly-used hard-real time constraints while preserving the control system performance and improving the computing resources use.

The effectiveness of the methodology is illustrated through a case study based on the flight control system of the F16 aircraft, and more especially on the pitch control. Beyond the analytic studies, the method has been successfully evaluated on a simulation model of the F-16 aircraft, under Matlab and TrueTime. Moreover, although the example focus on the allocation of time slots for control tasks in flight control systems, it can be applicable to other computer-controlled systems involving feedback controllers implemented by software tasks.

4.2 Perspectives

The following research directions represent possible extensions of the present work.

Real-time implementation feasibility. In this work, the problem of time allocation for control tasks was considered among others. As the allocated times for the control tasks are smaller than the WCET, it may happen that overruns occur occasionally. Several overrun handling strategies were proposed in this thesis but the real-time implementation feasibility of the various handlers was not examined. For example, the "Abort" solution proposed in case 1 (see section 2.2.1) is easy to handle from the control stability point of view. However, this solution may be difficult to implement, as it requires that, when aborted and restarted, the control task is correctly re-initialized. In particular, the context of the task, e.g., including the registers of recursive filters, must be restored coherently with the management of the new sensors inputs. On the other hand, the "Skip" case, where the control task runs to completion and skips the next execution in case of overrun, seems far simpler to implement using current software tools. However, as the execution time of the currently running control task can be sometimes several times larger than the allocated slot, the overrun handler must be guarded by a upper limit for acceptable successive skips,

which must be itself evaluated and implemented. Once again, trade-offs between the gains from the control point of view and the computing feasibility need to be carried out to find cost-effective solutions.

Finally the compatibility of this kind of flexible scheduling approach w.r.t. the current certification approach must be investigated. Conversely, considering the robustness of feedback controllers w.r.t. timing disturbances, new degrees of freedom for real-time control implementation appear, and the resulting new viewpoints about fault-tolerant control are expected to provide new inputs in the certification process.

Quality of Service statement As seen, the final part of the methodology and the analytic process consists in finding a trade-off between the gain in computing power that can be obtained from the reduction of the time allocated to control tasks and the loss of performance due to the occasional loss of samples. Solving a too general control/scheduling co-design problem in closed form is out of range of analysis, but restricted assumptions sets can be used to investigate more specific case studies. It is expected that well formalized cost functions might model the behavior of a real-time control system, and might lead to design effective and convincing flexible scheduling schemes, even for critical control systems. The idea is to gather the control performance (as a Quality of Control function) and the computing/networking performance (as a Quality of Service function) to minimize a unique cost function, able to capture both the control and computing requirements.

Improvements of the stability condition. In this work, the stability conditions are expressed in terms of asymptotic stability. This means that stability is guaranteed asymptotically with no consideration of other criteria. An extension of these conditions to express exponential stability would allow to consider integrating more QoS criteria. As any exponential stable system is also asymptotic stable, the exponential stability condition would moreover guarantee an exponential decay rate which can be interpreted as a speed of convergence of the system.

Handling of non-linearities and uncertainties In the current work the model uncertainties are aggregated in a unique multiplicative factor in the linearized model. In reality the error margin on the plant parameters are not equal, in particular because the various parameters represent different physical objects. As the plant uncertainties have a direct impact on the control robustness margin, it is worth studying more elaborated models, and to adapt the stability analysis method, to better discriminate the knowledge about the different parameters of a complex and heterogeneous plant.

Besides parametric uncertainties, complex systems such aircrafts are non-linear systems. In fact some non-linearities can be considered as varying parameters of a Linear Parameter Varying System (LPV). A main advantage of LPV systems is that well established control theory for linear systems, e.g. robust control design, can still be used for some non-linear plants. Compared with classical linearization, LPV modeling enlarges the range of validity of linearized models. Therefore it is appealing to apply our flexible real-time control and scheduling methodology to LPV models of complex systems.

APPENDIX A

Worst Case Execution Method: State of Art

Contents

The methods used to evaluate the WCET of a program is divided into 3 main parts. There are dynamics methods based on measurements or on probability patterns of execution time, static methods based on the analysis of the code and/or the executable software, and hybrid methods based on more or less the combination of both methods. We will give concise details on each methods in the following paragraphs.

A.1 Introduction to WCET Methods

In terms of control implementation, one of the most important requirement is about safety (looking back at the different legislation and requirement for safety critical systems presented 1.4). Among the many different regulations required to be done in the stringent development process of safety critical software, a timing analysis if required. However, the approach based on worst case is only a mean to attain the objective because it was the most viable and the requirements have not explicitly required that the analysis must use worst case execution time. This happens because it avoids to implement fault tolerance to deadline which might be more difficult. However, based on the design requirements of the designers of high-level system (meaning the whole control system from the end-to-end point of view), software designers and testers use the WCET (Worst Case Execution Time) leading to an overestimation of the need of the system in terms of size, computing power etc etc.

But before that, we will have a look at the different methods used for the computation of WCET. The chief characteristic of hard real-time systems is the requirement to complete the computation within a given time or by given deadline. To ensure reliability of execution, and ability to guarantee that the computing units (microprocessors or microcontrollers) are

able to hold the workload which they are assigned to, it is necessary to make a timing analysis. However, if working with WCET-based assumptions, finding a tight upper bound is important to avoid conservative implementations. This problem is in general undecidable due to the halting problem, there must be some conditions and constraints to satisfy to make it decidable.

In major practical cases, WCET-based approaches are used. There are three main approaches to WCET computation.

A.2 Dynamic Methods

Dynamic methods are the most commonly used method in the determination of WCET. This methods utilize a real program execution to obtain the *estimated WCET*. This method is called dynamic because of the use of real software with different input datas. The general principle is straightforward: the software to be analysed is executed with all possible set of input datas, and for every execution the running time is measured. This is why this method is also called "Measurement method". It is important to notice that the measurements must be done while running the software on the real microprocessor target or an adequate simulator of the target. In this context, *"input data"*, means the data which the program is waiting for to run correctly. For instance, if the code is a given subprogram, the input datas are the parameters of this subprogram. [Deverge 2007] give a comprehensive introduction to the measurement method for the WCET estimation. At first sight, many problems can occur in this methodology, but we will only give some of them:

- During the generation of all possible software input, it is hard to guarantee that every possible case has been taken into account. This leads to a problem of safety insurance and confidence. Moreover, if all cases were to be taken into account, the possibilities grow so fast that the resulting combinatorial explosion will render this analysis impossible. A solution to this problem is proposed by [Colin 2003] as to rely on the human expertise to generate the relevant test cases in order to obtain the WCET. However, we think that this is solution may be error prone and not safe. Another solution to this problem is to consider automatic test case generations. One of the related work on this field is proposed by [Wegener 2001] where the authors propose to choose the test cases according to special criteria automatically using a method called Evolutionary Testing. However, results are promising but not yet mature for safety critical systems. On the other hand, the practical interest of this work is that it permits to eliminate test cases by finding a lower bound of the WCET. It allows then a complementary association with static methods which give instead an upper pessimistic bound. Other application in many industrial practice solve this problem of test coverage issue by associating unit testing and timing analysis. This consider using the unit test cases (which are supposed to cover every execution path of the software if the test cases are well defined) along with an instrumented version of the program (for e.g a timer...) so as to permit measuring its execution time while the test cases are running.

- It is also difficult to demonstrate formally that at least one of every single execution based on each input has gone through the longest (worst case feasible) execution path of the program.

- Real conditions for a realistic execution is difficult to mock up. This means that for every execution context of the program, there can be a worse case that can happen in the real use. It becomes very expensive, in terms of resources to mock up these different conditions, compared to the obtained precision.

- The variability of the execution-time when the target processor is a complex processor (for e.g: with a cache memory) because the cache latency can be very small compared to the central memory latency. This leads either to an overestimation (which may not be bad) or to an over-optimistic execution time (which can be dangerous for safety critical).

- Adequate and precise hardware/simulators are not always available. An alternative can be the use of Instruction Set Simualtors ([Ratsiambahotra 2010] for precision on ISS).

Many variations and improvements of this method has been proposed to solve some of the issues above (for e.g [Lindgren 2000]. Among different issues, the main improvements can reside in the automatic generation of adequate test cases to ensure that these test cases can really cover the Worst case execution scenario but also to avoid a brute force approach based on generating all possible datas because it is know to be impossible [Colin 2003, Lokuciejewski 2011]. Improvements were also done (but mainly in many industrial practice as this method was greatly used in the) in the choice of a correct safety margin in order to improve the level of confidence that can be put this method without being too conservative [Lokuciejewski 2011].

A.3 Static Analysis Methods

Static program analysis is a generic method to determine properties of the dynamic behavior of a given task [Wilhelm 2008a]. The principle of static analysis is to make an off-line analysis of the program without executing it. The key idea is to compute the WCET from the structure of the code when it is combined with an abstract model of the hardware that models the internal structure of the execution. Unlike the measurement method, the static analysis considers all the possible values of the input data of the program, but envelops it using an abstraction. Thus, the large number of input, usually leading to combinatory explosion, is reduced into a smaller values envelopped inside the abstraction. Using the same type of abstraction, the worst case path of the code execution can also be determined.

(NB: it might be the source code as proposed in [Colin 2001], the executable code as suggested in [Wilhelm 2008b] or some intemediate code as in [Schoeberl 2006]. The basic concepts for static analysis include:

- *the flow analysis*: The aim of the flow analysis is to gather information about the exeuction path of the program. The exact path cannot in general be exactly determined [Wilhelm 2008a]. The input of this flow analysis step is a representation of the program such as its syntaxic tree graph and its CFG (control flow graph).

  Basic blocks and Control Flow Graph: Basic blocks are blocks of code the program for which the execution time can be computed easily because they are only composed of sequential instructions with one input and output path. The control flow graph describes all the possible relations between the basic blocks. In the example of figure A.1, the basic blocs are the boxes denoted by BB_x (where x is the id number of the block). The generation of the CFG is done at the level of the object code because the object furnishes the information on the execution time. (In figure A.1 an example of CGF determines the possible execution paths for the simple example given in this same figure).

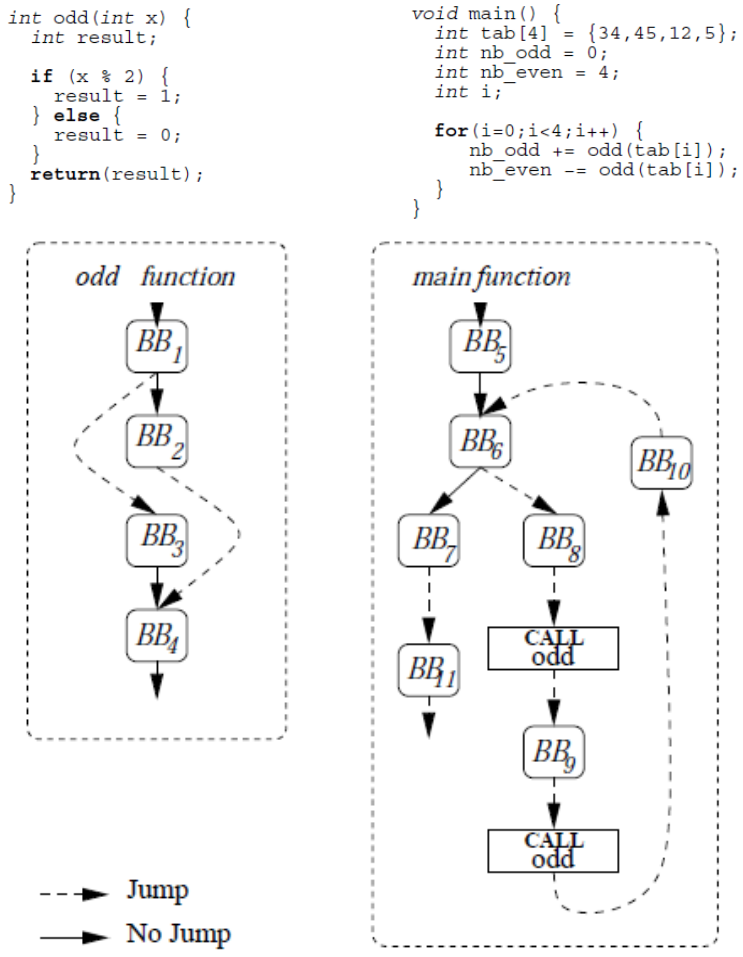

Figure A.1: Control Flow Graph Example

The set of basic blocks must be finite-numbered [Wilhelm 2008a] because termination of the program must be guaranteed.

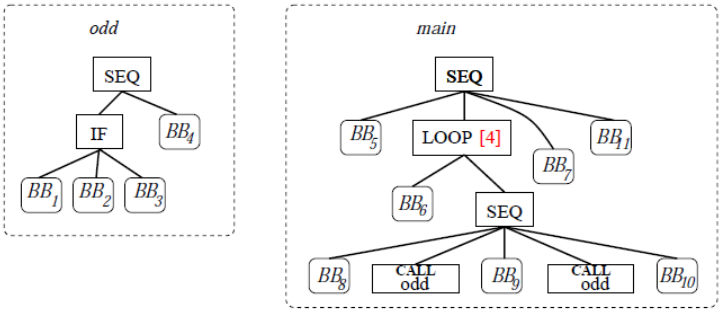

Figure A.2: Control Flow Graph Example

Syntaxic tree: this is a tree for which nodes represent the control structures of the high level code and the leaves represent basic blocks. For the same example presented in figure A.2. As seen, it contains more information than the CFG and then is more useful.

- Low-level analysis which evaluates the impact of the architecture of the hardware on the WCET. It contains steps such as the determination of some basic blocks in the code.

- the overall computation of the estimated WCET by combining the result of all previous components

This method is known to give a safe but overly pessimistic estimation [Colin 2003]. Many difficulties can be found currently:

- The success of this approach is based on the abstract hardware model. If correctly modelled, a safe upper bound of the WCET is found. However, it is known that the verification of correctness of the abstract models is hard [Lokuciejewski 2011]. Moreover, in some systems, it might be mandatory to take into account other parameters such as the I/O (Input Output) access time, the cache memory access time vs the central memory access. All of these considerations lead to a very complex and long process. However, these works have been done partly or fully (see for instance [Louise 2002] concerning multiple cache level considerations in the WCET and works having to do with the processor pipelines for WCET computing can be found in [Thesing 2004])

- The production of an overestimated result. In fact, static methods are known to be safe but not tight.

- Complexity: The method may work on a simple prototype program but as soon as the size of the program increases, the run time analysis can be very high. This difficulty also embeds complexity of hardwares. It is also a concern in this approach because the more complex the hardware, the more probable, the problem is undecidable.

- Undecidability of the problem [Thesing 2004]: This problems is acutally undecidable except for some restricted set of programs that satisfy some conditions linked with the method of programming (e.g: loop iterations must be known, no recursion). A solution has been proposed for some types of programs by applying a so-called timing schemes [Shaw 1989]

- Some implementations of static methods use annotation in order to simplify the process of computing the WCET. These annotations are human-made and thus error-prone.

A.4 Hybrid methods

The idea behind hybrid approaches is to use the safeness property provided by static analysis with the relative applicability of the dynamic methods. In fact, measurement method at least provides a lower bound of the WCET, as seen in section A.2 and the static method provides an upper bound. The key approach is to identify the so-called *single feasible paths* (SFP), which are execution paths that are specific in a term that they are invariant to changes of input data [Lokuciejewski 2011]. This identification is a static method using an abstract syntaxic tree. The execution times of these SFP is measured using a real hardware (or an adequate simulators). For input-dependant paths, the input data must be supplied for complete coverage of the paths. The execution times of these paths are also determined by dynamic method, and in order to be safe in the determination, the usual safety margin for measurement method is added. Finally all information is combined with techiniques from static methods to determine the worst case path. The real advantage is that it does not use abstract hardware models. Instead, the problems of measurements still exist.

On one hand, static methods, when facing the different aforementionned problems, produce no results or overly pessimistic results. On the other hand, the complexity of modern processors including complex features make cycle true simulation (needed for measurements) infeasible. As a result of all of those elements, this imposes either impractical limitations on the code (we have seen examples such as loop iterations must be known to permit analysability, no recursive function in order no to run the risk to have stack overflow, etc etc) Those constraints exist also on the hardware. It is possible with some hardware processor to force the system not to use the whole cache or part of it, to constrain a variable by annotation. One possibility to get around is to use statistical methods.

A.5 Probabilistic methods

It is a specific variation of the measurement method. Instead of giving a single value as the WCET, this method consist of providing a probability density function for the different execution time and the assumed WCET. While the probability values tend to approach zero, the absolute safety (100% guarantee) is reached only with a huge degree of conservatism. To cope with this problem, to have a certain degree of tolerance can be useful. This notion of risk is quite conform to what has been mentionned in section 1.4 where the notion of

appearance of fault in avionics critical systems was expressed by a probability (10^{-9}per hour of flight). In our case, the 100% guarantee is not physically achievable and a certain amount of risk is acceptable. This risk is constrained to be very small. A major problem in this method is the assumptions' validity. It has to be assumed that the method provides a description of the program behavior which correctly covers all the execution times for all modes of execution.

The principles of the method is as follows. The program to be analysed is run several times with random-input data and the end-to-end execution-time (meaning the time from task launches until the task ends) is measured. It is clear that this would not give the WCET value on the target processor in the general case. So the use of Extreme Value statistics is proposed by [Petters 2002]. The simplest instance of a solution is to approximate the probability density of an execution by a Gumbel Probability function. The formula is quite complex and we recommend to look at the reference for detail.

Possible use mainly focuses on approaches using test cases and then to use the existing results to deduce in a probabilistic manner other execution times not detected during the tests. The experiments done by the authors have given the following approximation.

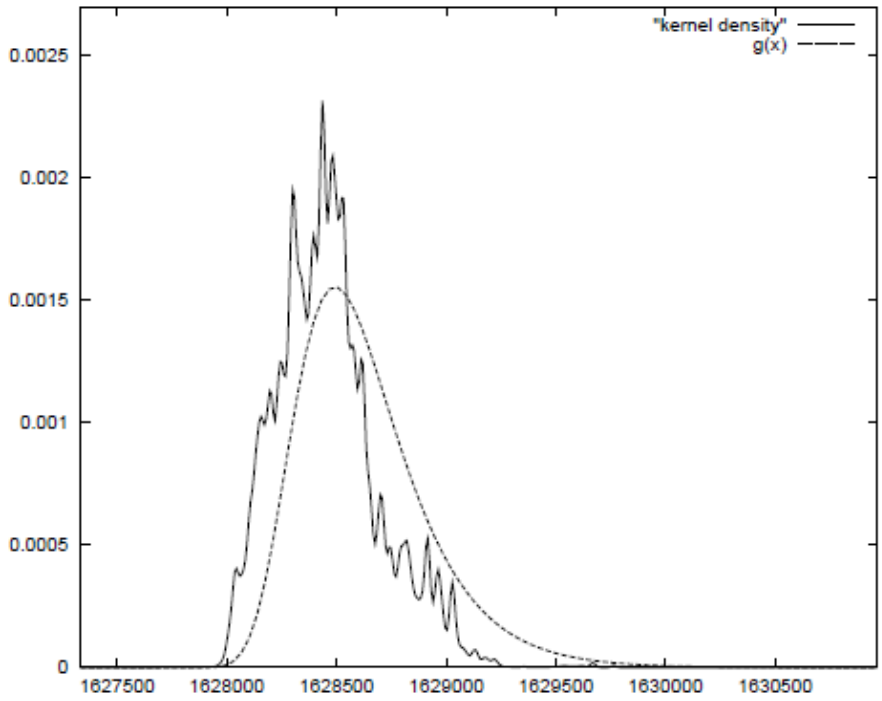

Figure A.3: Sample Measurement Data and Extreme Value Approximation

The major disadvantage of using this special density function is its nonzero probability except for time $\pm\infty$. It has been shown that the while the execution time exceeds 20σ (σ is the standard error) beyond the mean, and the probability is 2.06×10^{-9} then the "risk" still exist. Experiments show that probability reaches quickly 1.0×10^{-20} and at this time, the overestimation is around 10%

Interpreting a deadline overrun of an assumed value of the WCET as a fault, it becomes

possible to assess the deadline miss probability of a software system. This approach can bring solutions to some of the problems aforementionned but moreover, can be used also to insert some sort of fault- tolerance in the system as seen from end-to end. This has been used throughout this thesis for the case study.

A.6 Construction and interpretation of the probability distribution:

Some assumptions need to be clarified concerning these probability values. [Petters 2002] proposes to use similar techniques as in hardware systems dependability analysis, where a 100% guarantee is physically impossible and a certain amount of extremely small risk is acceptable.

Also, probabilistic analysis provides a continuum of worst-case execution times for different confidence levels. Thus a system may have a probability of 10^{-8} per hour of exceeding an execution time of 1.5 ms for instance, and probabilities of 10^{-9} per hour of exceeding 1.505 ms.

Now, the most difficult assumption to be satisfied is to justify the classical formulation *"per hour"*. The construction of this probability is considered as follows.

Considering the abstract processor used in the case study, in which the execution times of basic are considered independent and identically distributed (iid), it is possible to derive easily a probability rate. Of course this assumption is quite difficult to assume on new generations of processors considering the effects of caches memories and speculative executions.

Following [Burns 2011], let us take the simplest case, where all the execution time of all the abstract machine's instructions is either 1 unit of time (the Best Case Execution Time which could be equivalent to the execution time cost fo a cache hit) or 10 units of time (for the WCET). We note that the difference between the BCET and the WCET is 9 units of time, which is quite optimistic with respect to the reality. Let us also suppose that the probability of the execution to be equal to the WCET is of 10 % (again very optimistic, as in reality the WCET is considered as a rare event). Thus, the average instruction execution time for the considered abstract processor is computed as 1.9 unit of times.

A relevant and adequate formulation would be modelled as follows [Burns 2011]:

$$proba(A > E) \leq p \tag{A.1}$$

where A is an instance of the execution time of the task and E is the estimated worst case execution time of such task. Thus the formula A.1 would be interpreted as the probability $proba$ of an instance A to exceed the worst case E is less that p. Now practically speaking, if the reliability threshold $p = 10^{-5}$ (the probability that any execution time exceeds the estimated WCET E) then, by solving the equation A.1 with a given probability function $proba$, it possible to give a fixed value of E according to the desired reliability.

To illustrate this, let us give a table constructed in [Burns 2011] for such fictitious processor.

	$N = 10^5$	$N = 10^6$	$N = 10^7$	$N = 10^8$
Average ET	190000	1900000	19000000	190000000
WCET for $p = 10^{-3}$	192639	1908344	19026385	190083437
Average ET	190000	1900000	19000000	190000000
WCET for $p = 10^{-5}$	193642	1911516	19036415	190115153
Average ET	190000	1900000	19000000	190000000
WCET for $p = 10^{-7}$	194440	1914039	19044393	190140383
Average ET	190000	1900000	19000000	190000000
WCET for $p = 10^{-9}$	195122	1916195	19051211	190161941

Table A.1: Estimated WCET based on N measurements and a reliability objectve p [Burns 2011]

Without going into deep details about the constructions of the data of this table, the objective when giving this table was to show the relative proximity of a probabilistic estimated WCET of the execution time considered and the average case execution time. The analysis presented in this table also allows to justify once again that strictly considering worst case assumptions is very penalizing. More details on the construction of the probability distribution $proba$ is given in [Burns 2011] and will not be give in this work.

Example: Consider a control task that is composed of a N number of instruction and the previously mentionned assumptions on the processor. Thus the case where $N = 100000$ instructions (thus N number of measurements). Assuming that the code is linear thus the execution time is obtained by summing the cost of each instruction, then for a standard definition of the worst case, it leads to a worst case execution time of $100000 \times 10 = 1000000$ units of time, while the average being 19000. The probabilistic approach allows to implement a property of flexibility and tolerance by using probability laws such as the one constructed in [Burns 2011] and whose results are illustrated in table A.1.

Now the construction of the probability of deadline over an hour of execution is explained as follows. Let us consider a control task whose execution period is 10 ms meaning, that over an hour, the number of execution is equal to 360000 times. Now, if we take the case where the software task is composed of $N = 10^5$ basic instructions, in table A.1, the estimated WCET for such task should be 195122 in order to insure a failure rate of 10^-9. Now if we need to insure a failure rate of 10^-9 within an hour of utilization, we need to consider a single execution (let us denoted it X the probability that the single execution would overrun its estimated WCET). Thus X is given by the solution of the equation:

$$1 - (1 - X) = 10^{-9} \tag{A.2}$$

Again, this solution gives approximately a value of $X = 10^{-13}$ for the failure rate of a single instruction execution overan hour, which is equivalent to an estimated WCET of 196275 as given in [Burns 2011] using the probability distribution $proba$. The next step of

this analysis would be to map this type of approach over a real software and processor so that the presented analytical method be appliable in real-life applications.

The analysis described in this section was done on an abstract machine which had only two types of execution times (1 or 10 units of time). Thus, also the probability distribution of these execution times was quite simple. If instead, a more reaistic processor is taken, this technique can still be applied. The main information that is needed, according to [Burns 2011] is the variance of the instruction's execution time.

APPENDIX B

Weakened real-time control framework

To improve the average efficiency of embedded computers while preserving the control stability and performance, and relying on the robustness of feedback control laws, it is proposed to weaken the usual real-time constraints. An analysis framework is proposed in the sequel in order to fully achieve the analysis that will result in the optimal choice of the execution time that should be allocated to the control task of a considered control loop.

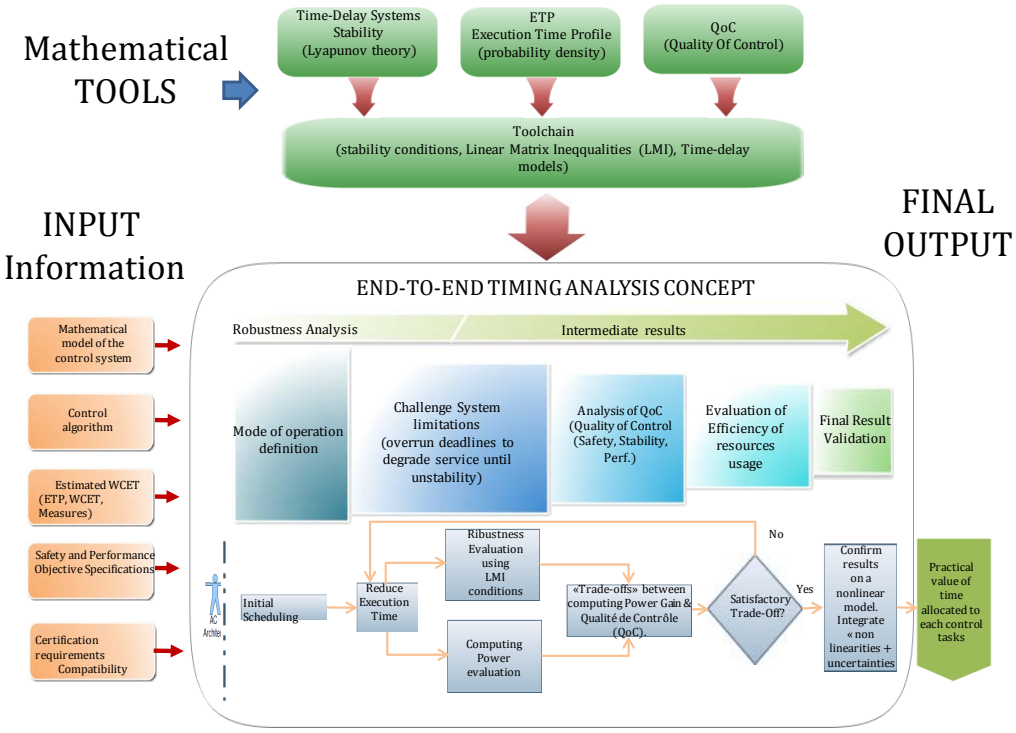

Figure B.1: Weakened analysis framework

Figure B.1, illustrates the overall analytical process that should be followed to derive the timing needs of a control loop.

In the preliminary phase of the analysis, the following elements need to be determined or specified:

- the mathematical model of the control system are determined. This model can be expressed as classical control system models such as state space models, transfer functions, ...
- The control algorithm is also necessary.
- The WCET estimation based on statistical studies of the processor target (for instance the Extreme Value approximation of the execution times as presented in section A) [Petters 2002].
- The performance and safety specifications that the system has to meet. This element allows to define the objectives of the system in terms of trade-off between safety and quality of control for instance. This may also include the certification specifications, meaning that certification bodies may ask for the achievable performance and safety at the worst case. However, instead of ensuring that the controller timing is respected at the worst possible case (the hard deadline paradigm), it would be more efficient in terms of control performance to formally that the system can allow for instance N numbers of deadline overruns (which can be managed by different safety procedures) without having any effects on safety.

The next step would be to work on the initial scheduling (which might be the usual worst case scheduling as presented in figure 1.14), where each task is allocated a processor time resource. In the worst possible case, this time as at least equal to the WCET of the task. During the next sequences of activites, the execution times are shortened until the system reaches unstability. The resulting robustness and stability of the system are then evaluated with respect to the computing time that has been gained. Then the quality of control and performance is analyzed and the trade-off between the quality of control and the processing resources usage resulting from the degradation of the system's performance is evaluated. It is checked whether the system stays in a satisfactory bound with respect to the performance and safety criterion intially specified.

If the trade-off is not satisfactory, then the whole step is performed again with the allocated time for the control tasks respectively increased on reduced, depending on the case. Two cases may appear, depending on the following situations. For instance, if the system after trade-off analysis is found wasting too much computing power, the whole procedures are repeated with a smaller allocated time. Conversely, if the system is no more compliant with safety specifications (e.g: it goes unstable without respecting the safety specifications. For instance the probability of going unstable should be less than 10^{-9} per hour of use), then the steps are repeated with larger allocated time.

Therefore at the end of the process, for a given control system, a known dicstribution of the execution times and a weakened real-time contraints, the problem to solved can be informally stated as follows:

- find N, the maximum value of consecutive data loss before loosing the closed-loop stability;
- find an adequate value of T_{slot} to fulfill a given trade-off between the control and the computing performances.

- evaluate the weakly-hard closed-loop robustness with respect to the plant's parameters uncertainties.

APPENDIX C

Sampled-data models

Contents

C.1 Computer-control theory

The most basic structure of a computer-controlled system has been introduced in 1.5.3. A schema proposed in section 1.5.3 has introduced the ideal case which would give the best performance. However, this schema is not realistic because of the difference between continuous- time design and discrete-time design. Indeed, physical systems typically evolve in continuous-time domain whereas computers and signal processing devices invariably operate in discrete -time domain.

A sampled-data system is defined simply as a control system where continuous- time plant is controlled with a digital device such as computer.

Looking back at figure 1.12 (reproduced in figure C.1), we see that the system is different from a classical continuous-time systems. From an architectural point of view, new subsystem elements have appeared (as compared to figure 1.17) and different types of signals are manipulated. It is remarked that a computer-controlled system contains both digital and continuous signal.

C.2 Advantages and drawbacks of using computers

From the perspective of figure C.1, such systems might be viewed as only an approximation of the continuous-time system by means of a set of points approximating the state (or the output) at a given time, and it may also be viewed as if the computer-control system's performance results would be only as good as those of the equivalent analog control. This implies that if the sampling period is very small, then the computer-controlled system would behave as the continuous system. The so- called *naive approach to computer-controlled system*, presented in [Wittenmark 2003], show that this statement is only true in very reasonnable assumption and that computer controlled systems have more potential that need to be fully understood and they present many subtantial advantages. Many problems with

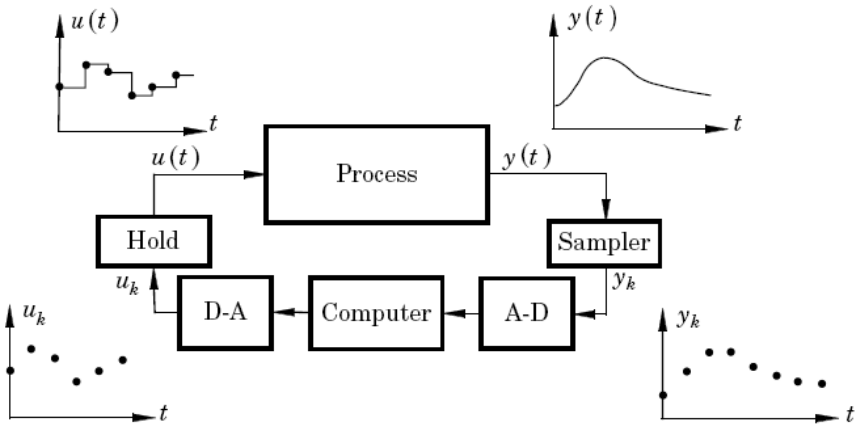

Figure C.1: Realistic sampled-data model

analog system implementations can be avoided by using computers, for instance the problems linked to the accuracy or drift of the mechanical or electronical components due to fatigue or elderness. Also, it becomes very easy to *try* new sophisticated calculations and complex control laws (because of the flexibility of the computer softwares for instance), as well as , it is easy to include logics and non-linear functions. On the other hand, the memory effects that computer provide by means of data structures such as *tables*, can be used to store data in order to accumulate a more or less long-term knwoledge about the system's properties. Moreover, there are phenomena that do not appear in continuous- time models that show up in computer-controlled systems.

On the other side, the future trend of computer systems lead some authors to speculate about the future [Wittenmark 2003], mainly about technological improvements of computers. Spectacular developments (such as the introduction of complex VLSI circuits, multicore, multiple-level caches...)as presented in [Louise 2002, Thesing 2004] have made it difficult to improve temporal analyazability. This means that, first: the ratio of price to performance is dropping subtantially. According to [Louise 2002] the enhancements brought to new generations of processors increases the uncertainty and decrease determinism and predictability of the execution timings of instructions. In fact, the execution time of a given instruction, when run on these processors, is no longer constant but variable. That means, using new processor generations doom designers to take even longer spare margins penalizing the system on resource usage.

From schdeuling and programming perspectives, these facts have made that programming and scheduling theory become the bottleneck of control system design (see Section 1.7).

From the control perspectives, theory has made significant advances since 1955 but only very few of those advances have made their way into the problematics of computer-controlled systems.

And from practical and industrial context, using a package provided by a vendor (processor and software COTS) requires little development effort in comparison to developing everything, which requires a major effort. However, those systems are optimized for

average-case performance and are difficult to validate for worst-case performance. In many cases, for instance, personal computer are starting to be used in conjunction with high level programming languages for process control. However, it would be very hard ,even impossible to write safe real-time systems that will be compliant with the safety directives presented in section 1.4.

The mixture of two signals cause practical and theoretical difficulties. In many cases, the vision of the system's behavior at the sampling instants is more than sufficient. This means that the whole system is only considered at the sampling instants. Such systems are called *discrete-time systems*. Again, as introduced in section 1.5.3, discrete models deal with a finite sequence of finite precision number while continuous-time systems deal with a non-countable sequence of numbers.

For a detailed explanation of the system and the signals presented in figure C.1, see section 1.5.3. Such systems, were traditionnaly called *Sampled-data systems*. All along this thesis, this term is used as a synonym for *Computer-controlled systems*.

C.3 Mathematical models

Physical laws allows to elaborate mathematical models of a physical plant. This model is precise enough to allow the evaluation of dynamical characteristics of the studied system. In control theory, synthesis of control laws, performance analysis and stability analysis are done on such models. On the other side, implementators also use such models as bases for the computer implementations.

The power and use of a model lies in the fact that it can be analyzed and simulated under hypothetical assumptions while these basis can afterward be used for synthesis of controllers. In a general case, any dynamical system can be represented by the state space representation (or state equations). In the standard form the mathematical description of the system is expressed as a set of n coupled first-order ordinary differential equations (or difference equations for discrete-time systems). It then facilitates numerical solutions to various problems.

From continuous-time to discrete-time systems

The sampling process has been defined as the act of taking a small part or a small quantity of something. In the context of signal processing and control system, sampling means taking the *continuous-time signal* and converting it by a sequence of numbers, as explained clearly in section 1.5.3. It is a fundamental property for computer- controlled systems because of the discrete- time nature of the digital computer.

Assuming that the plant is represented by the LTI continuous system.

$$\begin{cases} \dot{x}(t) &= A\,x(t) + B\,u(t) \\ y(t) &= C\,x(t) + D\,u(t) \end{cases} \tag{C.1}$$

where:

$x(\cdot)$ is the state vector, $x(.) \in \mathbb{R}^n$
$y(\cdot)$ is the output vector, $y(.) \in \mathbb{R}^q$
$u(\cdot)$ is the input (or control) vector, $u(.) \in \mathbb{R}^p$
$A(\cdot)$ is the state matrix, $\dim[A(\cdot)] = n \times n$
$B(\cdot)$ is the input matrix, $\dim[B(\cdot)] = n \times p$
$C(\cdot)$ is the output matrix,$\dim[C(\cdot)] = q \times n$
$D(\cdot)$ is the feedthrough (or feedforward) matrix, $\dim[D(\cdot)] = q \times p$

In this general formulation, all matrices are allowed to be time-variant (i.e., their elements can depend on time). However, in the common LTI (Linear Time Invariant) case, matrices will be constant.

The system is assumed to have n states, r inputs, and p outputs and where the variable t denotes the continuous time variable.

As seen in figure C.1, the input $y_k = y[s_k]$ and the output $u_k = u[s_k]$ of the computers are discrete-time signals while the input $u(t)$ and the ouput $y(t)$ of the plant are continuous-time signal.

As previously, explained, the main problem of computer-controlled systems is the mixture of continuous-time and discrete-time signals and the relationship between these sequences of values u_k and y_k. This problem can be solved bu observing the whole system only at the sampling instants s_k. This implies also that even the continuous-time plant is also looked upon from this perspective, that is to say by considering the discrete sequences u_k and y_k.

It is however emphasized that the physical plant is still in the continuous- time domain and getting the intersample behavior is impossible.

Such models are also called stroboscopic model as it gives only a snapshoot of the behavior of the whole system at timed instants. From this assumption, it is quite straightforward to get the desciption of the sampled-data system as a state spate model (this model can also be used to derive other equivalent models such as input/ouput models).

Before going straight into the equation of the sampled model, it is interesting to clarify some assumptions for simplicity. It assumed that periodic sampling is used, that the sampling period is denoted h and sampling instants are denoted $s_k = kh$ (meaning the s_k is the k^{th} sampling instant and for instance $u[s_k]$ denotes the k^{th} input sequence). It is also assumed that the hold device is a zero-order hold (see Section 1.5.3).

The explicit integration solution of the continuous-time linear state equation C.1 is given by:

$$x(t) = e^{A(t-t_0}\, x(t_0)) + \int_{t_0}^{t} e^{A(t-\tau)}\, Bu(\tau)d\tau \tag{C.2}$$

where $x(t_0)$ is the initial state.

If we assume that the input of the plant $u(t)$ is generated by a zero-order hold device from a sequence $u[k]$. This implies that $u(t) = u[s_k]\ s_k \leq t < t_{k+1}$.

Using this assumption and the result of equation C.2, it is obtained that:

$$\begin{aligned} x(t) &= e^{A(t-s_k)}\, x[s_k] + \int_{s_k}^{t} e^{A(t-\tau)} B u(\tau) d\tau \\ &= e^{A(t-s_k)}\, x[s_k] + \int_{s_k}^{t} e^{A(t-\tau)} d\tau B u[s_k] \\ &= e^{A(t-s_k)}\, x[s_k] + \int_{0}^{t-s_k} e^{A(\tau)} d\tau B u[s_k] \\ &= \Phi(t, s_k)\, x[s_k] + \Gamma(t, s_k)\, u[s_k] \end{aligned} \tag{C.3}$$

The second line holds because $u(t)$ is piecewise constant between sampling instants and takes the value of $u(s_k)$. Moreover, if the A-D and the D -A converters (refer to figure C.1) are perfetcly synchronized and the conversion times are assumed to be negligible, then the plant input $u(t)$ and the plant ouput $y(t)$ can be regarded as sampled at the same instants. This gives the equation of the discrete-time system as a function of the sampling instants s_k and t_{s+1}. Moreover, equation C.3, give the changes of the state over a sampling interval. then by considering things at the next sampling instant, t_{k+1}, we obtain:

$$\begin{aligned} x(t_{k+1}) &= \Phi(t_{s+1}, s_k)\, x[s_k] + \Gamma(t_{k+1}, s_k)\, u[s_k] \\ y(s_k) &= C\, x[s_k] + D\, u[s_k] \end{aligned} \tag{C.4}$$

Where

$$\begin{aligned} \Phi &= \Phi(t_{k+1}, s_k) = e^{A(t_{k+1}-s_k)} \\ \Gamma &= \Gamma(t_{k+1}, s_k) = \int_{0}^{t_{k+1}-s_k} e^{A\tau} d\tau\, B \end{aligned}$$

Notice that equation C.4 does not involve any approximations and gives the exact values of the state variable $x[t_{s+1}]$ and the output variable $y[s_k]$ at the sampling instants. This is the ZOH sampling of a discrete-time system. The less commonly used FOH (First Order Hold) sampling can be is rapidly presented in [Wittenmark 2003]. However, this type of hold device will not be addressed in this manuscript.

For a constant sampling period h, let's denote $s_k = kh = k$ (only the index of the sampling instant is enough to represent itself) and $x[s_k] = x(k)$, $y[t_k] = y(k)$ and $u[t_k] = u(k)$. Also denote that as h is constant, $t_{k+1} - t_k = h$

This leads to a simplified expression of equation C.4.

$$\begin{aligned} x(k+1) &= \Phi\, x(k) + \Gamma\, u(k) \\ y(k) &= C\, x(k) + D\, u(k) \end{aligned} \tag{C.5}$$

Where

$$\begin{aligned} \Phi &= e^{A\,h} \\ \Gamma &= \int_{0}^{h} e^{A\tau} d\tau\, B \end{aligned}$$

Remark:

The state vector between sampling times can be dervied from equation C.3, which makes it possible to investigate the intersample behavior of the system.

Note that in most cases, the D matrix equals to 0, because, in computer- controlled systems, the output y is measured first and the control signal $u(k)$ is then generated as a function of $y(k)$.

Also, it is common that, in a practical case, there is a significant delay between the A-D and the D-A operations.

Solution of the discrete-time systems

The previous section has exposed the manner to transform a continuous-time system into a discrete-time system. This step goes into the goal of this chapter which aims to clarify the timing problematics of computer-controlled system, because as previously explained, these systems are a mixture of continuous and discrete-time systems. In this paragraph, the objective is to show how to find the solution of such discrete-time equation for analysis purposes.

Considering a discrete-time system of the form C.5. Assuming that the initial state $x(k_0)$ and the input sequences $u(k_0), u(k_0+1), ...$ are knwon, the solution of such system can be derived by recursive appproach:

$$\begin{array}{rcccccl}
x(k_0+1) & = & \Phi x(k_0) & + & \Gamma u(k_0) & = & \Phi\, x(k_0) + \Gamma\, u(k_0) \\
x(k_0+2) & = & \Phi x(k_0+1) & + & \Gamma u(k_0+1) & = & \Phi^2\, x(k_0) + \Phi\Gamma\, u(k_0) + \Gamma\, u(k_0+1) \\
 & & & . & & = & . \\
 & & & . & & = & . \\
 & & & . & & = & . \\
x(k) & & & . & & = & \Phi^{k-k_0} x(k_0) + \Phi^{k-k_0-1}\Gamma\, u(k_0) + ... + \Gamma\, u(k-1) \\
 & & & . & & = & \Phi^{k-k_0} x(k_0) + \sum_{j=k_0}^{k-1} \Phi^{k-j-1}\Gamma\, u(j)
\end{array} \tag{C.6}$$

The solution consists of two distinct parts:

- one part, depending on the initial value of the state $x(t_0)$
- another part, depending on a weighted sum of the inputs over the interval $[k_0, k-1]$.

It shows that the matrix Φ (implying mainly its eigenvalues which are obtained through the *charactersitic equation*) will determine the properties of the solutions.

APPENDIX D

Résumé en français

D.1 Introduction

D.1.1 Contexte

Le processus de développement des systèmes avioniques suit des réglementations de sûreté de fonctionnement très strictes, incluant l'analyse du déterminisme et de la prédictibilité temporelle des systèmes. L'approche est basée sur la séparation des étapes de conception et d'implémentation ([Årzén 2000, Xia 2006]). D'une part, le concepteur utilise des hypothèses de périodes d'échantillonnages fixes (sans gigue et sans pertes) et des retards fixes et connus d'avance. D'autre part, la théorie de l'ordonnancement se concentre principalement sur la problématique du dimensionnement des ressources d'exécution pour respecter les contraintes de temps-réel fournies par les concepteurs ([Sha 2004]). L'approche la plus répandue consiste ainsi à considérer des périodes d'échantillonnage fixes et des échéances dures basées sur une hypothèse de connaissance du pire temps d'exécution (WCET pour Worst Case Execution Time en anglais) des tâches informatiques constituant le logiciel embarqué.

Un inconvénient majeur de cette hypothèse est la sous-utilisation des ressources de traitement induisant des difficultés à la fois techniques, économiques et industrielles. Une des plus grandes difficultés dans l'approche actuelle se trouve dans la détermination du WCET, qui est nécessaire pour prouver la satisfaction des contraintes de temps-réel dur du système. La précision du résultat dépend de la prédictibilité de l'unité de calcul utilisé. Cependant les nouvelles générations de processeurs ont une architecture peu déterministe rendant difficile la prédiction des durées d'exécution. Ainsi, simplement estimer le temps d'exécution d'un simple bloc de base n'est pas trivial à cause des anomalies temporelles qui peuvent résulter de l'ordonnancement dynamique des instructions, de l'influence sur l'état du pipeline des blocs de base exécutés avant et après ([Barre 2006]). Actuellement, plusieurs travaux se proposent de définir des méthodes de plus en plus fiables pour déterminer le WCET (e.g. [Khan 2009]) sans toutefois éliminer le conservatisme des résultats pour les nouvelles générations de processeurs ([Puschner 2008]).

De plus, en supposant la détermination des WCET résolue, cette approche basée uniquement sur le "pire cas" est fondamentalement pénalisante du point de vue de l'utilisation des ressources d'exécution et conduit à un sur- dimensionnement du système dont l'utilisation moyenne peut être très inférieure à sa capacité maximale. En fait, les systèmes de commande par rétroaction possèdent des propriétés de robustesse et de tolérance aux aléas temporels d'exécution remettant en cause les considérations de "temps-réel dur" utilisées habituellement. L'exploitation des ces propriétés de robustesse doit permettre de relâcher

les contraintes d'ordonnancement des tâches de commande et de mieux exploiter les capacités des ressources d'exécution des contrôleurs.

Pourtant, les méthodes de conception et d'analyse existantes actuelles ne fournissent pas de modèle assez flexible pour résoudre ce problème dans la conception de systèmes. De plus, la théorie de l'automatique classique ne propose pas de solution sur le problème d'utilisation efficace des ressources et la prise en compte des aspects de sûreté de fonctionnement pour les systèmes qualifiés de "critiques". Cependant, en considérant les systèmes de contrôle-commande en boucle fermée, des solutions plus flexibles peuvent être envisagées en exploitant les fonctionalités de base de la boucle de commande.

Ce résumé de thèse est organisé comme suit : le chapitre D.2 expose le problème d'ordonnancement flexible de boucles de commande.

Le chapitre D.3 formalise le problème de séparation d'interêt entre la communauté automatique et informatique. Ce même chapitre propose de nouvelles conditions de stabilité pour des systèmes soumis à des incertitudes paramétriques, à des retards et à des variations de la période d'échantillonnage et propose différents mécanismes de gestion de dépassement d'échéances.

Le chapitre D.5 présente un cas d'étude où la méthodologie exposée est appliquée au contrôle de l'axe de tangage d'un avion F-16. Enfin le chapitre D.6 conclut l'article et propose quelques pistes pour les travaux futurs.

D.1.2 Notations

Dans le reste de ce document, l'ensemble $\mathbb{R}^+$, $\mathbb{R}^{n\times n}$ et $\mathbb{S}^n$ dénoteront respectivement l'ensemble des réels positifs, l'ensemble des matrices carrées d'ordre n et l'ensemble des matrices symétriques de $\mathbb{R}^{n\times n}$. L'exposant 'T' indiquera la transposition des matrices. La notation $P > 0$, pour $P \in \mathbb{S}^n$, signifiera que P est une matrice symétrique définie positive. Pour toute matrice $A \in \mathbb{R}^{n\times n}$, la notation $\mathrm{He}\{A\} > 0$ signifiera que $A + A^T > 0$.

D.2 Problématique

D.2.1 Les systèmes à considérer et les problèmes d'ordonnancement

Le but d'un système de contrôle-commande est de commander un processus pour l'amener dans un état répondant aux besoins de l'utilisateur. La majorité de ce type de systèmes est considérée comme systèmes "embarquée ou enfouis". Ce sont des systèmes qui doivent réagir à des évènements extérieurs avec ou sans des contraintes temporelles (auquel cas, ils sont alors appelés systèmes embarqués temps réel) entre autres.

La structure classique de tels systèmes est représentée par la figure D.1

La procédure de commande est composée de 3 phases distinctes: les acquisitions des mesures, le calcul des signaux de commande et le rafraichissement des actionneurs. Ainsi ces opérations sont des séquences et sont donc des opérations parfaitement périodiques dans le cas idéal.

Dans le cas decrit ci-dessus où le contrôleur est un ordinateur embarqué, la loi de commande est un algorithme qui est implémenté sous forme de logiciel. Diverses méthodes

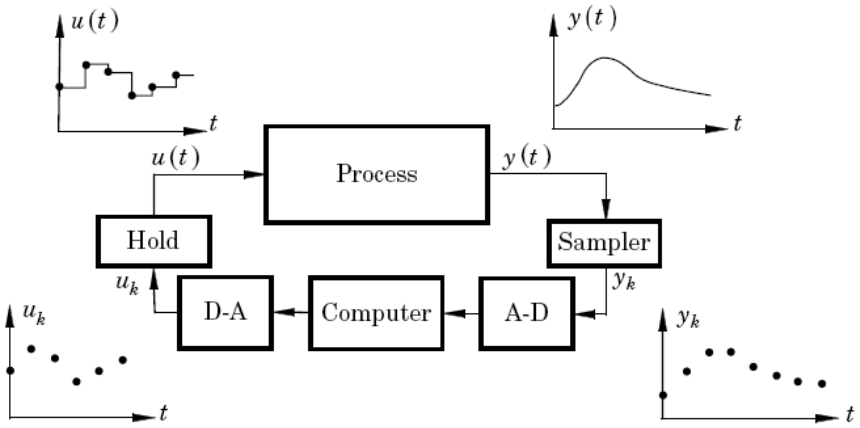

Figure D.1: Schéma d'un système de controle-commande digital

sont connues pour l'élaboration des lois de commandes en l'occurence les méthodes traditionnelles où les lois sont conçues en temps continu et ensuite discrétisés et la conception directement en temps discret. Toutefois, dans les deux cas, l'échantillonnage et les autres paramètres temporels sont supposés être parfaitement périodique et la latence est supposée être inexistante.

Cependant, dans la pratique, plusieurs considérations doivent être pris en compte lors de l'implémentation en particulier la notion d'ordonnancement.

Comme la loi de commande est pratiquement implémentée sous la forme de tâches logicielles, il est nécessaire de savoir comment organiser la séquence d'execution des différentes tâches afin de garantir le bon fonctionnement et le respect des contraintes temporelles.

En effet, le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, ...) et de contraintes portant sur l'utilisation et la disponibilité de ressources requises. Les méthodes utilisées pour l'élaboration d'une solution à un problème d'ordonnancement vise le plus souvent à satisfaire un ou plusieurs objectifs (par ex: coût, délais, qualités de service, optimisation de l'utilisation des ressources, stabilité, ...). Ces différents critères sont appelés des critères de performances.

Les approches traditionnelles de conception et d'implémentation considèrent la connaissance du pire temps d'execution: le WCET (Worst Case Execution Time). Un premier problème qui se pose est la difficulté de determination de cette grandeur (cf [Kirner 2005, Thesing 2004]).

Dans la section suivante, l'ordonnancement basée sur la connaissance du WCET est expliquée.

D.2.2 Ordonancement basée sur le pire temps d'execution

Actuellement, les systèmes de commande, comme les commandes de vols ou de freinages, sont généralement considérés comme des systèmes temps-réel durs. Il est souvent supposé

dès la phase de conception que les tâches logicielles devront s'exécuter de manière strictement périodique et determinisite. Des créneaux de temps préfixés sont alloués aux tâches de contrôle, et les dépassements d'échéances et le phénomène de gigue sont interdits. On considère que toute déviation par rapport à cet ordonnancement idéal est une panne du système. La connaissance supposée sûre des WCET est utilisée pour dimensionner le système. La figure D.2 montre un schéma d'exécution des tâches de contrôle dans ce contexte.

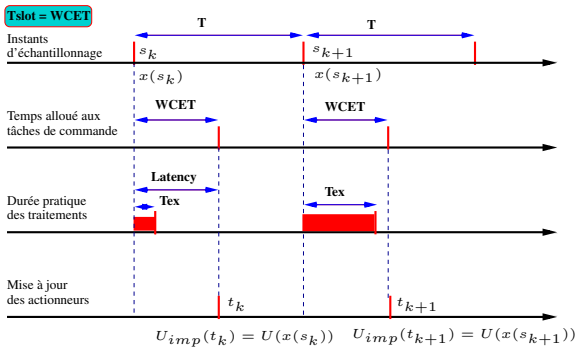

Figure D.2: Diagramme temporel d'exécution de tâche basé sur le WCET

Une tâche donnée s'exécute périodiquement sur une fenêtre de temps $T_{slot} = WCET$ allouée à son exécution. La tâche est déclenchée à une période $s_k - s_{k-1} = T$ par l'arrivée des mesures s_k, qui représentent l'état du système à l'instant s_k. La durée de traitement est de T_{ex}, qui est toujours inférieure au $WCET$. Pour éviter la gigue, le signal de contrôle $U(s_k))$ est seulement appliqué aux actionneurs à la fin de la période, c'est- à-dire à l'instant $s_k + WCET$.

$$\forall t \in [s_k + WCET, s_{k+1} + WCET[, \quad U = U(s_k)).$$

Ainsi, il s'agit d'un système de commande périodique soumis à un retard constant $T_{slot} = WCET$ et une période de T. Ce schéma d'exécution implémente bien l'hypothèse "temps-réel dur". Cependant, le traitement est achevé toujours avant la fin du temps alloué et une fraction de la puissance de calcul disponible est inutilisée. La perte en temps processeur est d'autant plus importante que le WCET s'écarte de la valeur moyenne des temps d'exécution T_{ex} observés. En effet, les hautes performances des nœuds de calculs modernes sont dues à l'introduction de mécanismes architecturaux tels que les pipelines, les mécanismes d'exécution spéculatifs et l'utilisation des mémoires caches à plusieurs niveaux qui réduisent le déterminisme temporel des processeurs. L'estimation du WCET devient de plus en plus difficile, et l'étalement de la distribution des temps d'exécution rend l'approche "pire cas"encore plus conservatrice. ([Souyris 2005]).

Une courbe typique de distribution du temps d'exécution (figure D.3) pour un processeur embarqué donné ([Hansen 2009]) montre que l'on s'attend à ce qu'une exécution proche du WCET soit un événement plutôt rare et qu'un ordonnancement basé sur une occupation moyenne du CPU plutôt que sur le pire cas donnerait des gains substantiels en terme de dimensionnement des équipements embarqués.

La validité de l'hypothèse habituelle de temps-réel "dur" mérite donc d'être réexaminée

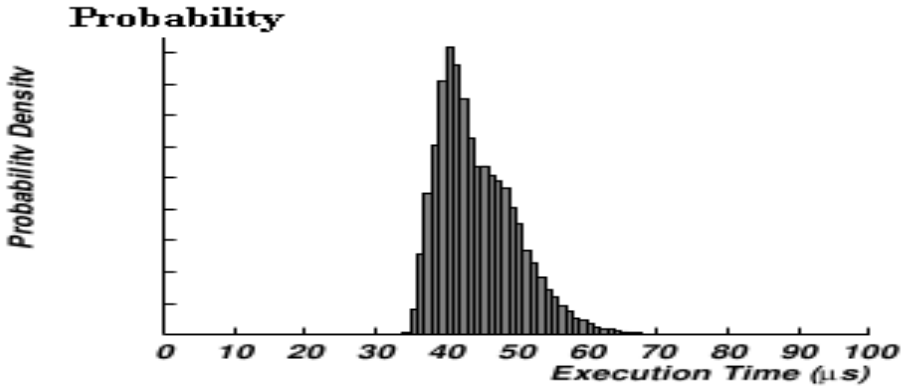

Figure D.3: Distribution caractéristique du temps d'exécution

soigneusement en tenant compte de la robustesse des systèmes de commande par rétroaction. C'est avec cette idée que la prochaine section aborde de la robustesse des systèmes à boucle de retour.

D.2.3 Robustesse des systèmes bouclés

La robustesse d'un système définit sa capacité à conserver sa stabilité ainsi que des performances spécifiées en dépit de déviations des paramètres réels par rapport au modèle nominal, c'est une préoccupation constante dans la conception de systèmes de commande.

L'évaluation de la robustesse en utilisant la simulation peut difficilement donner des résultats exhaustifs pour des systèmes complexes, en raison du nombre important de paramètres mal connus, susceptibles de varier dans le temps, et de la large combinatoire induite. La théorie de la commande robuste en automatique est une approche formelle permettant de concevoir des commandes robustes aux incertitudes des paramètres du modèle du procédé commandé. Cette notion de robustesse est moins courante en informatique : [Hamann 2006] propose par exemple une approche formelle d'évaluation de la robustesse des systèmes embarqués temps-réel, en définissant des métriques de performance informatique de systèmes sur puce.

Cependant le problème de la robustesse de systèmes commandés par rétroaction aux variations et incertitudes des paramètres temporels d'exécution est jusqu'ici peu abordé.

Il s'agit ici, en complément de la robustesse aux incertitudes du procédé, de prendre en compte les phénomènes de gigue ou de dépassements d'échéances d'exécution.

Pour des systèmes linéaires SISO, la robustesse est classiquement quantifiée par les notions de marge de phase, marge de retards et de marge de module. Il apparaît que la marge de phase implique une marge de retard (c'est-à-dire le retard maximum non modélisé que le système peut tolérer avant de devenir instable) mais aussi une marge de gigue qui est plus difficile à quantifier ([Cervin 2004]) mais qui peut être montrée expérimentalement ([Cervin 2003a]).

La robustesse des systèmes asservis peut adresser les perturbations dues aux pertes d'échantillons, comme expliqué dans [Felicioni 2010], où des pertes sélectives d'échantillons

sont causées volontairement afin d'alléger la charge des unités de traitement et l'engorgement du réseau de communication tout en préservant la stabilité en boucle fermée du système.

Le point intéressant est qu'un système robuste aux incertitudes de paramètres est aussi, de manière générale, robuste aux incertitudes temporelles. Les systèmes en boucle fermée, alors qu'ils sont souvent classés comme systèmes temps-réel durs, sont donc plutôt des systèmes temps-réel "faiblement durs" *(weakly hard real-time systems)*. Selon [Bernat 2001], ce sont des systèmes capables de supporter des déviations *spécifiées* des paramètres temps-réel d'exécution, tout en maintenant le niveau désiré de performance de commande.

D.2.4 Formulation du problème

Considérons le système linéaire représentant un système de commande avec une entrée échantillonnée et retardée :

$$\dot{x}(t) = (A + \Delta_\mu A(t))x(t) + (B + \Delta_\mu B(t))u(t), \tag{D.1}$$

où $x \in \mathbb{R}^n$ et $u \in \mathbb{R}^m$ sont respectivement les vecteurs d'états et d'entrée. Les matrices A et B sont supposées constantes et connues. Les matrices $\Delta_\mu A$ and $\Delta_\mu B$ représentent les incertitudes du modèle qui peuvent être constantes ou variant dans le temps. Ces incertitudes sont données sous la représentation polytopique suivante :

$$\Delta_\mu A(t) = \mu \sum_{i=1}^{M} \lambda_i(t) A_i,\ \Delta_\mu B(t) = \mu \sum_{i=1}^{M} \lambda_i(t) B_i,$$

où M correspond au nombre de sommets du polytope. Les matrices A_i et B_i sont constantes. Le scalaire $\mu \in R$, caractérise la grandeur de l'incertitude. Notons que si $\mu = 0$, alors, il n'existe aucune incertitude sur le système. La fonction $\lambda_i(.)$ est une fonction de pondération convexe, c'est-à-dire que, pour tout $i = 1, .., M$ et pour tout $t \geq 0$, $\lambda_i(t) \geq 0$ et $\sum_{i=1}^{M} \lambda_i(t) = 1$. On suppose que l'ordonnancement des tâches de commande induit un délai de calcul T_{slot} et un échantillonnage du signal de commande. Pour un gain donné, $K \in \mathbb{R}^{n \times m}$, la loi de commande est un retour d'état statique, constant par morceaux, et de la forme $u(t) = Kx(t_{k'} - T_{slot}), \forall t \in [t_{k'},\ t_{k'+1}]$.

Les instants $t_{k'}$ représentent les instants où la sortie du contrôleur est actualisée. Le système en boucle fermée peut ainsi s'écrire :

$$\forall t \in [t_{k'},\ t_{k'+1}] \dot{x}(t) = \bar{A}(t)x(t) + \bar{B}(t)Kx(t_{k'} - T_{slot}) \tag{D.2}$$

où $\bar{A}(t) = A + \Delta_\mu A(t)$ et $\bar{B}(t) = B + \Delta_\mu B(t)$.

Plusieurs travaux ont contribué à garantir la stabilité de ce type de système. Dans [Fridman 2004b], [Millán 2009] et [Liu 2009], les auteurs ont proposé une formulation agrégée des retards. Ils ont ensuite développé des critères de stabilité prenant en considération les effets du retard. Toutefois, ils n'ont pas considéré les différentes natures des retards de temps de calcul et d'échantillonnage. Quand $\mu = 0$, la discrétisation de ce type de système est facilement obtenue par l'intégration de l'équation différentielle (D.2) sur l'intervalle $[t_{k'},\ t_{k'} + \tau]$. Ainsi, $\forall \tau \in [0,\ \bar{T}]$, on obtient :

$$\begin{array}{l} x(t_{k'}+\tau) = \tilde{A}(\tau)x(t_{k'}) + \tilde{B}(\tau)Kx(t_{k'}-T_{slot}), \\ \tilde{A}(\tau) = e^{\bar{A}\tau}, \quad \tilde{B}(\tau) = \int_0^\tau e^{\bar{A}(\tau-\theta)}d\theta\bar{B}. \end{array} \tag{D.3}$$

On définit alors pour tout entier k', la fonction $\chi_{k'}^{T_{slot}} : [0,\ NT] \times [-T_{slot},\ 0] \to \mathbb{R}^n$ telle que $\forall\tau \in [0,\ NT]$ et $\theta \in [-T_{slot},\ 0]$, $\chi_{k'}(\tau,\theta) = x(t_{k'}+\tau+\theta)$. L'ensemble $\mathbb{K}_{NT}^{T_{slot}}$ représente l'ensemble des fonctions définies par $\chi_k^{T_{slot}}$, continues de $[0,\ NT] \times [-T_{slot},\ 0]$ vers $\mathbb{R}^n$.

Toutefois, la même méthode de discrétisation conduit à d'importantes difficultés quand le système contient des intervalles d'échantillonnage variables et des paramètres variants dans le temps. Ceci montre la nécessité d'introduire de nouvelles conditions de stabilité qui soient adaptées à ce type de problèmes. Dans la thèse, une méthode innovante est proposée pour évaluer la stabilité des systèmes soumis à des échantillonnages variables, à des retards constants et à des incertitudes de paramètres variants dans le temps.

D.3 Allègement des contraintes temps réels

Dans la partie précédente, les problématiques des ordonnancements basés sur le pire temps d'éxécution ont été mises en évidence. Une proposition de solution consiste à alléger les contraintes temps réel. Dans ce chapitre, il sera exposé le principal motivation de cette thèse qui conduit directement à définir ses objectifs généraux suivi d'une description simplifiée des contributions.

D.3.1 Mise en évidence de la séparation d'interêt

Cette section est intéressante car elle permettra de mettre en valeur les différentes contributions de la thèse.

Dans le cadre des systèmes de contrôle-commande, les méthodes de developpement mise en oeuvre sont des méthodes classiques qui consistent en plusieurs étapes séparées. Les différentes étapes classiques utilisées actuellement dans le monde industriel sont composées d'une étape de conception, et d'une étape d'implémentation, suivie ensuite de plusieurs étapes de vérification et de tests comme le montre la figure D.4.

Cependant, lors du developpement du système, il y a traditionnellement une forte séparation d'interêt entre la conception et l'implémentation.

D'une part, la communauté automaticienne suppose que les paramètres temporels sont parfaitement connus et constants. Ces hypothèses de départ ne reflètent pas correctement la réalité mais toutefois, conviennent parfaitement aux outils d'analyse et de synthèse dédiés aux systèmes échantillonnés.

Ces hypothèse ont conduit la communauté temps réel, d'autre part, à considérer les systèmes de contrôle commande comme étant des systèmes temps réel dur. L'implémentation la plus classique consiste à implementer physiquement les lois de commande avec des tâches périodiques. Un mapping entre les paramètres temporels de ces tâches sont faits avec les paramètres temporels de la loi de commande. Un exemple classique, dans un cas

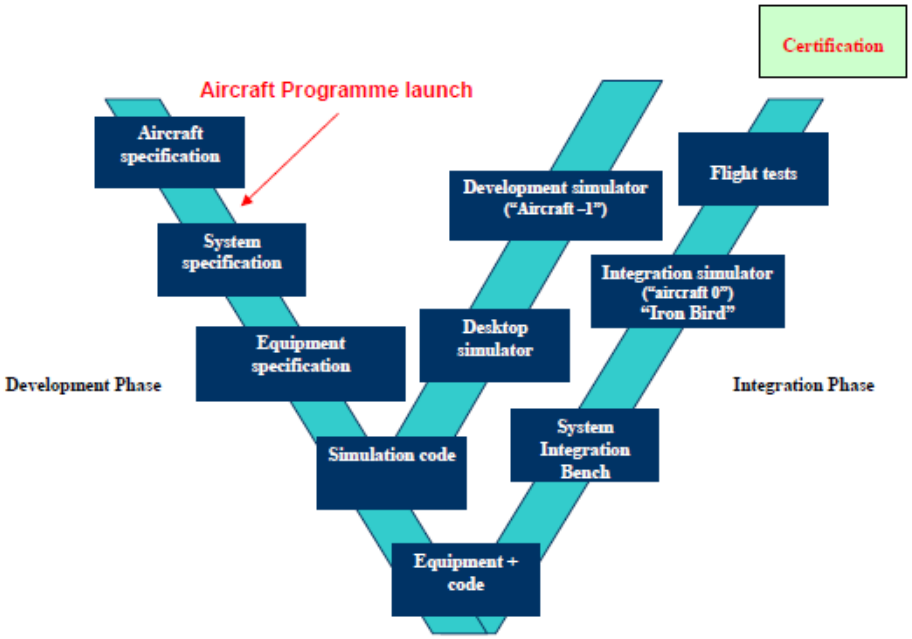

Figure D.4: V-Life cycle for aircraft development

simpliste, consiste à considérer la période d'échantillonnage du système comme étant égale à la périodicité de la tâche dans le cas d'une *tâche périodique [Liu 1973]*.

Cette séparation a comme avantage de permettre à chaque communauté de se concentrer sur leur domaines respectives et s'est avéré être viable dans plusieurs situations.

Cependant, cette manière de faire a conduit à une incompréhension mutuelle des besoins et contraintes de chaque communauté. Les conséquences directes impactent les performances des systèmes implémentés et la difficulté de mise en oeuvre. En effet, avec l'évolution et le développement de la technologie, de la connaissance et des composants, des exigences et des contraintes sans cesse grandissantes existent. Ceci conduit à la complexité des systèmes et requiert l'adaptation des méthodes. Dans la réalité, le fait de considérer que l'échantillonnage est parfait et qu'on peut complètement ignorer la variabilité des retards (et/ou considérer clairement que ces retards sont inexistants) n'est qu'une vision hypothétique et simplifié de la réalité.

Il a été montré (ex: [Louise 2002, Thesing 2004]) que la complexité des unités de calculs (pipeline à plusieurs niveaux, mémoire caches,...) introduite pour augmenter les performances de traitements augmente les difficultés pour calculer le WCET et des temps d'éxecution effective des tâches executées. Or, il est connu qu'un retard non considéré entre la mesure et le rafraichissement de la commande, une variation non prise en compte dans l'échantillonnage dégradent la qualité et la performance de la commande.

Pour pallier ces problèmes, la théorie de l'automatique a évolué en prenant en compte différents types de variabilités (retards, échantillonnages, pertes de paquets, ...).

Plusieurs travaux ([Seuret 2006, Naghshtabrizi 2006] proposent des approches inno-

vantes et intéressantes. Toutefois, ces travaux sont plus orientés vers les analyses de stabilité et ne considèrent pas la problématique de l'implémentation sur calculateur.

D'un autre côté, la communauté informatique temps réel s'efforce à réduire les variabilités des paramètres d'execution sur une cible donnée en améliorant les méthodes d'analyse de temps d'éxecution (par exemple: analyse des algorithmes de remplissage des mémoires caches [Louise 2002, Thesing 2004] , analyse statique de programmes [Reineke 2007, Cullmann 2010, Burguière 2006], ...

Il est donc souhaité d'intégrer la prise en compte de ces perturbations à la fois dans la conception et l'implémentation. C'est ainsi que des notions comme la conception conjointe commande-ordonnancement [Simon 2006a, Xia 2006] , le "control-aware computing" et le "computing-aware control" (cf: [Aubrun 2010]) ont vu le jour.

Ces notions mettent en avant la prise en compte de des contraintes d'implémentation informatique d'un système lors de la conception des lois de commande et vice versa considère lors de l'implémentation des contraintes de paramètres qui peuvent influer sur la performance de commande

D.3.2 Objectifs généraux de la thèse

Nous souhaitons améliorer l'utilisation des ressources de calcul tout en préservant la stabilité et les performances du système. Nous proposons donc de relâcher les contraintes temps-réel selon l'ordonnancement et les objectifs de commande suivants (Figure D.5) :

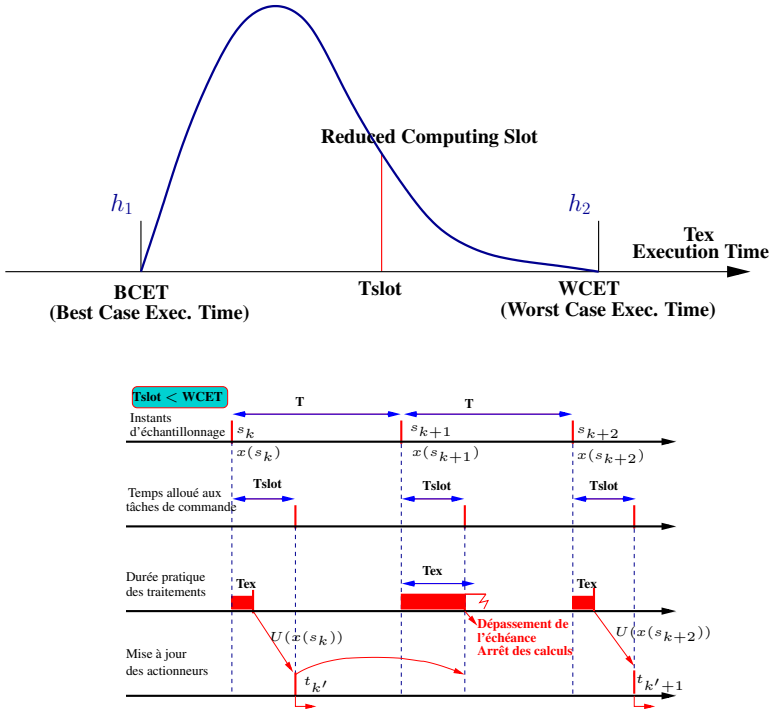

Figure D.5: Proposition de diagramme temporel d'exécution de tâche

L'échantillonnage des données capteurs est toujours effectué périodiquement (de péri-

ode T). Un temps $T_{slot} < WCET$ est maintenant alloué à la tâche de contrôle. Le signal de contrôle obtenu est envoyé aux actionneurs à la fin du temps T_{slot}, c'est-à-dire que $U(s_k))$ est envoyé à l'instant $s_k + T_{slot}$, où T_{slot} représente le retard tel que :

$$\forall t \in [s_k + T_{slot}, s_{k+1} + T_{slot}[, \quad U = U(s_k).$$

Cependant, avec une allocation de temps d'exécution réduite, il peut arriver que occasionnellement une tâche de contrôle dépasse son échéance. Il est alors proposé d'arrêter le traitement en cours, de maintenir la précédente valeur de commande $U(s_k))$ pendant la prochaine période, et de lancer un nouveau traitement utilisant la nouvelle mesure. Ainsi, si le non respect de l'échéance se fait à l'instant $s_k + T_{slot}$, le signal de contrôle est :

$$\forall t \in [s_k + T_{slot}, s_{k+2} + T_{slot}[, \quad U = U(s_k),$$

et pour N violations d'échéance consécutives :

$$\forall t \in [s_k + T_{slot}, s_{k+N} + T_{slot}[, \quad U = U(s_k).$$

En d'autres termes, un nouveau signal de commande est envoyé aux actionneurs, si et seulement si l'exécution de la tâche de contrôle s'est terminée correctement. La commande est appliquée à des instants non équidistants $t_{k'}$ tels que :

$$t_{k'} = s_k + T_{slot} \quad \text{si} \quad T_{ex} \leq T_{slot},$$

où k' est un entier positif qui représente le nombre des entrées calculées avant $s_k = kT$. Ainsi, les entrées du système peuvent être asynchrones puisque l'intervalle entre deux instants d'échantillonnage $t_{k'+1} - t_{k'}$ varie dans le temps mais reste toutefois bornée entre T et NT. Les nouveaux instants d'échantillonnage sont alors $t_{k'+1} - t_{k'} = \alpha T$, où α est un entier $[1, ..., N]$. L'échantillonnage asynchrone est déterminé par le couple (T, N).

Comme il a déjà été observé et exploité ([Cervin 2002, Felicioni 2010]), un système de commande en boucle fermée peut rester stable malgré des aléas temporels d'exécution, au prix d'une dégradation contrôlable des performances. Ainsi, étant donnés un système LTI, une loi de commande, une distribution des temps d'exécution sur l'unité de calcul et une spécification des paramètres d'ordonnancement nominaux, les problèmes à résoudre peuvent être formulés comme suit :

- Trouver N, le nombre maximal de dépassements d'échéances consécutifs avant instabilité du système en boucle fermée;
- Trouver une valeur adéquate de T_{slot} pour gérer le compromis entre qualité de commande et utilisation des ressources;
- Évaluer la robustesse du système en boucle fermée asynchrone par rapport aux incertitudes sur les paramètres du processus.

La méthodologie à appliquer afin d'atteindre les objectifs fixés sont résumés dans la figure D.6.

Pour améliorer l'efficacité de l'utilisation des ressources des ordinateurs embarqués, tout en maintenant la stabilité et la performance de commande, la thèse propose d'alléger les contraintes temps réel en se basant sur la robustesse des lois de commande en boucle fermée. Le processus d'analyse complète est proposé dans la suite afin de formaliser la manière de faire qui résultera au choix optimal du temps d'exécution qui sera alloué à une tâche de commande pour un système donné.

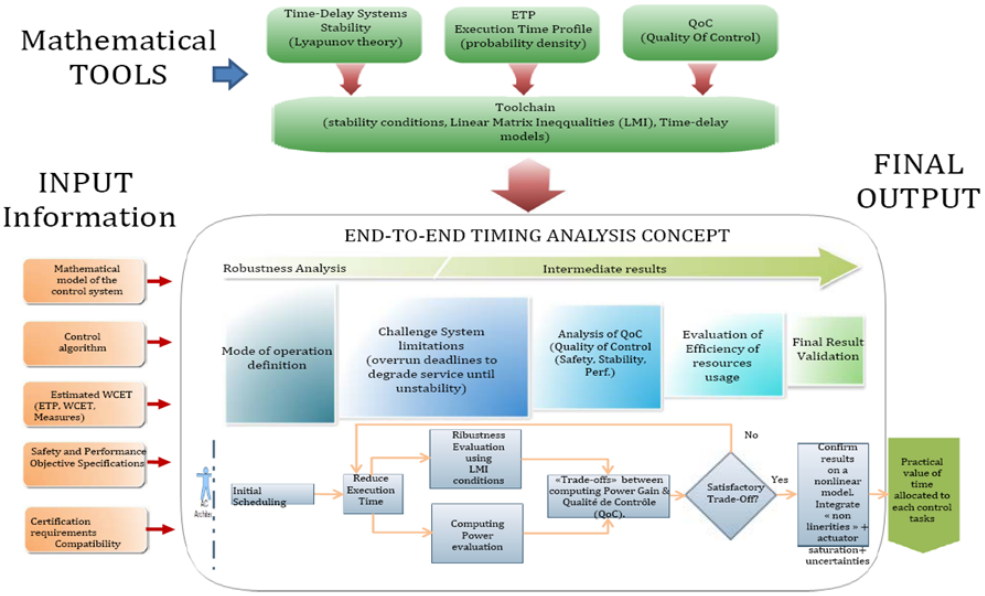

Figure D.6: Méthodologie d'analyse

Dans la phase préliminaire de l'analyse, les éléments suivants doivent être spécifiés:

- Le modèle mathématique du système. Ce modèle peut être exprimé sous des formes classiques (modèles d'état ou fonctions de transfert)
- les lois de commande
- une estimation du WCET afin d'avoir un temps alloué de référence pour les tâches de commande (remarque: une approche statistique comme présentée dans [Bernat 2003] du problème est idéale.
- une spécification des performances de commande et de la sureté que le système doit avoir. Ces éléments permettent de définir les objectifs du système sur le compromis qui doit être fait entre la sureté d'exécution (la sureté d'exécution peut être considéré comme étant la garantie de ne pas dépasser les échéances) et les performances de commande. Un autre exemple serait les spécifications de certification. En effet, les organismes de certification pourraient demander des justificatifs sur les performances atteignables, les quantifications des taux d'évènements indésirables, le fonctionnement au pire cas.... Ces genres d'informations permettront de définir les objectifs à atteindre par le système et ainsi permettront de formaliser le processus de

compromis à faire (à remarquer que cette formalisation n'a pas été faite dans le cadre de cette thèse et donc fera partie des perspectives de recherche [cf D.6]).

D.3.3 Analyse de stabilité

D.3.3.1 Systèmes nominaux

Dans cette section, une étude de la stabilité asymptotique des solutions du système échantillonné (D.1) avec $\mu = 0$ est présentée. Ici, la contribution est basée sur les conditions de stabilité asymptotique développées dans [Seuret 2011a].

Theorem 4 *Considérons un entier N et deux réels positifs T_{slot} et T. Supposons qu'il existe $Q > 0$, $R_1 > 0$ et $R_2 > 0 \in \mathbb{S}^n$, $P > 0$, $U > 0$, $S_1 \in \mathbb{S}^{2n}$ et $S_2, X \in \mathbb{R}^{2n\times 2n}$, $Y \in \mathbb{R}^{5n\times 2n}$ qui satisfont aux conditions suivantes, pour $j = 1, 2$, :*

$$\begin{aligned} \Psi_1(A,B) &= \Pi_1(T_{slot}) + T_j\Pi_2 + T_j\Pi_3 < 0, \\ \Psi_2(A,B) &= \begin{bmatrix} \Pi_1(T_{slot}) - T_j\Pi_3 & T_jY \\ T_jY^T & -T_jU \end{bmatrix} < 0, \end{aligned} \tag{D.4}$$

où $T_1 = T$, $T_2 = NT$ et

$$\begin{aligned} \Pi_1(T_{slot}) &= \mathit{He}\{N_1^TPN_0 - YN_{12} - N_2^TS_2N_{12}\} \\ &\quad + M_1^TQM_1 - M_2^TQM_2 - M_5^TR_1M_5 - N_{12}^TS_1N_{12} \\ &\quad + M_0^T(R_1 + T_{slot}R_2)M_0 - M_{12}^TR_2/T_{slot}M_{12} \\ \Pi_2 &= N_0^TUN_0 + \mathit{He}\{N_0^T(S_1N_{12} + S_2^TN_2)\}, \\ \Pi_3 &= N_2^TXN_2, \end{aligned}$$

et

$$\begin{aligned} &M_0(A,B) = [A\ 0\ 0\ BK\ 0]\,, && M_1 = [I\ 0\ 0\ 0\ 0]\,, \\ &M_2 = [0\ I\ 0\ 0\ 0]\,, && M_3 = [0\ 0\ I\ 0\ 0]\,, \\ &M_4 = [0\ 0\ 0\ I\ 0]\,, && M_5 = [0\ 0\ 0\ 0\ I] \\ &N_0 = [M_0^T(A,B)\ M_5^T]^T, && N_1 = [M_1^T\ M_2^T]^T, \\ &N_2 = [M_3^T\ M_4^T]^T, && M_{12} = M_1 - M_2, \\ &N_{12} = N_1 - N_2. \end{aligned}$$

Le système (D.2) est alors asymptotiquement stable pour tout échantillonnage asynchrone et retard définis respectivement par (T, N) et T_{slot}

Proof 4 *La preuve de ce théorème ne sera pas détaillée dans ce résumé mais est disponible dans [Seuret 2011a].*

Il est à noter que les conditions du théorème 4 incluent les propriétés de stabilité robuste par rapport au retard en entrée T_{slot}. Ceci signifie que (D.4) requiert que le système soit stable au moins pour un retard de transmission T_{slot} et une période $T = T_i$.

D.3.3.2 Systèmes soumis à des incertitudes

Dans cette section, nous considérons que μ n'est pas nul. Nous allons donc étendre le théorème précédent pour couvrir le cas des systèmes soumis à des incertitudes variant dans le temps. Dans le théorème précédent, les conditions dépendent linéairement des matrices du modèle continu du système. Le corollaire suivant présente une extension de ce théorème au cas des système incertains ou à temps variant.

Corollary 2 *Soient un entier N et les scalaires positifs T_{slot}, T et μ. Soient les matrices Q, R_1, R_2, P, U, S_1 et S_2 comme dans le théorème 4 et $X_i \in \mathbb{R}^{2n\times 2n}$, $Y_i \in \mathbb{R}^{5n\times 2n}$ qui satisfont aux conditions suivantes, pour $i = 1,\ldots,M$ et $j = 1,2$:*

$$\Psi_1(A_i, B_i) < 0, \quad \Psi_2(A_i, B_i) < 0, \tag{D.5}$$

où les éléments du polyto-pe $A_i = A + \mu A_i$ et $B_i = B + \mu B_i$. Le système (D.2) est alors asymptotiquement stable pour la période d'échantillonnage définie par T et le retard défini par T_{slot}.

Proof 5 *Considérons les conditions de stabilité décrites dans le théorème 4. En notant que*

$$M_0(\bar{A}(t), \bar{B}(t)) = \sum_{i=1}^{M} \lambda_i(t) M_0(A_i, B_i),$$

et en introduisant les matrices variables $Y(t) = \sum_{i=1}^{M} \lambda_i(t) Y_i$ et $t) = \sum_{i=1}^{M} \lambda_i(t) X_i$ et en utilisant le complément de Schur, on remarque que les deux conditions sont linéaires par rapport aux matrices M_{0i} and N_{0i} et donc, pour $j = 1,2$, $\Psi_j(\bar{A}(t), \bar{B}(t)) = \sum_{i=1}^{M} \lambda_i(t) \Psi_j(A_i, B_i)$.

D.4 Les scénarii d'allègement des contraintes

Comme il a été mentionné précédemment, et reporté dans [Cervin 2002], les systèmes de commandes sont cités comme étant des systèmes temps réel durs. Toutefois, cette affirmation est donnée sans argumentation tangible. Or, selon [Cervin 2003a, Cervin 2005, Simon 2006b], les systèmes en boucle fermée présentent une robustesse naturelle vis à vis des incertitudes liées aux paramètres du processus. Ces travaux, affirmations et experimentations nous permettent d'affirmer qu'un système en boucle fermée peut garder sa stabilité malgré la présence d'incertitudes ou la présence de perturbations.

Les perturbations temporelles et incertitudes de modèles sont considérés dans ce travail de thèse.

Afin de pallier à ces perturbations, la thèse propose des scénarii de gestion de violation de deadline. Comme mentionné dans [Cervin 2005], plusieurs modèles de gestion de dépassement sont proposés dans cette thèse, par exemple **Skip**, **Abort** ou **Queue**. Tous ces scénarii supposent que les tâches s'éxécutent de manière périodique et que les temps de calculs des lois T_{slot} sont plus petits que le WCET (voir figure D.7).

Pour être le plus proche possible des pratiques classiques, il faut que les tâches de contrôles soient périodiques et soient lancées au moment de l'arrivée des nouvelles mesures. En réduisant le temps alloué aux tâches en dessous de la valeur maximale, WCET, on permet l'utilisation plus efficace des ressources de calculs. De plus, comme les temps T_{slot}

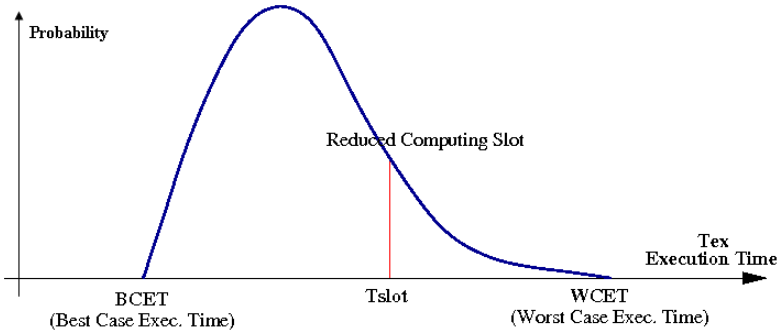

Figure D.7: Reduced computing slot

sont modélisés dans les systèmes en boucle fermée comme des latence d'E/S (Entrées-Sorties de bout en bout), la réduction de cette grandeur résultera naturellement aussi à de meilleures performances de commandes. Toutefois, la réduction de la latence occasionnera aussi des dépassements d'échéances ce qui entrainera la perte "des calculs en cours" dans le cas où on prend le scénario "Abort" (expliqué ci-dessous).

Il est attendu que les perturbations dûes à la variabilité des temps d'éxecution et aux pertes d'échantillons occasionnelles dans la boucle de commande soient compensés par le gain obtenus par la réduction du retard de bout en bout (et éventuellement de la réduction de la période d'échantillonnage qui s'ensuit).

1 Abort Les mesures sont supposées être toujours périodiques de période T, et leur arrivée activent le lancement des tâches de commandes. Le temps T_{slot} que l'on alloue maintenant aux tâches de commande est dans ce cas-ci strictement inférieur au WCET ($T_{slot} < WCET$). L'envoie des nouvelles commandes aux actionneurs arrivent à la fin de temps T_{slo}, c'est à dire $U(x(s_k))$ est envoyé à l'instant $s_k + T_{slot}$, et le retard est égal à une valeur constante T_{slot}

$$\forall t \in [s_k + T_{slot}, s_{k+1} + T_{slot}[, \quad U = U(x(s_k)).$$

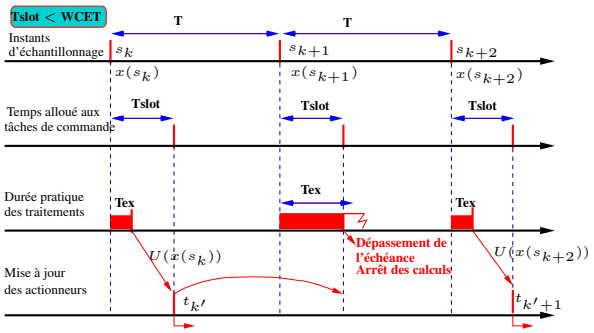

Figure D.8: Abort scheduling pattern

Toutefois, dans le cas présent, il arrive qu'occasionnellement une échéance temporelle est manquée : dans ce genre de cas, le scénario propose d'avorter ("de tuer" et non pas préempter) la tâche actuelle, de maintenir les anciennes valeurs des entrées (résultats de calculs pour la mise à jour des actionneurs) pour la prochaine période et recommencer un calcul propre avec les nouvelles mesures des capteurs. Dans ce cas, le signal de commande $U(x(s_k))$ est gardé pour une période en plus si le dépassement d'échéance arrive à l'instant $s_k + T_{slot}$. Ceci implique qu'un dépassement d'échéances entraine la perte de l'échantillon en cours.

$$\forall t \in [s_k + T_{slot}, s_{k+2} + T_{slot}[, \quad U = U(x(s_k)),$$

et pour N dépassements consecutives d'échéances:

$$\forall t \in [s_k + T_{slot}, s_{k+N} + T_{slot}[, \quad U = U(x(s_k)).$$

En d'autres mots, un signal de commande nouvellement calculé est envoyé aux actionneurs à des instants non équidistants $t_{k'}$ et ne sera envoyé que si et seulement si le calcul a été mené à termes avec succès.

$$t_{k'} = s_k + T_{slot} \quad \text{if} \quad T_{ex} \leq T_{slot},$$

où k' est un entier positif représentant le nombre de valeurs d'entrée avant $s_k = kT$.

Alors, l'entrée de la commande est considéré comme asynchrone (au sens signifiant non périodique) parce que la durée entre deux instants d'échantillonnage successifs $t_{k'+1} - t_{k'}$ devient variant dans le temps. Cependant cette grandeur est aussi bornée par T et NT. Ainsi, $t_{k'+1} - t_{k'} = \alpha T$, dans lequel l'entier $\alpha \in [1, ..., N]$ et les paramètres d'échantillonnage est déterminé par le couple (T, N).

2 Abort' Là encore, une fenêtre de temps $T_{slot} < WCET$ est allouée à l'exécution de la tâche de commande. Cependant, la période d'échantillonnage est réduite de la même quantité. Ainsi le reste du temps, qu'on notera T_{others} sera plus grande. Dans ce cas, une attente d'amélioration de la performance de commande est attendue à la fois par la réduction de latence mais aussi par la réduction de la période d'échantillonnage (Figure D.9).

A première vue, du point de vue de l'implémentation en informatique, ces deux scénarii peuvent être difficile à implémenter. En effet, dans le cas d'un dépassement d'échéances, il est nécéssaire d'avorter instantanément la tâche de commande en cours et de recommencer proprement une nouvelle instance dans la prochaine périodicité avec des nouvelles mesures. Ce modèle de fonctionnement est difficile à implémenter dans un cas pratique (ex: dans le cas d'un thread POSIX en langage C), dans lequel le corps de la thread est la fonction implémentant la loi de commande, ne peut pas être aussi facilement " tué" et recommencé dans les contextes des besoins des scénarii décrits.

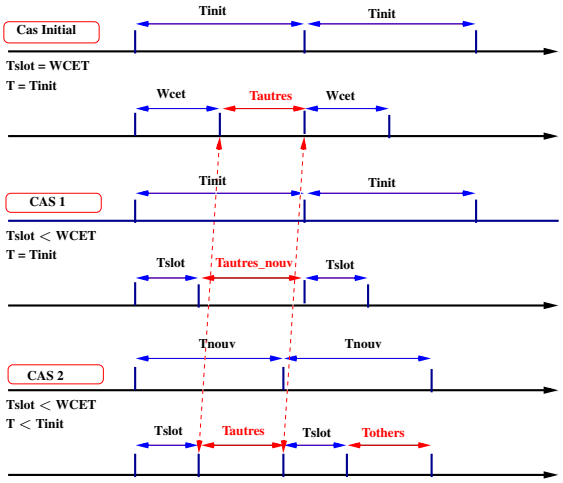

Figure D.9: Abort and Abort' scheduling cases

3 Skip Pour le cas nominal ce scénario est similaire au Cas 1, où la période d'échantillonnage de la tâche de commande est maintenue égale au cas initial. Cependant, en cas de dépassement d'échéances, plutôt que d'abandonner l'execution de l'instance en cours , la tâche continue son calcul à termes.

L'exécution de la tâche qui a commencé pendant l'intervalle de T_{slot_k} continue pendant la tranche suivante affecté à cette tâche ($T_{slot_{k+1}}$) , et éventuellement sur les créneaux suivants $T_{slot_{k+n}}$, jusqu'à sa finition.

Le dernier signal de commande calculé est maintenu comme valeur de rafraichissement des actionneurs jusqu'à ce qu'un nouveau signal de commande ait été envoyé aux actionneurs.

La figure D.10 montre le cas où $U(x(s_k)$ est maintenu jusqu'à $s_{k+2} + T_{slot}$ en raison du dépassement d'échéance dans l'intervalle de $T_{slotk+1}$, ce qui a pour conséquence que le calcul d'un nouveau signal de commande n'est pas fini.

Notez qu'ici le signal de commande envoyé à l'instant $s_{k+2} + T_{slot}$ est $U(x(s_{k+1}))$ et est calculée en utilisant les mesures s_{k+1} obtenu une période avant.

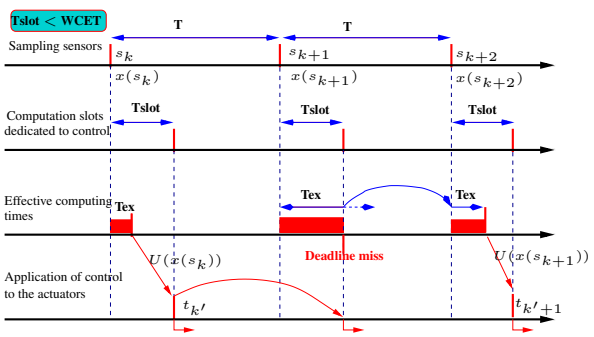

Figure D.10: Skip scheduling pattern

4 Spare Notez qu'une certaine réserve de temps de calcul peut être faite à l'intérieur de la fenêtre de temps T_{others}. Usuellement, ce temps en réserve peut être utilisé pour des activités annexes de l'unité de calcul, mais peut aussi être utilisé pour gérer des activités sporadiques, comme la gestion de dépassement d'échéances (Figure D.11).

Dans ce cas, si la fenêtre de temps libre est suffisamment grande pour terminer le calcul en cours d'exécution, le signal de commande peut être envoyé soit juste à la fin de la fenêtre de temps (cela pour minimiser la latence d'E/S) où à la fin de la période (dans l'idée de minimiser la gigue de sortie).

Les calculs qui ne sont pas finis dans cette fenêtre de temps pourront continuer dans la fenêtre de temps disponible dans la prochaine périodicité jusqu'à leurs complétions.

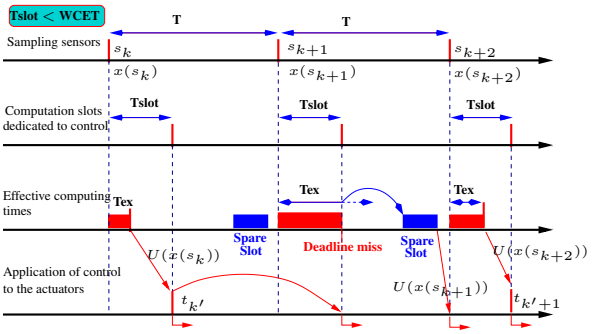

Figure D.11: Slipping on a spare slot

5 Skip' En combinant les deux cas précédent (2 et 3), on peut proposer de réduire la période d'échantillonage selon la réduction T_{slot} (comme dans le cas **Abort'**), et d'adopter le schéma **Skip** pour la gestion des dépassements d'échéance.

7-> Just-in-time Dans les cas précédents, le signal de commande nouvellement calculé est envoyé aux actionneurs et ensuite l'unité de calcul attend la fin du temps alloué T_{slot}. Ceci pour permettre la minimisation des gigue mais surtout pour assurer l'équidistance de l'échantillonage aux niveaux des actionneurs. Cependant, une alternative consiste à envoyer directement le signal de commande, dès sa disponibilité, aux actionneurs. En suivant l'idée exposé dans [Buttazzo 2007], le résultat attendu consiste en la compensation des perturbations dues aux gigues de sortie. Ce scénario **Just-in-time** peut être combiné avec un ou plusieurs des scénarii présentés auparavant dans cette section.

...

En réalité, la combinaison des différents scénarii avec différents gestionnaire de dépassement d'échéance est potentiellement nombreuse. La meilleure combinaison est le résultat d'un compromis à évaluer au cas par cas selon les algorithmes de commande proposés, les caractéristiques des processus à contrôler, les caractéristiques des ressources d'éxécution (processeur, mémoire, ...), des contraintes de sécurité et des différentes contraintes de certification ....

Variable	Nominal	$x_{cg} = 0.3\bar{c}$	$x_{cg} = 0.38\bar{c}$
V_T (ft/s)	502.0	502.0	502.0
α(rad)	0.03691	0.03936	0.03544
θ(rad)	0.03691	0.03936	0.03544
Q (rad/s)	0	0	0
Thtl (0-1)	0.1385	0.1485	0.1325
El (deg)	-0.7588	-1.931	-0.05590

Table D.1: Conditions de vols pour la simulation

D.5 Etudes de cas

D.5.1 Application analytique de la méthode

Cette étude de cas utilise l'approche précédente de robustesse à l'ordonnancement temps-réel relâché de la commande de tangage d'un avion. Nous considérerons seulement un des modes longitudinaux de l'avion, que l'on appelle "le mode oscillatoire à courte période".

Ce modèle est linéarisé avec des conditions nominales de vols données fournis par un ouvrage de référence du domaine ([Stevens 2003]) :

$$\begin{cases} E\dot{x} &= Fx + Gu, \\ y &= Cx. \end{cases} \tag{D.6}$$

Le vecteur d'état est $x = \begin{bmatrix} \alpha & \theta & q \end{bmatrix}$ où α est l'angle d'attaque, θ est l'angle de tangage, q est la vitesse de tangage, le vecteur d'entrée $u = \delta_E$ contient uniquement la déflection de la gouverne de profondeur de l'avion, dont les paramètres sont résumés dans le tableau D.1.

$$\begin{aligned} E &= \begin{bmatrix} V_T - Z_{\dot{\alpha}} & 0 & 0 \\ 0 & 1 & 0 \\ -M_{\dot{\alpha}} & 0 & 1 \end{bmatrix}, \; H = \begin{bmatrix} \frac{180}{\pi} & 0 & 0 \\ 0 & \frac{180}{\pi} & 0 \\ & 0 & 0 & 1 \end{bmatrix}, \\ G &= \begin{bmatrix} Z_{\delta_e} \\ 0 \\ M_{\delta_e} \end{bmatrix}, \quad F = \begin{bmatrix} Z_\alpha & -g_0' sin\gamma_e & V_T + Z_q \\ 0 & 0 & 1 \\ M_\alpha & 0 & M_q \end{bmatrix}. \end{aligned} \tag{D.7}$$

On note que la matrice E est toujours inversible dans les conditions normales de vols. Les paramètres du modèle sont données à partir du modèle standard d'avion fourni par ([Stevens 2003]).

En pratique, un filtre passe-bas est systématiquement rajouté à la lecture de l'angle d'attaque α. La dynamique de la gouverne de profondeur ainsi qu'un intégrateur agissant sur la consigne (pour assurer une erreur nulle en régime permanent) sont ajoutés au système initial. Le vecteur d'état devient

$\begin{bmatrix} \alpha & q & \delta_E & \alpha_F & \beta \end{bmatrix}$ où α_F est la mesure de α filtré, et β est la sortie de l'intégrateur. Le modèle d'état complet est :

$$\begin{cases} \dot{x} = Ax + Bu, \\ y = Cx, \end{cases} \tag{D.8}$$

où $A = E^{-1}F$ et $B = E^{-1}G$. Dans ce cas d'étude, les paramètres sont ceux d'un avion F16 bien documenté dans la littérature ([Stevens 2003] par exemple).

Les paramètres de vol nominaux à altitude nulle donnent le modèle numérique suivant :

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 & 0 & 0 \\ 0.82225 & -1.07741 & -0.17555 & 0 & 0 \\ 0 & 0 & -20.2 & 0 & 0 \\ 10.0 & 0 & 0 & -10 & 0 \\ 0 & -57.2958 & 0 & 0 & 0 \end{bmatrix},$$
$$B = \begin{bmatrix} 0 \\ 0 \\ 20.2 \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 0 & 57.2958 & 0 \\ 0 & 57.2958 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.9}$$

Un contrôleur par retour de sortie $u = Ky = -k_\alpha \alpha_F - k_q q - k_i \beta$ est conçu en utilisant les techniques de placement de pôle classique, donnant $K = [-0.04238; -0.4098; 0.8426]$. La période d'échantillonnage du contrôleur est fixée à $T = 0.08$sec et le temps d'exécution d'une tâche de contrôle supposé égal à $WCET = 0.02$ sec (en supposant qu'il existe 4 contrôleurs qui se partagent l'unité de traitement). Les matrices A_i et B_i, éléments de la combinaison convexe $\Delta_\mu A(t)$ et $\Delta_\mu B(t)$ sont définies dans un polytope à deux sommets définis par $A_i = (-1)^i A$ et $B_i = (-1)^i B$ pour $i = 1, 2$. Le principe consiste à commencer l'analyse en partant des hypothèses classiques "au pire cas" décrites dans la figure D.2, où une fenêtre de temps égale au $WCET$ est allouée à la tâche de contrôle avec une période T et ensuite à réduire pas à pas cette grandeur. Deux scénarios sont alors considérés dans l'étude (Figure D.12) :

Cas 1 : Une fenêtre de temps $T_{slot} < WCET$ est allouée à la tâche, et la période du système garde sa valeur initiale T_{init}. Dans ce cas, un gain en temps de traitement est obtenu (qui pourra être utilisé pour d'autres activités du processeur) avant l'arrivée de la prochaine activation. Des améliorations sont alors attendues grâce à la réduction du retard de $WCET$ à T_{slot};

Cas 2 : Un temps $T_{slot} < WCET$ est alloué à la tâche de contrôle, mais cette fois la période du système est aussi réduite de la même quantité, soit $T_{nouv} = T_{init} - (WCET - T_{slot})$. Dans ce cas, des améliorations sont attendues à la fois par la réduction du retard et de l'intervalle d'échantillonnage.

Les conditions de stabilité du théorème 4 sont utilisées pour trouver des liens entre le gain en temps de traitement (que l'on note $\varepsilon = \frac{T_{slot}}{WCET}$) et N, le nombre maximum de dépassements d'échéances (et donc de perte d'échantillon) consécutifs tolérables avant instabilité du système. L'analyse du Cas 1 (partie gauche de la figure D.13), montre que la tolérance aux dépassements d'échéance du contrôleur, mesurée par N, croît quand T_{slot} et donc le retard sont réduits. Cette expérience montre aussi qu'une grande incertitude des paramètres du système (μ croissant)réduit la robustesse du contrôleur aux pertes d'échantillons. Dans le Cas 2, (partie droite de la figure D.13), la robustesse du système aux

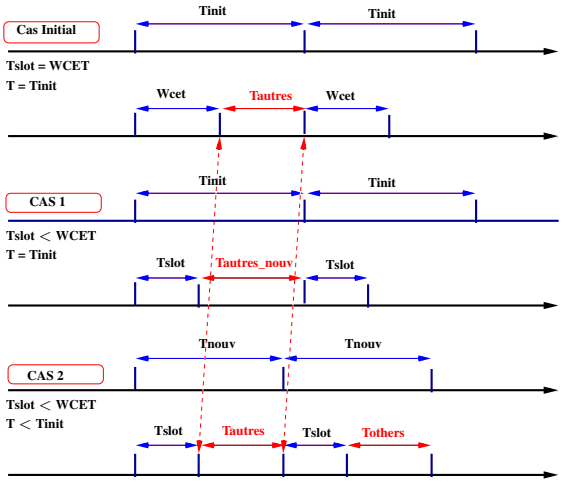

Figure D.12: Ordonnancements proposés

dépassements d'échéances est encore améliorée du fait que ce scénario d'ordonnancement induit à la fois une réduction de l'intervalle d'échantillonnage et du retard.

Cependant, la réduction de T_{slot} augmente manifestement le risque de perte d'échantillon. Pour une distribution de temps d'exécution donnée, la probabilité de dépasser une échéance varie de 1 à 0 lorsque le facteur d'ordonnancement ε varie de 1 ($T_{slot} = WCET$) à une valeur minimale $T_{slot} = BCET$ (Best Case Execution Time).

En supposant que les durées d'exécution des tâches sur des intervalles successifs sont des variables indépendantes, la probabilité d'occurrence d'un évènement où l'on atteint le nombre maximal de dépassements d'échéances consécutives admissibles est donnée dans la figure D.14 pour les deux scénarios décrits et pour différentes valeurs de μ.

La propriétés d'indépendance est requise par les différents hypothèses considérés dans de la méthode de construction du profil d'exécution sur l'unité de calcul considéré (figure D.14). Cette méthode se base théoriquement sur la notion de EVT (Extreme Value Theory), comme présenté dans [Petters 2007].

Cependant cette propriété d'indépendance des temps de calcul est à vérifier. En effet, il est possible que des jeux d'entrées identiques donnent des temps d'exécution semblables. Seul une validation expérimentale permettra de valider ou non cette hypothèse.

Cette valeur peut être utilisée comme métrique de conception en alternative à l'approche du pire cas. Ainsi, pour un scénario d'ordonnancement considéré et une hypothèse sur les incertitudes du système, il est facile de calculer la valeur de ε correspondant à un taux de défaillance spécifié.

D.5.2 Validation par simulation

Selon la figure D.6, il est important de vérifier que les résultats obtenus tout au long du processus d'analyse sont corrects. Ces vérifications se feront en premier lieu par simulation, et ensuite de préférence par experimentation sur cible directe.

La première étape consiste à vérifier que les résultats obtenus dans le calcul des LMI

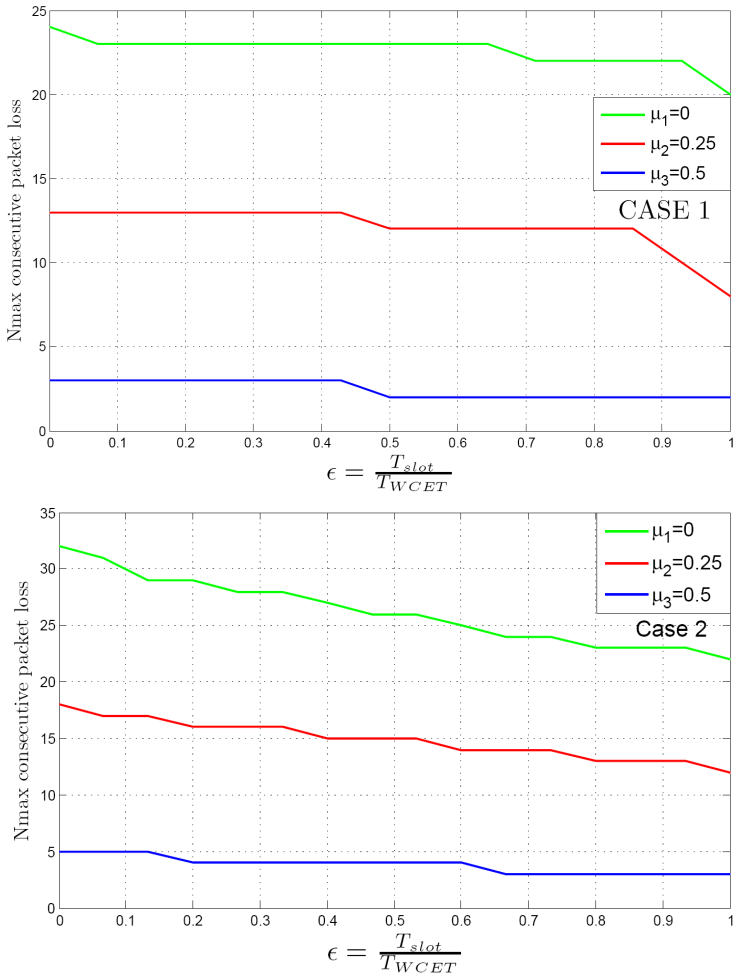

Figure D.13: Nombre maximum de dépassements d'échéances consécutifs admissibles

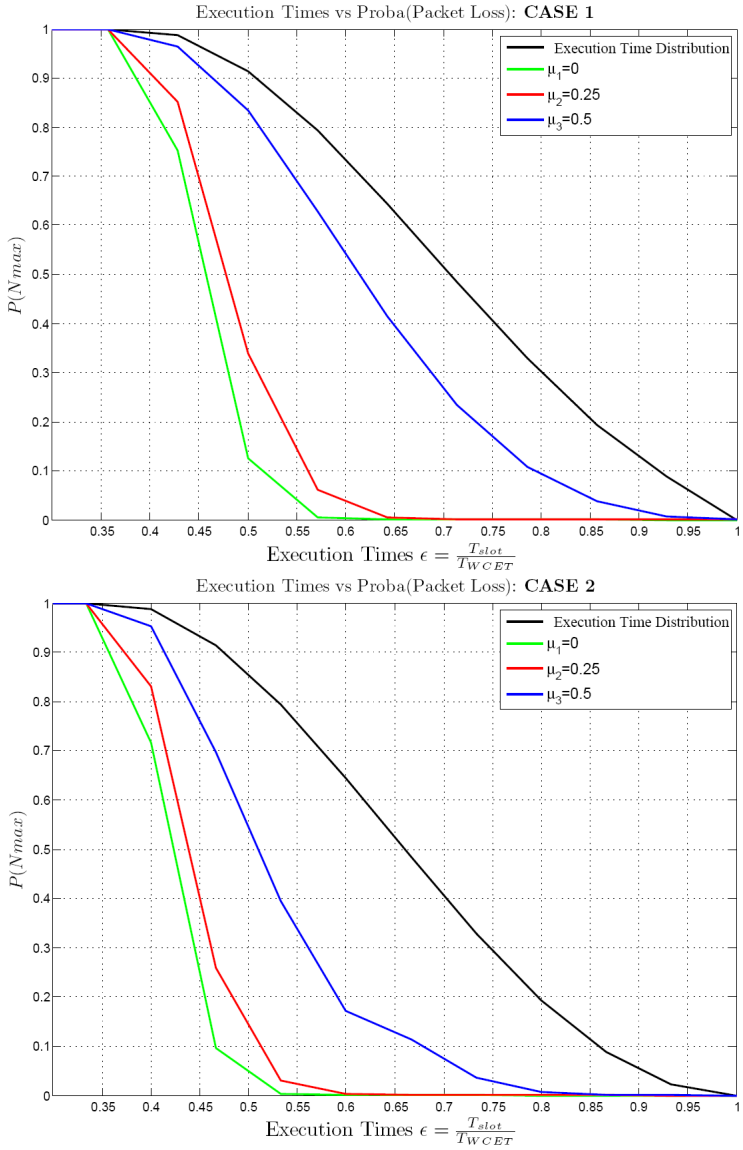

Figure D.14: Probabilité d'atteindre le nombre maximum de dépassement d'échéance consécutive admissible

dans la section D.3.3 appliqués à l'étude de cas présenté dans la section D.5.1. L'outil TrueTime [Cervin 2009]a été choisi pour faire la simulation. Le choix est justifié par le fait que cet outil est orienté conception conjointe conception ordonnancement ([Cervin 2003a]. De plus les fonctionnalités disponible de l'outil permettent de simuler, par programmation, des pertes d'échantillonnage, des avortements de tâches temps réels, ...

Simulation sur modèle linéaire

Le modèle de simulation est exactement le même modèle que celui utilisé pour les calculs. La différence réside dans le fait que cette fois, il est implémenté sous l'outil Mathworks Simulink.

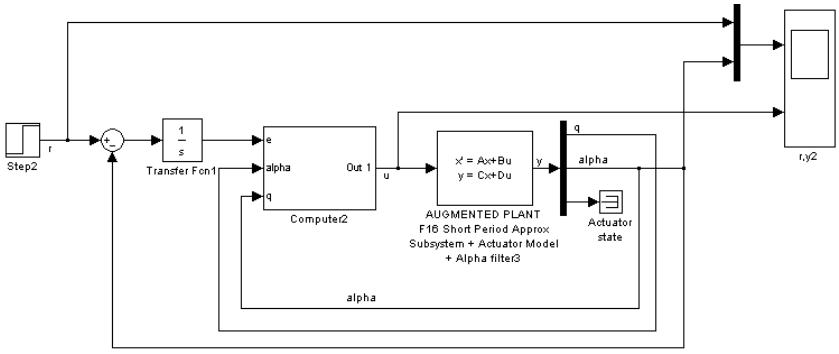

Figure D.15: Pitch Control System with a TrueTime Controller

Le modèle de simulation en TrueTime & Simulink est donné par la figure D.15

Ce modèle intègre un objet TrueTime Kernel pour simuler l'ordinateur embarqué et par le biais duquel, il est possible d'implémenter des tâches de commandes avec les paramètres d'éxecution nécéssaires et les scénarios de gestion de dépassement d'échéances souhaités.

Le modèle de processus à contrôler intégré dans ce modèle est composé d'un composant simulink "State Space Blockset". Cet objet permet d'insérer directement les matrices d'états du systèmes dont les valeurs ont été reprise à partir du modèle numérique de la section D.5.1.

Les différents cas de figure exposés dans la section D.5.1 ont été relancés. Dans ce résumé, il ne sera présenté que le cas où $\mu = 0$. Le reste de la simulation étant présente dans le rapport de thèse complet.

Ce premier cas avec $\mu = 0$ présente la simulation dans le cas où les paramètres du processus ne présente aucune incertitude. Ainsi, seuls le nombre maximal de pertes d'échantillons admissibles est analysé par les figures.

En analysant les différentes figures, D.20, on peut voir que les résultats des calculs analytiques sont plus ou moins conformes. Toutefois, on peut aussi noter que même si le système est asymptotiquement stable, les performances sont insatisfaisantes pour des valeurs trop grandes de N. Il est à noter que le pessimisme des résultats analytiques comparés aux valeurs obtenus par la simulations proviennent du conservatisme des critères de stabilité obtenus par la méthode de Lyapunov Krasovskii.

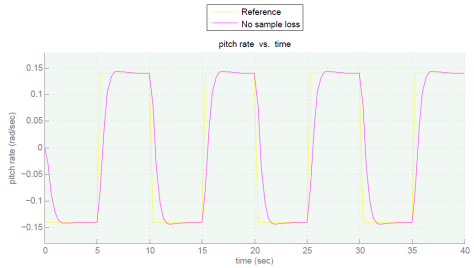

Figure D.16: No sample loss

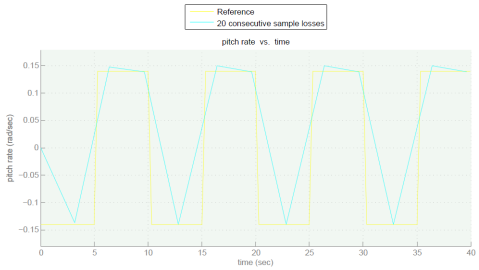

Figure D.17: 20 consecutive sample losses

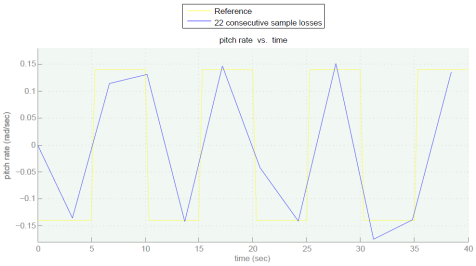

Figure D.18: 22 consecutive sample losses

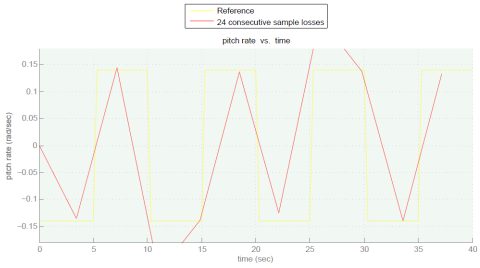

Figure D.19: 24 consecutive sample losses

Figure D.20: Simulation results for $\mu = 0$

Simulation sur modèle non linéaire

L'idée de cette section est de comprendre les limites du système étudié auparavant par rapport aux incertitudes liées aux non linéarités du modèle, aux couplages avec les différents axes de l'avion et aux interferences lors de l'intégration du modèle de tangage étudié dans un modèle d'avion complet.

Le modèle d'avion considéré est toujours celui du F-16. Il a été écrit en langage C avec interfaçage Matlab/Simulink afin d'être réutilisé avec le modèle étudié dans la section D.5.2.

Ce modèle est présenté par la figure D.21

Pour respecter les hypothèses de vitesse constante et pour éviter des couplage indésirables entre les axes (tangage, roulis et lacet), un contrôleur de poussée moteur simulé avec un PID a été intégré au modèle. Les valeurs intiales des autres entrées ont été mis à neutre (valeur 0) vu de l'exterieur pour ne pas les mettre en l'air, mais les états internes des ces entrées sont initialisés à la valeur d'équilibre de l'avion dans les conditions nominales de vols données par le tableau D.2:

Dans cette simulation nonlinéaire, il est possible de confirmer les résultats obtenus par les LMIs et la simulation linéaire, en prenant en compte les perturbations exterieures et les nonlinéarités du système.

Les résultats pour le cas 1 sont présentés par la figure D.26

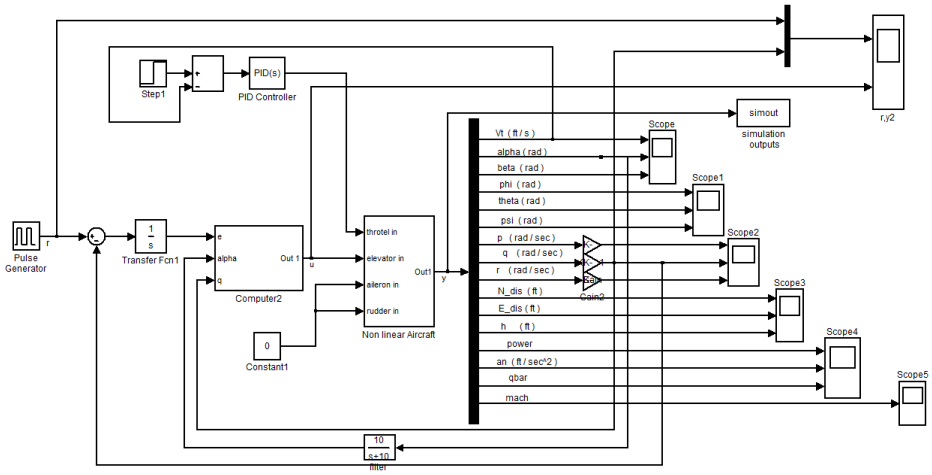

Figure D.21: Nonlinear simulation of aircraft control systems

Variable	Nominal	$x_{cg} = 0.3\bar{c}$	$x_{cg} = 0.38\bar{c}$
V_T (ft/s)	502.0	502.0	502.0
α(rad)	0.03691	0.03936	0.03544
β(rad)	$4, 0.10^{-9}$	$4, 1.10^{-9}$	$3, 1.10^{-9}$
ϕ(rad)	0	0	0
θ(rad)	0.03691	0.03936	0.03544
P (rad/s)	0	0	0
Q (rad/s)	0	0	0
R (rad/s)	0	0	
Thtl (0-1)	0.1385	0.1485	0.1325
El (deg)	-0.7588	-1.931	-0.05590
Ail (deg)	$-1, 27.10^{-7}$	$-7, 0.10^{-8}$	$-5, 1.10^{-7}$
Rdr (deg)	$6, 2.10^{-7}$	$8, 3.10^{-7}$	$4, 3.10^{-6}$
Nominal conditions: $h = 0$, $\bar{q} = 300$ psf, $x_{cg} = 0.35\bar{c}$, $\dot{\phi} \equiv \dot{\theta} \equiv \dot{\psi} \equiv \gamma \equiv 0$			

Table D.2: Trimmed flight conditions for the F-16 simulation model [Stevens 2003]

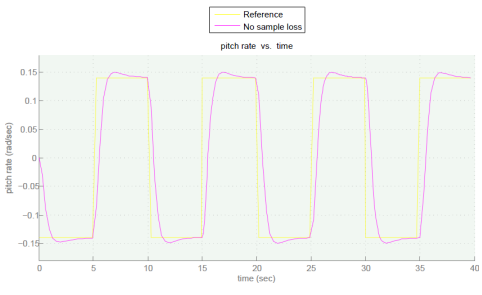

Figure D.22: No sample loss

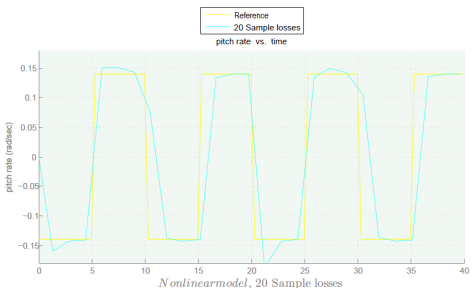

Figure D.23: 20 consecutive sample losses

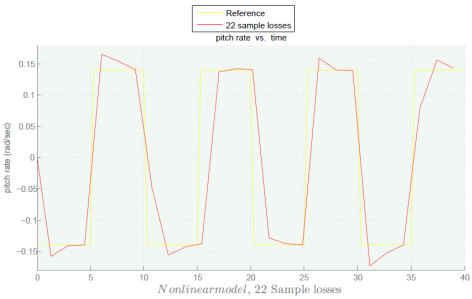

Figure D.24: 22 consecutive sample losses

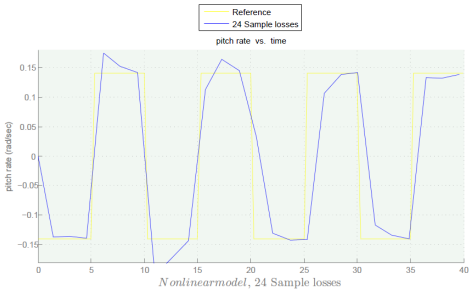

Figure D.25: 24 consecutive sample losses

Figure D.26: Nonlinear simulation results

D.6 Conclusion

Dans cette thèse, une nouvelle méthodologie pour analyser les systèmes de contrôle commande numérique a été proposée. La problématique principale consiste en la détermination au plus précis du temps alloué pour chaque tâche composant le logiciel de commande. Il a été montré que ce problème est à la limite du domaine de l'automatique (ex: la conception des lois de commandes et la définition de leur paramètre temporels) et de l'informatique temps réel (l'implémentation par logiciel et l'implantation sur calculateur de ces lois de commandes).

L'allocation de temps pour les tâches de commandes de systèmes sûrs est d'un grand interêt dans des domaines comme l'avionique, l'automobile car cela implique des enjeux légaux (ex: pour la certification), des enjeux commerciaux (ex: parce que le marché des composants évoluent et que l'entreprise doit y faire face) et impliquent des conséquences sur le système final (masse, consommation electrique, technologie de refroidissement, gestion d'obsolescence,...). Traditionnellement, les concepteurs de lois de commande avaient une approche classique qui consistait à considérer que les paramètres temporelles des lois de commandes sont connus et constants. Ainsi la communauté informaticienne qui implémentait ces lois sur calculateurs allouait à chaque tâche le maximum de temps possible, le WCET. Cela permettait de respecter les exigences des concepteurs en matière de paramètres temporels, mais rendait l'utilisation du système non optimum en matière de performance et d'utilisation efficace des ressources.

En considérant la robustesse naturelle des systèmes de commandes en boucle fermée

(feedback loop), cette approche de conception et de réalisation a été revisitée dans cette thèse. Un modèle spécifique de système a été considéré: les systèmes échantillonnés. En effet, ces systèmes modélisent très précisement les modèles à temps discrèt (computer-controlled systems), car les ordinateurs effectuent leurs calculs en temps discret tout en gardant un procédé en temps continu. Ce type de systèmes permet aussi d'étudier les retards de différents types (variable ou constants). c'est ainsi que les temps d'éxécution des tâches de commandes sont considérés comme un type très spécial de retards: les retards en entrée du système. En se basant sur des travaux récents [Seuret 2011a], la nouvelle méthodologie proposée a été étoffé avec une nouvelle méthode d'analyse de stabilité utilisant la méthode de Lyapunov Krasovskii.

Une caractérisitique importante de la méthode est l'intégration de tolérance aux fautes. En effet, contrairement aux systèmes conçus avec la méthode classique de séparation d'interêt [Xia 2006, Simon 2006b],

l'objectif général est d'assurer la stabilité du système tout en gardant une utilisation efficace des ressources de calculs. Cela implique des méthodes de gestion de charges proposées dans le chapitre D.3.

Un cas d'étude a été proposé pour vérifier l'applicabilité et l'efficacité de la méthodologie. Le système considéré est l'axe de tangage d'un avion F16. Au delà des études analytiques de modélisation, la méthode a été évaluée avec succès en suivant le processus d'analyse proposé par la figure D.6.

- **La faisabilité des scénarios de gestion de dépassement**. Divers scénarios ont été proposés dans la section D.4 mais la faisabilité de ces scénarii dans le cadre d'une implémentation n'a pas été étudié. A titre d'exemple, la solution "Abort" proposée dans le cas 1 est la plus simple à implémenter du point de vue pratique. Toutefois, ce scénario peut être difficile à implémenter correctement sous forme de logiciel.

  D'un point de vue de l'implémentation informatique, le cas d'un dépassement d'échéances où il est nécéssaire d'avorter instantanément la tâche de commande en cours et de recommencer proprement une nouvelle instance dans un nouveau contexte peut être difficile à implémenter

  Ce modèle de fonctionnement est difficile à implémenter dans un cas pratique (ex: dans le cas d'un thread POSIX en langage C). En effet, suivant le langage, il n'est pas forcément facile de tuer et de relancer des tâches.

  Ce type d'analyse de faisabilité devra donc être appliqué pour chaque scénario envisagé afin de donner des perspectives d'application directe.

- **Formalisation du "QoS"**. Là encore, la séparation d'interêt est mise en évidence. En effet, la communauté automatique se concentre plus sur l'analyse de stabilité et la stabilisation des systèmes tandis que la communauté informatique s'efforce d'améliorer le determinisme du système en poussant plus loin les méthodes d'obtention du WCET.

  Comme il a été vu tout au long la thèse, la solution finale n'est pas du domaine du temps réel dur (ni du temps réel souple) mais un compromis entre le gain en performance que l'on peut avoir en réduisant le temps alloué aux tâches de commandes

tout en acceptant des dégradations de performances de temps en temps par la perte d'échantillons. L'idée maîtresse est de formaliser la notion de performance de commande (comme une fonction mathématique) qui permettra de donner un coût objectif capable de capturer les besoins à la fois en qualité de commande et en puissance de calcul. Une telle fonction devra en plus intégrer la notion de sureté de fonction (par exemple pour permettre la mesure de la sureté du système dans une configuration donnée).

Bibliography

[Abdelzaher 2000] Tarek F. Abdelzaher, Ella M. Atkins and Kang G. Shin. *QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control*. IEEE Transactions on Computers, vol. 49, no. 11, pages 1170–1183, 2000. (Cited on page 37.)

[Andrianiaina 2011a] Patrick Jocelyn Andrianiaina, Alexandre Seuret and Daniel Simon. *Robust control under weakened real-time constraints*. In 50th IEEE Conference on Decision and Control and European Control Conference CDC/ECC, Orlando, United States, December 2011. (Cited on page 1.)

[Andrianiaina 2011b] Patrick Jocelyn Andrianiaina, Daniel Simon and Alexandre Seuret. *Robust system control method with short execution deadlines*, Sep 2011. (Patent n° EP11306103.0 claimed by Airbus Operations SAS). (Cited on page 1.)

[Andrianiaina 2011c] Patrick Jocelyn Andrianiaina, Daniel Simon, Alexandre Seuret, Jean-Michel Crayssac and Jean-Claude Laperche. *Weakening Real-time Constraints for Embedded Control Systems*. Rapport de recherche RR-7831, INRIA, December 2011. (Cited on pages 1, 31 and 32.)

[Andrianiaina 2012] Patrick Jocelyn Andrianiaina, Daniel Simon and Alexandre Seuret. *Commande robuste avec relâchement des contraintes temps-réel*. In Conférence Internationale Francophone d'Automatique CIFA2012, Grenoble, France, Jul 2012. (Cited on page 1.)

[Årzén 1999] K.-E. Årzén, B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson and L. Sha. *Integrated Control and Scheduling*. Technical Report ISRN LUTFD2/TFRT--7586--SE, Department of Automatic Control, Lund Institute of Technology, Sweden, August 1999. (Cited on pages 2, 19 and 20.)

[Årzén 2000] K.-E. Årzén, A. Cervin, J. Eker and L. Sha. *An Introduction to Control and Scheduling Co-Design*. In 39th IEEE Conference on Decision and Control, Sydney, Australia, December 2000. (Cited on page 125.)

[Aubrun 2010] Christophe Aubrun, Daniel Simon and Ye-Qiong Song. Co-design approaches to dependable networked control systems. ISTE Wiley, 2010. (Cited on pages 31, 35, 36, 37 and 133.)

[Ayav 2006] Tolga Ayav, Pascal Fradet and Alain Girault. *Implementing Fault-Tolerance in Real-Time Systems by Program Transformations*. Research Report RR-5919, INRIA, 2006. (Cited on page 36.)

[Barre 2006] Jonathan Barre, Cédric Landet, Christine Rochange and Pascal Sainrat. *Modeling Instruction-Level Parallelism for WCET Evaluation*. In 12th IEEE Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA

2006), pages 61–67, Sydney, Australia, August 2006. IEEE Computer Society. (Cited on page 125.)

[Bedin França 2011] Ricardo Bedin França, Denis Favre-Felix, Xavier Leroy, Marc Pantel and Jean Souyris. *Towards Formally Verified Optimizing Compilation in Flight Control Software*. In Philipp Lucas, Lothar Thiele, Benoit Triquet, Theo Ungerer and Reinhard Wilhelm, editeurs, Predictability and Performance in Embedded Systems : PPES 2011, volume 18 of *OpenAccess Series in Informatics (OASIcs)*, pages 59–68, Grenoble, France, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. (Cited on pages 10 and 28.)

[Bernat 2001] Guillem Bernat, Alan Burns and Albert Llamosí. *Weakly Hard Real-Time Systems*. IEEE Trans. Computers, vol. 50, no. 4, pages 308–321, 2001. (Cited on pages 4, 32, 50 and 130.)

[Bernat 2002] G. Bernat, A. Colin and S-M. Petters. *WCET Analysis of Probabilistic Hard Real-Time Systems*. In 23rd IEEE Real-Time Systems Symposium RTSS'02, Washington DC, USA, 2002. (Cited on page 4.)

[Bernat 2003] Guillem Bernat, Antoine Colin and Stefan Petters. *pWCET: a Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems*. Technical report, University of York, Department of Computer Science, January 2003. (Cited on page 135.)

[Bertrand 2009] D. Bertrand, S. Faucou and Y. Trinquet. *An analysis of the AUTOSAR OS timing protection mechanism*. In 14th IEEE international conference on Emerging technologies & factory automation ETFA'09, Palma de Mallorca, Spain, 2009. (Cited on page 10.)

[Briat 2012] Corentin Briat and Alexandre Seuret. *Robust stability of impulsive systems: A functional-based approach*. In Proceedings of the 4th IFAC conference series on Analysis and Design of Hybrid Systems (ADHS'2012), page 6, Eindhoven, Pays-Bas, June 2012. (Cited on page 63.)

[Briére 1993] Dominique Briére and Pascal Traverse. *AIRBUS A320/A330/A340 electrical flight controls - A family of fault-tolerant systems*. In Proceedings of the 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS) 1993, pages 616 – 623, Toulouse, France, June 1993. IEEE Computer Society. (Cited on page 11.)

[Burguière 2006] C. Burguière and Ch. Rochange. *History-based Schemes and Implicit Path Enumeration*. In 6th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis, Dagstuhl, Germany, 2006. (Cited on page 133.)

[Burns 2011] A. Burns and D. Griffin. *Predictability as an Emergent Behaviour*. In Robert I. Davis and Linh T.X. Phan, editeurs, 4th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems, pages 27–29, Vienna, Austria, November 2011. (Cited on pages 112, 113 and 114.)

[Busenberg 1991] S.N. Busenberg and M. Martelli. Delay differential equations and dynamical systems: proceedings of a conference in honor of kenneth cooke held in claremont, california, jan. 13-16, 1990. Lecture notes in mathematics. Springer-Verlag, 1991. (Cited on page 56.)

[Buttazzo 2007] G. Buttazzo and A. Cervin. *Comparative Assessment and Evaluation of Jitter Control Methods*. In Proc. 15th International Conference on Real-Time and Network Systems, Nancy, France, March 2007. (Cited on pages 44, 49 and 141.)

[Cary 2001] Cary and Virginia Williamsburg, editeurs. The Avionics Handbook. CRC Press, 2001. (Cited on pages 9, 14, 15 and 18.)

[Cervin 2000] A. Cervin. Towards the integration of control and real-time scheduling design. Licentiate thesis tfrt-3226, Dpt. of Automatic Control, Lund University, Sweden, May 2000. (Cited on pages 31 and 45.)

[Cervin 2001] Anton Cervin. *Analyzing of the Effects of Missed Deadlines in Control Systems*. In ARTES Graduate Student Conference, Uppsala, Sweden, March 2001. (Cited on page 44.)

[Cervin 2002] A. Cervin, J. Eker, B. Bernhardsson and K.-E. Årzén. *Feedback-Feedforward Scheduling of Control Tasks*. Real-Time Systems, vol. 23, no. 1–2, pages 25–53, July 2002. (Cited on pages 45, 134 and 137.)

[Cervin 2003a] A. Cervin. *Integrated Control and Real-Time Scheduling*. PhD thesis, Dpt of Automatic Control, Lund Inst. of Technology, Sweden, April 2003. (Cited on pages 129, 137 and 147.)

[Cervin 2003b] A. Cervin. *Using Jitterbug to Derive Control Loop Timing Requirements*. In CERTS'03 – Co-Design of Embedded Real-Time Systems Workshop, Porto, Portugal, July 2003. (Cited on page 38.)

[Cervin 2004] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén and G. Buttazzo. *The Jitter Margin and Its Application in the Design of Real-Time Control Systems*. In 10th International Conference on Real-Time and Embedded Computing Systems and Applications, Göteborg, Sweden, August 2004. (Cited on pages 31 and 129.)

[Cervin 2005] Anton Cervin. *Analysis of Overrun Strategies in Periodic Control Tasks*. In Proc. 16th IFAC World Congress, Prague, Czech Republic, July 2005. (Cited on page 137.)

[Cervin 2006] A. Cervin, K-E. Årzén, D. Henriksson, M. Lluesma Camps, P. Balbastre, I. Ripoll and A. Crespo. *Control Loop Timing Analysis Using TrueTime and Jitterbug*. In 2006 IEEE Computer Aided Control Systems Design Symposium, October 2006. (Cited on pages 38 and 94.)

[Cervin 2009] Anton Cervin, Dan Henriksson and Martin Ohlin. *TRUETIME 2.0 beta Reference Manual*, January 2009. http://www.control.lth.se/truetime. (Cited on pages 93, 94 and 147.)

[Chen 1995] T. Chen and B.A. Francis. Optimal sampled-data control systems. Springer-Verlag, Berlin, Germany, 1995. (Cited on page 59.)

[Colin 2001] A. Colin and I. Puaut. *A modular and retargetable framework for tree-based WCET analysis*. In Proc. of the 13th Euromicro Conference on Real-Time Systems, pages 37–44, Delft, The Netherlands, June 2001. (Cited on page 107.)

[Colin 2003] Antoine Colin, Isabelle Puaut, Christine Rochange and Pascal Sainrat. *Calcul de majorants de pire temps d'exécution : état de l'art* . Technique et Science Informatique, vol. 22, no. 5, pages 651–677, 2003. (Cited on pages 106, 107 and 109.)

[Cullmann 2010] Christoph Cullmann, Christian Ferdinand, Gernot Gebhard, Daniel Grund, Claire Maiza, Jan Reineke, Benoît Triquet, Simon Wegener and Reinhard Wilhelm. *Predictability Considerations in the Design of Multi-Core Embedded Systems*. Ingénieurs de l'Automobile, vol. 807, pages 36–42, September 2010. (Cited on page 133.)

[Deverge 2007] Jean-François Deverge and Isabelle Puaut. *Safe measurement-based WCET estimation*. In Reinhard Wilhelm, editeur, 5th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis, Dagstuhl, Germany, 2007. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. (Cited on page 106.)

[DO-178B 1993] DO-178B. Rtca, inc., document rtca/do-178b [electronic resource]. U.S. Dept. of Transportation, Federal Aviation Administration, [Washington, D.C.] :, 1993. (Cited on pages 3 and 15.)

[DO-254 2005] DO-254. Rtca, inc., document rtca/do-254, design assurance guidance for airborne electronic hardware [electronic resource]. U.S. Dept. of Transportation, Federal Aviation Administration, [Washington, D.C.] :, 2005. (Cited on pages 3, 15 and 17.)

[EASA 2009] EASA. *Certification Specifications for Large Aeroplanes CS-25*. Technical Report Amendment 7, European Aviation Safety Agency, 2009. (Cited on page 14.)

[Falla 1997] Mike Falla. *Advances in Safety Critical Systems - Results and Achievements from the DTI/EPSRC R&D Programme in Safety Critical Systems*. Technical report, School of Computing and Communication - Lancaster University (compiled and edited by Mike Falla), 1997. (Cited on page 12.)

[Favre 1994] C. Favre. *Fly-by-wire for commercial aircraft: the Airbus experience*. International Journal Of Control, vol. 59, no. 1, pages 139–157, 1994. (Cited on pages 11 and 12.)

[Felicioni 2010] Flavia Felicioni, Ning Jia, Françoise Simonot Lion and Ye-Qiong Song. *Overload Management Through Selective Data Dropping*. In Co-design Approaches for Dependable Networked Control Systems, pages 187–224. ISTE - Wiley, 2010. (Cited on pages 50, 129 and 134.)

[Fridman 1988] Emilia Fridman, Yu V. Micheev and Sobolev V.A. *Asymptotic analysis of digital control systems*. Automation and Remote Control, vol. 49, no. 9, pages 1175–1180, 1988. (Cited on page 60.)

[Fridman 2004a] E. Fridman. *Stability of linear functional differential equations: A new Lyapunov technique*. In Proceedings of Mathematical Theory of Networks and Systems, September 2004. (Cited on page 57.)

[Fridman 2004b] Emilia Fridman, Alexandre Seuret and Jean-Pierre Richard. *Robust sampled-data stabilization of linear systems - An input delay approach*. Automatica, 2004. (Cited on pages 59, 60, 63, 68 and 130.)

[Fridman 2006] E. Fridman and U. Shaked. *Input-output approach to stability and L_2-gain analysis of systems with time-varying delays*. Systems and Control Letters, vol. 55, pages 1041–1053, 2006. (Cited on page 62.)

[Fridman 2010] E. Fridman. *A refined input delay approach to sampled-data control*. Automatica, vol. 46, no. 2, pages 421–427, 2010. (Cited on page 61.)

[Fujioka 2009] H. Fujioka. *Stability analysis of systems with aperiodic sample-and-hold devices*. Automatica, vol. 45, no. 3, pages 771–775, 2009. (Cited on pages 59 and 60.)

[Goupil 2010] Ph. Goupil and A. Marcos. Fault tolerant flight control, chapitre ch. 19: Industrial Review, pages 521–536. Numeéro 399 de LNCIS. Springer-Verlag, 2010. (Cited on pages 10, 44 and 45.)

[Gu 2003] K. Gu, V. Kharitonov and J. Chen. Stability of time-delay systems. Control engineering. Birkhäuser, 2003. (Cited on pages 54, 55 and 60.)

[Hamann 2006] Arne Hamann, Razvan Racu and Rolf Ernst. *A formal approach to robustness maximization of complex heterogeneous embedded systems*. In Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis, (CODES+ISSS 2006, Seoul, Korea, October 2006. ACM Journal. (Cited on pages 50 and 129.)

[Hansen 2009] Jeffery Hansen, Scott Hissam and Gabriel A. Moreno. *Statistical-based WCET estimation and validation*. In 9th International Workshop on Worst-Case Execution Time (WCET) Analysis ECRTS'09, Dagstuhl, Germany, 2009. (Cited on pages 29 and 128.)

[Heath 2003] S. Heath. Embedded systems design. EDN series for design engineers. Newnes, 2003. (Cited on page 21.)

[Hespanha 2007] J.P. Hespanha, P. Naghshtabrizi and Y. Xu. *A survey of recent results in networked control systems*. Proceedings of the IEEE, vol. 95, pages 138–162, 2007. (Cited on pages 53 and 63.)

[Hetel 2006] L. Hetel, J. Daafouz and C. Iung. *Stabilization of Arbitrary Switched Linear Systems With Unknown Time-Varying Delays*. IEEE Trans. on Automatic Control, vol. 51, no. 10, pages 1668–1674, 2006. (Cited on page 60.)

[Jensen 1906] J. Jensen. *Sur les fonctions convexes et les inégalités entre les valeurs moyennes*. Acta Mathematica, vol. 30, pages 175–193, 1906. 10.1007/BF02418571. (Cited on page 65.)

[Kao 2007] Chung-Yao Kao and Anders Rantzer. *Stability analysis of systems with uncertain time-varying delays*. Automatica, vol. 43, pages 959–970, June 2007. (Cited on page 56.)

[Khan 2009] Usman Khan and Iain Bate. *WCET Analysis of Modern Processors Using Multi-Criteria Optimisation*. In Proceedings of the 2009 1st International Symposium on Search Based Software Engineering, SSBSE '09, pages 103–112, Washington, DC, USA, 2009. IEEE Computer Society. (Not cited.)

[Kirner 2005] R. Kirner and P. Puschner. *Classification of WCET Analysis Techniques*. In 8th IEEE International Symposium on Object-oriented Real-time distributed Computing ISORC'05, pages 190–199, Seattle, Washington, May 2005. (Cited on page 127.)

[Kolmanovskii 1996] Vladimir Borisovich Kolmanovskii and Leonid Efimovich Shaïkhet. Control of systems with aftereffect. Translations of mathematical monographs. American Mathematical Society, 1996. (Cited on pages 58 and 59.)

[Kolmanovskii 1999] V.B. Kolmanovskii and A.D. Myshkis. Applied theory of functional differential equations. Kluwer, 1999. (Cited on pages 55 and 60.)

[Krishna 1983] C. Mani Krishna and Kang G. Shin. *Performance Measures for Multiprocessor Controllers*. In Performance '83, Proceedings of the 9th International Symposium on Computer Performance Modelling, Measurement and Evaluation, pages 229–250, College Park, Maryland USA, 1983. (Cited on page 36.)

[Li 2010] Xiaoting Li, Jean-Luc Scharbarg and Christian Fraboul. *Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis*. In Proceedings of 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2010, pages 1–8, Bilbao, Spain, September 2010. IEEE. (Cited on page 4.)

[Lindgren 2000] Markus Lindgren, Hans Hansson and Henrik Thane. *Using Measurements to Derive the Worst-Case Execution Time*. In 7th International Workshop on Real-Time Computing and Applications Symposium (RTCSA 2000), 12-14 December 2000, Cheju Island, South Korea, pages 15–22. IEEE Computer Society, December 2000. (Cited on page 107.)

[Liu 1973] C. L. Liu and James W. Layland. *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*. J. ACM, vol. 20, no. 1, pages 46–61, January 1973. (Cited on pages 27 and 132.)

[Liu 2009] K. Liu and E. Fridman. *Stability analysis of networked control systems: a discontiuous Lyapunov functional approach*. In Joint 48^{th} IEEE Conference on Decision and Control and 28^{th} Chinese Control Conference, Shanghai, China, 2009. (Cited on pages 68 and 130.)

[Liu 2010] K. Liu, V. Suplin and E. Fridman. *Stability of linear systems with general sawtooth delay*. IMA J. of Math. Control and Information, vol. 27, no. 4, pages 419–436, 2010. (Cited on page 61.)

[Liu 2012] K. Liu and E. Fridman. *Wirtinger's Inequality and Lyapunov-Based Sampled-Data Stabilization*. Automatica, vol. 48, 2012. (Cited on page 68.)

[Lokuciejewski 2011] Paul Lokuciejewski and Peter Marwedel. Worst-case execution time aware compilation techniques for real-time systems. Springer, 2011. (Cited on pages 107, 109 and 110.)

[Louise 2002] S. Louise. *Calcul de majorants sûrs de temps d'exécution au pire pour des tâches d'application Temps-réel critique pour des sytèmes disposant de caches mémoire*. PhD thesis, Paris XI Orsay, 2002. (Cited on pages 2, 109, 120, 132 and 133.)

[Martí 2002] Pau Martí. *Analysis and Design of real-time scheduling in Control Systems*. PhD thesis, Universitat Politècnica de Catalunya, june 2002. (Cited on page 34.)

[McLean 1990] D. McLean. Automatic flight control systems. Prentice-Hall international series in systems and control engineering. Prentice Hall, 1990. (Cited on pages 72 and 73.)

[McRuer 1974] D.T. McRuer, I.L. Ashkenas and D. Graham. Aircraft dynamics and automatic control. Princeton University Press, 1974. (Cited on page 82.)

[Midonnet 2010] Serge Midonnet, Damien Masson and Rémi Lassalle. *Slack-Time Computation for Temporal Robustness in Embedded Systems*. IEEE embedded systems letters, vol. 2, no. 4, pages 119–122, December 2010. (Cited on page 29.)

[Millán 2009] P. Millán, L. Orihuela, C. Vivas and F.R. Rubio. *Improved delay-dependent stability criterion for uncertain networked control systems with induced time-varying delays*. In 1^{rst} IFAC Workshop on Distributed estimation and Control of Networked control Systems, Necsys'09, Venise, Italy, 2009. (Cited on pages 68 and 130.)

[Mirkin 2007] L. Mirkin. *Some remarks on the use of time-varying delay to model sample-and-hold circuits*. IEEE Trans. on Automatic Control, vol. 52, no. 6, pages 1009–1112, 2007. (Cited on page 60.)

[Moravec 1998] Hans Moravec. *When will computer hardware match the human brain*. Journal of Transhumanism, vol. 1, 1998. Received Dec. 1997. (Cited on page 1.)

[Mueller 2006] Frank Mueller. *Challenges for Cyber-Physical Systems: Security, Timing Analysis and Soft Error Protection*. In High-Confidence Software Platforms for Cyber-Physical Systems (HCSP-CPS) Workshop, Alexandria, Virginia, November-December 2006. (Cited on page 4.)

[Naghshtabrizi 2006] P. Naghshtabrizi, J.P. Hespanha and A.R. Teel. *On the robust stability and stabilization of sampled-data systems: A hybrid system approach*. In Proceedings of the 45^{th} IEEE Conference on Decision and Control, San Diego, CA, USA, 2006. (Cited on page 132.)

[Naghshtabrizi 2008] P. Naghshtabrizi, J.P. Hespanha and A.R. Teel. *Exponential stability of impulsive systems with application to uncertain sampled-data systems*. Systems and Control Letters, vol. 57, no. 5, pages 378–385, 2008. (Cited on pages 60 and 68.)

[Naghshtabrizi 2009] P. Naghshtabrizi, J.P. Hespanha and A.R. Teel. *Stability of Delay Impulsive Systems with Application to Networked Control Systems*. Trans. of the Inst. of Measurement and Control, Special Issue on Hybrid and Switched Systems, 2009. (Cited on pages 62, 65 and 68.)

[Nguyen 1979] L T Nguyen, M E Ogburn, W P Gilbert, K S Kibler, P W Brown and P L Deal. Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability, volume 12854. National Aeronautics and Space Administration, Langley Research Center, December 1979. (Not cited.)

[Niculescu 2001] S.I. Niculescu. Delay effects on stability: a robust control approach, volume 269 of *Lecture notes in control and information sciences*. Springer, 2001. (Cited on pages 54 and 55.)

[of Automotive Engineers (SAE) 1996a] Society of Automotive Engineers (SAE). *SAE ARP4754 Certification Considerations for Highly-Integrated Or Complex Aircraft Systems*, nov 1996. (Cited on pages 15 and 16.)

[of Automotive Engineers (SAE) 1996b] Society of Automotive Engineers (SAE). *SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, December 1996. (Cited on pages 15 and 17.)

[Oishi 2009] Y. Oishi and H. Fujioka. *Stability and Stabilization of Aperiodic Sampled-Data Control Systems: An Approach Using Robust Linear Matrix Inequalities*. In Joint 48^{th} IEEE Conference on Decision and Control and 28^{th} Chinese Control Conference, pages 8142 – 8147, December 16-18 2009. (Cited on page 60.)

[Parr 1999] Gordon R. Parr and R. Edwards. *Integrated modular avionics*. Air and Space Europe, vol. 1, no. 2, pages 72 – 75, 1999. (Cited on page 4.)

[Peet 2009] M.M. Peet, A. Papachristodoulou and S. Lall. *Positive Forms and Stability of Linear Time-Delay Systems*. SIAM Journal on Control and Optimization, vol. 47, no. 6, pages 3227–3258, 2009. (Cited on page 62.)

[Peters 2003] Mark Peters and Micheal A. Konyak. The engineering analysis and design of the aircraft dynamics model for the faa target generation facility. Federal Aviation Administration System Resources Cooporation. Seagull Tecnology, Inc., 2003. (Cited on pages 73 and 74.)

[Petters 2002] Stefan M. Petters. *How much Worst Case is Needed in WCET Estimation?* In 2nd International Workshop on Worst Case Execution Time Analysis 2002, Vienna, Austria, Jun 18 2002. Satellite Workshop of 14th Euromicro International Conference on Real-Time Systems (ECRTS2002). (Cited on pages 111, 112 and 116.)

[Petters 2007] Stefan M. Petters. *Execution-Time Profiles*. Technical report, NICTA, Sydney, Australia, Jan 2007. (Cited on page 144.)

[Philippe 2011] Goupil Philippe. *AIRBUS state of the art and practices on FDI and FTC in flight control system*. In 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, pages 524 – 539, Barcelona, Spain, July 2011. (Cited on pages 12, 13, 14, 15 and 16.)

[Puschner 2008] P. Puschner and M. Schoeberl. *On Composable System Timing, Task Timing, and WCET Analysis*. In 8th International Workshop on Worst-Case Execution Time (WCET) Analysis, Prague, Czech Republic, July 2008. (Cited on pages 23 and 125.)

[Ratsiambahotra 2010] Tahiry Ratsiambahotra. *Contribution à la simulation de processeur : conception d'un générateur de librairie de simulateurs fonctionnels*. Thèse de doctorat, Université de Toulouse, Toulouse, France, septembre 2010. (Not cited.)

[Reineke 2007] Jan Reineke, Daniel Grund, Christoph Berg and Reinhard Wilhelm. *Timing Predictability of Cache Replacement Policies*. Real-Time Systems, vol. 37, no. 2, pages 99–122, November 2007. (Cited on page 133.)

[Richard 2003] J Richard. *Time-delay systems: an overview of some recent advances and open problems*. Automatica, vol. 39, no. 10, pages 1667–1694, 2003. (Cited on pages 54 and 55.)

[Robert 2010] D. Robert, O. Sename and D. Simon. *An H_∞/LPV design for sampling varying controllers : experimentation with a T inverted pendulum*. IEEE Trans. on Control Systems Technology, vol. 18, no. 3, pages 741–749, May 2010. (Cited on page 38.)

[Rochange 2007] Ch. Rochange, editeur. 7th intl. workshop on worst-case execution time (wcet) analysis, Pisa, Italy, jul 2007. (Cited on pages 23 and 29.)

[RTCA-EUROCEAE 1992] RTCA-EUROCEAE. *DO-178B/ED-12B - Software considerations in airborne systems and equipment certification considerations sur le logiciel en vue de la certification des systemes et equipements de bord*, 1992. (Cited on pages 15 and 16.)

[Schoeberl 2006] Martin Schoeberl and Rasmus Pedersen. *WCET analysis for a Java processor*. In Proceedings of the 4th international workshop on Java technologies for real-time and embedded systems, JTRES '06, pages 202–211, New York, NY, USA, 2006. ACM. (Cited on page 107.)

[Seto 1996] D. Seto, J.P. Lehoczky, L. Sha and K.G. Shin. *On task schedulability in real-time control systems*. In Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS '96), pages 13 –21, Washington DC, USA, December 1996. IEEE Computer Society. (Cited on pages 36 and 38.)

[Seuret 2006] Alexandre Seuret. *Commande et observation des systèmes à retards variables : théorie et applications*. PhD thesis, École centrale de Lille, 2006. (Cited on pages 60 and 132.)

[Seuret 2009] A. Seuret. *Stability analysis for sampled-data systems with a time-varying period*. In Joint 48^{th} IEEE Conference on Decision and Control and 28^{th} Chinese Control Conference, pages 8130 – 8135, December 2009. (Cited on page 61.)

[Seuret 2011a] Alexandre Seuret. *Stability analysis of networked control systems with asynchronous sampling and input delay*. In Proceedings of the 2011 American Control Conference - ACC 2011, page 6, San Francisco, Californie, États-Unis, June 2011. (Cited on pages 46, 62, 63, 136 and 151.)

[Seuret 2011b] Alexandre Seuret and Matthew M. Peet. *SOS for sampled-data systems*. In 18th IFAC World Congress, page 6, Milan, Italie, August 2011. (Cited on page 63.)

[Seuret 2012] Alexandre Seuret. *A novel stability analysis of linear systems under asynchronous samplings*. Automatica, pages 177–182, January 2012. (Cited on page 61.)

[Sha 2004] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky and A. K. Mok. *Real-Time Scheduling Theory: A Historical Perspective*. Real-Time Systems, vol. 28, no. 2–3, pages 101–155, November 2004. (Cited on pages 23 and 125.)

[Shaw 1989] Alan Shaw. *Reasoning about Time in Higher-Level Language Software*. IEEE Transactions on Software Engineering, vol. 15, pages 875–889, 1989. (Cited on page 110.)

[Shin 1985] K.-G. Shin, C.M. Krishna and Y.-H. Lee. *A unified Method for Evaluating realtime computer controllers and its application*. IEEE Trans. on Automatic Control, vol. 30, no. 4, april 1985. (Cited on page 31.)

[Shin 1992] K.-G. Shin and H. Kim. *Derivation and Application of Hard Deadlines for Realtime Control Systems*. IEEE Trans. on Systems, Man and Cybernetics, vol. 22, no. 6, november/december 1992. (Cited on pages 31 and 32.)

[Shin 1993] K.G. Shin and H. Kim. *Hard deadlines in real-time control systems*. Control Engineering Practice, vol. 1, no. 4, pages 623 – 628, 1993. (Cited on page 36.)

[Shin 1995] Kang G. Shin and Xianzhong Cui. *Computing Time Delay and Its Effects on Real-Time Control Systems*. IEEE Trans. on Control Systems Technology, vol. 3, pages 218–224, June 1995. (Cited on pages 32 and 45.)

[Shin 1999] K.G. Shin and C.L. Meissner. *Adaptation and Graceful Degradation of Control System Performance by Task Reallocation and Period Adjustment*. In 11th Euromicro Conference on Real-Time Systems, York, England, 1999. (Cited on pages 32 and 37.)

[Silva 1993] João Gabriel Silva, Mario Rela and Antonio Magalhães. *Are deadlines that hard?* In Fifth European Workshop on Dependable Computing, Dependability, Decentralization and Distribution, Lisboa, Portugal, 1993. (Cited on page 31.)

[Simon 2006a] Daniel Simon, David Robert and Olivier Sename. *Control and Real-time Scheduling Co-design : Application to Robust Robot Control*. In 3rd Taiwanese-French Conference on Information Technology TFIT'06, Nancy, France, 2006. (Cited on page 133.)

[Simon 2006b] Daniel Simon, Olivier Sename and David Robert. *Conception conjointe commande/ordonnancement et ordonnancement régulé*. In Nicolas Navet, editeur, Systèmes temps-réel, volume tome 2: Ordonnancement, Réseaux, Qualité de Service of *IC2*. Hermes, 2006. (Cited on pages 137 and 151.)

[Simon 2009] D. Simon, A. Seuret, P. Hokayem, J. Lygeros and E. Camacho. *State of the art in control/computing co-design*. Technical Report Public Deliverable D04-01, FeedNetBack IST FP7 project, 2009. (Cited on pages 34, 36 and 45.)

[Sipahi 2011] R. Sipahi, S.-I. Niculescu, C.T. Abdallah, W. Michiels and K. Gu. *Stability and Stabilization of Systems with Time Delay*. Control Systems, IEEE, vol. 31, no. 1, pages 38 – 65, 2011. (Cited on pages 54 and 56.)

[Souyris 2005] Jean Souyris, Erwan Le Pavec, Guillaume Himbert, Victor Jégu and Guillaume Borios. *Computing the worst case execution time of an avionics program by abstract interpretation*. In 5th Intl. Workshop on Worst-Case Execution Time (WCET2005) Analysis, pages 21–24, Palma de Mallorca, Balearic Islands, Spain, July 2005. (Cited on pages 10, 29 and 38.)

[Stankovic 1988] John A. Stankovic. *Real-Time Computing System: The Next Generation*. Technical report, University of Massachusetts, Amherst, MA, USA, 1988. (Not cited.)

[Stengel 2004] R.F. Stengel. Flight dynamics. Princeton University Press, 2004. (Cited on pages 72, 74, 76, 80, 81 and 83.)

[Stevens 2003] B.L. Stevens and F.L. Lewis. Aircraft control and simulation. Wiley-Interscience, October 2003. (Cited on pages 72, 76, 79, 81, 83, 85, 89, 142, 143 and 149.)

[Suh 2008] Y.S. Suh. *Stability and stabilization of nonuniform sampling systems*. Automatica, vol. 44, no. 12, pages 3222–3226, 2008. (Cited on page 60.)

[Thesing 2004] Stephan Thesing. *Safe and Precise WCET Determination by Abstract Interpretation of Pipeline Models*. PhD thesis, Saarland University, jul 2004. (Cited on pages 2, 109, 110, 120, 127, 132 and 133.)

[Traverse 2004] Pascal Traverse, Isabelle Lacaze and Jean Souyris. *Airbus fly-by-wire - A total approach to dependability*. In Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, pages 191–212, Toulouse, France, August 2004. (Cited on page 18.)

[Törngren 1998] Martin Törngren. *Fundamentals of implementing Real-Time Control Applications in Distributed Computer Systems*. J. of Real-Time Systems, vol. 14, pages 219–250, 1998. (Cited on pages 2 and 19.)

[Wegener 2001] Joachim Wegener and Frank Mueller. *A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints*. In IEEE Real Time Technology and Applications Symposium, volume 21, pages 144–154, Norwell, MA, USA, November 2001. IEEE. (Cited on page 106.)

[Wilhelm 2008a] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat and Per Stenström. *The worst-case execution-time problem: Overview of methods and survey of tools*. ACM Transaction on Embedded Computing Systems, vol. 7, pages 36:1–36:53, May 2008. (Cited on page 108.)

[Wilhelm 2008b] Reinhard Wilhelm and Björn Wachter. *Abstract Interpretation with Applications to Timing Validation*. In Aarti Gupta and Sharad Malik, editeurs, Computer Aided Verification, volume 5123 of *Lecture Notes in Computer Science*, pages 22–36. Springer Berlin Heidelberg, 2008. (Cited on page 107.)

[Wittenmark 2003] Björn Wittenmark, Karl Johan Åström and Karl-Erik Årzén. *Computer Control: An Overview*. Technical report, Lund Institute of Technology, Sweden, 2003. (Cited on pages 26, 30, 33, 119, 120 and 123.)

[Xia 2006] F. Xia and Y. Sun. *Control-Scheduling Codesign: A Perspective on Integrating Control and Computing*. Dynamics of Continuous, Discrete and Impulsive Systems - Series B, vol. Special Issue on ICSCA 06, pages 1352–1358, 2006. (Cited on pages 125, 133 and 151.)

[Zampieri 2008] S. Zampieri. *A survey of recent results in Networked Control Systems*. In Proc. of the 17^{th} IFAC World Congress, pages 2886–2894, Seoul, Korea, July 2008. (Cited on page 53.)

[Zhang 2001a] W. Zhang and MS. Branicky. *Stability of networked control systems with time-varying transmission period*. In Proceedings Of The Annual Allerton Conference On Communication Control And Computing, volume 21, pages 1205–1214, Monticello, Illinois, October 2001. (Cited on page 59.)

[Zhang 2001b] W. Zhang, M.S. Branicky and S.M. Phillips. *Stability of Networked Control Systems*. IEEE Control Systems Magazine, no. 21, pages 84–99, 2001. (Cited on page 59.)

UNIVERSITÉ DE GRENOBLE

Robust control under slackened real-time constraints

Abstract:

The development process of critical avionics products are done under strict safety regulations. These regulations include determinism and predictability of the systems' timing. The overall approach is based on a separation of concerns between control design and implementation. One of the toughest challenges in the current approach is the determination of the WCET, in order to correctly size the system. In this thesis, a weakened implementation scheme for real-time feedback controllers is proposed to reduce the conservatism due to traditional worst-case considerations, while preserving the stability and control performance. The methodology is tested to the pitch control of an aircraft model, showing that weakening the real-time constraints allows for saving computing power while preserving the system's stability and quality of control.

Keywords:

Control systems, avionics, architecture, worst case execution time, time-delay system, fault tolerance, real-time contraints

Commande robuste avec relâchement des contraintes temps-réel

Abstract:

Le processus de développement des systèmes avioniques suit des réglementations de sûreté de fonctionnement très strictes, incluant l'analyse du déterminisme et de la prédictibilité temporelle des systèmes. L'approche est basée sur la séparation des étapes de conception et d'implémentation. Une des plus grandes difficultés dans l'approche actuelle se trouve dans la détermination du WCET, qui est nécessaire pour prouver la satisfaction des contraintes de temps-réel dur du système. Dans cette thèse, une méthodologie de relâchement de contraintes temps-réels pour les systèmes de commandes digital est proposé. L'objectif est de réduire le conservatisme des approches traditionnelles basés sur le pire temps d'exécution, tout en préservant la stabilité et les performances de commandes. L'approche a été appliqué au système de commande de tangage d'un avion, ce qui a permi de montrer que le relâchement des contraintes temps réels améliore l'utilisation de la puissance de calcul disponible tout en préservant la stabilité et la qualité de commande du système.

Mots-clés:

Systèmes de commandes, systèmes avioniques, architecture, pire temps d'éxécution, systèmes à retards, tolerance aux fautes, temps-réels