An Imprecise Computation Framework for Fault Tolerant Control Design

ADDISON RIOS-BOLIVAR[§] LUIS PARRAGUEZ[‡] FRANCISCO HIDROBO[§]
MARGARITA HERAOUI[‡] JULIMA ANATO[‡] FRANCKLIN RIVAS[§]

[§]Universidad de Los Andes
Facultad de Ingeniería
La Hechicera, Mérida 5101
VENEZUELA
ilich,hidrobo,rivas@ula.ve

[‡]Universidad de Oriente
Departamento de Ingeniería Eléctrica
Núcleo Anzoátegui, Puerto La Cruz
VENEZUELA
parraguezl,anato,heraouim@cantv.net

Abstract: In this work the existing relations between the imprecise computation and the fault tolerant control (FTC) are analyzed. From those relations, there are constructed FTC systems according to the model of imprecise computation. The found relations establish that the obligatory tasks in the model of imprecise computation correspond with the mission of maintaining the stability of the systems under FTC according to the redundancy degree that assure the structural properties of the systems. On the other hand, the optional tasks in imprecise computation correspond to performance criteria of the systems under FTC are satisfied, which can be degraded under adverse conditions of operation of the system. These correspondences are probed in an example.

Key–Words: Imprecise Computation, Fault Tolerant Control, Real-Time, Fault Detection, Anytime Computation.

1 Introduction

The advances in the system theory and in the control mechanisms design, from the technologies of information and communication (TIC), have allowed that the industrial complexes satisfy high levels of productivity, which are quantified by performance indices. In spite of these advances, the complex systems do not totally fulfill the objectives for which they were designed, reason why bad operation situations can be generated whose effects can be very severe: losses of production, environmental damages, losses of human lives, etc. Therefore, jointly with the falsified and complex systems that allow to reach excellent production indices, specialized systems are required in order to guarantee rigorous objectives of security, reliability and availability.

The conjunction of the mechanisms for obtaining performance indices for security and reliability conforming the fault tolerant systems, which leave from the identification of the root cause of the anomalous operations.

Definition 1.1 *For a controlled system, the Fault Tolerance (FT) is defined as the ability of the system to maintain the control objectives, in spite of the occurrence of a failure, thus, under anomalous operation can be accepted a degradation of performance index.*

An important system for identifying the problems of operation in real time and with the required speed, corresponds to the filters for fault detection and isolation (FDI). These filters allow to find the root cause by means of the identification of the components or functional blocks of the system that do not operate of nominal way. The information that is generated from the FDI filters allows, besides recognizing the functional devices or blocks that operate outside their ratings, to establish an operation level satisfying performance objectives, in spite of the presence of failures.

The Fault Tolerance (FT) has as goal designing a control system with a determined structure in order to guarantee the *stability* and a satisfactory *performance*, not only when components are under nominal operation, but in case that the components (sensors, actuators, or process ele-

ments), fail [5]. Thus, the FT is defined in reference to the system objectives (stability, performance) and the given faults. The designed control systems under this conception is known as *fault tolerant control system*.

A FTC includes fault diagnosis in real time and decisions taking, in order to avoid or to lessen the adverse consequences of those failures, from the reconfiguration or intelligent change of the control systems. Thus, the FTC problem consists in to design, under a given degree of redundancy, intelligent control systems such that integrity of the system is guaranteed and certain performance indices are satisfied with performance [4].

For the analysis of the FT, it is necessary to define when, for a given system and under a failure situation, is still possible to reach the main targets of operation of the controlled system. The FT synthesis tries to provide to the system a hardware architecture and the software mechanisms that would allow, as far as possible, to reach certain performance objectives not only under normal operation, but also in adverse situations [17].

Under those premises, the FT is defined with reference to:

1. One or several objectives of the system (Structural Properties).

2. One or several given faults.

In Control Engineering traditionally two types of objectives referred to their structural properties are considered. Each objective is associate with the system control the estimation of its variables in real time, from measurements. These properties are the controllability and the observability [4].

The controllability is referred as the ability system states or a functional of the states, to be controlled by the inputs. The observability talks about to the ability of the system states of the systems or a functional of the states, of being estimated from outputs.

Thus, it is easy to infer that in the design of a FTC system it is necessary to guarantee that in normal operation the control guarantees the stability and the performance objectives of the system, and that under the fault conditions, for which it was designed, guarantees of *obligatory* way the stability and of *optional* way the best performance than can be obtained.

On the other hand, for the implantation of FTC systems some computer applications are required, which must operate in *real time*. It constitutes a challenge in the context of the TICs [15].

In a *Real Time System* (RTS) is not only important the logic validity of the answers which provides, but this must be generated before a determined temporary period ends [18]. Obtaining an answer out of the right time can be more damaging than obtaining an imprecise answer but within the period [2, 10, 3]. Then we clearly highlight that a RTS is a computational system in which the tasks, or at least a part of them, have a maximum temporary delivery period known as delivery period (*deadline*).

Similarly, the term *task* refers a computational work unit that must be planned and executed for the system. A *task* can be a computation of a control action, the transmission of a message, the execution of a command, the recovery of a file, etc. As a result of its execution each task gives some kind of information or service. The breach of a *delivery period*, in a critical system, is considered a failure and, therefore, ends in a unacceptable condition [9].

In many situations is preferable a low quality answer (approximated), but in time, than a high quality result (precise), but late. For Example, for a system to avoid collisions is much better to push in time a message of warning, along with an estimated ubication of the conflict of traffic, than to specify an exact evasive action, but late. Another example is the voice transmissions; can result acceptable to hear a low quality message, but long silences might be intolerable. Likewise, a not optimum control action in any sense, under emergency conditions, can keep annotated, instead of looking for computations precise when the process becomes unstable.

In this sense, some relations between the FTC and the objectives that are considered in the model of Imprecise Computation with the purpose of to orient a mechanism based on the fulfillment of obligatory tasks (stability condition of controlled systems), and optional tasks (certain performance indices), can be established in order to design fault tolerant control systems [17].

2 Imprecise Computation

The Model of Imprecise Computation sets out to solve situations of transient computer overloads and to reinforce the Fault Tolerance in Systems of Time Real Critics (*Hard-RTS*), essentially with restrictions of time [11]. To appreciate the basic idea of the technique of *Imprecise Computation* is highlighted the fact that, often, the global

behavior of a system may be tolerable, even in presence of temporary failures, if the more important tasks are completed on time. Therefore, instead of making the operating system handle all the tasks in the same way, the programmer may identify some as *obligatory*, meaning with this that they must be completed in their respective ranges of time, and other tasks as *optionals*, indicating that this last can be skipped without causing any failure considered as intolerable. In that order, the operating system, in overload conditions, can skip the less important tasks, trying to execute on time the important ones.

From the point of view of the Real Time community, the *Imprecise Computation* turns out to be, in principle, a generalization of *Anytime Computing* [21], as it is known in the field of Artificial Intelligence. The *Anytime Algorithms* are those where the quality of the results is improve gradually as long as the time of execution increases. Hence there is a compromise between the quality of computation and the consumption of resources.

We emphasize clearly that a RTS is a computational system in which the tasks, or at least one part of them, has a temporary term of delivery maximum well-known as *deadline*. The breach of a *deadline*, in a critical system, is considered as failure, which is an unacceptable condition [9].

In that order of ideas, there exist situations where a low answer in quality is preferable (approximated), but in time, that a result of great quality (precise), but inopportunely, as it is the particular case of the controlled processes [2]. In industrial processes controlled by computers during the initialization of the system or in an operation of emergency, occur transients increases at the computational load, that degrade the quality, accuracy and opportunity for the controller of the system to respond. To face this situation there have been developed different techniques, since those that cover the over-design of the computational capacity until the reduction of the accuracy of the computational answer and, therefore, of the time required to obtain it, providing an acceptable degradation of the answer.

Thus, the technique of Imprecise Computation is based on the observations before mentioned, as well as of the fact that a good approximate result can, often, to obtain by far less computer resources, especially time of processor. A system of imprecise computation allows choosing the computational accuracy of each task, in order to reach to the purposes of the handler of the system with a transient overload of the computation, annotating the time of the services to keep the operation in the conditions imposed for the planning that has been established. When the load is normal, the objective is to provide some balance preestablished between accuracy an time of respond. If the load is high the objective of the system is to keep the time of respond annotated through the reduction of the computational accuracy.

The characteristic of the variable accuracy of an imprecise computation system is implemented through the structuration of the tasks, so that each task has two parts: The first is mandatory and the other is optional. The computation of the mandatory part must keep an approximated result, and the computation of the optional part must refine this result. An approximated result is considered acceptable if the answer of the controlled system is bounded facing a failure or an overload.

Within the RTS are the Critical Real Time systems, in which, the lost of a period can cause a catastrophic and unrecoverable failure. In the control systems, the lost of a period, we should say, the sending of the control action out of the right time, can cause the reach of the process to an unstable state. This situation can be considered as a critical failure.

Therefore, an application of real time control is structured as a set of task that interact with the environment getting data from the sensors and sending control actions to the actuators of the process. As it has been said previously, it is not enough that the actions of the system are correct logically, but, moreover, they must be produced within a determined interval of time. This is because the system is connected to an external process that receives stimulus, which it must respond with enough quickness to avoid the evolution to an undesirable state. This external stimulus can appear in moments of time not predictable. Therefore, the design of the system can get complicated, because the implementation is not as simple as decompose the system in many periodic tasks as control loops, executing a simple algorithm as a PID [2].

In a RTS, a *transients overloading* is the inability to plan and execute computational tasks before they their deadline be reached. Denotes the state of a system where the requirements of services exceed the availability of resources during a limited time, causing the lost of the deadline; it means, the answer will be available after the limit set as peremptory. This phenomenon

can occur during the initialization of the system, emergencies or in presence of failures at the environment that reduce the availability of computational resources [19]. Thus, it is necessary the *Imprecise Computation*.

2.1 Basic model of Imprecise Computation

The *workload* used to characterize the Imprecise Computation is based on the classical deterministic models of the real time application that define an application as a T set of not expulsive tasks.

$$T = \{T_1, T_2, ..., T_n\} \quad (1)$$

The tasks can present, as it was specified before, data/control dependencies that impose precedence restrictions on the order of its execution that is denoted by the operator "$<$".

Definition 2.1 *A task T_i is* predecessor *of another task T_j, and T_j a* successor *of T_i, defined by*

$$T_i < T_j \quad (2)$$

if T_j can not start the execution until T_i has ended.

Definition 2.2 *A task T_i is* immediately predecessor *of another task T_j, and T_j an* immediate successor *of T_i, if $T_i < T_j$ and there is not another task T_k such that $T_i < T_k < T_j$.*

Definition 2.3 *Two tasks T_i and T_j are* independents *when there is no precedence relationship $T_i < T_j$ or $T_j < T_i$) and they can be executed in any order.*

Definition 2.4 *One says that a task is monotonous if the quality of its intermediate results does not fall in the measurement that is executed in the time.*

In the basic model of imprecise computation, each task T_i in the system is defined by the following parameters:

1. *Arrival Time (ready time)* r_i is the time instant in which the task T_i is ready to start its execution.
2. *Delivery Period (deadline)* d_i: maximum time instant in which T_j must be finished and deliver a result.
3. *Processing time* τ_i: is the required time, in the worst case, to execute the task completely.
4. *Assigned Time of Processor* σ_i: is the time of processor that assigns the planner to the execution of a task.
5. *Relative weight* ω_i is a positive rational number that indicates the relative importance of the task.

Each task T_i is decomposed in two tasks, also called sub-tasks or parts (see Figure 1):

1. Mandatory Part M_i.
2. Optional Part O_i.

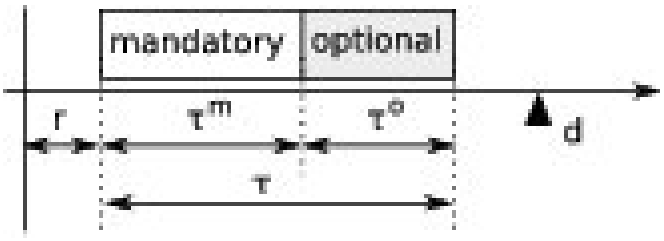

Figure 1: Imprecise task.

The *arrival time* and the *delivery period* of the tasks M_i and O_i are the same for T_i. M_i is an immediate predecessor of O_i. If it is denoted by τ_i^m the time of the mandatory task and by τ_i^o the time of the optional part of the process, where τ_i^m and τ_i^o are rational numbers, then

$$\tau_i^m + \tau_i^o = \tau_i \quad (3)$$

A *planning* is an assignation of the processor for the task in T, in not associated intervals of time. A task is said planned, in a interval of time, if the processor is assigned to the task of that interval. The quantity of time of the processor assigned to the task is the sum of the not associated intervals of time (see Figure 2). In any valid planning, for systems with just a processor, the assigned time of the processor (σ_i), is assigned only to a task at the time, and every task is assigned after its arrival time, and all the predecessor tasks have been finished, and the quantity of the time for the processor assigned to the mandatory part is equal to τ_i^m and the completion of this must occur for the most at the delivery time d_i. The optional part of a *monotonous task* can be completed at any time instant $t \leq d_i$. From here,

the quantity of time assigned σ_i for the processor to a monotonous task must be in the interval

$$\tau_i^m \leq \sigma_i \leq \tau_i \quad (4)$$

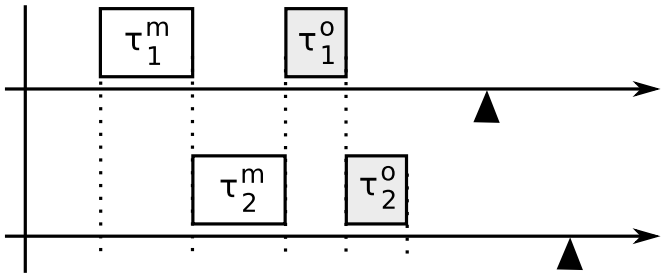

Figure 2: Imprecise Tasks.

The quantity of time for the processor assigned to a task with *0/1* restrictions can be τ_i or τ_i^m, according to if there will executed or not its optional part. When $\sigma_i = \tau_i$, we say the task is *precisely planned*; on the contrary, if the quantity of time for the processor assigned is less than the time for the processor required for the total computation task, $\sigma_i < \tau_i$, then we are talking about *imprecise planning*.

A task T_i is *completed* in a planning, if its mandatory part is finished in a traditional sense. A *valid planning* is a reachable planning if all the tasks are finished before its delivery time. The set of T tasks can be considered as able to be planned if, at least, it has a reachable planning. The global performance of the system under imprecise computation can be measured from the evaluation of errors, [1, 7, 6, 8, 20].

3 FTC and Imprecise Computation

Normal operation of any control system settles down by the fulfillment of certain relations between the different variables from the controlled process. These relations allow to define, with a high degree of precision, the behavior to future of the system, which can be verified from the measured data of the process. From the operational point of view, those same relations also allow the characterization of the performance of the system in a temporary space, being able to establish what it has been denominated like operation in real time. This means that the supervision and monitoring of a productive process can be realized based on the answers that it generates within established time intervals, according to the relations between the process variables [15].

On the basis of the previous, from an performance model, the anomalous and normal operation regions can be characterized and taking into account the answers from the process in certain times previously established. Consequently, a region of normal operation is that one where all the relations between the variables of the process are satisfied within the established times. This means that the region of normal operation characterizes in that all the tasks (relations between variables) obligatory are executed according to its times and margins of time for optional tasks are allowed, in which those factors are included that can disturb the operation of the process (exogenous signals and/or uncertainties), without reducing their global performance based on the operational objectives, (see Figure 3).

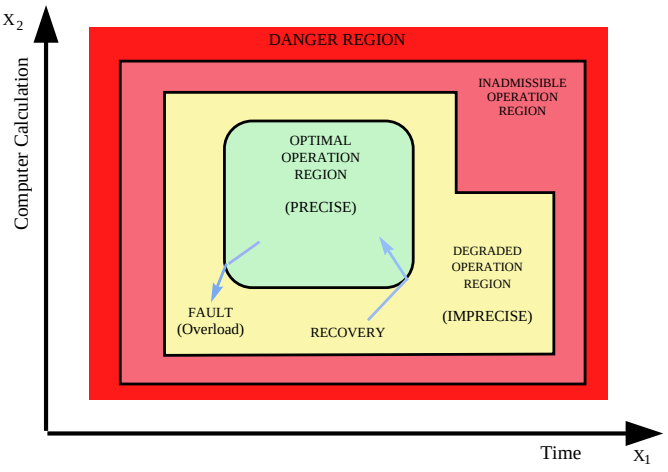

Figure 3: Process Operation Regions.

At the time of occurrence of an anomaly, so that the relations between variables are not satisfied, then the process operates in an imprecision region with respect to satisfy, in time, the execution of optional tasks. It is at this moment that is necessary to take some actions to diagnose the magnitude and consequences of the anomaly, in order to recover the normal operation of the processes. In this case, the actions are oriented of way to guarantee the execution of the obligatory tasks, still under the circumstances of operation under failure, according to the magnitude of the anomaly and in agreement with the system robustness. If the magnitude and consequences of the failures are such that are not possible to guarantee the obligatory tasks, it enters an inadmissible region that, generally, brings like result the

stopping of the execution of the tasks. Thus, from the point of view of the FTC, it is necessary to evaluate the operation of the processes when the imprecise tasks are admitted, and there exists the possibility of the recovery towards the region of precise operation. This demand to analyze the propagation of the failure and the level of redundancy that would allow to maintain, on the one hand, the certain possibility of the execution of the obligatory tasks and a certain temporary degree for the execution of optional tasks.

The impact of the failure is evaluated from a fault model. At the same time, the diagnosis and the minimization of their effects must be established. Thus, the FTC based on imprecise computation would allow the construction of mechanisms, based on obligatory and optional tasks, that they still assure the availability and the performance the systems in adverse conditions.

From the analytical redundancy, the fault diagnosis is based on the availability of a fault model. That model allows, on the one hand, to establish operating characteristics, that are the same that is used for the design of the mechanisms of detection by means of the residual analysis. On the other hand, the model is used for, quantitatively and qualitatively, to define levels of severity and impact, from a propagation analysis [16].

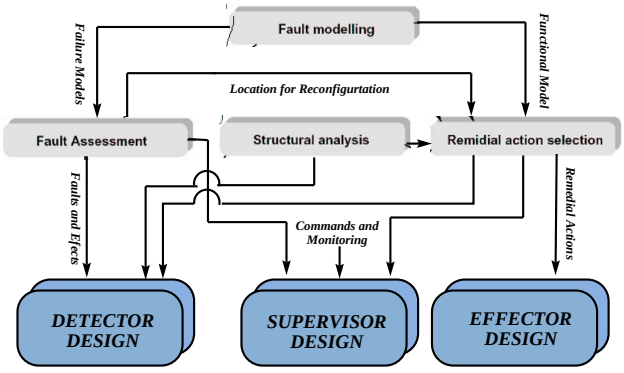

Figure 4: Modelo de falla e impacto.

At the same time, the model, jointly with its valuation and from a structural analysis, allows to the definition and construction of the mechanism of detection and diagnosis. The design of the fault detector is obtained according to the techniques and tools established in previous sections. Thus, the model and the fault propagation also will govern the design of the actions to take in situations of anomalous operation, such that as the supervision supports of the fiability system centered in the fault diagnosis. One of the fundamental tasks of the supervision, in the context of the FTC, is to discern on the mechanisms of reconfiguration of the control action.

On the base of the imprecise computation, the fault model is defined by the structuring and temporary planning of the tasks. That is to say, the model is associated to the execution or not of the obligatory and optional tasks according to the assigned times. Everything what is outside those time intervals constitutes as failure, whose effects will have to be mitigated with the actions be executed.

3.1 Passive FTC

For FTC based on imprecise computation, some conditions of passive and active FTC possible can be established, such as in other techniques. Thus, *the passive FTC is defined as the robust computation, that is, a computational system is designed (control system), which under certain conditions of failures, executes the obligatory tasks that are coordinated for to guarantee the stability of the controlled system. In the measurement of the possibilities, the totality or some optional tasks will be executed, which will guarantee a degree of acceptable performance, although degraded in a certain quantitative and/or qualitative level.*

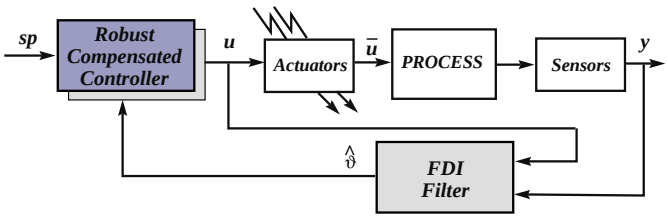

Figure 5: Passive FTC.

Such as is showed in Figure 5, the design of a robust compensated control system is required, which must satisfy multi-objectives according to the robust stability and performance conditions [14]. It is evident that the greater commitment for the design of the robust control system is based on the capacity of the planning of the tasks that, as far as possible, must be constructed for a maximum utility, according to the metric criteria associated to the performance indices, (to see the Section 2.1). This means that of the planned tasks, besides the obligatory ones, those optional ones

must be executed, in order to guarantee the performance index, which is constructed on the base of a criterion of optimization of maximum utility.

3.2 Active FTC

In the context of the Active FTC under imprecise computation, *a computational system must be to designed, under a given degree of redundancy, such that the system integrity is guaranteed, which means that the obligatory tasks must be executed, and the performance is satisfactory, still degrading quantitatively and/or qualitatively the optional tasks, through the planning involving the reconfiguration or accommodation.*

Again, the design consists in the construction of a detector system and a supervisor system, which operating of simultaneous way to diminish the failure effects, taking the corresponding actions. As is showed in the Figure 6, the detector and the supervisor correspond to the supervision-diagnosis block and the effector is related to the reconfiguration mechanisms. In this last one, the actions are taken, from the impact of the failure, of its propagation and the level of redundancy available. Thus, this mechanism allows to guarantee the integrity of the operation according to the obligatory part, and as optional tasks must be suppressed or be delayed, assuring a certain performance level.

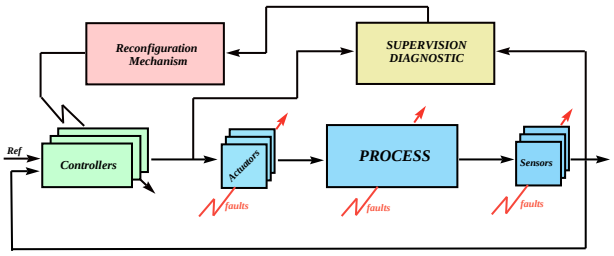

Figure 6: Active FTC.

Needing the tasks and actions within the operational framework in real time, the Figure 7 allows to distinguish as it is the handling of signals between the different functional blocks. There exist regular operations and possible operations. These last ones are carried out under the appearance of some detected anomaly.

As the tasks in time real are high-priority, the calculation and application of the control action that allows the stability of the systems will correspond to an obligatory action, since operational integrity depends on such task. All those other

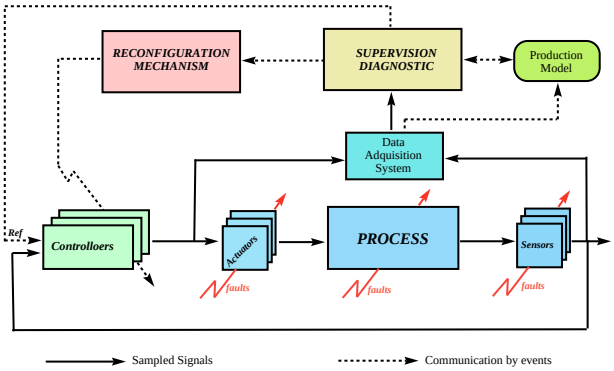

Figure 7: Active FTC in Real Time.

control actions referring to maintain a performance level will be optional tasks, in this vision of the FTC. According to the magnitude of the failure, the remedial action could simply be the change of the reference (see Figure 7), which is related to the accommodation. The action could also be the reconfiguration or reconstruction of the controller, which must guarantee the functional integrity.

4 Example

Lets consider the system

$$\begin{aligned} \dot{x}(t) &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} -1 \\ 0 \end{pmatrix} \nu(t) \\ y(t) &= \begin{pmatrix} 1 & 1 \end{pmatrix} x(t); \end{aligned} \tag{5}$$

where the fault mode $\nu(t)$ has the same form that the control signal $u(t)$, that is, $\nu(t) = u(t)$ unless the fault appears in a unknown time. This characterization describes actuator faults, where, at the time in that the anomaly is presented, the structure of the system changes, conserving structural properties such as the controllability of this system. This condition establishes an analytical redundancy in the actuator.

In order to control the system without fault, a PID controller has been designed, with $K_p = 2$, $K_I = 10$ and $K_D = 0$. This type of control requires the calculation of a proportional action and an integral action, which allow to assure the stability of closed loop (obligatory task), and some conditions of performance (optional tasks). In the absence of fault, the system responses are showed in the Figure 8.

For the fault diagnosis a filter based on state observers has been designed, [13]. The filter dy-

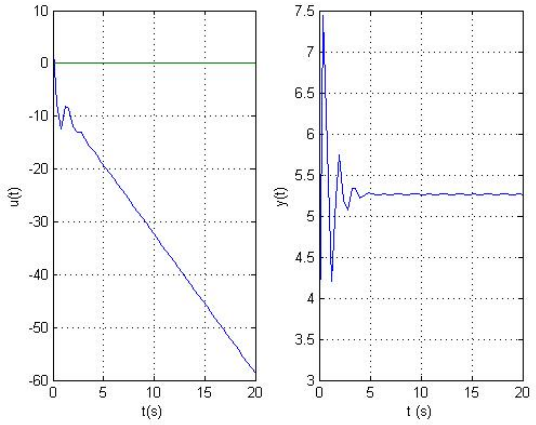

Figure 8: Controlled system signals without faults.

namic is given by

$$\begin{aligned}\dot{\hat{x}}(t) &= \begin{pmatrix}0 & 1\\ 1 & 2\end{pmatrix}\hat{x}(t) + \begin{pmatrix}1\\ 1\end{pmatrix}u(t) + \begin{pmatrix}1\\ 4\end{pmatrix}(y(t)-\hat{y}(t))\\ \hat{y}(t) &= \begin{pmatrix}1 & 1\end{pmatrix}\hat{x}(t);\end{aligned} \quad (6)$$

which allows the fault diagnosis, because the dynamics of the estimation error is asymptotically stable.

For the simulation aims, the fault appears in $t = 10s$, from which the actuator changes since the control matrix will be, as a result of the anomaly $B = \begin{pmatrix}1\\ 0\end{pmatrix}$. With that structure, the PID controller, who before the fault has regulated satisfactorily the system, it cannot stabilize the system in closed loop, such as is showed in the Figure 9.

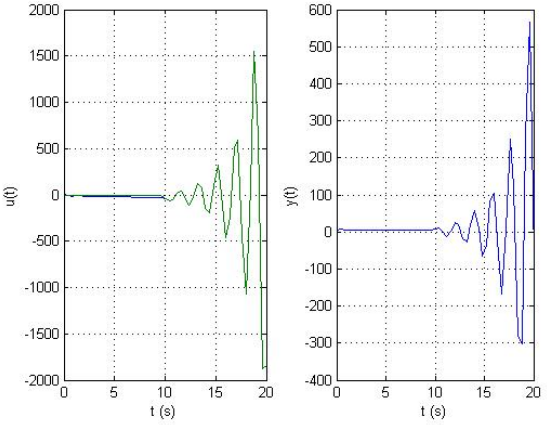

Figure 9: System signals without tolerant control.

Consequently, the fault diagnosis filter generates residues that will be used to diminish the failure effects allowing to take action in that sense.

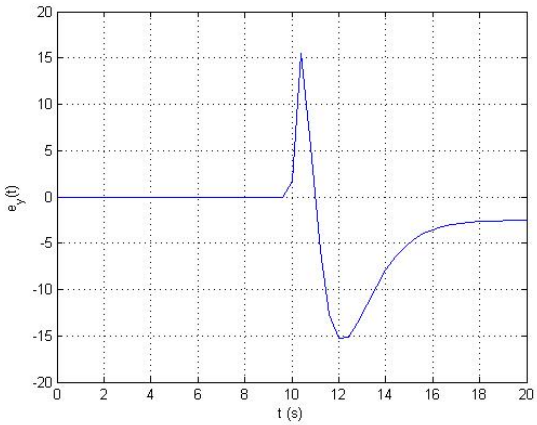

Figure 10: Residual signals.

On the contrary, a FTC has been designed, which allows to redefine the structure of the controller, in order to maintain the obligatory operation requirements

In the moment of the fault detection, the controller is switched to an action appropriate, in this case, that action is obtained by a controller by static output feedback $u(t) = K_y y(t)$, where $K_y = -10$ is the feedback gain, which allows to assure the obligatory task of stability in closed loop, executing itself as well, with a smaller requirement of calculations. The switching is realized through the evaluation of the residues (see Figure 10), which are generated by the fault detection filter.

The results of the FTC are shown in the Figure 11. In this way, the effectiveness of the method is demonstrated because the performance requirements are satisfied. The reconfiguration mechanism can be constructed from a Neuronal Network (NN), such as has appeared in [12].

5 Conclusions

In this work a relation analysis of the Fault Tolerant Control (FTC) and the imprecise computation has been presented. From the study of the fault tolerant control systems and of the imprecise computation some relations have been established, that allow to orient schemes for the construction of FTC systems according to the techniques of imprecise computation. These relations

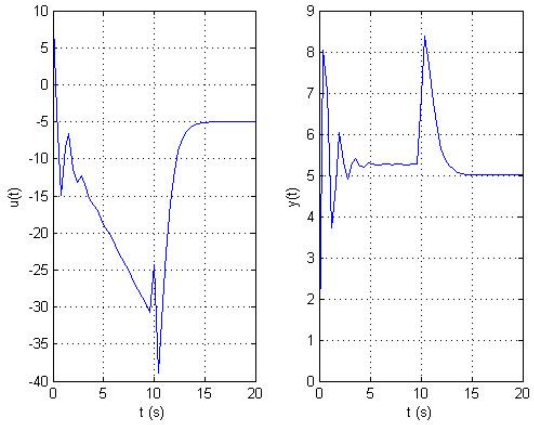

Figure 11: System signals with tolerant control.

characterize that the obligatory tasks in imprecise computation correspond to the tasks that guarantee the stability of the systems under FTC and according to the degree of redundancy, which is assured by the structural properties of the systems. Whereas the optional tasks correspond to satisfy performance criteria of the controlled systems, which can be degraded according to the operation of the systems under adverse conditions. The results have been validated from a numerical example.

Acknowledgements: The research was supported by the FONACIT under the grant **2005000170**, and by the CDCHT-ULA, through grant **I-1103-08-02-A**.

References:

[1] S. Aldarmi and A. Burns. Dynamic cpu scheduling with imprecise knowledge of computation -time. Technical Report YCS-314, Department of Computer Science, University of York, U.K., 1999.

[2] Patricia Balbastre. *Modelo de tareas para la integración del control y la planificación en sistemas de tiempo real*. PhD thesis, Universidad Politécnica de Valencia, 2002.

[3] Raktim Bhattacharya and Gary J. Balas. Anytime control algorithm: Model reduction approach. *Computer*, 1993.

[4] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and fault-tolerant control*. Springer-Verlag, Berlin, 2003.

[5] Mogens Blanke, Marcel Staroswiecki, and N.E. Wu. Concepts and methods in fault-tolerant control. In IEEE, editor, *Proceedings of the 2001 American Control Conference*, volume 4, pages 2606–2620, Arlington, VA, USA, 2001.

[6] Giorgio Buttazzo. Research trends in real-time computing for embedded systems. *ACM SIGBED Review*, 3(3), July 2006.

[7] Yanching Chu. On fixed priority preemptive scheduling for imprecise computation. Technical report, Real-time Systems Research Group, Department of Computer Science, University of York, UK, 2006.

[8] Peng Jun-jie, Zhu Xiao-gang, Duan Jun, and Xu Chang-qing. Scheduling imprecise computation tasks by maximizing minimum reward. *Journal of Communication and Computer*, 2(9), Sep. 2005.

[9] J. Liu, K. Lin, R. Bettati, D. Hull, and A. Yu. Use of imprecise computation to enhance dependability of real-time systems, 1994.

[10] Jane W.S. Liu, Kwei-Jay Lin, Wei-Kuan Shih, Albert Chuang shi Yu, Jen-Yao Chung, and Wei Zhao. Algorithms for scheduling imprecise computations. *Computer*, 24(5):58 – 68, May 1991.

[11] Jane W.S. Liu and Wei-Kuan Shih. Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. *IEEE Transactions on Computers*, 44(3):pp. 466–471, Mar 1995.

[12] Lisbeth Pérez, Francklin Rivas, Addison Ríos, and Gloria Mousalli. Faults tolerant control application using neural networks. In *Proc. 3rd WSEAS Int. Conf. on Dynamical Systems and Control (CONTROL '07)*, volume 1, pages CD–Rom, Arcachon, France, 2007.

[13] A. Ríos-Bolívar. *Sur la Synthèse de Filtres de Détection de Défaillances*. PhD thesis, Université Paul Sabatier, Toulouse, 2001.

[14] A. Ríos-Bolívar and S. Godoy. Multiobjective approaches for robust anti-windup controller synthesis in LTI systems. *WSEAS Transaction on Systems*, 5(1):164–171, 2006.

[15] A. Ríos-Bolívar, F. Hidrobo, M. Cerrada, and J. Aguilar. Control and supervision system development with intelligent agents. *WSEAS Transaction on Systems*, 6(1):141–148, 2007.

[16] A. Ríos-Bolívar and R. Márquez. Fault tolerance for robust anti-windup compensation implementation. *WSEAS Transaction on Systems*, 5(9):2197–2203, 2006.

[17] A. Ríos-Bolívar, L. Parraguez, F. Hidrobo, M. Heraoui, J. Anato, and F. Rivas. Fault tolerant control: An imprecise computation setting. In Prof. Shengyong Chen, editor, *Proc. of the 9th WSEAS International Conference on Robotics, Control and Manufacturing Technology*, volume 1 of *Electrical and Computer Engineering Series*, pages 44–49, Hangzhou, China, 2009. Published by WSEAS Press.

[18] John A. Stankovic. The integration of scheduling and fault tolerance in real-time systems. Technical Report 49, UM-CS, 1992.

[19] Marcus Thuresson. Imprecise computation - a real-time scheduling problem. Master's thesis, Department of Computer Science, University of Skovde, Sweden, 1999.

[20] José M. Urriza, Ricardo Cayssials, and Javier D. Orozco. Optimización on-line de sistemas de tiempo real con computación imprecisa basados en recompensas. Technical report, Universidad Nacional del Sur, Laboratorio de Sistemas Digitales, Bahía Blanca, Argentina, 2003.

[21] S. Zilberstein and S. Russell. *Imprecise and Approximate Computation*, chapter Approximate Reasoning Using Anytime Algorithms. Kluwer Academic Publishers, 1995.