Discrete Event Based Simulation and Control of Continuous Systems

Ernesto Kofman

Thesis presented in partial fullfilment of the requirements for the degree of

Doctor en Ingeniería

Director: Sergio Junco

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Abstract

This Thesis introduces the fundamentals and the theory of a new way to approximate differential equations applied to numerical integration and digital control.

Replacing the classic time discretization with state quantization –an approximation approach already available in the literature– two new numerical integration methods are introduced. Due to this kind of discretization technique, the resulting simulation models are discrete event systems instead of a discrete time as in all classic numerical methods. This fact yields many practical advantages like an efficient sparsity exploitation and an important computational cost reduction in hybrid system simulation.

From a theoretical point of view, it is shown that the methods can be analyzed as continuous systems with bounded perturbations and thus, stability, convergence and error bound properties are proven showing also some interesting advantages with respect to classic approaches.

The application of the new first order method to the discretization of continuous controllers with the addition of an asynchronous sampling scheme allow to define a new digital control methodology in which the time discretization is ideally avoided. As a result, this new technique improves considerably the dynamic response of digital control systems, reduces the quantization effects, the computational costs and the information traffic between plant and controller.

When it comes to theoretical properties, the new control scheme can ensure stability, convergence and error bound properties which can be applied to linear, non linear and time varying cases. Based on these properties, different design algorithms are also provided.

Acknowledgment

Every Thesis begins with a mention to the Director. In this case, my acknowledgement is not ony due to our work in the last four years. If I had not met Sergio, my work would not been involved with teaching and research (and now I would be probably working at the industry, with a good salary, instead).

Following with the academic side, an important part of what I learned during the last years (from technical and mathematical details to using LaTeX) is due to Marimar Seron. She also gave the impulse I needed to choose the subject of this Thesis.

I also want to thank Julio Braslavsky for his remarks and for the review of many topics which then were part of this work (mainly those topics related to control).

In that sense, I also owe thanks to Juan Carlos Gómez who, among many other things, helped me with a careful revision of more than one article supporting this Thesis.

Many of the ideas which form part of the results of this Thesis were conceived after talking with François Cellier. Besides that, his invitation to coauthor his book *Continuous System Simulation* was one of the most important impulses I received to continue with this work.

I also want to mention the very important help that I received from Bernie Zeigler. He not only revised and collaborated with the correction of my articles but he also opened the doors of a great part of the simulation community to make my work known.

Although I was only referring to the help I received in the academical sense, it would be unfair to forget the friendship and all what I received from these people in the human aspect.

Nothing of this work had been done without the support of my parents, Julia and Hugo. They gave me –and they still do– everything what can be expected regarding love, advise, help and –the most important thing– the freedom to choose and build my own way.

When I mention my family I must also thank Queca, my Grandmother. She, with her knishes and cookies, with her infinite dedication and with her example, is one of the lights which guide every day of my life.

I cannot forget mentioning the support of my brothers, Diego and Marco, and I must also thank the unconditional friendship of Monica (and Rami, of course). I do not want to skip a mention to my friends that always are with

me, leaving aside the geographic distance: Damián, Cabeza, Lottar, Dieguito, Martín, Diego, Gastón, Betty, Momia, Hernán.

Finally, my greatest thanks to Juliana, for sharing each moment of our lifes.

Contents

Chapter 1

Introduction

The resolution of most modern engineering problems could not be conceived without the help of simulation. Due to risk and cost reasons, the direct experimentation on real systems is leaving his place to the experimentation on simulation models (there are exceptions, of course). Nowadays, we can hardly find a design problem which can be carried without the help of a computer.

The increasing complexity of man–made systems and the need of more and more accurate and faster results stimulated the development of hundreds of new simulation techniques in the last 40 years.

Simultaneously, the appearence of modern computers and its amazing evolution gave the necessary tool which allows the easy and efficient implementation of the most complex simulation methods.

Due to these facts, computer simulation constitutes a discipline itself. As every other discipline, it is divided in several sub–disciplines which deal with different specific problems.

Engineering problems involve dealing with physical systems. Since most physical laws are described by differential equations, simulation in engineering is in fact related to numerical resolution of differential equations. This is also called *Continuous System Simulation*.

However, modern egineering systems usually include the presence of digital devices –digital controller for instance– whose description does not fit in the form of a differential equation. This fact adds more complexity to our subject and leads to the family of the *Hybrid Systems*.

In many applications, the simulations have to be performed in real–time. Typical examples are the so called *man–in–the–loop* systems (flight simulators for instance) and, in general, simulations which interact with systems of the real world.

Digital control systems can be considered as part of the last category since they have to interact with a real plant. Taking into account that plants are usually described by differential equations and consequently most controllers are designed to satisfy a continuous law, the digital implementation of continuous controllers can be seen as a problem of real–time simulation. Thus, it has to be

solved following some discretization techniques.

The goal of this Thesis is to develop a completely new family of numerical methods for ordinary differential equations –which can be also applied to hybrid and *differential algebraic equation* systems– and to use the same ideas in the implementation of digital controllers.

The main innovation in the numerical methods developed here is the avoidance of time discretization. All existing numerical methods for ordinary differential equations are based on the calculation of an approximate solution in some discrete instant of time. Here, we replace that discretization by the quantization of the state variables.

As a result, the simulation model becomes discrete event instead of discrete time on one hand. This fact produces, in some cases, an important reduction of the computational costs (specially in hybrid systems).

On the other hand, this new approximation yields strong theoretical properties related to stability, convergence, error bound, etc.

Both kind of advantages –practical and theoretical– are also verified in the application to digital control.

1.1 Outline of the Thesis

This Thesis is conceived as a self content work where each chapter is based in the previous ones although this does not always coincide with the chronological order in which the subjects were developed.

On one hand, it is assumed that all the basic concepts –i.e. the concepts which are usually learned at the undergraduate level in Control and Numerical Methods– are already known by the reader. On the other hand, other new subjects and tools are only introduced for its use in the Thesis.

In that way, the Thesis does not include neither a complete theory nor a state of the art description about DEVS[1], perturbation theory, numerical integration, or other concepts involved.

Several original theoretical results, including theorems and proofs, are included in the Thesis. In order to give the reader the possibility of skipping them, most are concentrated in a chapter.

When it comes to notation, the classic notations of control and DEVS theory were simulataneously used. Thus, many symbols are often redefined to be used in different contexts. In that way, the meaning of each symbol must be seen according to the last definition made.

Taking into account all these principles, the Thesis is organized as follows:

This first introductory chapter gives a description of the whole Thesis, not only by enummerating the results but also trying to relate them with the state of the art.

The second chapter introduces the seed of the main ideas in which the rest of the work is based on. There, the original concepts about quantization and

[1]The list of abbreviations used in this Thesis can be found in Appendix A.

DEVS are introduced starting from a motivating example in Section 2.1. Then, Sections 2.2 to 2.4 develop an *ad–hoc* theory of DEVS which is then used in the rest of the Thesis. After that, Section 2.5 shows the relationship between the first example and DEVS introducing the concept of *Quantized Systems* (QS), which was the first idea to approximate differential equations with DEVS models.

At that point, the state–of–the–art description is almost complete and then, Section 2.6 describes the main problem –called *illegitimacy*– shown by Quantized Systems. Discovering this problem and finding a solution were probably the most important motivations of this work.

The third chapter starts describing the solution to the illegitimacy problem, which consists in adding hysteresis to the quantization. Based on the use of hysteretic quantization functions –formally defined in Section 3.1– the *Quantized State Systems* (QSS) and the *QSS–method* are introduced in Section 3.2. This method is the first general discrete event numerical method for integration of Ordinary Differential Equations (ODEs).

Based on the study of the trajectory forms (Section 3.3), the DEVS model of the QSS is deduced in Section 3.4. This model also shows the practical way to implement the method. After introducing a simple simulation example –where some qualities as the sparsity exploitations become evident– Sections 3.5 and 3.6 explain some practical aspects about the use of the QSS–method. Then, the chapter finishes concluding about the need of a deeper theoretical analysis.

Chapter 4 is dedicated to the study of the main theoretical properties of the approximation. After explaining the relatioship between the QSS–method and the theory of perturbed systems, the convergence property is proven in the theorem included in Section 4.2. Further, the general stability properties are studied making use of a Lyapunov approach in Section 4.3. The main result here (Theorem 4.2) concludes that –under certain conditions– the ultimately boundedness of the approximate QSS solutions can be ensured.

Despite the importance and generality of that stability result, its use is quite complicated and conservative (mainly due to the presence of Lyapunov functions). Thus, the analysis is carried to the field of Linear Time Invariant (LTI) Systems where the properties of classic numerical method are usually studied. Here, in order to avoid conservative results, a new way to establish the bound of the perturbation effects is introduced (Section 4.5) which is then compared with the classic Lyapunov approach for LTI systems (previously introduced in Section 4.4). This comparison shows the convenience of th new approach.

Based on this new approach, not only the stability but also the global error bound properties of the QSS–method in LTI systems are shown in Section 4.6. These theoretical results are then applied to the choice of the appropriate quantization and hysteresis according to the required accuracy. In spite of the good properties proven, the theoretical study also concludes that a good accuracy cannot be achieved without an important amount of calculations and therefore, a higher order approximation becomes necessary.

The fifth chapter introduces then the *Second Order Quantized State Systems* (QSS2) and the *QSS2–method* following a similar procedure to the one

used when the QSS–method was presented. After studying the trajectories in Section 5.2, the corresponding DEVS model is deduced (Section 5.3) and then the theoretical properties studied in Chapter 4 are extended to the new method. Finally, the simulation of some examples –which shows some practical advantages– is followed with a theoretical and empirical comparison between both introduced methods.

Chapter 6 is concerned with the extension of the QSS and QSS2 methods to some special cases. Sections 6.1 and 6.2 study the use of these methods in *Differential Algebraic Equation* (DAE) Systems, providing a general methodology for the index 1 case. There, a very simple *block–oriented* solution which permits the direct simulation on block diagrams containing algebraic loops is also presented. The simulation examples illustrate an advantage: the method only iterates with the implicit algebraic equations in some particular steps.

Section 6.3 shows the use of the new methods in *Hybrid Systems*. Here, the knowledge of the complete QSS and QSS2 trajectories together with the intrinsec asyncronous behavior of the methods give them several advantages for discontinuity handling. These advantages are illustrated with two examples which also include a comparative analysis against classic methods.

This chapter finishes introducing the application of the QSS–method to the simulation of *Bond Graphs* (BG). The resulting scheme –called *Quantized Bond Graph* (QBG) – gives a very simple way to perform a direct simulation on a Bond Graph model of a physical system. There, Section 6.5 sketches a new way to deal with higher index DAEs resulting from BG models based on a switching behavior and avoiding iterative algorithms.

The seventh chapter introduces the use of the QSS–method in real time control applications. The QSS approximation of a previously designed continuous controller implemented with an asynchronous sampling methodology defines a new digital control scheme which –in theory– avoids the time discretization. This asynchronous digital control method is called *Quantized State Control* (QSC) and it is formally defined in Section 7.2.

Similarly to the simulation methods, QSC can be analyzed as a perturbed version of the original control system in order to deduce its theoretical properties. Making use of this, Sections 7.4 to 7.7 study the stability and the ultimate bounds of QSC nonlinear systems, going from the particular case of Time Invariant (TI) Systems to the general Time Varying cases. There, two practical design procedures –whose use is also illustrated with simulation examples– are developed from the corresponding stability theorems. Finally, the convergence of QSC system trajectories to the Continuous Control System (CCS) trajectories when the quantization goes to zero is shown in Section 7.8.

In Chapter 8, the particular case of QSC applied to LTI Systems is studied and some practical aspects of the methodology are discussed. After introducing the LTI QSC model, Section 8.2 introduces the stability and error bound results based on the non conservative tools developed in Chapter 4. These results are then translated into practical design rules which are applied in two new examples where some advantages over classic digital discrete time control can be observed. Final remarks about practical implementation problems and practical

advantages are presented in Sections 8.4 and 8.5.

The last chapter of the Thesis is finally dedicated to the discussion of unsolved problems, open questions, future research and general conclusions.

1.2 Original Contributions

From the last pages of the second Chapter to the end of the Thesis most of the results are original.

The main contribution is the development of a new formal way to approximate differential equations, which not only includes methods and applications but also a wide variety of analysis tools.

The first original contributions were discovering the *illegitimacy* of Quantized Systems and finding the solution based on the use of hysteresis and the definition of hysteretic quantization functions and Quantized State Systems.

All the work about Quantized State Systems is also original. There, the study of the trajectory forms, the deduction of the DEVS models, the practical issues related to the incorporation of input signals, startup and interpolation were all developed as part of this work.

Another contribution was to realize the relationship between QSS and perturbed systems. Based on this, the theoretical study of stability and convergence was made converting the QSS–method in a well posed integration technique.

In this theoretical study a colateral contribution was a new and less conservative way to analyze the ultimate bound of perturbed LTI systems which can be used not only in the context of quantization but also in more general problems related to perturbations. The use of this new analysis tool in the QSS–method allowed to establish a practical global error bound.

Another original result was the definition of the second order method (QSS2) and all the work made about it: deduction of the trajectory forms, construction of the DEVS model and study of their theoretical properties.

The extension of the methods to be used in DAEs, Hybrid Systems and Bond Graphs was also original as well as all the theoretical and practical study included there.

When it comes to control, the definition of QSC is the first asynchronous digital scheme which can be seen as an approximation of continuous controllers. Except the asynchronous sampling technique, all the study about QSC (definitions, theoretical properties and practical remarks) is completely original.

1.3 Related Works and Relevance of the Results

Among the works which have some relation with this Thesis, two different cases should be distinguished.

On one hand, there are works which use a similar methodology, trying to relate DEVS and differential equations. On the other hand, there are other

works –based on different methods and tools- which attempt to give a solution to similar problems.

With respect to the first group, there is not yet an important amount of work in the literature.

The first ideas and definitions are due to Bernard Zeigler. After defining DEVS [69] in the seventies, the concept of Quantized Systems took more than 20 years to be formally defined [67]. In the context of this line, some interesting applications can be found in [64].

Following a similar goal –i.e. relating DEVS and ODEs– Norbert Giambiasi gave his own approach based on the event representation of trajectories and the definition of GDEVS [17], which was also applied to Bond Graph models in [49]. Although the event representation of piecewise linear trajectories in QSS2 was made using some concepts developed there, GDEVS is based on the knowledge of the ODE solutions and then it cannot be used as a general simulation method.

There is also some recent work of Jean–Sebastien Balduc [2], but –despite the proposal is quite interesting – the research has not arrived yet to results which go much further than Zeigler's ideas.

This Thesis can be seen, in part, as a continuation of Zeigler's work in the area. Here, the main problem (illegitimacy) was discovered and solved and the QSS–method appears then as the first general discrete event integration algorithm for differential equations. However, as it was already mentioned, solving the illegitimacy problem was just the beginning. The work was then extended to a wide variety of theoretical and practical fields.

Although the problem of real–time DEVS has been being considered since 10 years ago [65], there are not precedents about its use in continuous control. Anyway, there was already an idea about the use of quantization to approximate a linear continuous controller using a finite state automata [45]. However, due to the fact that Finite Automata are not as general as DEVS the resulting models are non–deterministic unless they use a very sophisticate quantization.

When it comes to the second group –i.e. the related works which point to the same problems with different tools– there is all the literature on numerical integration and digital control. However, the problems for which this work tries to offer better solutions are in fact much more bounded.

It is impossible anyway to mention and to know all what is being done to solve all these problems. Thus, the works mentioned here will be just the ones which were more related with the more important results of this Thesis.

One of the most remarkable features of the QSS and QSS2 methods is the way in which they exploit sparsity. Many efforts are being doing in the field in order to take advantages of this fact. One of the most efficient simulation tools for numerical ODE integration is Matlab, whose algorithms are provided of special routines which tries to make use of the structure in each matrix multiplication and inversion [59].

However, in the QSS and QSS2 cases the sparsity exploitation is just due to the intrinsic behavior of the methodology. Thus, it is not necessary to use any special routine. Moreover, when a part of a system does not perform changes (here each integrator acts independently) it does not expend computation time

and it does not cause any calculation in the rest of the simulation model.

The global error bounds of different methods is usually studied to prove their convergence when the step size goes to zero [18]. Besides these cases, variable step methods are usually conceived to have this error bounded according to the desired accuracy. In the new methods –which are not provided of any kind of adaptive rules– the global error bound in LTI systems can be calculated by a closed formula which holds for any time and for any input trajectory.

Another area in which QSS and QSS2 showed a good performance is in the simulation of Hybrid Systems. These cases have been always a problem for classic discrete time methods. Here, one of the most difficult issues is the event detection. There is an important number of recent publications which are pointed to find efficient solutions to this problem [53, 62, 58, 13].

QSS and QSS2 have clear advantages in these cases. On one hand, the system trajectories are exactly known during the intersample times. Moreover, they are piecewise linear or parabolic. Thus, finding the exact time at which the discontinuities occur is a trivial problem. On the other hand, the methods are asynchronous and they accept events at any time. Thus, the implementation is very simple and it does not require to modify anything in order to take into account the events.

When it comes to control applications, QSC is defined as an asynchronous digital control scheme based on quantization and one of the most important qualities is that it takes into account the quantization effects of converters at the design time.

The study of quantization effects in sampled data control systems was studied during several years by Anthony Michel's group. They have results on LTI systems [48, 47, 14], concluding about ultimate bounds and errors. Some work was also done with nonlinear plants [20] and with multirate controllers [21].

Some works also attempt to deal with the quantization at the design stage with different goals. In [11] the problem of stabilizing a discrete time linear system taking into account the quantization in the state measurement is studied. The idea is also extended to continuous systems in [5].

Finally, there are some results which use quantization to reduce the amount of information which is transmitted between sensors, controllers and actuators [12].

In all these problems, QSC offers also new solutions. When it comes to quantization effects, their estimation is bounded by a very simple closed formula in LTI systems while in nonlinear cases the bound can be established by a Lyapunov analysis. All these concepts can be taken into account with design purposes.

In the case of information reduction, the advantages are amazing. QSC can work transmitting only a single bit at each sampling.

1.4 Supporting Publications

Most of the results included in this Thesis were already published in journals and conference proceedings, while the rest are still in press or under review.

The first results were discovering the illegitimacy of Quantized Systems, solving it with the addition of hysteresis and the definition of QSS, the deduction of the trajectory forms, the construction of the DEVS model and the proof of the general stability properties. These results were first published in a local conference [27] and then in an internation journal [39], where the convergence property was also included (Sections 2.6 to 4.3 of the Thesis).

The second step was the extension of the results to Bond Graphs models and the definition of Quantized Bond Graphs, whose first version was presented in [26] and then extended for its publication in an international conference [40] (Sections 6.4 and 6.5).

The comparison between the QS approach and the QSS method and the rules for the hysteresis choice were included in an international conference paper [41] (Section 4.7).

After that, the error bound properties of QSS in LTI systems (Section 4.6) were published in the proceedings of a local meeting [28].

Simultaneously, the first results on QSC with Time Invariant plants including the study of stability, design algorithm, convergence and practical remarks were presented as a two parts paper [29, 30] in a local conference (Sections 7.1–7.5, 7.8, and 8.4–8.5). These results are also published in an international journal [37].

The following step was the definition of the second order method QSS2 and the study of their properties (Chapter 5). The results were published in a journal paper [31], where the error bound analysis in LTI systems was also included (Section 4.6).

The non–conservative estimation of ultimate bounds in LTI perturbed systems and its comparison with the classic Lyapunov analysis (Sections 4.4–4.5) was presented in a local control conference [33] and then submitted to a journal (Automatica) as a technical note (it is still under review). The application of the previous result to QSC in LTI systems (Sections 8.1 and 8.3) was also published in the local control conference [35].

A journal paper with the application of QSS and QSS2 to DAEs (Sections 6.1–6.2) was accepted in *Simulation* [34]. The extension of those methods to Hybrid Systems (Section 6.3) was sent to a journal as a full paper [32], but it is still in the review process. In the same situation is the paper [36] which extends QSC for Time Varying plants (Sections 7.6–7.7) and study their properties in LTI systems (Sections 8.1 and 8.3) .

Finally, all the results concerning simulation (Chapters 2 to 6) are included in a coauthored textbook [7] which is still in preparation.

Chapter 2

Quantization and DEVS

The literature on numerical integration of Ordinary Differential Equations –see for instance [55, 19, 18]– shows a wide variety of techniques.

The methods can be either explicit, implicit or linearly implicit (according to the next–step formula), fixed or variable step, fixed or variable order, one or multiple step, etc.

In spite of their differences all those methods have something in common: they discretize the time. In other words, the resulting simulation model (i.e. the system implemented in the computer program) is always a Discrete Time System. Here, the name Discrete Time Systems refers to systems which change in a synchronous way only in some given instants of time.

The problems carried by this kind of behavior in the simulation of Continuous Systems are related to the lost of the simulation *control* between successive discrete instants. Thus, the error could grow to undesired values and, in some cases, it can produce unstability. It is also possible having input changes and even structure changes in some instants of time which do not correspond to those discrete instants.

It is known that the use of methods with step control and implicit formulas allows –sometimes– to handle these problems. However, all these solutions imply using algorithms whose implementation is not straightforward –unless we have tools like Dymola or other comercial software– and they are completely useless in some contexts (in Real-Time Simulation for instance).

Taking into account these facts, it is natural trying to do something in order to avoid the time discretization.

However, the simulation model has to be implemented in a digital device. Then, it is clear that discretization is necessary since only a finite number of changes in the model can be computed for each finite interval of time. Thus, at first glance, the time discretization avoidance seems to be impossible.

In spite of this remark there are other variables which can be discretized as it will be shown in this chapter.

2.1 An Introductory Example

Consider the first order system[1]:

$$\dot{x}(t) = -x(t) + 10\mu(t - 1.76) \tag{2.1a}$$

with the initial condition

$$x(t_0 = 0) = 10 \tag{2.1b}$$

An attempt to simulate it using Euler or any other classic method with a step size $h = 0.1$ –which is appropiated according to the system speed– falls in the case where the input changes at an instant of time which does not coincide with the discrete time.

Let us see now what happens with the following *Continuous Time System*:

$$\dot{x}(t) = -\text{floor}(x(t)) + 10\mu(t - 1.76) \tag{2.2a}$$

or

$$\dot{x}(t) = -q(t) + 10\mu(t - 1.76) \tag{2.2b}$$

where $q(t) \triangleq \text{floor}(x(t))$.

Although the system defined by Eq.(2.2) is nonlinear and it does not satisfy the properties usually observed in ODE integration (Lipchitz conditions, continuity, etc.), it can be easily solved.

When $0 < t < 1/9$ we have $q(t) = 9$ and $\dot{x}(t) = -9$. During this interval, $x(t)$ goes from 10 to 9 with a constant slope (-9). Then, during the interval $1/9 < t < 1/9 + 1/8$ we have $q(t) = 8$ and $\dot{x}(t) = -8$. Now, $x(t)$ goes from 9 to 8 (also with constant slope).

This analysis continues in the same way and in time $t = 1.329$ it results that $x(t) = 3$. If the input do not change, in $t = 1.829$ we would have $x(t = 1.829) = 2$. However, at time $t = 1.76$ (when $x = 2.138$) the input changes and we have now $\dot{x}(t) = 8$. The derivative then changes again when $x(t) = 3$, i.e. in time $t = 1.8678$ (this time can be calculated as $1.76 + (3 - 2.138)/8$).

The calculations continue until we have $x(t) = q(t) = 10$ and in that moment the derivative $\dot{x}(t)$ becomes zero and the system will not change any more. Figure 2.1 shows the trajectories of $x(t)$ and $q(t)$.

This strange *simulation* was completed in only 17 steps and –ignoring the round-off problems– the exact solution of (2.2) was obtained.

This solution and the solution of the original system (2.1) are compared in Figure 2.2.

The solutions of the true system and the modified one are clearly similar. Apparently, if variable x is replaced by $\text{floor}(x)$ in the right hand of a first order differential equation, a method to simulate it is obtained.

This idea could be generalized to be used in a system of order n replacing all the state variables by its floor value in the right hand of the equation.

However, it is necessary to explore the discrete nature of system (2.2) first and to introduce some tools for its representation and simulation.

[1] μ stands for the unit step.

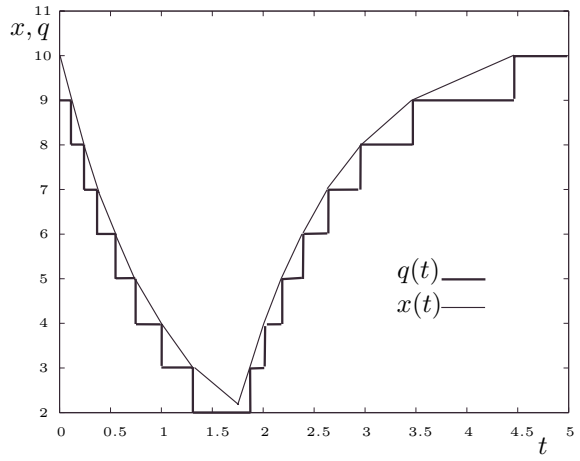

Figure 2.1: Trajectories in system (2.2)

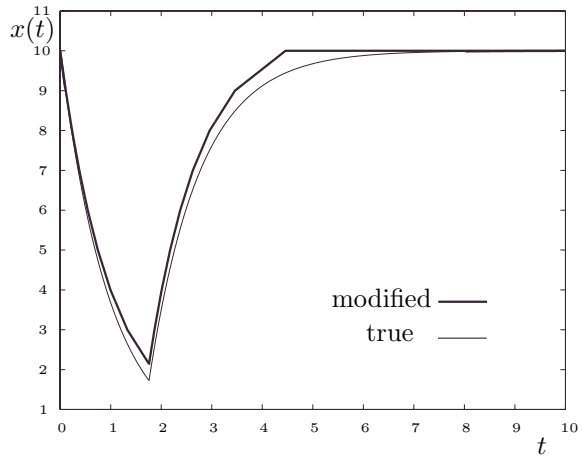

Figure 2.2: State trajectories in the systems (2.1) and (2.2)

2.2 Discrete Event Systems and DEVS

The simulation of a Differential Equation using any previously existing method can be expressed by a difference equation in the form:

$$x(t_{k+1}) = f(x(t_k), t_k) \tag{2.3}$$

where the difference $t_{k+1} - t_k$ can be either constant or variable, and function f can be explicit or implicit. As a consequence, the simulation program has an

iterative code which advances the time according to the next step size. Thus, it is said that those simulation methods produce *Discrete Time Models* of simulation.

System (2.2) can be seen itself as a simulation model because it can be exactly simulated with only 17 steps. However, it does not fit in the form of Eq. (2.3). The problem here is the asynchronous way in which it deals with the input change at $t = 1.76$.

Evidently, there is a system here which is discrete in some way but it belongs to a different category than *Discrete Time*. As it will be seen, our strange system can be represented by a *Discrete Event System*.

The word 'Discrete Events' is usually associated with very popular formalisms like State Automatas, Petri Nets, Event Graphs, Statecharts, etc. Unfortunately, none of them can represent this kind of systems in a general situation. Those graphical languages are limited to system with a finite number of possible states while in this case a more general tool is required. Anyway, such a general formalism exists and it is known as DEVS (Discrete EVent System specification).

The DEVS formalism [69, 66] was developed by Bernard Zeigler in the mid-seventies. The use of DEVS related with continuous systems is not yet very common and it is almost unknown by the numerical method and control communities. However, DEVS is widely used in computer sciences where it received a very important theoretical and practical development.

DEVS allows to represent all the systems whose input/output behavior can be described by sequences of events with the condition that the state has a finite number of changes in any finite interval of time.

An *event* is the representation of an instantaneous change in some part of a system. It can be characterized by a value and an ocurrence time. The value can be a number, a vector, a word, or in general, an element of a given set.

The trajectory defined by a sequence of events adopts the value ϕ (or *No Event*) for all the time values except in the instants in which there are events. In these instants, the trajectory takes the value corresponding to the event. Figure 2.3 shows an event trajectory which takes the value x_1 in time t_1, then the value x_2 at time t_2, etc.

A DEVS model processes an input event trajectory and, according to that trajectory and its own initial conditions provokes an output event trajectory. This Input/Output behavior is represented in Figure 2.4.

The behavior of a DEVS model is expressed in a way which is very common in automata theory.

A DEVS *atomic* model is defined by the following structure:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- X is the set of input event values, i.e., the set of all possible values that and input event can adopt.
- Y is the set of output event values.

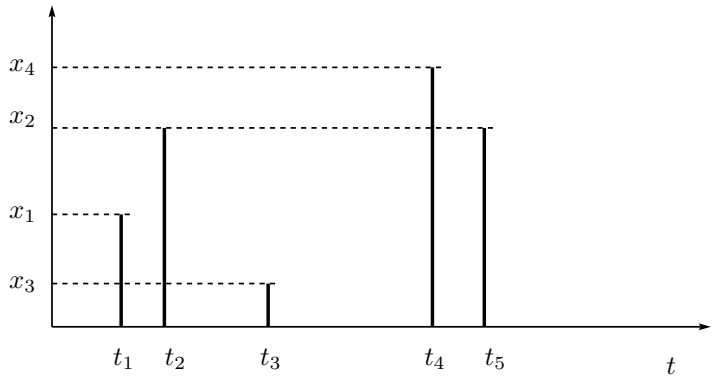

Figure 2.3: An event trajectory

Figure 2.4: Input/Output behavior of a DEVS model

- S is the set of state values.
- δ_{int}, δ_{ext}, λ and ta are functions which define the system dynamics.

Each possible state s ($s \in S$) has an associated *Time Advance* calculated by the *Time Advance Function* $ta(s)$ ($ta(s) : S \rightarrow \mathbb{R}_0^+$). The *Time Advance* is a non-negative real number saying how long the system remains in a given state in absence of input events.

Thus, if the state adopts the value s_1 at time t_1, after $ta(s_1)$ units of time (i.e. at time $ta(s_1) + t_1$) the system performs an *internal transition* going to a new state s_2. The new state is calculated as $s_2 = \delta_{\text{int}}(s_1)$. The function δ_{int} ($\delta_{\text{int}} : S \rightarrow S$) is called *Internal Transition Function*.

When the state goes from s_1 to s_2 an output event is produced with value $y_1 = \lambda(s_1)$. The function λ ($\lambda : S \rightarrow Y$) is called *Output Function*. The functions ta, δ_{int} and λ define the autonomous behavior of a DEVS model.

When an input event arrives the state changes instantaneously. The new state value depends not only on the input event value but also on the previous state value and the elapsed time since the last transition. If the system goes to the state s_3 at time t_3 and then an input event arrives at time $t_3 + e$ with value x_1, the new state is calculated as $s_4 = \delta_{\text{ext}}(s_3, e, x_1)$ (note that $ta(s_3) > e$). In this case, it is said that the system performs an *external transition*. The function δ_{ext} ($\delta_{\text{ext}} : S \times \mathbb{R}_0^+ \times X \rightarrow S$) is called *External Transition Function*.

No output event is produced during an external transition.

Example 2.1. *DEVS model of a static scalar function.*

Consider a system which receives a piecewise constant trajectory $u(t)$ represented by a sequence of events with the consecutive values. The system output is another sequence of events which represents the function $y(t) = f(u(t))$ where $f(u)$ is a known real–valued function.

A possible DEVS model corresponding to this behaviour is given by the following structure:

$$
\begin{aligned}
&M_1 = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta), \text{ where} \\
&X = Y = \mathbb{R} \\
&S = \mathbb{R} \times \mathbb{R}_0^+ \\
&\delta_{int}(s) = \delta_{int}(u, \sigma) = (u, \infty) \\
&\delta_{ext}(s, e, x) = \delta_{ext}((u, \sigma), e, x) = (x, 0) \\
&\lambda(s) = \lambda(u, \sigma) = f(u) \\
&ta(s) = ta(u, \sigma) = \sigma
\end{aligned}
$$

Note that the state is composed by two real numbers. The first one (u) contains the last input value and the second one (σ) has the time advance. In most DEVS models that variable σ, equal to the time advance, is put as part of the state. In that way, the modeling task becomes easier.

This DEVS model has a *static* behavior since it only does something when an event arrives. It was mentioned that no output event is produced during an external transition. However, in this example a trick was made to produce the the output event: the time advance is set to zero when an event arrives. Then, the internal transition takes place immediatly and the output event is produced.

2.3 Coupled DEVS models

As it was mentioned, DEVS is a very general formalism and it can describe very complex systems. However, the representation of a complex system based only on the transition and time advance functions is too difficult. The reason is that in those functions all the possible situations in the system must me imagined and described.

Furtunately, complex systems can be usually thought as the coupling of simpler ones. Through the coupling, the output events of some subsystems are converted into input events of other subsystems. The theory guarantees that the coupling of DEVS models defines a new DEVS model (i.e. DEVS is closed under coupling) and the complex systems can be represented by DEVS in a hierarchical way [66].

There are basically two different ways of coupling DEVS models. The first one is the most general and uses *translation functions* between subsystems. The second one is based on the use of input and output ports. This last way is the

one which will be used in this Thesis since it is simpler and more adequate to the simulation of continuous systems.

The use of ports requires adding to the input and output events a new number, word or symbol representing the port in which the event is coming. The following example –which is a modification of the Example 2.1– illustrates this idea.

Example 2.2. *DEVS model of a static function*

Consider the system of Example 2.1, but suppose that it receives n piecewise constant inputs, $u_1(t), \ldots, u_n(t)$ *and the output* $y(t)$ *is now calculated as* $y = f(u_1, \ldots, u_n)$.

Then, a DEVS atomic model with ports which can represent this behavior is given by the following structure:

$$
\begin{aligned}
&M_2 = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta), \text{ where} \\
&X = Y = \mathbb{R} \times \mathbb{N} \\
&S = \mathbb{R}^n \times \mathbb{R}_0^+ \\
&\delta_{int}(s) = \delta_{int}(u_1, \ldots, u_n, \sigma) = (u_1, \ldots, u_n, \infty) \\
&\delta_{ext}(s, e, x) = \delta_{ext}((u_1, \ldots, u_n, \sigma), e, (x_v, p)) = (\tilde{u}_1, \ldots, \tilde{u}_n, 0) \\
&\lambda(s) = \lambda(u_1, \ldots, u_n, \sigma) = (f(u_1, \ldots, u_n), 1) \\
&ta(s) = ta(u_1, \ldots, u_n, \sigma) = \sigma
\end{aligned}
$$

where

$$
\tilde{u}_i = \begin{cases} x_v & \text{if } i = p \\ u_i & \text{otherwise} \end{cases}
$$

As it can be seen, in this last example the input and output events contain a natural number which indicates the corresponding port.

Then, the coupling between different systems is indicated by enumerating the connections to describe it. An *internal connection* involves an input and an output port corresponding to different models. In the context of hierarchical coupling, there are also connections from the output ports of the subsystems to the output ports of the network –which are called *external output connections*– and connections from the input ports of the network to the input ports of the subsystems (*external input connections*).

Figure 2.5 shows a coupled DEVS model N which is the result of coupling the models M_a and M_b. There, the output port 2 of M_a is connected to the input port 1 of M_b. This connection can be represented by $[(M_a, 2), (M_b, 1)]$. Other connections are $[(M_b, 1), (M_a, 1)]$, $[(N, 1), (M_a, 1)]$, $[(M_b, 1), (N, 2)]$, etc. According to the closure property, the model N can be also used as an atomic DEVS and it can be coupled with other atomic or coupled models.

The DEVS theory uses a formal structure to represent coupled DEVS models with ports. The structure includes the subsystems, the connections, the network input and output sets and a *tie-breaking* function to manage the presence of simultaneous events. The connections are divided into three sets: one set composed by the connections between subsystems (internal connections), other

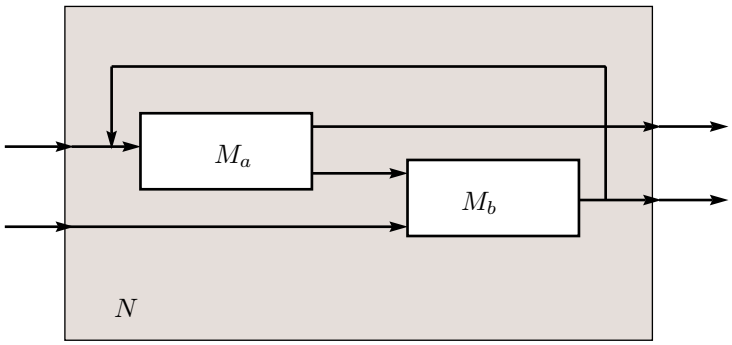

Figure 2.5: Coupled DEVS model

set which contains the connections from the network to the subsystems (external input connections) and the last one has the connections from the subsystems to the network (external output connections).

The *tie-breaking* function can be avoided with the use of *Parallel-DEVS*, which is an extension of the DEVS formalism that allows dealing with simultaneous events.

However, these last concepts –the coupled DEVS formal structure and the parallel-DEVS formalism– will not be develped here since they are not necessary to use DEVS as a tool for continuous system simulation. Anyway, the complete theory of DEVS can be found in the second edition of Zeigler's book [66].

2.4 Simulation of DEVS models

One of the most important features of DEVS is that very complex models can be simulated in a very easy and efficient way.

In the last years, several software tools have been developed for the simulation of DEVS models. Some of those tools offer libraries, graphical interfaces and different facilities for the user. There are several free software packages for DEVS simulation and the most popular are DEVS-Java [68] and DEVSim++ [25].

It should be also mentioned here a software tool conceived in the context of this work which not only constitutes a general purpose DEVS simulation environment but also implements the main ideas which will be developed in this Thesis. This tool is called PowerDEVS [51] and it was developed by Esteban Pagliero and Marcelo Lapadula as a Diploma Work at the FCEIA, UNR.

Besides these tools, DEVS models can be simulated with a simple ad-hoc program written in any language. In fact, the simulation of a DEVS model is not much more complicated than the simulation of a Discrete Time Model. The problem is that there are models which are composed by many subsystems and

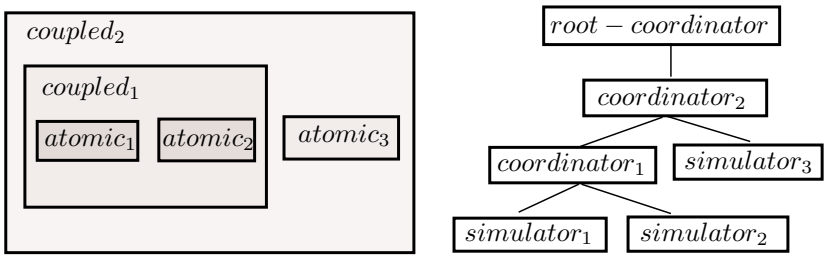

Figure 2.6: Hierarchical model and simulation scheme

the ad-hoc programming may become a very hard task.

The basic idea for the simulation of a coupled DEVS model can be described by the following steps:

1. Look for the atomic model that, according to its time advance and elapsed time, is the next to perform an internal transition. Call it d^* and let tn be the time of the mentioned transition

2. Advance the simulation time t to $t = tn$ and execute the internal transition function of d^*

3. Propagate the output event produced by d^* to all the atomic models connected to it executing the corresponding external transition functions. Then, go back to the step 1

One of the simplest ways to implement these steps is writing a program with a hierarchical structure equivalent to the hierarchical structure of the model to be simulated. This is the method developed in [66] where a routine called *DEVS-simulator* is associated to each *atomic DEVS model* and a different routine called *DEVS-coordinator* is related to each *coupled DEVS model*. At the top of the hierarchy there is a routine called *DEVS-root-coordinator* which manages the global simulation time. Figure 2.6 illustrates this idea over a coupled DEVS model

The simulators and coordinators of consecutive layers communicates each other with messages. The coordinators send messages to their children so they execute the transition functions. When a simulator executes a transition, it calculates its next state and –when the transition is internal– it sends the output value to its parent coordinator. In all the cases, the simulator state will coincide with its associated atomic DEVS model state.

When a coordinator executes a transition, it sends messages to some of their children so they execute their corresponding transition functions. When an output event produced by one of its children has to be propagated outside the coupled model, the coordinator sends a message to its own parent coordinator carrying the output value.

Each simulator or coordinator has a local variable tn which indicates the time when its next internal transition will occur. In the simulators, that variable is calculated using the time advance function of the corresponding atomic model. In the coordinators, it is calculated as the minimum tn of their children. Thus, the tn of the coordinator in the top is the time in which the next event of the entire system will occur. Then, the root coordinator only looks at this time, advances the global time t to this value and then it sends a message to its child so it performs the next transition (and then it repeats this cycle until the end of the simulation).

The details and the pseudo–codes associated to the simulators, coordinators and root coordinators are included in Appendix B.

One of the most interesting properties shown by this kind of simulation is the independence between different simulators associated to different atomic models. In that way, when an atomic model has its time advance set to a big value (or infinite), it does not affect at all to the rest of the models and the simulation does not spend any calculation with it. It will be seen that this fact will result in an important advantage for the simulation of *sparse* systems.

There are many other possibilities to implement a simulation of DEVS models. The main problem with the methdology described is that, due to the hierarchical structure, an important traffic of messages between the higher layers and the lower ones can be present. All these messages and their corresponding computational time can be avoided with the use of a flat structure of simulation. The way of transforming a hierarchical simulation into a flat one is rather simple in DEVS [24]. In fact, most of the software tools we mentioned implement the simulation based on a flat code.

2.5 Quantized Systems and DEVS

In the examples 2.1 and 2.2 it was shown that piecewise constant trajectories can be represented by sequences of events. This simple idea constitutes the basis for the use of DEVS in the simulation of continuous systems.

In those example, it was also shown that a DEVS model can represent the behavior of a static function with piecewise constant input trajectories. The only problem is that most continuous system trajectories are not piecewise constant. However, the system can be modified in order to have such kind of trajectories. In fact, that is what was done to System (2.1) using the function floor to convert it into System (2.2).

Coming back to that example, System (2.2) can be divided in the following way:

$$\dot{x}(t) = d_x(t) \tag{2.4a}$$

$$q(t) = \text{floor}(x(t)) \tag{2.4b}$$

and

$$d_x(t) = -q(t) + u(t) \tag{2.5}$$

where $u(t) = 10\mu(t - 1.76)$.

The system can be then represented using the Block Diagram of Figure 2.7.

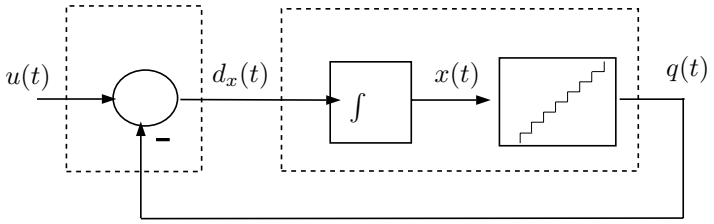

Figure 2.7: Block Diagram Representation of (2.4)-(2.5)

As we mentioned before, Subsystem (2.5) –which is modeled by a static function– can be represented by the DEVS model M_2 in Example 2.2 (page 15).

Subsystem (2.4) is a dynamic equation which has a piecewise constant input trajectory $d_x(t)$ and a piecewise constant output trajectory $q(t)$. It can be exactly represented using the DEVS model that follows:

$$
\begin{aligned}
&M_3 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = Y = \mathbb{R} \times \mathbb{N} \\
&S = \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{R}^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, d_x, q + \text{sgn}(d_x), \frac{1}{|d_x|}) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(x, d_x, q, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, q, \tilde{\sigma}) \\
&\lambda(s) = \lambda(x, d_x, q, \sigma) = (q + \text{sgn}(d_x), 1) \\
&ta(s) = ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

where

$$
\tilde{\sigma} = \begin{cases} \frac{q + 1 - x}{x_v} & \text{if } x_v > 0 \\ \frac{q - x}{x_v} & \text{if } x_v < 0 \\ \infty & \text{otherwise} \end{cases}
$$

Subsystem (2.4) –which corresponds to the integrator with the stairway block in the Block Diagram of Figure 2.7– is what Zeigler called *Quantized Integrator* [67, 66]. There, the function floor acts as a *quantization function*. A *quantization function* maps all real numbers into a discrete set of real values.

A quantizer is a system that relates its input and output by a *quantization function*. Then, the stairway block is a particular case of a quantizer. Although the DEVS model M_3 represents a particular *Quantized Integrator* the DEVS

model correponding to a general one –i.e.. with a general quantizer– is not very different.

The complete system (2.2) was called *Quantized System* and it can be exactly represented by the DEVS model resulting from the coupling of the atomic models M_2 and M_3.

This was the idea sketched by Zeigler to approximate and simulate continuous systems using DEVS.

As it was seen, a DEVS model representation of general *Quantized Integrators* can be obtained. This idea will work when their input trajectories are piecewise constant. It is also clear that the DEVS model of any static function (with the same condition for the input trajectories) can be also built.

Taking also into account that DEVS is closed under coupling, it is natural to think that a coupled DEVS model representing a general Quantized System can be easily obtained.

In order to do it, a general time invariant system should be considered:

$$\begin{array}{rcl} \dot{x_1} & = & f_1(x_1, x_2, \cdots, x_n, u_1, \cdots, u_m) \\ \dot{x_2} & = & f_2(x_1, x_2, \cdots, x_n, u_1, \cdots, u_m) \\ & \vdots & \\ \dot{x_n} & = & f_n(x_1, x_2, \cdots, x_n, u_1, \cdots, u_m) \end{array} \tag{2.6}$$

This last equation can be can transformed into:

$$\begin{array}{rcl} \dot{x_1} & = & f_1(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ \dot{x_2} & = & f_2(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ & \vdots & \\ \dot{x_n} & = & f_n(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \end{array} \tag{2.7}$$

where each $q_i(t)$ is related to $x_i(t)$ by some quantization function.

Considering that the input functions $u_j(t)$ are piecewise constant, each term at the right hand of (2.7) can only adopt values in a finite set.

The variables q_i are called *quantized variables*. This system can be represented by the block diagram of Figure 2.8, where $\mathbf{q}$ and $\mathbf{u}$ were defined as the vectors formed with the quantized and input variables respectively.

Each subsystem in Figure 2.8 can be exactly represented by a DEVS model since they are composed by a static function and a quantized integrator. These DEVS models can be coupled and according to the closure under coupling property the complete system will define a DEVS model.

Thus, when a system is modified with the addition of quantizers at the output of the integrators, it can be exactly simulated by a DEVS model.

This idea is the formalization of the first approximation to a discrete event based method for continuous system simulation. With this method –ignoring the round–off errors– System (2.7) can be exactly simulated. Taking into account that this system seems to be an *approximation* to the original System (2.6), it can be thought that a numerical method which avoids the time discretization was finally obtained.

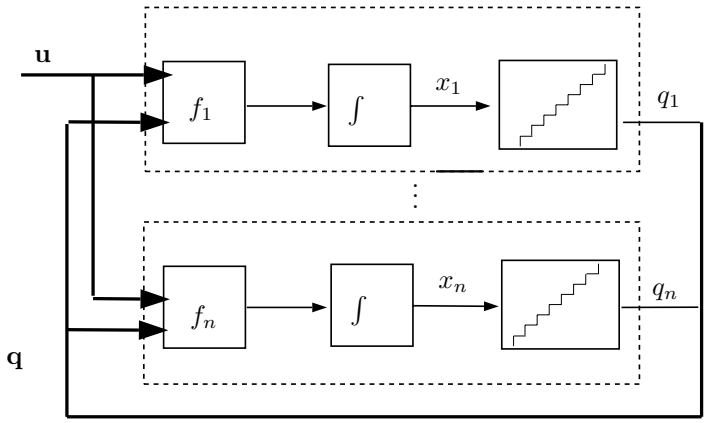

Figure 2.8: Block Diagram Representation of (2.7)

However, as it will be seen in the next section, this idea does not work in general cases.

2.6 Illegitimacy of Quantized Systems

A DEVS model is said to be legitimate when it cannot perform an infinite number of transitions in a finite interval of time [66].

Legitimacy is the property which ensures that a DEVS model can be simulated. Otherwise –when infinite transitions can occur in a finite interval– the system can only be simulated until that condition is reached. In that case, it is said that the system is illegitimate.

DEVS theory distinguishes two cases of illegitimacy. The first one is when there are infinite transitions in the same instant of time (i.e. a loop between different states with the time advance equal to zero). This kind of illegitimacy is also common in other discrete event formalisms (timed event graphs for instance).

The second case occurs when the system goes through an infinite sequence of states in which the time advance decreases. In that case, if the summatory of the serie defined by the time advance values converges to a finite number there is also an infinite number of events in a finite interval of time. Those cases are also called Zeno systems in reference to Zeno's paradox of Achilles and the Tortoise.

It can be easily checked that the atomic DEVS models M_2 and M_3 are legitimate. Unfortunately, legitimacy is a property which is not closed under coupling. As a result, the coupling of legitimate models might result in an illegitimate coupled model.

This fact opens the possibility that a DEVS model like the shown in Fig.2.8 results illegitimate. In fact, this is what happens in most cases.

However, the illegitimacy of Quantized Systems is not a problem of DEVS. It is related to the solutions of (2.7). There, the trajetories $q_i(t)$ are not necessarily piecewise constant. Sometimes, they can have an infinite number of changes in a finite interval of time which produces an infinite number of events in the corresponding DEVS model.

This problem can be observed just taking $u(t) = 10.5\mu(t - 1.76)$ in System (2.5)–(2.4). The trajectories until $t = 1.76$ are exactly the same as the shown in Figure 2.1. When the step is applied, the trajectory starts growing a bit faster and when $x(t) = q(t) = 10$ the state trajectory continues growing with a slope $\dot{x}(t) = 0.5$. Then, after 2 units of time we obtain $x(t) = q(t) = 11$ but immediatly the slope becomes negative ($\dot{x}(t) = -0.5$). Thus, $x(t)$ starts falling, $q(t)$ goes back to 10, the derivative becomes again positive and we obtain a cyclic behavior. The problem is that the cycle has a period equal to zero and there are infinte changes in $q(t)$ in the same instant of time.

This anomalous behaviour can be also observed in the resulting DEVS model. When the DEVS model corresponding to the integrator performs an internal transition, it also produces an output event which represents the change in $q(t)$. This event is propagated by the internal feed-back –see Figure 2.7– and it produces a new external transition in the integrator which changes the time advance to zero. Thus, the integrator performs another internal transition and the cycle continues forever.

This case belongs to the first kind of illegitimacy above mentioned. Here, the system can be only simulated until the condition $x(t) = 10$ is reached.

It could be conjectured that illegitimacy only occurs when the system approachs the equilibrium point. If that were the case, the illegitimacy condition could be detected finishing the simulation at that moment. However, that conjecture is only true in first order systems.

In higher order systems this can of behavior can be also observed far away from the equilibrium points. Moreover, Zeno–like illegitimacy can also occur as it is shown in the following counter–example:

Example 2.3. *Achilles and the Tortoise.*

Consider the second order system:

$$\begin{aligned} \dot{x_1} &= -0.5 \cdot x_1 + 1.5 \cdot x_2 \\ \dot{x_2} &= -x_1 \end{aligned} \tag{2.8}$$

Let us apply the quantization function:

$$q_i = 2 \cdot \mathit{floor}(\frac{x_i - 1}{2}) + 1 \tag{2.9}$$

This quantization (see Figure 2.9) over both variables divides the state space as Fig.2.10 shows:

Then, the resulting Quantized System is:

$$\begin{aligned} \dot{x_1} &= -0.5 \cdot q_1 + 1.5 \cdot q_2 \\ \dot{x_2} &= -q_1 \end{aligned} \tag{2.10}$$

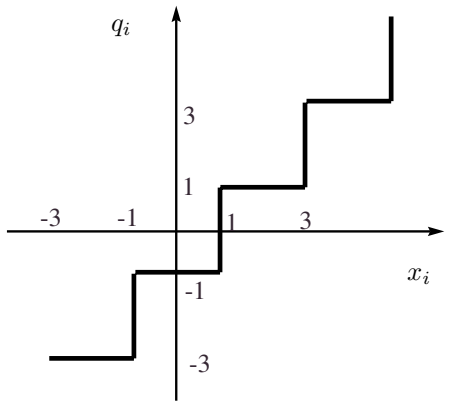

Figure 2.9: An *odd* quantization function

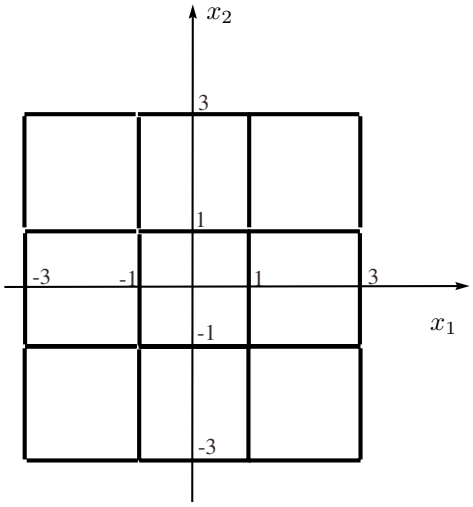

Figure 2.10: Partition of the state space

Now, let us analyze the solution of (2.10) *from the initial condition* $x_1 = 0$, $x_2 = 2$.

The derivative of the state vector of (2.10) *in a point is given by rigth hand of that equation, using* (2.9). *This is equal to the derivative of the continuous system* (2.8) *evaluated in the bottom left corner of the square cointaining that point. For instance, at the point* $(0, 2)$ *the derivative is given by the right hand of* (2.8) *at the point* $(-1, 1)$, *that is* $(2, 1)$.

Then, if the initial condition is $(0, 2)$ *the trajectory will go following the di-*

rection $(2, 1)$ *until it reaches the next square (here we have a transition since there is a change in the quantized variable* q_1 *corresponding to* x_1*). The transition will occur at the time* $t = 0.5$*(the speed in* x_1 *is 2 and the distance to the square is 1). The point in which the trajectory reaches the new square is* $(1, 2.5)$*.*

After this transition, the derivative is calculated at the point $(1, 1)$*. The direction is now* $(1, -1)$*. After* 1.5 *units of time the system will reach the point* $(2.5, 1)$ *arriving to a new square. The new direction is* $(-2, -1)$ *(calculated at the point* $(1, -1)$*) and after* 0.75 *units of time the system will reach the point* $(1, 0.25)$ *in the bound of a new square. Then, the direction is* $(-1, 1)$ *and after* 0.75 *units of time the system reaches the initial square at the point* $(0.25, 1)$*. Then, after* 0.375 *units of time the system goes back to the second square, arriving at the point* $(1, 1.375)$*.*

The elapsed time from the first time the system reaches the second square to the second arrival to that square is 3.375*. Then, it can be easily seen that the system will follow again a similar cycle but starting from the new initial condition* $(1, 1.375)$ *and it will take* $3.375/4 = 0.84375$ *units of time. Each cycle will be done four times faster than the previous one. Then, the sum of all the cycle times will converge to* 4.5 *units of time. Since the first transition occurs at time* 0.5*, before 5 unit of time the system performs an infinite number of transitions.*

Figure 2.11 shows that trajectory in the space state while Fig.2.12 shows the temporal evolution of the quantized variable q_1*.*

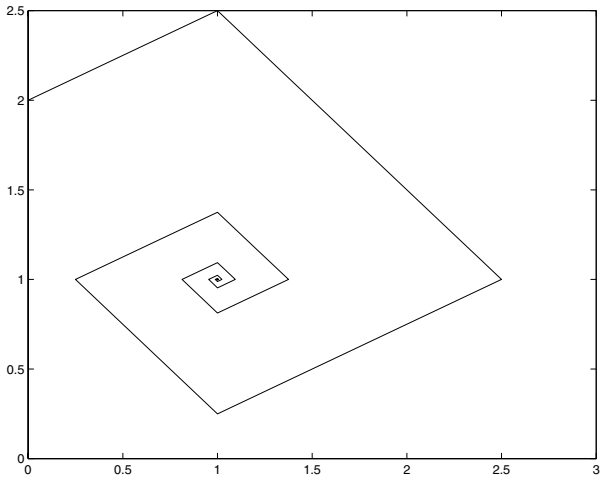

Figure 2.11: State Space trajectory with infinite transitions

As a result of this behavior, the simulation will be stuck after 5 units of time.

This last example not only exhibits illegitimacy, but it also shows that sometimes the illegitimacy condition cannot be easily detected. It is difficult –if not impossible– to write a routine which distinguishes this case as illegitimate since

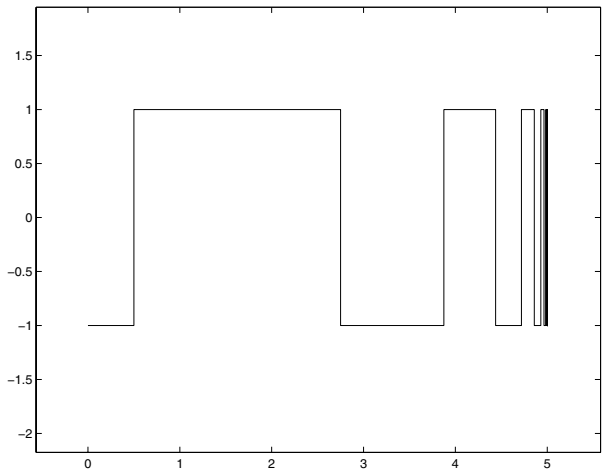

Figure 2.12: Quantized variable trajectory with infinite transitions

the transitions does not occur at the same time.

Unfortunately, illegitimate Quantized Systems are very common. As it was already mentioned, in most continuous system the use of this kind of quantization yields illegitimate DEVS models. Consecuently, the approach introduced cannot constitute a simulation method since it does not work in most cases.

2.7 DEVS and Continuous Systems Simulation

In spite of the illegitimacy problems, Zeigler's idea of discretizing state variables is very original and it yields very significant properties and qualities.

In fact, it was the first attempt to produce a formal transformation of a continuous system in order to obtain a discrete event one.

There was also another idea which should be also mentioned at least. Following a similar goal, Norbert Giambiasi gave his own approach based on the event representation of piecewise polynomial trajectories and the definition of GDEVS [17]. However, this *solution-based* approximation requieres the knowledge of the continuous system response to some particular input trajectories, which is not available in most cases. Because of this and also due to impossibility of formalizing the approach, these ideas will not be discussed here.

Coming back to Zeigler's approach, the main motivation of this Thesis was the discovering of illegitimacy in Quantized Systems and the original goal was trying to solve it.

Thus, the next chapter is completely dedicated to describe the solution which was found and to develop the resulting numerical algorithm, which is the first general discrete event integration method for ordinary differential equations.

After that, the theoretical properties, extensions and applications will be

introduced in the following chapters.

Chapter 3

Quantized State Systems

The previous and introductory chapter was a sort of description about the relationship between discrete event systems and numerical methods for ODEs.

There, two counter–examples were included showing the impossibility of simulating general continuous systems by using simple quantization because of illegitimacy.

In spite of this problem, the idea of approximating a differential equation by a discrete event system is still very attractive since it can offer many advantages with respect to a discrete time approach. For instance, due to the asynchronous behavior, the DEVS models can be implemented in parallel in a very easy and efficient way.

Many modern technical systems are described with models which combine discrete and continuous parts. The simulation of those *Hybrid Systems* requires the use of special techniques and a discrete event approximation of the continuous part would result in a unique discrete event simulation model. In fact, this idea –combining continuous and discrete simulation in a unique method– was the motivation of Herbert Praehofer's PhD Thesis [54]. However, at the beginning of the 90s there were not discrete event approximation methods and Praehofer's work was limited to express in terms of DEVS some existing discrete time integration methods.

As it will be seen, the use of discrete event approximations yields an important reduction of the computational costs in the simulation of hybrid systems. But the advantages of a discrete event integration method do not finish here. Besides other practical advantages it will be shown that some theoretical properties resulting from these kind of approximation are completely original in the field of numerical integration theory.

This chapter introduces most of the key ideas which allowed the formulation of the first general discrete event integration method for differential equations.

3.1 Hysteretic Quantization

If the infinitely fast oscillations in System (2.4)–(2.5) are analyzed, it can be seen that they are due to the changes in $q(t)$. An infinitesimal variation in $x(t)$ can produce, due to the quantization, an important oscillation with an infinitely fast frequency in $q(t)$.

A possible solution might consist in adding some delay after a change in $q(t)$ to avoid those infinitely fast oscillations. However, adding such delays is equivalent, in some way, to introduce time discretization. During the delays the *control* over the simulation is lost and we come back to the problems of the discrete time algorithms.

A different solution consists in the use of hysteresis in the quantization. If a hysteretic characteristic is added to the relationship between $x(t)$ and $q(t)$, the oscillations in $q(t)$ can be only produced by *large* oscillations in $x(t)$ which need a minimum time interval to occur due to the continuity in the state trajectories.

The existence of a minimum time interval between events is a sufficient condition for legitimacy [66]. Then, this simple idea –adding hysteresis to the quantization– is a good solution to fix the illegitimacy problem.

The formalization of the idea is given by the definition of the *hysteretic quantization functions* .

Definition 3.1. *Hysteretic Quantization Function.*

Let $Q = \{Q_0, Q_1, ..., Q_r\}$ be a set of real numbers where $Q_{k-1} < Q_k$ with $1 \leq k \leq r$. Let Ω be the set of piecewise continuous real valued trajectories and let $x \in \Omega$ be a continuous trajectory. Let $b : \Omega \to \Omega$ be a mapping and let $q = b(x)$ where the trajectory q satisfies:

$$q(t) = \begin{cases} Q_m & \text{if } t = t_0 \\ Q_{j+1} & \text{if } x(t) = Q_{j+1} \wedge q(t^-) = Q_j \wedge j < r \\ Q_{j-1} & \text{if } x(t) = Q_j - \varepsilon \wedge q(t^-) = Q_j \wedge j > 0 \\ q(t^-) & \text{otherwise} \end{cases} \tag{3.1}$$

and

$$m = \begin{cases} 0 & \text{if } x(t_0) < Q_0 \\ r & \text{if } x(t_0) \geq Q_r \\ k & \text{if } Q_k \leq x(t_0) < Q_{k+1} \end{cases}$$

Then, the map b is a hysteretic quantization function.

The discrete values Q_j are called *quantization levels* and the distance $\Delta Q \triangleq Q_{j+1} - Q_j$ is defined as the *quantum*, which is usually constant. The width of the hysteresis window is ε. The values Q_0 and Q_r are the lower and upper saturation values. Figure 3.1 shows a typical quantization function with uniform quantization intervals.

The use of hysteretic quantization functions instead of *memoryless* quantization yields the Quantized State Systems method (or QSS–method for short) for ODE integration.

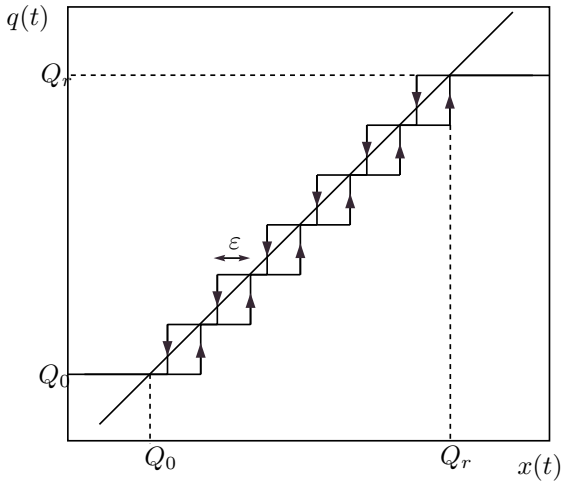

Figure 3.1: Quantization Function with Hysteresis

3.2 QSS–method

The QSS–method follows the idea of the generalization of Quantized Systems (Section 2.5). The only difference here is the use of hysteresis in the quantization.

Then, the QSS–method can be defined as follows:

Definition 3.2. *QSS-method.*

Given a time invariant state equation system:

$$\dot{x}(t) = f(x(t), u(t)) \tag{3.2}$$

where $x \in \mathbb{R}^n$, $u \in R^m$ and $f : \mathbb{R}^n \to \mathbb{R}^n$, the QSS–method approximates it by a system:

$$\dot{x}(t) = f(q(t), u(t)) \tag{3.3}$$

where $q(t)$ and $x(t)$ are related componentwise by hysteretic quantization functions (i.e. each quantized variable $q_i(t)$ is related to the corresponding state variable $x_i(t)$ by a hysteretic quantization function).

The resulting system (3.3) is called *Quantized State System* (QSS).

Figure 3.2 shows the block diagram representation of a generic QSS.

The QSS–method requires to choose quantization functions like the one shown in Figure 3.1. After choosing one of such functions for each state variable, a QSS is obtained.

In the next sections it will be proven that the resulting QSS is *equivalent* to a legitimate DEVS model (i.e. it can be exactly simulated by a legitimate DEVS).

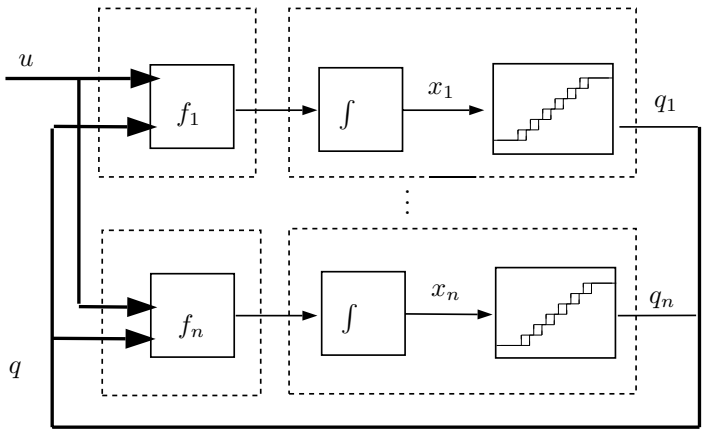

Figure 3.2: Block Diagram Representation of a QSS

Only after that, it will be possible to claim that *the QSS–method approximates a Differential Equation System by a legitimate DEVS model.*

3.3 Trajectories in QSS

Taking into account that QSS tries to be a discrete event approximation of a continuous system, their trajectories should have some particular properties. Since DEVS only processes events, each QSS trajectory should have an equivalent event trajectory.

Then, QSS must have piecewise constant, piecewise linear or piecewise something trajectories. Otherwise, their segments could never be represented by a finite sequence of values.

The non–hysteretic Quantized System approach failed in this goal.The trajectories there were not necessarily piecewise constant or linear as it could be seen in the counter–examples of Section 2.6.

However, the use of hysteresis in QSS solves those problems. Provided that the input trajectories are piecewise constant and bounded and function f is continuous and bounded in any bounded domain, the following properties are satisfied:

- The quantized variables have piecewise constant trajectories
- The state variable derivatives have also piecewise constant trajectories
- The state variables have continuous piecewise linear trajectories

The following theorems give the necessary conditions and prove the mentioned properties.

Theorem 3.1. *Quantized Trajectories in QSS*

Given the QSS defined in (3.3) *with f continuous and bounded in any bounded domain and $u(t)$ being bounded and piecewise constant, the trajectories of $q(t)$ are piecewise constant.*

Proof. Let $q_i(t)$ be an arbitrary component of q. Assume that it is related to $x_i(t)$ by the hysteretic quantization function given by (3.1). It follows from (3.1) that:

$$Q_0 \leq q_i(t) \leq Q_r \tag{3.4}$$

If we assume that all the quantized variables are related to the corresponding state variables by similar hysteretic quantization functions, Inequality (3.4) also implies that $\|q\|$ is bounded. From the hypothesis made about f there exists a positive number R such that

$$-R \leq \dot{x_i} \leq R \tag{3.5}$$

After integrating the inequality above we have

$$x_i(0) - R(t - t_0) \leq x_i(t) \leq x_i(t_0) + R(t - t_0) \tag{3.6}$$

Inequality (3.6) shows that the state variables have bounded trajectories in any finite time interval. Moreover, from (3.5) it follows that the state variables have also continuous trajectories.

Assume that in certain time t there is a change in q_i. If that change was produced because x_i was increasing its value, it results that:

$$x_i(t) = q_i(t^+) = Q_j \quad (0 < j \leq r)$$

Then, it follows from (3.5) and (3.1) that:

$$q_i(t + \Delta t) \neq q_i(t^+) \Rightarrow \Delta t \geq \frac{\min(Q_{j+1} - Q_j, \varepsilon)}{R} \triangleq \Delta t_{min} \tag{3.7}$$

If we assume that x_i was initially falling, then we have

$$x_i(t) = Q_{j+1} - \varepsilon = q_i(t^+) - \varepsilon$$

and we get again (3.7).

This implies that variable q_i needs a time interval greater than Δt_{min} to change its value twice. Since this value is constant (it does not depend on the state), it results that q_i has a piecewise constant trajectory.

Taking into account that we made the analysis for a generic component, we conclude that q is piecewise constante. □

Theorem 3.2. *State Derivative Trajectories in QSS.*

In a QSS verifying the hypothesis of Theorem 3.1 the trajectories of the state variable derivatives are piecewise constant.

Proof. It is straightforward from Theorem 3.1 since $q(t)$ and $u(t)$ are piecewise constant and f is a static function. □

Theorem 3.3. *State Trajectories in QSS.*

In a QSS verifying the hypothesis of Theorem 3.1 the trajectories of the state variables are continuous and piecewise linear.

Proof. It is straightforward from Theorem 3.2. □

The definition of Δt_{min} in (3.7) shows the effect of the hysteresis in the QSS trajectories. When ε is zero we cannot ensure the existence of a minimum time between changes in q_i.

Figure 3.3 shows typical trajectories in a QSS.

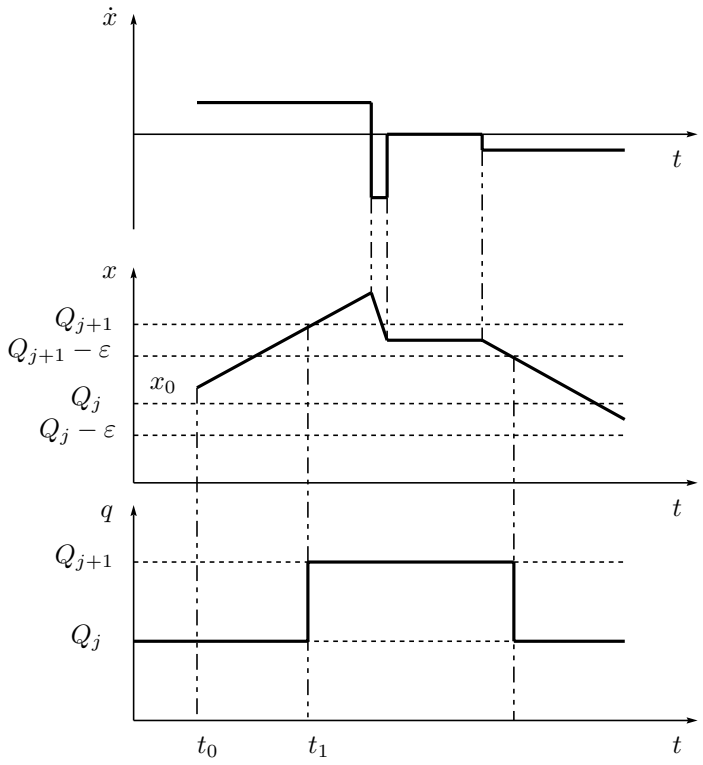

Figure 3.3: Typical trajectories in a QSS

3.4 DEVS model of a QSS

Taking into account that the components of $q(t)$ and $\dot{x}(t)$ are piecewise constant, they can be represented by sequences of events. Thus, following the procedure of Section 2.5, the QSS of Figure 3.2 can be divided into static functions and quantized integrators (now *hysteretic quantized integrators*).

The static functions can be still represented by the DEVS model M_2 in Example 2.2 (page 15) but the DEVS model M_3 corresponding to the quantized integrators (page 19) must be modified taking into account the presence of hysteresis.

With this modification, a hysteretic quantized integrator –which can be thought as a system constituted by Eqs (2.4a) and (3.1)– can be represented by the DEVS model:

$$
\begin{aligned}
&M_4 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where}\\
&X = Y = \mathbb{R} \times \mathbb{N}\\
&S = \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{R}^+\\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, j, \sigma) = (x + \sigma \cdot d_x, d_x, j + \text{sgn}(d_x), \sigma_1)\\
&\delta_{\text{ext}}(s, e, x_u) = \delta_{\text{ext}}(x, d_x, j, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, j, \sigma_2)\\
&\lambda(s) = \lambda(x, d_x, j, \sigma) = (Q_{j+\text{sgn}(d_x)}, 1)\\
&ta(s) = ta(x, d_x, j, \sigma) = \sigma
\end{aligned}
$$

whith

$$
\sigma_1 = \begin{cases} \dfrac{Q_{j+2} - (x + \sigma \cdot d_x)}{d_x} & \text{if } d_x > 0 \\ \dfrac{(x + \sigma \cdot d_x) - (Q_{j-1} - \varepsilon)}{|d_x|} & \text{if } d_x < 0 \\ \infty & \text{if } d_x = 0 \end{cases}
$$

$$
\sigma_2 = \begin{cases} \dfrac{Q_{j+1} - (x + e \cdot x_v)}{x_v} & \text{if } x_v > 0 \\ \dfrac{(x + e \cdot x_v) - (Q_j - \varepsilon)}{|x_v|} & \text{if } x_v < 0 \\ \infty & \text{if } x_v = 0 \end{cases}
$$

Then, a complete QSS can be represented by a coupled DEVS constituted by atomic models M_2 and M_4 according to Figure 3.2. If the round–off errors are ignored, that coupled DEVS model can exactly simulate the QSS behavior.

Example 3.1. *QSS simulation of a second order system.*

Consider the second order system:

$$
\begin{aligned}
\dot{x_1}(t) &= x_2(t)\\
\dot{x_2}(t) &= 1 - x_1(t) - x_2(t)
\end{aligned}
\tag{3.8}
$$

with the initial condition

$$x_1(0) = 0,\ x_2(0) = 0 \tag{3.9}$$

Using a uniform quantum $Q_{k+1} - Q_k = \Delta Q = 0.05$ *and hysteresis width* $\epsilon = 0.05$ *in both state variables, the resulting quantized state system:*

$$\begin{array}{rcl} \dot{x_1}(t) & = & q_2(t) \\ \dot{x_2}(t) & = & 1 - q_1(t) - q_2(t) \end{array} \tag{3.10}$$

can be simulated by a coupled DEVS model, composed by two atomic models like M_4 *–correponding to the quantized integrators– and two atomic models like* M_2*, which calculate the static functions* $f_1(q_1, q_2) = q_2$ *and* $f_2(q_1, q_2) = 1 - q_1 - q_2$*. Let us call these subsystems* QI_1*,* QI_2*,* F_1 *and* F_2 *respectively.*

The coupling can be then expressed by the connections: $[(QI_1, 1), (F_1, 1)]$*,* $[(QI_1, 1), (F_2, 1)]$*,* $[(QI_2, 1), (F_1, 2)]$*,* $[(QI_2, 1), (F_2, 2)]$*,* $[(F_2, 1), (QI_2, 1)]$ *and* $[(F_1, 1), (QI_1, 1)]$*.*

Figure 3.4 represents the coupled system.

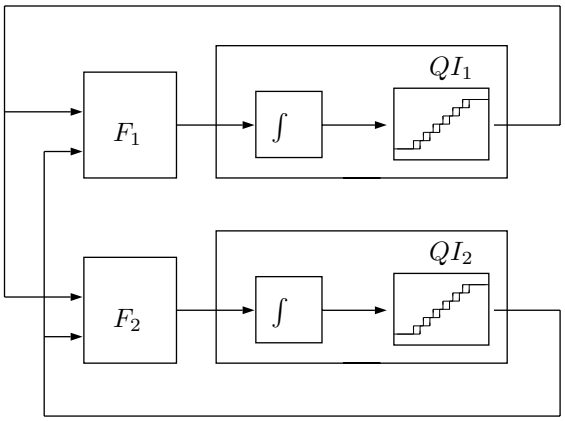

Figure 3.4: Block Diagram Representation of (3.10)

Observe that, due to the fact that function f_1 *does not depend on the variable* q_1*, the connection* $[(QI_1, 1), (F_1, 1)]$ *is not necessary. Moreover, taking into account that* $f_1(q_1, q_2) = q_2$ *the subsystem* F_1 *and the connections which invlove it can be replaced by a direct connection from* QI_2 *to* QI_1*. These simplifications can reduce considerably the computational cost of the implementation.*

The simulation results are shown in Figure 3.5. The first simulation was completed with 30 internal transitions at each quantized integrator, which gives a total of 60 steps. It can be seen in that figure the piecewise linear trajectories of $x_1(t)$ *and* $x_2(t)$ *as well as the piecewise constant trajectories of* $q_1(t)$ *and* $q_2(t)$*.*

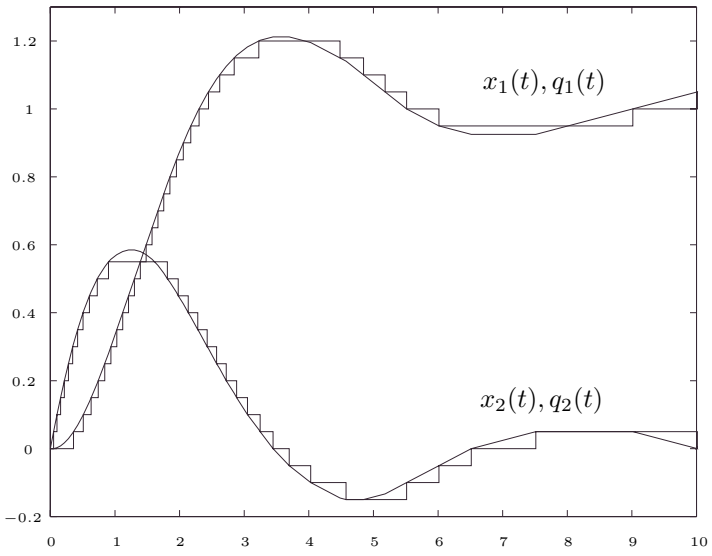

Figure 3.5: Trajectories in System (3.10)

The presence of the hysteresis can be easily observed when the slope of the state variables changes its sign. Near those points, there are different values of q for the same value of x.

The simplifications which were mentioned in the connections can be applied to general systems where some static functions do not depend on all the state variables. In that way, the QSS method can exploit the structural properties of the system to reduce the computational costs. When the system is *sparse* the QSS simulation results particularly efficient since each step involves calculations at very few integrators.

Discrete time algorithms can also make use of these sparsity properties. However, they require from specific techniques of matrix manipulation. In the QSS–method this is just an intrinsec property.

3.5 Input signals in the QSS–method

When the QSS–method was introduced, it was mentioned that it allows the simulation of time invariant systems with piecewise constant input signals. However, it was not said how these signals can be incorporated into the simulation model.

In the DEVS simulation model, each event represents a change in a piecewise constant trajectory. Then, the input trajectories must be incorporated as sequences of events. In the block diagram of Figure 3.2, the input signals $u(t)$

seem to come from the external world and it is not clear where the corresponding sequences of events should be generated.

In the context of DEVS simulation, all the events must come from an atomic DEVS model. Thus, it is evident that some new DEVS models that generates those sequences of events should be created and coupled with the rest of the system in order to simulate it.

Suppose that there is a piecewise constant input signal $u(t)$ which adopts values $v_1, v_2, \ldots, v_j, \ldots$ at times $t_1, t_2, \ldots, t_j, \ldots$ respectively. Then, a DEVS model which produces the events corresponding to this signal –i.e. a DEVS event generator– can be built as follows:

$$
\begin{aligned}
&M_5 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \phi \\
&Y = \mathbb{R} \times \mathbb{N} \\
&S = \mathbb{N} \times \mathbb{R}^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(j, \sigma) = (j+1, t_{j+1} - t_j) \\
&\lambda(s) = \lambda(j, \sigma) = (v_j, 1) \\
&ta(s) = ta(j, \sigma) = \sigma
\end{aligned}
$$

Notice that in this model the external transition function δ_{ext} is not defined since it cannot be called due to the fact that the system does not receive input events.

An interesting advantage of the QSS–method is that it deals with the input trajectory changes in an asynchronous way. The event indicating a change in the signal is always processed in the correct instant of time, producing instantaneous changes in the slopes of the state variable trajectories which are directly affected.

This is the intrinsic behavior of the method and it is obtained without modifying the DEVS models corresponding to the quantized integrators and static functions. Discrete time methods require an special treatment in order to perform a step in the exact instant in which an input change occurs. This issue will be treated later in a deeper way showing the advantages of the QSS–method in discontinuous systems.

Up to here, only piecewise constant input trajectories were considered. However, in many applications, the input signals have more general forms. Anyway, they can be approximated by piecewise constant trajectories with the addition of quantization functions and then, they can be represented by DEVS models. For instance, the following DEVS generates an event trajectory which follows the form of a sine function with angular frequency ω and amplitude A using a quantum $A\Delta u$.

$$
\begin{aligned}
&M_6 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \phi \\
&Y = \mathbb{R} \times \mathbb{N} \\
&S = \mathbb{R} \times \mathbb{R}^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(\tau, \sigma) = (\tilde{\tau}, \tilde{\sigma}) \\
&\lambda(s) = \lambda(\tau, \sigma) = (A\sin(\omega\tau), 1) \\
&ta(s) = ta(\tau, \sigma) = \sigma
\end{aligned}
$$

with

$$
\tilde{\sigma} = \begin{cases} \dfrac{\arcsin[\sin(\omega\tau) + \Delta u]}{\omega} - \tau & \text{if } (\sin(\omega\tau) + \Delta u \leq 1 \wedge \cos(\omega\tau) > 0) \\ & \quad \vee \sin(\omega\tau) - \Delta u < -1 \\ \dfrac{\text{sgn}(\tau)\pi - \arcsin[\sin(\omega\tau) - \Delta u]}{\omega} - \tau & \text{otherwise} \end{cases}
$$

and

$$
\tilde{\tau} = \begin{cases} \tau + \tilde{\sigma} & \text{if } \omega(\tau + \tilde{\sigma}) < \pi \\ \tau + \tilde{\sigma} - \dfrac{2\pi}{\omega} & \text{otherwise} \end{cases}
$$

The trajectory generated by this model with parameters $A = 2$, $\omega = 3$ and $A\Delta u = 0.2$ is shown in Figure 3.6.

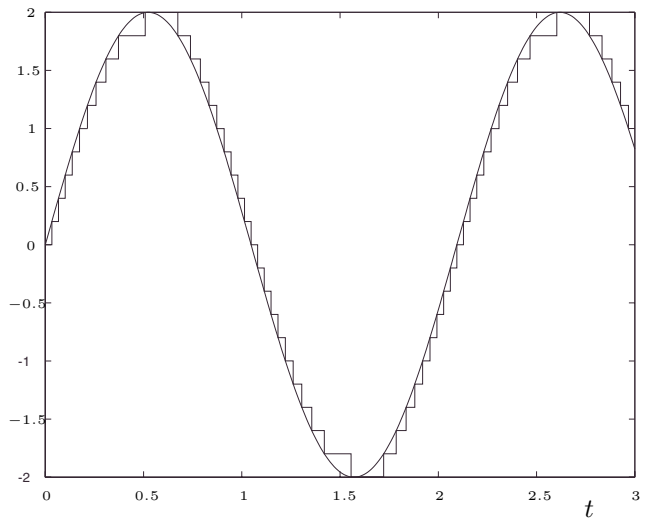

Figure 3.6: Piecewise constant sine trajectory

A piecewise constant trajectory can be also obtained using a constant time step. However, the previous approximation is better in the QSS–method since

the quantization in the values ensures that the difference between the continuous signal an the piecewise constant one is always less than the quantum. This fact can be easily noticed in Figure 3.6 whereas if a constant step approximation had been taken, the maximum error would have depended on the relationship between the time step and the signal frequency.

3.6 Startup and Output Interpolation

The QSS–method startup consists in giving appropriate initial conditions to the atomic DEVS models which perform the simulation.

The quantized integrator state can be written as $s = (x, d_x, k, \sigma)$ (see model M_4 in page 33) where x is the state variable, d_x is its derivative, k is the index corresponding to the quantized variable q_k and σ is the time advance.

It is clear that the variable representing the state should be initially chosen as $x = x_i(t_0)$ and j must be taken so that $Q_j \leq x_i(t_0) \leq Q_{j+1}$. With respect to d_x and σ, they could be calculated according to $f_i(q, u)$. However, there is a much simpler solution.

If we choose $\sigma = 0$, all the quantized integrators will perform internal transitions at the beginning of the simulation and then, the models associated to the static functions f_i will calculate the derivatives producing –instantaneously– output events. After that, the quantized integrators will receive the correct values of d_x and they will calculate the corresponding σ in the external transition.

However, this behavior will be only observed if the quantized integrators receive the input events –which come from the static functions– after they performed the internal transitions. The problem is that after the first quantized integrator performs the transition, not only the other quantized integrators but also some static models will have their σ equal to 0 (because they perform an external transition due to the quantized integrator output event).

Thus, it is necessary to establish priorities between the components in order to ensure that when some models schedule their next transition for the same instant of time, the quantized integrators are which perform it first. This can be easily done using the tie-breaking function *Select* mentioned in Section 2.3.

These considerations also solve the problem of the static model initial conditions. They can be arbitrarily chosen since the static models will receive input events coming from the quantized integrators at the beginning of the simulation and then their external transition functions will set the appropriate states.

Finally, the input signal generator models must start with their σ equal to 0 and the rest of the state must be chosen so that the first output event corresponds to the signal initial value.

A very important property of the QSS–method is related to the fact that the state trajectory has a very particular form. It was already mentioned in Section 3.3 that the state trajectories in a QSS are piecewise linear. Moreover, they are also continuous.

Hence, if the values adopted by a variable x_i at the event times are known, they can interpolated with segments in order to obtain the exact evolution of

(3.3) at any time. Moreover, to do that the only things which are needed are the values after the external transitions because in the internal transitions the slopes do not change.

Considering this, the problem of *output interpolation* has a straightforward solution in the QSS–method.

3.7 Quantization, Hysteresis and Errors

After the definition of the QSS–method in Section 3.2 the following remarks were focused in the DEVS implementation of Quantized State Systems. Taking into account the considerations of Sections 3.4 to 3.6, it becomes clear how to build and simulate the DEVS model corresponding to any QSS.

Then, given a continuous system like (3.2) and knowing which is the quantum and hysteresis to be used, the DEVS simulation can be easily performed.

However, it was not mentioned any word about the key issue of the method: the choice of the quantization and hysteresis to be applied at each state variable.

In classic discrete time methods, the simulation parameters (step size, error tolerance, etc.) are chosen according to stability and accuracy considerations.

Up to here, it was not said anything about these matters –stability, error, etc.– related to the QSS–method.

Since the goal is to simulate System (3.2), the accuracy of the simulation will be connected to the similarity between this system and (3.3).

Taking into account that the only difference between both systems is the presence of the quantization functions, it is natural to expect that the error depends on the size of the quantization intervals.

However, this assertion requires a deeper study about the effects of the quantization. All these theoretical issues will be left for the next chapter.

Anyway, we anticipate here that –under certain conditions– there is a linear relationship between the quantum and the error and this fact will provide the basic rule for the choice of the quantization and hysteresis.

Chapter 4

Theoretical Properties of QSS

In the previous chapter, it was claimed that the QSS–method was a general simulation method for ODE. Although it was shown that it can be applied to general time invariant ODEs, it was not proven any property telling that the QSS solutions are in some way *similar* to the exat ones.

The properties which are typically studied in numerial analysis are consistence, convergence, stability and error bound. Those properties allow to estimate under which conditions the resulting *approximate* solution is in fact a good *approximation* to the continuous trajectories. They also help in the choice of the parameters (step–size, error tolerance, etc.) according to the desired accuracy.

The local error –the error in one step– is usually obtained from a Taylor series expansion, while the global error –the error after several steps– is only theoretically estimated making use of Lipschitz conditions (see [18] for instance). The convergence property (i.e. the property for which the error goes to zero when the step size goes to zero) is then obtained as the limit case of the global error bound.

Since all the existing methods are *discrete time* ones, their stability properties are usually studied based on difference equations theory. The usual trick here is to find the relationship between the eigenvalues of the original differential equation and the roots of the discrete time one (this procedure is obviously limited to the analysis in linear systems[1]).

In the case of the QSS–method this last idea is completely useless. We already mentioned that the Quantized State Systems do not fit in the form of a difference equation. Similarly, the use of Taylor expansions does not lead to any result.

However, there is a much more powerful tool wich can be used here: the perturbation theory. Based on that, it will be possible not only to get similar

[1]There are other ways to study properties which can be applied to nonlinear systems. We only refer here to the usual tools of numerical analysis.

results to the classic ones, but also to calculate practical global error bounds and to establish stability conditions including nonlinear cases.

Making use of those stability and error bound conditions –and after some extra considerations– the problem which was open at the end of the previous chapter –i.e. the choice of the appropriate quantization and hysteresis width– will be finally solved.

4.1 QSS and Perturbation Theory

As it was mentioned, the usual tools for the analysis of numerical stability are based on the discrete time systems theory. The basic idea is to obtain the difference equation corresponding to a given method applied to a differential equation and then to relate the eigenvalues of both systems.

This idea, which is useful for discrete time methods, cannot be applied to the QSS–method because the resulting simulation model is a discrete event system which does not fit in a difference equation representation.

A first attempt to solve this problem would consist in looking for a discrete event systems theory which allows such kind of stability analysis. In fact, there is a nice mathematical theory based on the use of *max-plus algebra* which permits expressing some discrete event systems by difference equations in the context of that algebra [1]. This theory also arrives to stability results based on the study of eigenvalues and it is completely analogous to the discrete time systems theory. However, it can be only applied to systems described by Petri Nets and unfortunately, the QSS–method produces DEVS models which do not have Petri Net equivalents.

A different way to study the QSS dynamics is using the representation (3.2) for the original differential equation and (3.3) for its QSS approximation.

Let us define $\Delta x(t) \triangleq q(t) - x(t)$. Thus, the last equation can be written as

$$\dot{x}(t) = f(x(t) + \Delta x(t), u(t)) \tag{4.1}$$

and now, the simulation model (3.3) can be seen as a perturbed version of the original system (3.2).

The hysteretic quantization functions have a fundamental property. If two variables $q_i(t)$ and $x_i(t)$ are related by a quantization function like (3.1), then

$$Q_0 < x_i(t) < Q_r \Rightarrow |q_i(t) - x_i(t)| \leq \max(\Delta Q, \varepsilon) \tag{4.2}$$

where $\Delta Q \triangleq \max(Q_{j+1} - Q_j),\ 0 \leq j \leq r$, is the maximum quantum (it was mentioned in the previous chapter that the quantum is usually constant).

The property given by (4.2) implies that each component of the perturbation Δx is bounded by the corresponding hysteresis width and quantum size. Thus, the accuracy and stability analysis can be based on the effects of a bounded perturbation.

As it was said, the study of numerical stability is usually restricted to linear systems (their study in nonlinear systems is quite difficult). It will be seen

that this problem disappears from the QSS–method because the perturbation analysis can be easily applied to nonlinear systems. Moreover, when the method is used with a linear system it will be also possible to establish a global bound for the error and the problem of the approximation accuracy can be treated not only locally but also globally.

Despite these advantages, a new problem appears in the QSS–method. Let us illustrate it with the following example.

Example 4.1. *Oscillations in the QSS–method.*

Consider the first order system:

$$\dot{x}(t) = -x(t) + 9.5 \tag{4.3}$$

with the initial condition $x(0) = 0$.

The simulation using the QSS–method with quantum $\Delta Q = 1$ *and hysteresis width* $\varepsilon = 1$ *is shown in Figure 4.1*

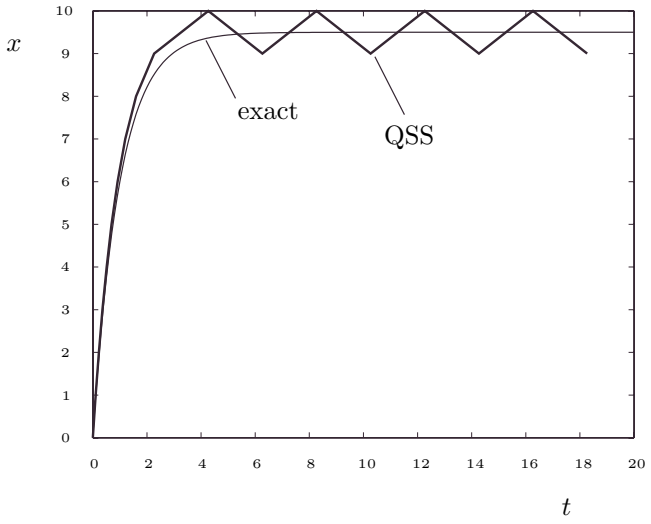

Figure 4.1: QSS simulation of (4.3)

Although the system (4.3) is asymptotically stable, the QSS simulation finishes into a limit cycle. The equilibrium point $\bar{x} = 9.5$ is no longer an equilibrium point in the resulting QSS and we cannot claim that the stability is conserved by the method.

However, the QSS solution never goes far away from the exact solution and it finishes with an oscillation near the equilibrium point. Taking into account that the goal was just simulating the system, this result is not bad.

The trajectory given by the QSS–method is called *ultimately bounded* [23]. In general, the QSS–method cannot assure stability according to the classic

definition. Thus, in the context of this method *stability* will refer in fact to ultimately boundedness of the solutions.

4.2 Convergence of QSS–method

The convergence property tells that the solutions of (4.1) go to the solutions of (3.2) when the maximum quantum ΔQ and the hysteresis width ε go to zero.

The importance of this property is in the fact that an arbitrarily small error can be achieved in the simulation when a sufficiently small quantization is used.

The following theorem give sufficient conditions for the convergence of the QSS–method.

Theorem 4.1. *Convergence of QSS–method.*

Consider System (3.2) *and its associated QSS* (3.3)*. Let* D *be the* non-saturation region *defined by*

$$D = \{x = (x_1, ..., x_n)/Q_{0_i} < x_i < Q_{r_i}\} \tag{4.4}$$

Assume that the input $u(t) \in D_u$ *being* D_u *a bounded region and suppose that function* $f(x, u)$ *is locally Lipschitz on* $D \times D_u$*. Let* $\phi(t)$ *be the solution of* (3.2) *from the initial condition* $x(0) = x_0$ *and let* $\phi_1(t)$ *be a solution of the related QSS* (3.3) *starting in the same initial condition* x_0*. Assume that* $\phi(t) \in D_1$ *where* $D_1 \subset D$ *(the continuous system solution is in the non-saturation region). Then,* $\phi_1(t) \to \phi(t)$ *when the quantization intervals go to* 0.[2]

Proof. Let S be $\mathbb{R}^n - D$ and let F be

$$F \triangleq \sup_{u \in D_u} (\sup_{x \in D} \|f(x, u)\|)$$

Let d be defined by

$$d \triangleq \inf_{x \in S} (\inf_{t \in [0, \infty]} \|\phi(t) - x\|) \tag{4.5}$$

Taking into account the assumptions on f and $\phi(t)$, a positive constant t_1 can be found satisfying

$$t_1 < \frac{d}{F} \tag{4.6}$$

It can be easily seen that during the interval $[0, t_1]$ the trajectory of $\phi_1(t)$ will remain inside D.

Using the perturbation notation (4.1) instead of (3.3) and taking into account that when $0 \leq t \leq t_1$ the trajectories are inside the non–saturation region, the fundamental property (4.2) stands for all the components and then $\Delta x(t)$ satisfies

$$\|\Delta x(t)\| \leq \Delta_x \quad (0 \leq t \leq t_1) \tag{4.7}$$

[2]Without modifying the saturation bounds (i.e. the number of quantization levels goes to ∞).

being Δ_x a constant defined by the quantization intervals. Let $t \in [0, t_1]$. It follows from (4.1), (3.2) and the fact that $\phi_1(0) = \phi(0)$ that:

$$\phi_1(t) - \phi(t) = \int_0^t (f(\phi_1(\tau) + \Delta x(t), u(\tau)) - f(\phi(\tau), u(\tau)))d\tau$$

Thus, appliyng the euclidean norm we obtain

$$\|\phi_1(t) - \phi(t)\| = \| \int_0^t (f(\phi_1(\tau) + \Delta x(\tau), u(\tau)) - f(\phi(\tau), u(\tau)))d\tau\|$$

and then,

$$\|\phi_1(t) - \phi(t)\| \leq \int_0^t \|f(\phi_1(\tau) + \Delta x(\tau), u(\tau)) - f(\phi(\tau), u(\tau))\|d\tau \tag{4.8}$$

Let M be the Lipschitz constant of function f on $D \times D_u$. Since the argument of the function f in (4.8) is inside that region we have

$$\begin{aligned}
\|\phi_1(t) - \phi(t)\| &\leq \int_0^t M\|\phi_1(\tau) + \Delta x(\tau) - \phi(\tau)\|d\tau \\
\Rightarrow \|\phi_1(t) - \phi(t)\| &\leq \int_0^t M(\|\phi_1(\tau) - \phi(\tau)\| + \|\Delta x(\tau)\|)d\tau \\
\Rightarrow \|\phi_1(t) - \phi(t)\| &\leq \int_0^t M(\|\phi_1(\tau) - \phi(\tau)\| + \Delta_x)d\tau \\
\Rightarrow \|\phi_1(t) - \phi(t)\| &\leq \int_0^t M\|\phi_1(\tau) - \phi(\tau)\|d\tau + Mt\Delta_x
\end{aligned}$$

The functions ϕ and ϕ_1 are continuous, as well as the term $Mt\Delta_x$. Since M is positive, it is possible to apply the *Gronwall–Bellman Inequality* [23], resulting in

$$\begin{aligned}
\|\phi_1(t) - \phi(t)\| &\leq Mt\Delta_x + \int_0^t M^2 s\Delta_x e^{\int_s^t M d\tau} ds \\
\Rightarrow \|\phi_1(t) - \phi(t)\| &\leq (e^{Mt} - 1)\Delta_x
\end{aligned}$$

Then, since M and t_1 do not depend on Δ_x, for $0 \leq t \leq t_1$ we have

$$\lim_{\Delta_x \to 0} \|\phi_1(t) - \phi(t)\| = 0 \tag{4.9}$$

From (4.5) we have

$$d \leq \inf_{x \in S}(\|\phi(t_1) - x\|) \tag{4.10}$$

Taking into account (4.6), (4.9) and (4.10), it is possible to find a sufficiently small quantization such that

$$t_1 < \frac{\inf_{x \in S}(\|\phi_1(t_1) - x\|)}{F}$$

This inequality implies that the solution $\phi_1(t)$ does not leave the region D during the interval $[t_1, 2t_1]$. Then, the validity of equations (4.7) to (4.9) holds for the interval $[0, 2t_1]$. Repeating that argument it results that (4.9) holds for all t. □

4.3 General Stability Properties of QSS

Although the convergence constitutes an important theoretical property, it does not give any quantitative information about the relationship between the quantum and the error and it does not establish any condition for the stability domain.

The main result of this section –Theorem 4.2– follows this last goal by giving sufficient conditions to find the quantization which *conserves*[3] the stability properties of the original continuous system.

For the stability analysis an autonomous system will be considered (which can eventually represent a non–autonomous system with a constant input trajectory):

$$\dot{x}(t) = f(x(t)) \tag{4.11}$$

whose corresponding QSS is given by

$$\dot{x}(t) = f(q(t)) \tag{4.12}$$

where $q(t)$ and $x(t)$ are componentwise related by hysteretic quantization functions.

A first property of this approximation is given by the following lemma

Lemma 4.1. *QSS Equilibrium Points Consider the autonomous continuous system* (4.11) *and its associated Quantized State System* (4.12)*.*

Then, the point $x = \bar{x}$ is an equilibrium point of (4.11) *if and only if the point $q = \bar{x}$ is an equilibrium point of* (4.12)*.*

The proof of this lemma is straightforward and, as it was already mentioned, it also holds for system with constant inputs.

Lemma 4.1 implies that the quantized variables of the QSS have the same possible values in the equilibrium than the state variables of the original system. However, it does not imply that state variables in the quantized system will have such values or that the quantized variables can reach them.

Anyway, taking into account that the difference between $q(t)$ and $x(t)$ is bounded in the QSS the value of the state variables in an eventual equilibrium situation will be near the right one.

Equation (4.12) can be rewritten using the perturbation notation:

$$\dot{x}(t) = f(x(t) + \Delta x(t)) \tag{4.13}$$

with $\Delta x(t) \triangleq q(t) - x(t)$.

Now, the theorem which relates the stability of systems (4.11) and (4.12) can be introduced.

As before, this theorem assumes that the system is autonomous and it studies the trajectories around an equilibrium point in the origin but it can be easily extended to systems with constant inputs and with different equilibrium points.

[3] As it was mentioned before, the term stability refers in fact to ultimately boundedness of solutions

Theorem 4.2. *Stability of QSS.*

Consider a system as the one defined in (4.11) *that has an equilibrium point in the origin with the function f being continuously differentiable.*

Assume that it is also possible to find a Lyapunov function $V(x)$, which is continuous in an open region D including the origin, and has a negative definite time derivative along the trajectories of (4.11)*. Let $D_1 \subset D$ be a region limited by a level surface of function V. Then, given an arbitrary open region $D_2 \subset D_1$ also limited by a level surface of V it is always possible to find a quantization so that any trajectory of the resulting associated QSS starting into D_1 converges to the interior of D_2.*

Proof. Let D_3 be the region defined by $D_3 \triangleq D_1 - D_2$. Since $\dot{V}(x)$ is negative definite, it exists a positive number s such that:

$$\dot{V}(x) < -s, \forall x \in D_3 \tag{4.14}$$

Define:

$$\alpha(x, \Delta x) \triangleq \nabla V(x)^T \cdot f(x + \Delta x) \tag{4.15}$$

This is a continuous function in Δx since it is the scalar product of a constant vector and a continuous function. It is also verified that:

$$\alpha(x, 0) = \dot{V}(x) \tag{4.16}$$

Now, consider the following function

$$\alpha_M(\Delta x) \triangleq \sup_{x \in D_3} (\alpha(x, \Delta x)) \tag{4.17}$$

It can be easily seen that this function is continuous and verifies

$$\alpha_M(0) < -s \tag{4.18}$$

Thus, given a constant s_1 ($0 < s_1 < s$), a positive number ρ can be found satisfying

$$\alpha_M(\Delta x) < -s_1 < 0 \tag{4.19}$$

if

$$\|\Delta x\| < \rho \tag{4.20}$$

The condition given by (4.20) can be satisfied with the choice of an adequate quantization taking into account (4.2).

Let $\phi(t)$ be a solution of equation (4.13) starting from the initial condition $\phi(t = 0) = x_0 \in D_3$. Consider that the quantization was done in order to satisfy the condition given by (4.20). From (4.13) and (4.15) it follows that:

$$\alpha(\phi, \Delta x) = \nabla V(\phi)^T \cdot \dot{\phi}$$

Using (4.17) and (4.19) in the equation above, it can be seen that:

$$\frac{\partial V}{\partial x}(\phi) \cdot \dot{\phi} < -s_1 \tag{4.21}$$

This condition will be satisfied at least during certain time while $\phi(t)$ remains inside D_3 (the existence of this time is guaranteed by the continuity of $\phi(t)$).

After integrating both sides of the Inequality (4.21), we have:

$$\begin{aligned} \int_0^t \frac{\partial V}{\partial x}(\phi) \cdot \dot{\phi} \cdot dt &< \int_0^t -s_1 \cdot dt \\ V(\phi(t)) - V(\phi(0)) &< -s_1 \cdot t \\ V(\phi(t)) &< V(x_0) - s_1 \cdot t \end{aligned}$$

This implies that V evaluated along the QSS solution is bounded by a strictly decreasing function while that solution remains inside D_3. Since the value $V(x_0)$ is smaller than the value that V takes in the bound of D_1, it is clear that the trajectory will never leave D_1.

Let V_1 be the value that V takes in the bound of region D_2. Then, it can be easily seen that the trajectory will reach the region D_2 in a finite time t_1 with:

$$t_1 < \frac{V(x_0) - V_1}{s_1}$$

completing the proof. □

A first remark is that the proof requires the choice of the quantization intervals satisfying (4.20). In order to achieve that, it is necessary to leave the saturation region outside region D_3.

Then, considering the same constant quantum ΔQ and hysteresis window ε for all the quantized variables, each component of Δx is less than $\max(\Delta Q, \varepsilon)$. Thus, $\|\Delta x\|$ is less than $\sqrt{n}$ times that value, being n the dimension of the state space. Thus, the condition given by (4.20) can be achieved by taking:

$$\max(\Delta Q, \varepsilon) < \frac{\rho}{\sqrt{n}}$$

Instead of ensuring stability, Theorem 4.2 shows that the QSS–method can be implemented achieving a given final error. As it was mentioned before, the QSS–method only can guarantee ultimately boundedness of solutions. This fact can be easily understood taking into account that the perturbation term $\Delta x(t)$ does not dissapear when the time goes to ∞. These kind of perturbations are known as *nonvanishing perturbations* [23].

When the Lyapunov function V is also known, function α can be evaluated to obtain ρ. In that way, this result also shows the way of doing the quantization in order to obtain a final error bounded to some arbitrary value (given by the choice of region D_2).

In spite of the theoretical importance of these results, the relationship between quantization, stability and error bound cannot be easily deduced from Theorem 4.2.

On one hand, the choice of the quantization requires the knowledge of a Lyapunov function for the continuous system. There are converse theorems

which ensure the existence of those functions in asymptotically stable systems [23]. However, the way to obtain those functions is sometimes very difficult if not impossible. Then, the result is not so useful from a practical point of view.

On the other hand, only a bound for the final error was obtained. Up to here, there is no information about the relationship between the quantum and the error bound at a given instant of time.

The main reason of these problems is the fact that we were working with general nonlinear systems. It is necessary to deal with more particular cases in order to get more practical and stronger results.

Like in all existing numerical methods, the most interesting properties of the QSS–method result from the study of LTI systems.

Taking into account what was already done, a first attempt to study the QSS properties in LTI system would consist in taking the Lyapunov–based theorem and modifying it using the particular properties of linear systems.

However, that kind of study is a problem of ultimate bound estimation and the use of Lyapunov–based approaches usually leads to very conservative results. The reason of this can be found in the fact that the structure of the system and perturbation are lost when the Lyapunov analysis is performed.

Thus, before studying the properties of the QSS–method in LTI systems some new tools should be developed in order to analyze the behavior of those systems in presence of *nonvanishing perturbations*.

The next section is dedicated to the particular Lyapunov analysis and then the results are compared with the new methodology introduced in Section 4.5. After that, these results are applied to the study of the QSS properties.

4.4 LTI Perturbed Systems: Lyapunov Approach

We shall consider the LTI nominal system

$$\dot{x}(t) = f(t, x) = Ax(t) + Bu(t) \tag{4.22}$$

where $A \in \mathbb{R}^{n\times n}$ is a Hurwitz matrix, $u \in \mathbb{R}^m$ and $B \in \mathbb{R}^{n\times m}$.

As it was seen in the previous sections, the QSS–method introduces bounded perturbations in the state variables. Similarly, when continuous input signals are approximated by their quantized versions, the effect can be represented by the presence of bounded perturbations in the input variables.

In that way, the goal of this section is to study the effects of bounded state and input perturbations in System (4.22).

It was already mentioned that one of the most important consequences carried by the presence of nonvanishing perturbations is that the asymptotic stability dissapears, leaving its place to the ultimately boundedness of solutions.

The main problem is then to estimate a quantitative relationship between the perturbation bounds and the resulting ultimate bound. In our particular problem –the QSS–method– those bounds are respectively the quantum and the final error.

Taking into account the problem studied –to find the ultimate bound of a perturbed system– it must be assumed that the non–perturbed system (4.22) is asymptotically stable and it has a constant input.

Due to the linearity and without loss of generality it can be supossed that $u(t) = 0$ and then, in presence of general input and state perturbations, System (4.22) can be written as

$$\dot{x}(t) = A(x(t) + \Delta x(t)) + B\Delta u(t) \tag{4.23}$$

It will be assumed that the perturbation components satisfy

$$|\Delta x_i(t)| \leq \Delta x_{max_i} \quad 1 \leq i \leq n \tag{4.24}$$

and

$$|\Delta u_j(t)| \leq \Delta u_{max_j} \quad 1 \leq j \leq m \tag{4.25}$$

where Δx_{max_i} and Δu_{max_j} are non-negative constants (which in the QSS–method represent the quantum).

In order to simplify the next equations, a new notation will be introduced:

The symbol $|\cdot|$ will indicate the componentwise module of a matrix or vector. If T is a matrix with components $T_{1,1}, \ldots, T_{n,m}$, then $|T|$ will be a new matrix of the same size than T with components $|T_{1,1}|, \ldots, |T_{n,m}|$.

For vectors having the same dimension, the vector inequality $x \leq y$ implies that $x_i \leq y_i$ for every component of x and y.

According to these definitions, the following property will be satisfied:

$$|T \cdot x| \leq |T| \cdot |x|$$

Using the notation defined above, the inequalities (4.24) and (4.25) can be rewritten as

$$|\Delta x(t)| \leq \Delta x_{max} \tag{4.26}$$

and

$$|\Delta u(t)| \leq \Delta u_{max} \tag{4.27}$$

Using this new notation, the problem is to find a ultimate bound for System (4.23) where the perturbations satisfy (4.26) and (4.27).

In Section 4.3 a similar study for general nonlinear systems was performed making use of a Lyapunov's based technique.

The classic way of studying this problem in LTI systems is just to follow that technique improving it with the particular properties resulting from the system linearity.

In order to justify the use of a new approach, the ultimate bound of system (4.23) will be obtained following that Lyapunov's analysis and then it will be compared with the new one in the next section.

The study will be done for norm 2 and norm ∞. Although a bound in norm ∞ can be obtained from the norm 2 –using the fact that the first one is always less or equal than the second one– this bound could result even more conservative.

Let $U(x) = x^T P x$ where $P = P^T > 0$ satisfies

$$A^T P + PA = -Q \tag{4.28}$$

with $Q = Q^T > 0$. Then

$$\begin{aligned} \dot{U}(x) &= \dot{x}^T P x + x^T P \dot{x} \\ &= [A(x + \Delta x) + B\Delta u]^T P x + x^T P [A(x + \Delta x) + B\Delta u] \end{aligned}$$

and

$$\dot{U}(x) = -x^T Q x + 2x^T PA\Delta x + 2x^T PB\Delta u \tag{4.29}$$

It is known that

$$\begin{aligned} 2x^T PA\Delta x &\leq 2\|x\| \cdot \|PA\| \cdot \|\Delta x\| \\ &\leq 2\|x\| \cdot \|PA\| \cdot \|\Delta x_{max}\| \end{aligned} \tag{4.30}$$

and similarly

$$\begin{aligned} 2x^T PB\Delta u &\leq 2\|x\| \cdot \|PB\| \cdot \|\Delta u\| \\ &\leq 2\|x\| \cdot \|PB\| \cdot \|\Delta u_{max}\| \end{aligned} \tag{4.31}$$

Up to here, the analysis was the same to both norms. Now, the study should continue using particular properties of each norm.

In norm 2, we know that:

$$x^T Q x \geq \|x\|_2^2 \lambda_{min}(Q) \tag{4.32}$$

Then, if

$$\|x\|_2 \geq \rho \quad \triangleq \quad \frac{2}{\lambda_{min}(Q)} (\|PA\|_2 \cdot \|\Delta x_{max}\|_2 + \|PB\|_2 \cdot \|\Delta u_{max}\|_2) \tag{4.33}$$

it results from (4.30) and (4.31) that

$$x^T Q x \geq \|x\|_2^2 \lambda_{min}(Q) \geq 2x^T PA\Delta x + 2x^T PB\Delta u$$

and from (4.29) we have

$$\dot{U}(x) < 0$$

Let

$$c \triangleq \max_{\|x\|_2 = \rho} U(x) = \rho^2 \lambda_{max}(P) \tag{4.34}$$

Then, it can be ensured that the trajectories will finish inside the level surface given by

$$U(x) = c = \rho^2 \lambda_{max}(P)$$

and the ultimate bound will be μ_2 so that

$$\min_{\|x\|_2 = \mu_2} U(x) = \mu_2^2 \lambda_{min}(P) = c$$

this is

$$\mu_2 = \sqrt{\frac{c}{\lambda_{min}(P)}}$$

using (4.34) and (4.33) in the last equation, we have

$$\mu_2 = \sqrt{\frac{\lambda_{max}(P)}{\lambda_{min}(P)}} \cdot \frac{2}{\lambda_{min}(Q)} \cdot (\|PA\|_2 \cdot \|\Delta x_{max}\|_2 + \|PB\|_2 \cdot \|\Delta u_{max}\|_2) \quad (4.35)$$

The analysis in norm ∞ is almost identical to the norm 2. The only difference is that the lower bound given by (4.32) does not stand for norm ∞. To obtain a similar inequality, the following result will be used.

Lemma 4.2. *Constrained Quadratic [44].*
Let $U(x) = x^t Q x$, where $Q = Q^T > 0$, and let C be a row vector in $\mathbb{R}^n$ and r be a nonzero scalar. Then the minimum of $U(\cdot)$ along a hyperplane $\{x|Cx = r\}$ is given by: $r^2/CQ^{-1}C^T$.

Using this lemma, the minimum of the quadratic function when the component $x_i = r$ is $r^2/e_i Q^{-1} e_i^T = r^2/(Q^{-1})_{i,i}$. Thus, it results that

$$\min_{\|x\|_\infty = r} x^T Q x = \frac{r^2}{\max_i (Q^{-1})_{i,i}}$$

and Equation (4.32) can be replaced by

$$x^T Q x \geq \frac{\|x\|_\infty^2}{\max_i (Q^{-1})_{i,i}} \quad (4.36)$$

Then, following a similar idea to the analysis in norm 2, the bound obtained is

$$\mu_\infty \quad = \quad 2b_q \sqrt{b_p \|P\|_\infty} \cdot (\|PA\|_\infty \|\Delta x_{max}\|_\infty + \|PB\|_\infty \|\Delta u_{max}\|_\infty) \quad (4.37)$$

where

$$b_q \triangleq \max_{1 \leq i \leq n} (Q^{-1})_{i,i}$$

and

$$b_p \triangleq \max_{1 \leq i \leq n} (P^{-1})_{i,i}$$

4.5 LTI Perturbed Systems: Non Conservative Approach

In order to obtain a less conservative result, it is necessary to exploit the structure of the system. Then, a completely different idea to the Lyapunov analysis has to be followed.

To make use of the structure, the decoupled system will be studied (Lemma 4.3 and Corollary 4.1). Then, the results will be brought back to the original system in Theorem 4.3.

This way of working will allow to use the geometrical properties of the system and perturbations. As it will be seen later, this idea results in a less conservative estimation of the ultimate bound.

Lemma 4.3. *Scalar Perturbed System*

Consider the following first order equation with complex coefficient

$$\dot{x} = a(x + \Delta x) + B\Delta u \tag{4.38}$$

where a, x, $\Delta x \in \mathbb{C}$, $\Delta u \in \mathbb{C}^k$ *and* $B \in \mathbb{C}^{1\times k}$. *Assume also that* $\mathbb{R}e(a) < 0$, $|\Delta x| \leq \Delta x_{max}$ *and* $|\Delta u| \leq \Delta u_{max}$.

Let $x(t)$ *be a solution of* (4.38) *from the initial condition* $x(t_0) = 0$. *Then, for all* $t \geq t_0$ *it results that*

$$|x(t)| \leq \left|\frac{a}{\mathbb{R}e(a)}\right| \Delta x_{max} + \left|\frac{B}{\mathbb{R}e(a)}\right| \Delta u_{max}$$

Proof. Let $x = \rho \cdot e^{j\theta}$ with ρ, $\theta \in \mathbb{R}$. With this, Equation (4.38) becomes

$$\dot{\rho} \cdot e^{j\theta} + j\rho \cdot e^{j\theta} \cdot \dot{\theta} = a(\rho \cdot e^{j\theta} + \Delta x) + B\Delta u$$

Then, we have

$$\dot{\rho} + j\rho \cdot \dot{\theta} = a(\rho + \Delta x \cdot e^{-j\theta}) + B\Delta u \cdot e^{-j\theta}$$

Taking the real part of the equation above it results that

$$\dot{\rho} = \mathbb{R}\mathrm{e}(a)\rho + \mathbb{R}\mathrm{e}(a\Delta x \cdot e^{-j\theta}) + \mathbb{R}\mathrm{e}(B\Delta u \cdot e^{-j\theta})$$

and

$$\dot{\rho} \leq \mathbb{R}\mathrm{e}(a)\rho + |a|\Delta x_{max} + |B|\Delta u_{max}$$

Thus, when

$$\rho = |x(t)| = \frac{|a|\Delta x_{max} + |B|\Delta u_{max}}{|\mathbb{R}\mathrm{e}(a)|}$$

it results that $\dot{\rho} \leq 0$ and $|x(t)|$ cannot become greater than the given bound. □

Applying Lemma 4.3 to each component of a decoupled system, the following corollary is obtained

Corollary 4.1. *Decoupled Perturbed System.*

Consider System (4.23) *where* x, $\Delta x \in \mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$, $\Delta u \in \mathbb{C}^k$ *and* $B \in \mathbb{C}^{n\times k}$. *Assume further than* A *is a diagonal matrix with* $\mathbb{R}e(A_{i,i}) < 0$ *and consider the inequalities* (4.26) *and* (4.27). *Let* $x(t)$ *be a solution of* (4.23) *from* $x(t_0) = 0$. *Then, for all* $t \geq t_0$ *we have*

$$|x(t)| \leq |\mathbb{R}e(A)^{-1}A|\Delta x_{max} + |\mathbb{R}e(A)^{-1}B|\Delta u_{max}$$

With this corollary, the following theorem can be derived

Theorem 4.3. *Bound of Trajectories Starting from the Origin*

Consider System (4.23) *where* x, $\Delta x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $\Delta u \in \mathbb{R}^k$ *and* $B \in \mathbb{R}^{n\times k}$. *Assume that* A *is a diagonalizable Hurwitz matrix and suposse that the perturbation terms satisfy* (4.26) *and* (4.27). *Let* $x(t)$ *be a solution of the system starting from* $x(t_0) = 0$. *Then, for all* $t \geq t_0$

$$|x(t)| \leq |V| \cdot (|\mathbb{R}e(\Lambda)^{-1}\Lambda||V^{-1}|\Delta x_{max} + |\mathbb{R}e(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \tag{4.39}$$

where Λ *is a diagonal eigenvalues matrix of* A *and* V *is an associated matrix of eigenvectors, that is*

$$V^{-1}AV = \Lambda \tag{4.40}$$

Proof. Let $x = Vz$. It results from (4.23) that

$$V\dot{z} = A(Vz + \Delta x) + B\Delta u$$

Then,

$$\dot{z} = \Lambda(z + V^{-1}\Delta x) + V^{-1}B\Delta u \tag{4.41}$$

Taking into account the restrictions in $|\Delta x|$ and $|\Delta u|$, we have

$$|V^{-1}\Delta x| \leq |V^{-1}|\Delta x_{max}$$

and

$$|V^{-1}B\Delta u| \leq |V^{-1}B|\Delta u_{max}$$

Since Λ is a diagonal matrix with $\mathbb{R}\mathrm{e}(\Lambda_{i,i}) < 0$ (since A is Hurwitz) and taking into account the last inequalities, System (4.41) satisfies the hypothesis of Corollary 4.1.

Then, for all $t \geq t_0$ we can ensure that

$$|z(t)| \leq |\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta x_{max} + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u_{max}$$

and finally, we have

$$\begin{aligned} |x(t)| &= |Vz(t)| \leq |V||z(t)| \leq \\ &\leq |V|(|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta x_{max} + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \end{aligned}$$

which completes the proof. □

Theorem 4.3 calculates a bound for the trajectories of a LTI perturbed system assuming that the trajectories start from the origin.

Now, based on this last result, Theorem 4.4 gives an estimation of the ultimate bound by components and in terms of a norm p for a LTI perturbed system with arbitrary initial conditions.

Theorem 4.4. *Ultimate Bound of LTI Perturbed Systems.*

System (4.23)*, under the hypothesis of Theorem 4.3 is globally ultimately bounded with ultimate bound*

$$\mu = \left\| |V| \cdot (|\mathbb{R}e(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta x_{max} + |\mathbb{R}e(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \right\| \quad (4.42)$$

Moreover, it exists a finite time $t_1 = t_1(c, x_0)$ so that for each positive constant c the solutions satisfy

$$|x(t)| \leq (1+c)|V| \cdot (|\mathbb{R}e(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta x_{max} + |\mathbb{R}e(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \quad (4.43)$$

for all $t \geq t_1$ and for an arbitrary initial condition x_0.

Proof. Let $x(t)$ be a solution of (4.23) starting from an arbitrary initial condition $x(0) = x_0$, and let $\tilde{x}(t)$ be a solution of the nominal system

$$\dot{\tilde{x}} = A\tilde{x} \quad (4.44)$$

starting from the same initial condition.

Let $e(t) = x(t) - \tilde{x}(t)$. Then, it results that $e(0) = 0$ and $e(t)$ satisfies Equation (4.23), which implies that it satisfies the hypothesis of Theorem 4.3. Then,

$$|e(t)| \leq |V| \cdot (|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta x_{max} + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \quad (4.45)$$

Since A is Hurwitz, the nominal system (4.44) is exponencially stable. Then, for each positive constant c a finite time t_1 exists so that for all $t > t_1$ we have

$$|\tilde{x}(t)| \leq c|V| \cdot (|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta x_{max} + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \quad (4.46)$$

Then, for $t > t_1$,

$$\begin{aligned} x(t) &= e(t) + \tilde{x}(t) \\ |x(t)| &\leq |e(t)| + |\tilde{x}(t)| \end{aligned}$$

and replacing with (4.45) and (4.46) we arrive to (4.43), which completes the proof. □

The results, Theorem 4.4 and Eq.(4.42), provide an alternative to the bounds given by (4.35) and (4.37).

The following examples illustrate the advantages of the new approach.

Example 4.2. *Ultimate Bound of a Stiff Perturbed System*

Consider the perturbed system

$$\dot{x}(t) = \begin{bmatrix} 0 & 100 \\ -100 & -10001 \end{bmatrix} \cdot [x(t) + \Delta x(t)]$$

where the perturbation components are bounded by

$$|\Delta x_1(t)| \leq 0.01; \;\; |\Delta x_2(t)| \leq 0.0001$$

The Lyapunov-based formula (4.35) *is minimized taking*[4]

$$Q = \begin{bmatrix} 1 & -1.4631 \\ -1.4631 & 197.568 \end{bmatrix}$$

obtaining the ultimate bound $\mu_2 = 20.1861$. *Similarly, the minimum of* (4.37) *is obtained for*

$$Q = \begin{bmatrix} 1 & -1.448 \\ -1.448 & 196.124 \end{bmatrix}$$

which gives $\mu_\infty = 14.45$.

For the same example, the new approach –Theorem 4.4 and Equation (4.42)*– gives* $\tilde{\mu}_2 = 0.0100085$ *(bound in norm 2) and* $\tilde{\mu}_\infty = 0.01$. *The estimation is about 2000 times less conservative in norm 2 and 1400 times in norm* ∞.

Moreover, Theorem 4.4 concludes that according to (4.43)*, after certain time* t_1 *we will have*

$$|x_1(t)| < (1+c)0.01; \; |x_2(t)| < (1+c)0.0003$$

for any given positive constant c. If the goal were knowing about the second component, this result is even much better than the obtained with (4.42).

The calculations were made using the eigenvectors and eigenvalues matrices obtained with function 'eig' of Matlab:

$$V = \begin{bmatrix} 1 & -0.01 \\ -0.01 & 1 \end{bmatrix}; \; \Lambda = \begin{bmatrix} -1 & 0 \\ 0 & -10000 \end{bmatrix}$$

The reason for the poor performance of the Lyapunov analysis is, in part, due to the big difference between the system eigenvalues (the system is stiff). Thus, the eigenvalues of matrix P are also very different which implies that the level surfaces of $U(x)$ are very flat ellipses. Hence, the radius ρ which defines the ball where the Lyapunov function derivative is negative differs significantly from the radius μ_2, which defines the ball cointaining all the level surfaces that passes by the ball with radius ρ (see Figure 4.2).

However, the bad performance is also due to the lost of the problem structure when the Lyapunov function is used. In this way, the analysis cannot obtain any profit from the fact that the perturbation term acts in each direction with different amplitude.

Example 4.3. *A Stiff System with Input Perturbations.*

Consider now the same system than before with an input perturbation.

$$\dot{x}(t) = \begin{bmatrix} 0 & 100 \\ -100 & -10001 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot d(t)$$

[4]The minimum was obtained with Nelder–Mead simplex method (function 'fminsearch' of Matlab).

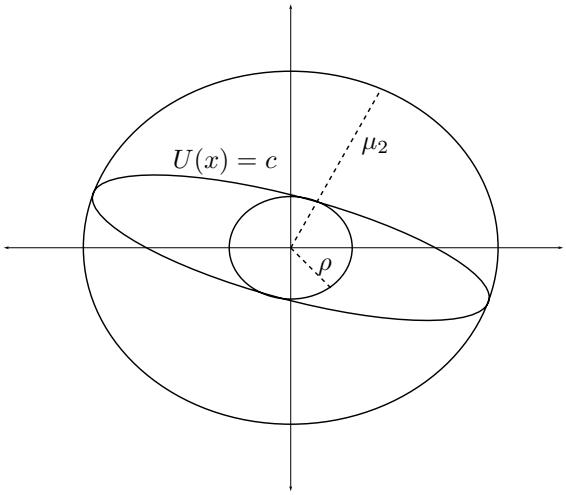

Figure 4.2: Lyapunov Bound in Norm 2

where $|d(t)| \leq 1$.

The minimum of the Lyapunov bound given by (4.35) *is obtained with*

$$Q = \begin{bmatrix} 1 & -2.2822 \\ -2.2822 & 266.581 \end{bmatrix}$$

which results in an estimation of $\mu_2 = 26.7327$.

Similarly (4.37) *has a minimum for*

$$Q = \begin{bmatrix} 1 & -1.7681 \\ -1.7681 & 195.484 \end{bmatrix}$$

giving $\mu_\infty = 21.593$.

Now, the use of (4.42) *gives* $\tilde{\mu}_2 = 1.00015$ *and* $\tilde{\mu}_\infty = 1.0001$. *Then, the results with the new approach are more than 20 times better.*

Theorem 4.4 also ensures that, according to (4.43), *after some time we will have*

$$|x_1(t)| < 1.0001; \;\; |x_2(t)| < (1 + c)0.01$$

and the estimation of the bound in the second component becomes 2000 times better than the given by the Lapunov analysis.

The advantages of the new approach in stiff systems is quite clear taking into account what was illustrated by Figure 4.2.

The last example of this section will illustrate the use of the new methodology in a very simple non-stiff second order system with complex eigenvalues in presence of input perturbations. This is not the case of Figure 4.2 and then the advantage of the method here is only due to the conservation of the problem structure.

Example 4.4. *A Non–Stiff Perturbed System*
Consider the system

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot d(t)$$

where $|d(t)| \leq 1$.

The bounds obtained via Lyapunov's analysis are $\mu_2 = 5.1098$ *and* $\mu_\infty = 3.7535$ *obtained with the optimal matrices* Q*:*

$$Q = \begin{bmatrix} 1 & -0.0421 \\ -0.0421 & 1.1177 \end{bmatrix}$$

and

$$Q = \begin{bmatrix} 1 & -0.475 \\ -0.475 & 1.0703 \end{bmatrix}$$

respectively.

On the other hand, the bounds obtained with the new approach are $\tilde{\mu}_2 = 3.266$ *and* $\tilde{\mu}_\infty = 2.3094$*, which still are less conservative.*

The eigenvectors and eigenvalues matrices used here were also the ones obtained with Matlab:

$$V = \begin{bmatrix} 0.612 - j0.354 & 0.612 + j0.354 \\ j0.707 & j0.707 \end{bmatrix}$$

and

$$\Lambda = \begin{bmatrix} -0.5 + j0.866 & 0 \\ 0 & -0.5 - j0.866 \end{bmatrix}$$

Although this new approach was developed in order to study the QSS–method properties, it can be also applied in a much more general context.

In fact, these results will be reused in the second order approximation and in the asynchronous control methodology. But they can be also applied to many other problems which require ultimate bound estimation in presence of bounded nonvanishing perturbations.

Nonvanishing perturbations can also represent effects of unknown disturbance signals (see [42] for instance), unmodeled dynamics [43] and errors in networked control systems [63]. Then, the new methodology may also provide a useful tool in applications corresponding to all these cases.

4.6 QSS–method in LTI Systems

The stability of numerical methods is always studied in linear systems. As it was already exaplined, the idea is to relate the eigenvalues of the original continuous systems with the eigenvalues of the resulting difference equation. In that way, the analysis arrives to conditions –usually about the step size– so that the stable eigenvalues of the continuous system –i.e. in the left half–plane– are mapped into stable eigenvalues –i.e. inside the unit circle– in the difference equation.

As it is already known, this idea cannot be applied to the QSS–method because here there is no difference equation to study.

In Section 4.3 general conditions were found in order to ensure ultimately boundedness of the QSS solutions. There, a Lyapunov function was required for the analysis and then the result was not really practical. Although that general result has very important theoretical consequences, it is necessary to go to the particular case of LTI systems in order to obtain more practical conditions.

Let us then consider the following LTI system:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \tag{4.47}$$

with the same definitions made in (4.22)

The use of the QSS–method transforms it into:

$$\dot{x}(t) = A \cdot q(t) + B \cdot u_q(t) \tag{4.48}$$

where $q(t)$ and $x(t)$ are related componentwise by hysteretic quantization functions. Here, it is considered that $u_q(t)$ is the quantized version of $u(t)$ (in case it is not piecewise constant).

Then, defining $\Delta x(t) \triangleq q(t) - x(t)$ and $\Delta u(t) \triangleq u_q(t) - u(t)$ Eq.(4.48) can be rewritten as

$$\dot{x}(t) = A(x(t) + \Delta x(t)) + B(u(t) + \Delta u(t)) \tag{4.49}$$

where each component of the perturbation terms Δx and Δq is bounded by the corresponding quantum (according to Eq.(4.2)). Thus, provided that the trajectories do not reach the saturation bounds, we have that $\Delta x(t)$ and $\Delta u(t)$ satisfy (4.26) and (4.27) respectively.

Based on this last remark and making use of the results of the previous section, the following theorem gives the fundamental error bound property of the QSS–method in LTI systems.

Theorem 4.5. *Global Error bound of the QSS–Method*

Consider the LTI system (4.47) *where matrix A is Hurwitz and diagonalizable. Let $\phi(t)$ be a solution of that system and let $\phi_1(t)$ be a solution calculated by the QSS–method from the same initial condition using a quantization so that* (4.26) *and* (4.27) *are satisfied.*

Then, the error $e(t) \triangleq \phi_1(t) - \phi(t)$ is always bounded by:

$$|e(t)| \leq |V| \cdot (|\mathbb{R}e(\Lambda)^{-1}\Lambda||V^{-1}|\Delta x_{max} + |\mathbb{R}e(\Lambda)^{-1}V^{-1}B|\Delta u_{max}) \tag{4.50}$$

where Λ is a diagonal eigenvalues matrix of A and V is an associated matrix of eigenvectors (i.e., they verify Eq.(4.40)*).*

Proof. From the definition of $e(t)$, using (4.47) and (4.49) it results that

$$\dot{e}(t) = A \cdot e(t) + B \cdot \Delta u(t) \tag{4.51}$$

Since both solutions start from the same initial condition we have that $e(t_0) = 0$ and then, System (4.51) verifies all the hypothesis of Theorem 4.3. Thus, Inequality (4.39) takes the form of (4.50) completing the proof. □

Theorem 4.5 gives the relationship between quantization and error. Inequality (4.50) can be also seen as a closed formula to choose the quantization according to the desired error.

A remarkable fact is that the formula does not depend neither on the input trajectory nor on the initial condition. Besides being an amazing theoretical property, it gives to the method very important practical advantages.

It can be easily seen that the error bound is proportional to the quantum and, for any quantum adopted, the error is always bounded. Then, the method is always *stable*. In that way the QSS–method –using only explicit formulas– shares some properties with implicit discrete time methods.

It is also important to notice that Inequality (4.50) is an analytical expression for the maximum global error . As we already mentioned, discrete time methods lack of similar formulas.

Finally, an interesting fact is that in explicit discrete time algorithms the stability depends on a relationship between the location of the eigenvalues –the system *speed*– and the step size.

Here, if a constant input is considered it results that $\Delta u_m ax = 0$ and then the error is only connected to geometrical properties of the system: the eigenvectors and the angles of the eigenvalues (note that the product $|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda|$ is a diagonal matrix where each entry on the diagonal is the inverse of the cosine of an eigenvalue angle). This is coherent with the fact that the method discretizes the state variables.

4.7 Quantum and Hysteresis Choice

The QSS–method requires the choice of an adequate quantum and hysteresis size. Although we mentioned that the error is always bounded –at least in LTI systems– an appropriate quantization must be chosen in order to obtain a decent simulation result.

It was also mentioned that Inequality (4.50) can be used for the quantization design. Given a desired error bound, it is not difficult to find appropriate values for Δx_{max} and Δu_{max} so the inequality is satisfied. Let us illustrate this in a simple example.

Example 4.5. *Choosing the Quantum.*
Consider the system:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 - x_2 + u(t) \end{aligned} \tag{4.52}$$

A set of matrices of eigenvalues and eigenvectors (calculated with Matlab) are

$$\Lambda = \begin{bmatrix} -0.5 + 0.866i & 0 \\ 0 & -0.5 - 0.866i \end{bmatrix}$$

and

$$V = \begin{bmatrix} 0.6124 - 0.3536i & 0.6124 + 0.3536i \\ 0.7071i & -0.7071i \end{bmatrix}$$

Then,

$$T \triangleq |V||\mathbb{R}e(\Lambda)^{-1}\Lambda||V^{-1}| = \begin{bmatrix} 2.3094 & 2.3094 \\ 2.3094 & 2.3094 \end{bmatrix} \tag{4.53}$$

Let us consider that the goal is to simulate (4.52) *for an arbitrary initial condition and input trajectory with an error less or equal than* 0.1 *in each variable. Then, a quantum*

$$\Delta Q = \begin{bmatrix} 0.05/2.3094 \\ 0.05/2.3094 \end{bmatrix} = \begin{bmatrix} 0.0217 \\ 0.0217 \end{bmatrix} \tag{4.54}$$

is enough small to ensure that the error cannot be bigger than the given bound.

Although Inequality (4.50) gives an idea about the relationship between the quantum, the hysteresis and the error bound, sometimes it might result quite conservative (despite the fact that it is less conservative than the Lyapunov–based analysis).

In fact, using a quantum and hysteresis equal to 0.05 in each variable of System (4.52) and taking $u(t) = 1$, the simulation shown in Figure 3.5 (page 35) is obtained. The predicted error bound is 0.23094 in each variable. However, the maximum error in that simulation results quite smaller than this bound.

Up to here, the hysteresis width was chosen equal to the quantum size. However, no reason was given for this election.

On one hand, Inequality (4.2) tells that the perturbation at each variable is bounded by the maximum between the quantum and the hysteresis width. Then, a hysteresis width bigger than the quantum should not be used because in that way the error would be increased.

On the other hand, the minimum time between succesive internal transitions in a quantized integrator –given by Eq.(3.7) in page 31– depends on the minimum between the quantum and hysteresis width. Thus, the use of a hysteresis smaller than the quantum would increase the number of calculations.

The following example illustrates these ideas.

Example 4.6. *Choosing the Hysteresis.*

Consider system the first order system (4.3).

Figure 4.3 shows simulation results with quantum $\Delta q = 1$ *and hysteresis widths* $\varepsilon = 1$, $\varepsilon = 0.6$ *and* $\varepsilon = 0.1$. *In all the cases the maximum error is bounded by the same value. In theory, the bound is equal to* 1 *but in the simulations we can observe that the maximum error is always* 0.5.

However, the final oscillation frequency increases as well as the hysteresis width becomes smaller. In fact, that frequency can be calculated as

$$f = \frac{1}{2\varepsilon}$$

Then, it is clear that when the hysteresis is chosen to be equal to the quantum, the frequency is reduced without increasing the error. The result of reducing the oscillation frequency is a reduction in the number of steps performed by the algorithm and a consequent reduction of the computational costs.

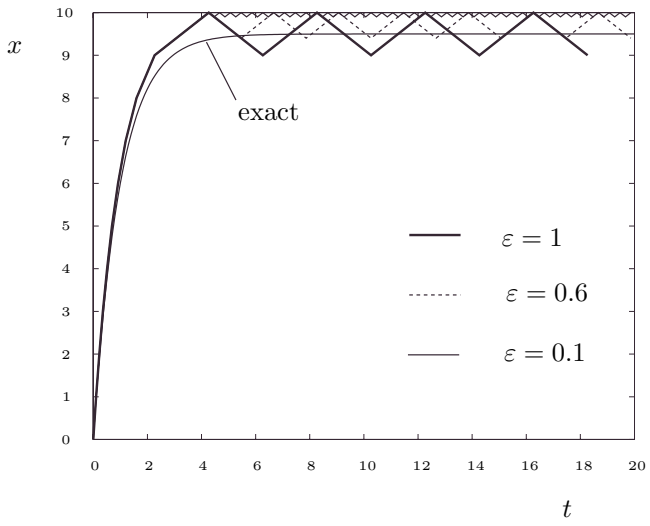

Figure 4.3: Simulation of (4.3) with different hysteresis values

4.8 Limitations of the QSS–method

Along this chapter, several theoretical properties were studied concluding about the good qualities of the QSS approximation. Particularly, the results of Section 4.6 shows that the QSS–method has very strong properties which are at the level of variable step and implicit discrete time methods.

However, there is a problem which was not mentioned up to here. On one hand, Inequality (4.50) says that the error bound is proportional to the quantum. On the other hand, the model M_4 in page 33 tells that the time advance σ is inversely proportional to the quantum. Thus, the number of step will result approximately proportional to the required accuracy.

It means that any attempt to increase the accuracy in 10 times, will result in an increment in about 10 times of the number of calculations.

This fact can be easily understood taking into account that QSS only performs a first order approximation and then a good accuracy cannot be obtained without increasing significantly the number of calculations.

This limitation will be –partially– solved in the next chapter with a second order approximation.

There are, of course, more limitation. The most important is probably the one related to stiff systems. These cases will be studied at the end (we anticipate now that there are some encouraging results but the problem is still open).

The last problem is related –again– to the choice of the quantization. Despite Inequality (4.50) could be useful, nobody wants to calculate eigenvalues and eigenvectors before simulating. The only good solution to this is the use of adaptive quantization, which was not still developed.

Chapter 5

Second Order QSS

The previous chapter finished with a brief analysis on the limitations of the QSS–method. The first problem mentioned there was related to the fact that the method performs only a first order approximation.

As a result, there is an approximate linear relationship between the inverse of the global error bound and the number of steps. Thus, any attempt to reduce the error results in a proportional increase of the computational costs and it is quite expensive to obtain accurate results.

In this chapter, a second order approximation will be developed which permits arriving to more accurate results without increasing considerably the number of calculations.

In QSS, the quantizers approximate continuous trajectories by piecewise constant ones. Thus, the information about the higher order derivatives of the state variables is lost. Here, the main idea is to use piecewise linear trajectories instead of piecewise constant in order to make use of the state second order derivatives.

This new numerical approximation will be stated so that it allows to ensure that it satisfy most of the practical and theoretical properties showed by the QSS–method.

On one hand, the coupling scheme of the resulting DEVS atomic models will be the same than before, exploiting sparsity and making use of their asynchronous features. On the other hand, the approximatting simulation model will be still seen as a perturbed version of the original system. In that way, similar theoretical properties to the shown for the QSS–method will be obtained.

5.1 QSS2–Method

The basic idea of the second order method is the use of *first order quantizers* replacing the simple hysteretic quantizers which were placed to define QSS.

A hysteretic quantizer whose quantum and hysteresis width are equal to each other –it was already mentioned that this is the best choice– can be seen

as a system that produces a piecewise constant output trajectory which only changes when the difference between that output and the input reaches certain threshold (i.e. the quantum).

Following this idea, a first order quantizer is defined as a system which produces a piecewise linear output trajectory having discontinuities only when its difference with the input reaches the quantum. This behavior is illustrated in Figure 5.1.

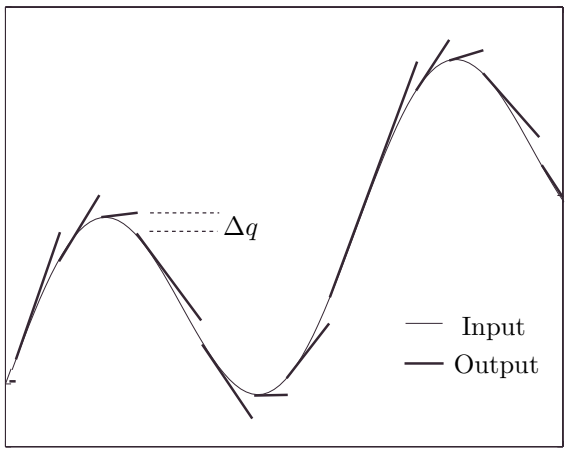

Figure 5.1: Input and Output trajectories in a *First Order* quantizer

This idea can be formalized as follows:

Definition 5.1. *First Order Quantization Function.*

Two trajectories $x_i(t)$ and $q_i(t)$ are related by a first–order quantization function if they satisfy:

$$q_i(t) = \begin{cases} x_i(t) & \text{if} \quad t = t_0 \vee |q_i(t^-) - x_i(t^-)| = \Delta q \\ q_i(t_j) + m_j(t - t_j) & \text{otherwise} \end{cases} \tag{5.1}$$

with the sequence $t_0, \ldots, t_j, \ldots$ defined as

$$t_{j+1} = \min(t | t > t_j \wedge |x_i(t_j) + m_j(t - t_j) - x_i(t)| = \Delta Q)$$

and the slopes are

$$m_0 = 0,\ m_j = \dot{x}_i(t_j^-) \quad j = 1, \ldots, k, \ldots \tag{5.2}$$

The constant ΔQ will be called again the *quantum* and variables $x_i(t)$ and $q_i(t)$ satisfy

$$|q_i(t) - x_i(t)| \leq \Delta Q_i \quad \forall t \tag{5.3}$$

Notice that this inequality is just a particular case of (4.2) with a constant quantum equal to the hysteresis width.

Taking into account that the difference between the new approximation and the QSS–method is the use of first order quantizers instead of hysteretic quantization functions, the new method can be defined as follows:

Definition 5.2. *QSS2–Method.*
Given a time invariant state equation system like (3.2)*, the QSS2–method approximates it by a system like* (3.3) *where the components of* $q(t)$ *and* $x(t)$ *are related componentwise by first–order quantization functions.*

The resulting system (3.3) will be called *Second Order Quantized State System* (or QSS2 for short). Figure 5.2 shows the corresponding block diagram.

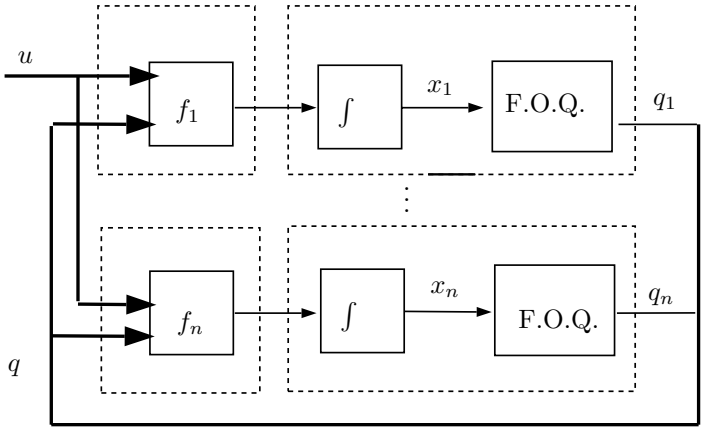

Figure 5.2: Block Diagram Representation of a QSS2

As before, the components of $q(t)$ will be called *quantized variables*.

In QSS it was necessary to add hysteresis in order to ensure that the quantized variable trajectories become piecewise constant avoiding illegitimacy. In the QSS2 definition the hysteresis seems to be absent. However, hysteresis is implicitly present in the definition of first–order quantization function. Indeed, the absolute value in Equation (5.1) expresses that hysteretic behavior.

5.2 Trajectories in QSS2

It is quite obvious that the output of a first order quantization function is a piecewise linear trajectory. In fact, it was defined in order to obtain such kind of trajectories. Thus, it can be expected that the quantized variables of a QSS2 are piecewise linear (this fact will be formaly proven later on).

In QSS not only the quantized variables but also the state derivatives were piecewise constant. In that way, it could be affirmed that the state variables had piecewise linear trajectories and these feactures were used to build the DEVS model.

However, all that kind of particular trajectories in QSS2 cannot be found. Although the quantized variables are piecewise linear, it does not mean that the state derivatives are the same even if the inputs have piecewise linear trajectories. The reason is that a nonlinear function applied to a piecewise linear trajectory gives a trajectory which is not necessarily piecewise linear.

The only case in which it can be claimed that the state derivative trajectories are piecewise linear is in LTI systems and then, in that case, the state variables have piecewise parabolic trajectories.

The following theorems prove all these facts.

Theorem 5.1. *Quantized Trajectories in QSS2*

Given the QSS2 defined in (3.3) *with f continuous and bounded in any bounded domain and $u(t)$ being bounded, the trajectories of $q(t)$ are piecewise linear while they remain inside a bounded region.*

Proof. Let $q_i(t)$ and $x_i(t)$ denote the trajectory of the i-th component of $q(t)$ and $x(t)$ respectively. Since $x_i(t)$ and $q_i(t)$ are related by a first order quantization function, the last trajectory can be written as

$$q_i(t) = \begin{cases} x_i(t) & \text{if} \quad t = t_0 \vee |q(t^-) - x(t^-)| = \Delta q \\ q_i(t_j) + m_j(t - t_j) & \text{otherwise} \end{cases} \tag{5.4}$$

with the sequence $t_0, \ldots, t_j, \ldots$ defined according to

$$t_{j+1} = \min(t | t > t_j \wedge |x_i(t_j) + m_j(t - t_j) - x_i(t)| = \Delta q_i) \tag{5.5}$$

and where, according to (5.2), we have

$$m_0 = 0, \; m_j = \dot{x_i}(t_j^-) \quad j = 1, \ldots, k, \ldots \tag{5.6}$$

Although Equation (5.4) implies that $q_i(t)$ is formed by sections of lines, in order to assure that it is piecewise linear it is also necessary to prove that the sequence $t_0, \ldots, t_j, \ldots$, does not contain an infinite number of components in a finite interval of time.

Since it was assumed that $q(t)$ is bounded over some interval of time $[t_0, t_f]$, and taking into account the hypothesis made about f and the fact that $u(t)$ is bounded, we have

$$|f_i(q(t), u(t))| \leq F_i \quad t_0 \leq t \leq t_f \tag{5.7}$$

From (5.6) we have

$$|m_j| \leq |f_i(q(t), u(t))| \leq F_i \tag{5.8}$$

Thus, the slope results always bounded by a constant F_i. From (5.5) we have

$$\begin{aligned} \Delta q_i &= |x_i(t_j) + m_j(t_{j+1} - t_j) - x_i(t_{j+1})| \\ &\leq \frac{|x_i(t_{j+1}) - x_i(t_j)|}{t_{j+1} - t_j}(t_{j+1} - t_j) + |m_j|(t_{j+1} - t_j) \\ &\leq |m_{j+1}|(t_{j+1} - t_j) + |m_j|(t_{j+1} - t_j) \\ &\leq 2F_i(t_{j+1} - t_j) \end{aligned}$$

Finally, it results that

$$t_{j+1} - t_j \geq \frac{\Delta q_i}{2F_i}$$

This inequality implies that the sequence $t_0, \ldots, t_j, \ldots$, has a minimum time between successive components and it is impossible to have an infinite number of components in a finite interval of time. As a consequence of this, the trajectories of $q(t)$ are piecewise linear. □

Theorem 5.2. *State Derivative Trajectories in QSS2.*

In a QSS2 related to a LTI system with bounded piecewise linear inputs, the trajectories of the derivatives of the state variables are piecewise linear when the quantized variables remain in a bounded region.

Proof. In LTI systems we have $f = Ax + Bu$ and then, the hypothesis of continuity and boundedness formulated in Theorem 5.1 are satisfied. Taking into account the assumptions on the input and quantized variable trajectories it can be easily seen that the mentioned theorem holds and then, the quantized variables have piecewise linear trajectories.

A LTI system can be expressed by Eq.(4.47) (page 59) and its associated QSS2 takes the form of (4.48).

Since $q(t)$ and $u(t)$ have piecewise linear trajectories, it is clear that the trajectories of $\dot{x}(t)$ are also piecewise linear □

A corollary of this theorem is that in the case of QSS2 related to LTI systems, the state variables have piecewise parabolic trajectories, as it was mentioned before.

The piecewise constant and piecewise linear trajectories of QSS allowed to find its DEVS representation. In a similar way, the piecewise linear and piecewise parabolic trajectories will allow to find a DEVS model that exactly represents the behaviour of a QSS2 associated to a LTI system.

However, taking into account that the trajectory forms are not conserved in nonlinear systems, it will be only possible to simulate exactly the QSS2 approximations of LTI systems.

Anyway, as it will be seen in the next section, the QSS2–method can be also applied to general nonlinear systems but in that case the simulation results will not coincide exactly with the solutions of (3.3).

5.3 DEVS Representation of QSS2

The DEVS model of QSS built in Section 3.4 was based on the use of events representing the changes in the values of the piecewise constant variables.

Here, in a similar way, the changes in the slopes and values of the piecewise linear trajectories will be represented by events carrying the new values and slopes of the corresponding variables.

At this point, Giambiasi's work must be mentioned again, since his definition of GDEVS [17] gives a way of representing piecewise polynomial trajectories by segments of events.

We shall start focusing on LTI systems where –provided that the input trajectories are piecewise linear– the state derivatives result piecewise linear and then the state variables have continuous piecewise parabolic trajectories.

Using these facts, the same procedure used in Section 3.4 to build the DEVS model of a QSS can be followed. There, the main idea was splitting the model into quantized integrators and static functions.

In spite of the similarity between QSS and QSS2, the atomic models in the latter approximation will result quite different since they should calculate and take into account not only the values but also the slopes of the trajectories. Moreover, the events will carry both, values and slopes.

As before, the quantized integrators in QSS2 will be formed by an integrator and a first–order quantizer. They will be called *second–order quantized integrators* . The reason for this name is that they calculate the state trajectories using their first and second derivatives (i.e. the state derivative values and slopes).

A DEVS model which can represent the behavior of a second–order quantized integrator is the following one

$$
\begin{aligned}
&M_7 = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where:}\\
&X = \mathbb{R}^2 \times \mathbb{N}\\
&S = \mathbb{R}^5 \times \mathbb{R}^+\\
&Y = \mathbb{R}^2 \times \mathbb{N}\\
&\delta_{\text{int}}(u, m_u, x, q, m_q, \sigma) = (u + m_u \cdot \sigma, m_u, \tilde{q}, \tilde{q}, u + m_u \cdot \sigma, \sigma_1)\\
&\delta_{\text{ext}}(u, m_u, x, q, m_q, \sigma, e, v, m_v, p) = (v, m_v, \tilde{x}, q + m_q \cdot e, m_q, \sigma_2)\\
&\lambda(u, m_u, x, q, m_q, \sigma) = (x + u \cdot \sigma + \frac{m_u}{2}\sigma^2, u + m_u \cdot \sigma, 1)\\
&ta(u, m_u, x, q, m_q, \sigma) = \sigma
\end{aligned}
$$

where

$$
\tilde{q} = x + u \cdot \sigma + \frac{m_u}{2}\sigma^2; \;\; \tilde{x} = x + u \cdot e + \frac{m_u}{2}e^2
$$

$$
\sigma_1 = \begin{cases} \sqrt{\frac{2\Delta q}{m_u}} & \text{if} \quad m_u \neq 0 \\ \infty & \text{otherwise} \end{cases} \tag{5.9}
$$

and σ_2 can be calculated as the least positive solution of

$$
|x + u \cdot e + \frac{m_u}{2}e^2 + v \cdot \sigma_2 + \frac{m_v}{2}\sigma_2^2 - (q + m_q \cdot e + m_q\sigma_2)| = \Delta q \tag{5.10}
$$

It can be easily seen that model M_7 gives an exact representation of a second–order quantized integrator with piecewise linear input trajectories.

Equations (5.9) and (5.10) calculate the time advance, that is, the time in which the difference between the piecewise parabolic state trajectory ($x(t)$) and the piecewise linear quantized trajectory ($q(t)$) reaches the quantum (Δq).

It is clear that in a QSS2 simulation the integrators should be represented with models like M_7 instead of M_4. To represent the static functions in the QSS–method, the model M_2 was used. However, this last model does not take into account the slopes. Then, the representation of static functions in QSS2 requires using a different DEVS model.

Before trying to build the new DEVS model for the static functions it is it is necessary to take into account the possible presence of a nonlinear relationship.

Each component f_j of a static function $f(q,u)$ will receive the piecewise linear trajectories of the quantized and input variables.

Let us define $z \triangleq [q^T, u^T]^T$. Each component of z has a piecewise linear trajectory:

$$z_i(t) = z_i(t_k) + m_{z_i}(t_k) \cdot (t - t_k)$$

Then, the static function output can be written as

$$\dot{x}_j(t) = f_j(z(t)) = f_j(z_1(t_k) + m_{z_1}(t_k) \cdot (t - t_k), \ldots, z_l(t_k) + m_{z_l}(t_k) \cdot (t - t_k))$$

where $l \triangleq n + m$ is the number of components of $z(t)$.

Defining $m_z \triangleq [m_{z_1}, \ldots, m_{z_l}]^T$, the last equation can be rewritten as

$$\dot{x}_j(t) = f_j(z(t)) = f_j(z(t_k) + m_z(t_k) \cdot (t - t_k))$$

which can be expanded in Taylor series as follows

$$\dot{x}_j(t) = f_j(z(t)) = f_j(z(t_k)) + \left[\frac{\partial f_j}{\partial z}(z(t_k))\right]^T \cdot m_z(t_k) \cdot (t - t_k) + \ldots \quad (5.11)$$

Then, a piecewise linear approximation of the output can be obtained by taking the two first terms of (5.11).

This idea requires an estimation of the partial derivatives of function f_j, which must be recalculated in all the changes of the piecewise linear trajectories.

Taking into account these considerations, a DEVS model which can be associated to a generic static function $f_j(z) = f_j(z_1, \ldots, z_l)$ taking into account input and output values and slopes can be written according to

$$\begin{aligned}
&M_8 = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where:}\\
&X = \mathbb{R}^2 \times \mathbb{N}\\
&S = \mathbb{R}^{3l} \times \mathbb{R}^+\\
&Y = \mathbb{R}^2 \times \mathbb{N}\\
&\delta_{\text{int}}(z, m_z, c, \sigma) = (z, m_z, c, \infty)\\
&\delta_{\text{ext}}(z, m_z, c, \sigma, e, v, m_v, p) = (\tilde{z}, \tilde{m}_z, \tilde{c}, 0)\\
&\lambda(z, m_z, c, \sigma) = (f_j(z), m_f, 1)\\
&ta(z, m_z, c, \sigma) = \sigma
\end{aligned}$$

where $z = (z_1, \ldots, z_l)$ and $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_l)$ are the input values.

Similarly, $m_z = (m_{z_1}, \ldots, m_{z_l})$ and $\tilde{m}_z = (\tilde{m}_{z_1}, \ldots, \tilde{m}_{z_l})$ represent the input slopes.

The coefficients $c = (c_1, \ldots, c_l)$ and $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_l)$ estimate the partial derivatives $\frac{\partial f_j}{\partial z_i}$ which are used to calculate the output slope according to

$$m_f = \sum_{i=1}^{n} c_i m_{z_i}$$

After the external transition function, the components of $\tilde{z}$, $\tilde{m}_z$ and $\tilde{c}$ are calculated according to

$$\tilde{z}_i = \begin{cases} v & \text{if} \quad p = i \\ z_i + m_{z_i} e & \text{otherwise} \end{cases}$$

$$\tilde{m}_{z_i} = \begin{cases} m_v & \text{if} \quad p = i \\ m_{z_i} & \text{otherwise} \end{cases}$$

$$\tilde{c}_i = \begin{cases} \dfrac{f_j(z + m_z e) - f_j(\tilde{z})}{z_i + m_{z_i} e - \tilde{z}_i} & \text{if} \quad p = i \wedge z_i + m_{z_i} e - \tilde{z}_i \neq 0 \\ c_i & \text{otherwise} \end{cases} \tag{5.12}$$

If the function $f(z)$ is linear, this DEVS model exactly represents its behavior when the components of $z(t)$ are piecewise linear.

In the nonlinear case, the DEVS model approximates the Taylor expansion (5.11).

Then, through the coupling DEVS models like M_7 and M_8 as it is shown in Figure 5.2, the QSS2–method can be used to simulate any time invariant ODE system.

5.4 Properties of the QSS2–Method

It was already mentioned that the QSS2–method shares some properties with QSS. The reason can be easily understood. Two variables, $x_i(t)$ and $q_i(t)$ related by a first–order quantization function satisfy (5.3).

This inequality is just a particular case of (4.2) with a constant quantum equal to the hysteresis width. The QSS properties were deduced using this inequality –which implies that the method only introduces a bounded perturbation– and the representation of (4.1). Taking into account that this representation is also correct for QSS2, the conclusion is that the QSS2–method satisfies the same convergence, stability and error bound properties than QSS.

However, in nonlinear systems the QSS2 definition and what the DEVS model simulates do not coincide. Thus, the analysis only ensures that those properties hold in the simulation of LTI systems. Although there are many reasons which allow to conjecture that convergence and stability are still verified in the simulation of nonlinear systems, there is not yet a formal proof of this.

When it comes to the error bound, Inequality (4.50) is verified by the QSS2–method since they were deduced for LTI systems.

Besides the theoretical properties deduced from the perturbation analysis, it was seen that the QSS–method also exhibits some practical advantages related

to the incorporation of input signals and the exploitation of sparsity. Let us see then what happens with these practical issues in the QSS2–method.

The sparsity exploitation is straightforward. The simulation structure is conserved from QSS and then each transition only involves calculation at the integrators and static functions directly connected to the integrator which performed it.

When it comes to the input signals, now the situation is better than before. Here, it can be ensured that not only the inputs are processed as soon as they change but also it is possible to exactly represent piecewise linear instead of just piecewise constant trajectories.

The following example illustrates these advantages:

Example 5.1. *A Transmission Line Simulation*

The circuit of Figure 5.3 represents an RLC transmission line. This model can be used to study the performance of integrated circuits transmitting data at a very fast rate. Although the length of the wires is only of a few centimeters, the high frequency of the signal produces that the delays introduced by the wires cannot be ignored and the transmission line theory must be applied.

The transmission line models are described as systems of partial differential equations. However, they can be approximated by lumped models *where the distributed effects of capacity, inductance and resistance are represented by a cascade of single capacitors, inductors and resistors, as Figure 5.3 shows. In*

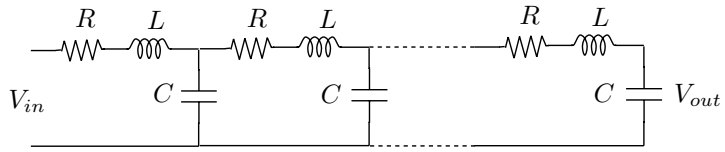

Figure 5.3: RLC Transmission Line

order to constitute a good approximation, the RLC model must be formed by several sections. As a consequence of this, it results in a sparse high order ordinary differential equation.

In [22] an example composed by five sections of RLC circuits is introduced. The resistance, inductance and capacitance values used there corresponds to real parameters. The model obtained is a tenth order linear system, where the evolution matrix (A) can be written as follows:

$$\boldsymbol{A} = \begin{bmatrix} -R/L & -1/L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/C & 0 & -1/C & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/L & -R/L & -1/L & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/C & 0 & -1/C & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/C & 0 \end{bmatrix}$$

A typical input trajectory in these digital systems is a trapezoidal wave representing the "0" and "1" levels as well as the rising and falling times. Since a trapezoidal wave is a piecewise linear trajectory, we can generate it exactly using the following DEVS model

$$
\begin{aligned}
&M_9 = (X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta), \text{ where:}\\
&X = \phi\\
&S = \mathbb{N} \times \mathbb{R}^+\\
&Y = \mathbb{R}^2 \times \mathbb{N}\\
&\delta_{int}(k, \sigma) = (\tilde{k}, t_{\tilde{k}})\\
&\lambda(k, \sigma) = (u_{\tilde{k}}, mu_{\tilde{k}}, 1)\\
&ta(k, \sigma) = \sigma
\end{aligned}
$$

where $\tilde{k} = (k + 1 \mod 4)$ *is the next cycle index which has 4 phases: The low level (index 0), the rising phase (1), the high level (2) and the falling phase (3). The duration of each phase is given by* t_k.

During the low level the output is u_0 *and at the high level phase it is* u_2. *Then, the slopes* mu_0 *and* mu_2 *are zero. During the rising time we have* $u_1 = u_0$ *and the slope is* $mu_1 = (u_2 - u_0)/t_1$. *Similarly, during the falling time we have* $u_3 = u_2$ *and* $mu_3 = (u_0 - u_2)/t_3$.

Then, the DEVS generator representing the input trajectory produces only 4 events in each cycle. This is an important advantage, since the presence of the input wave only adds a few extra calculations. Moreover, since the representation is exact, it does not introduce any error, i.e. we can estimate the error bound using Inequality (4.50) *with* $\Delta u_{max} = 0$.

The simulation was performed using parameters $R = 80\Omega$, $C = 0.2pF$ *and* $L = 20nH$. *(These parameters correspond to a transmission line of one centimeter divided in five sections, where the resistance, capacitance and inductance are* $400\Omega/cm$, $1pF/cm$ *and* $100nH/cm$ *respectively)*

The trapezoidal input has the rising and falling times $t_1 = t_3 = 10ps$, *while the low and high times are* $t_0 = t_2 = 1ns$. *The low and high levels are* $0V$ *and* $2.5V$ *respectively.*

The quantization adopted was $\Delta v = 4mV$ *in the state variables representing voltages and* $\Delta i = 10\mu A$ *in the state variables representing currents. That quantization, according to* (4.50) *ensures that the maximum error is smaller than* $250mV$ *in the variable* V_{out}.

The input and output trajectories are shown in Figure 5.4. The simulation took a total of 2536 *steps (between* 198 *and* 319 *internal transitions at each integrator) to obtain the first* $3.2ns$ *of the system trajectories.*

The experiment was repeated using a quantization 100 *times smaller, which ensures having a maximum error in* V_{out} *of* $2.5mV$. *This new simulation was performed with a total of* 26883 *internal transitions.*

The comparision of the results of both experiments (Figure 5.5) shows that the difference between the trajectories never becomes greater than $14.5mV$ *which implies that the error in the first simulation was not greater than* $17mV$. *In this*

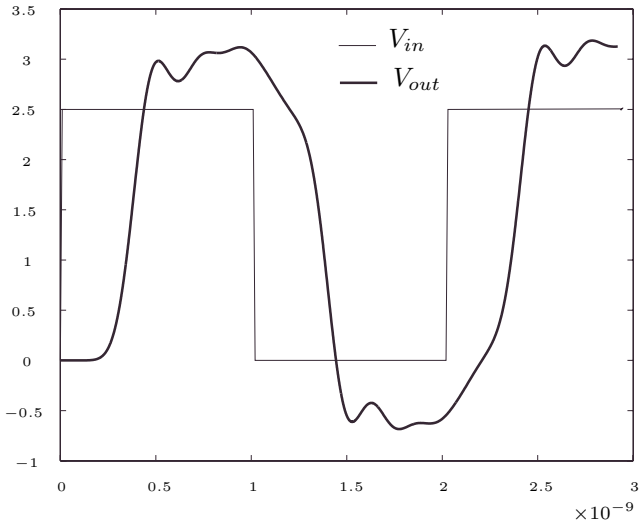

Figure 5.4: QSS2 simulation results in a RLC transmission line

case, the theoretical prediction of the error (the bound was $250mV$*) was quite conservative (this can be easily understood taking into account that the theoretical bound of the error holds for any input trajectory and for any initial condition).*

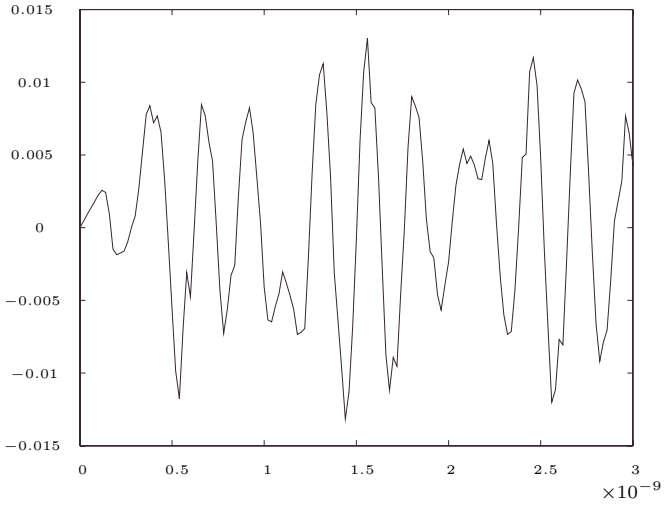

Figure 5.5: Difference in V_{out} using two different quanta

Although the number of steps (2536) seems to be quite big, it is important to take into account that each step only involves calculations at three integrators (the integrator that performs the internal transition and the two integrators di-

rectly connected to its output). This is due to the particular form of the matrix A (which is sparse).

It is important to mention that any fixed step discrete time method should use a step size of about 1*ps in order to obtain a good representation of the input signal since its rising time is very short (mainly if we are interested in the behavior of the system during the rising time). As a result, the number of steps would be* 3200 *or greater with each step involving calculations over all the state variables. Even in the case that tools for dealing with sparse matrices are used all the state variable will change at each step.*

It was already mentioned that the QSS2–method properties only stand in LTI systems but it was conjectured that in nonlinear systems they should be quite similar. The reason of this conjecture has to do with the validity of the linearized models, which can be justified in general simulation problems taking into account that there are only small changes in the variables during successive steps. Thus, after each step, the trajectories stay inside the region where the linearized model constitutes a good approximation.

The following example illustrates these ideas through the use of the QSS2–method in a nonlinear system.

Example 5.2. *Lotka–Volterra's model.*

The famous second order Lotka–Volterra's model is a nonlinear system of differential equations that tries to represent the evolution of the population of two species, prey and predator, in a common habitat.

The following set of state equations constitutes one possible representation of the mentioned system

$$\begin{cases} \dot{x_1} &= \epsilon x_1 + \alpha x_1 x_2 - \sigma x_1^2 \\ \dot{x_2} &= -m x_2 + \beta x_1 x_2 \end{cases}$$

where the variables x_1 *and* x_2 *represent the population of preys and predators respectively.*

The system was simulated using QSS2 with a quantization of $\Delta q_1 = \Delta q_2 = 0.001$*. The parameters taken were* $\epsilon = 0.1$*,* $\alpha = 0.01$*,* $\sigma = 0.01$*,* $m = 0.4$*,* $\beta = 0.5$ *and the initial condition adopted was* $x_1(0) = x_2(0) = 10$*. The results are shown in Figures 5.6 and 5.7.*

The simulation was performed with 211 *internal transitions in the first integrator and* 264 *in the second, which gives a total of* 475 *steps. These results were compared with the trajectories obtained using Heun's method, the classic second order fixed step method. The step size used in Heun's method was* 0.63 *so that it performed the same number of steps. Figure 5.8 shows a part of those trajectories and the "true" trajectory. The "true" trajectory was obtained using the fifth order Dormand-Prince method with a step size of* 0.1*.*

The absolute error of the QSS2 and Heun's method during the simulation is shown in Figure 5.9. The use of QSS2 achieves a considerable reduction of the maximum error. However, the error does not converge to zero as in Heun's method.

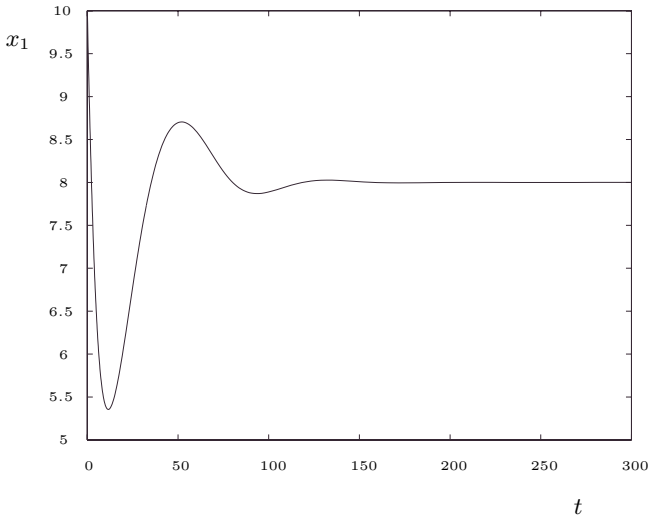

Figure 5.6: Number of preys in Lotka Volterra's model

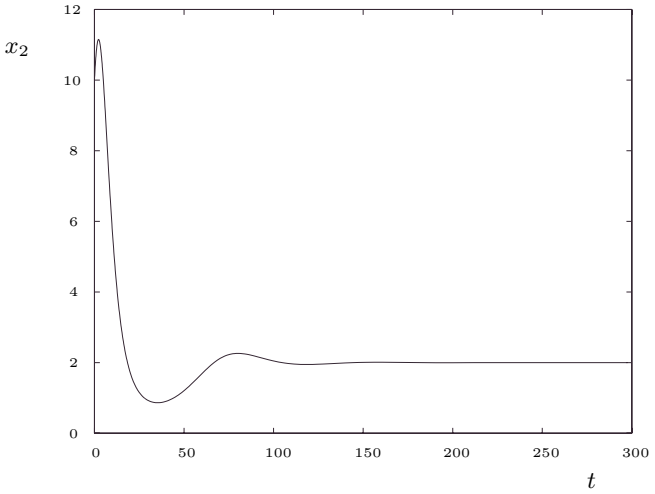

Figure 5.7: Number of predators in Lotka-Volterra's model

In spite of the lack of convergence, the results obtained with QSS2 simulation are much more reliable than the results with Heun's method since the error is bounded during the whole simulation. Here it can be seen that this property deduced for LTI systems in Section 4.6 is still verified in this nonlinear example.

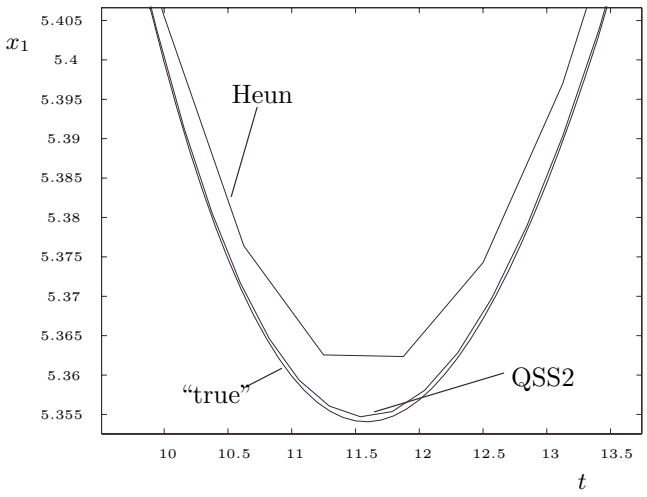

Figure 5.8: Number of preys according to Heun's method, QSS2 and the "true" trajectory.

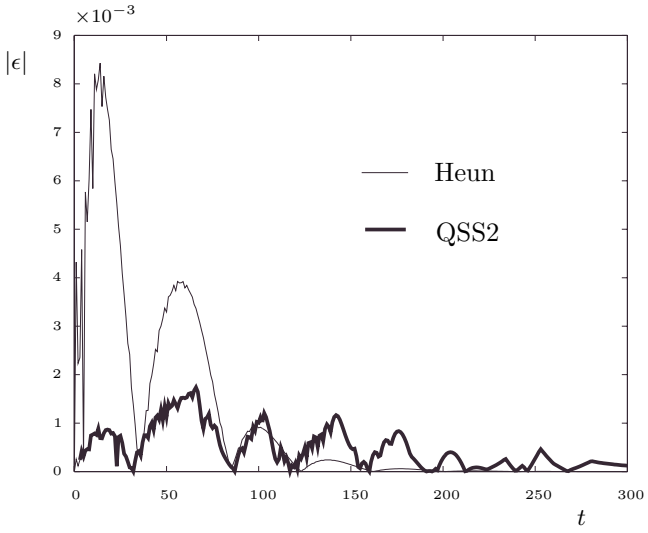

Figure 5.9: Error in Heun and QSS2 simulation of Lotka–Volterra's model

5.5 QSS vs. QSS2

It was mentioned that the error bound (4.50) holds for QSS and QSS2. Then, if the same quantum is used in QSS and QSS2 methods it should be expected to have the same error bound.

This last remark opens a question: Where is the advantage of using the

QSS2–method if it has the same error than QSS?.

The answer is that the simulation using QSS2 requires much less steps than a simulation using QSS for the same quantization (i.e. for the same error bound).

The cause of the reduction of the number of steps can be found in the comparision of the Figures 5.1 and 5.10. The number of steps used by a QSS to represent the same trajectory with the same quantum (Δq) is much greater than the used by the QSS2. The key is that using the information of the slope in QSS2 the time required to obtain a difference equal to Δq with respect to the input trajectory becomes much greater.

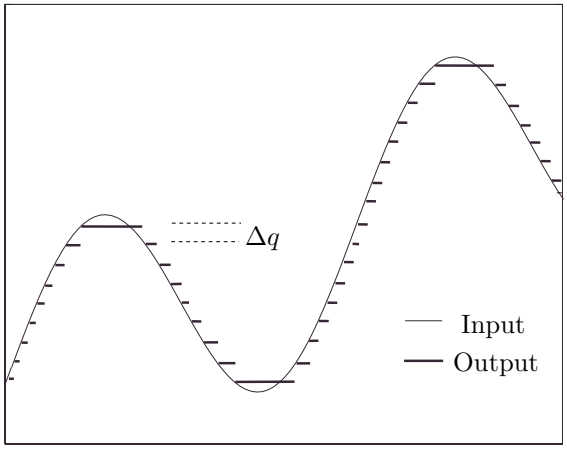

Figure 5.10: Input and Output trajectories in a *Zero Order* quantizer

In Lotka-Volterra's example, the simulation via QSS using the same quantization would have taken more than 10000 internal transitions at each integrator[1]. This value, compared against the 264 and 211 steps performed by the QSS2 simulation shows clearly the advantage of the new method.

The difference between the performance of QSS and QSS2 becomes greater as well as the quantization is taken smaller. In fact the number of internal transitions in QSS is approximately proportional to the inverse of the quantum while in QSS2 it is approximately proportional to the square root of that number (see Equation (5.9)). This relationship is verified in the example of the transmission line, where the use of a quantization 100 times smaller resulted in an increment of about 10 times in the number of steps.

However, it is clear that each transition in the QSS2–method involves more calculations than in QSS. Thus, if it is not important obtaining a good accuracy, the QSS method might result more efficient.

The following example illustrates these facts.

Example 5.3. *QSS vs. QSS2*

[1]This number can be obtained comparing the amplitude of the trajectories and the quantum 0.001

Consider the second order LTI system

$$\begin{cases} \dot{x_1} &= x_2 \\ \dot{x_2} &= 1 - x_1 - x_2 \end{cases}$$

The system was simulated using both methods and different quanta. In all cases, it was taken $\Delta q_1 = \Delta q_2$.

Figure 5.11 compares the CPU time taken by both methods. QSS shows a better performance for big quanta but when the quantization becomes smaller QSS2 reduces considerably the computational costs. In fact, for $\Delta q = 0.0001$ *the second order method is 40 times faster than QSS.*

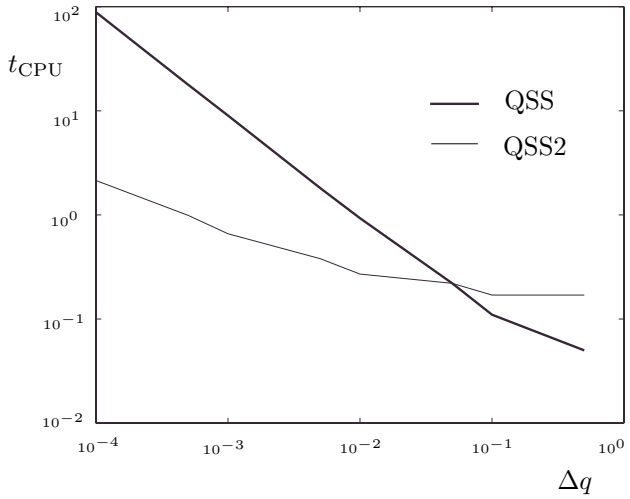

Figure 5.11: CPU time vs. quantum in QSS and QSS2

Chapter 6

Extensions of QSS and QSS2 Methods

Up to here, the Thesis attempted to develope a sort of theory about discrete event approximation of ordinary differential equations.

However, ODEs constitute only one fraction in the continuous systems world. Leaving the partial derivative problems aside, there are many continuous systems which cannot be represented by ODEs.

Many physical and egineering problems yield models where, in spite of having a differential equation nature, the functions involved in the corresponding ODE are not explicitely defined. These problems are called Differential Algebraic Equations (DAEs) and constitute a more general class than the ODEs.

In other cases –very common in modern technical systems– a single set of ODEs is not enough to represent the complete behavior. Those problems usually have some components which should be described by difference equations or discrete event systems (digital devices, for instance). Since these discrete components interact with the continuous parts of the system –which are typically described by ODEs or DAEs– those ODEs or DAEs show a *discontinuous* nature.

These last cases are called *Hybrid Systems* and, as in the case of the DAEs, their numerical integration has new problems and difficulties.

This chapter focuses in the application of QSS and QSS2 methods to the simulation of Differential Algebraic Equations and Hybrid Systems, taking also into account the particular case of physiscal systems modeled by Bond Graphs.

In all the cases some interesting advantages over the classic approaches will be found.

As it was anticipated, those advantages will result notably significant in the particular case of Hybrid Systems due to the discrete event nature of the approximation. There, an important reduction of the computational costs will be observed with a remarkable increase of accuracy and simplicity.

6.1 QSS and QSS2 in DAE Systems

There are many continuous systems where an explicit ODE formulation cannot be easily obtained [10]. Indeed, there are cases in which that representation does not exist. These systems, where only implicitely defined state equations can be written, are called Differential Algebraic Equations. The difficulty carried by DAE simulation is that it requires the use of some special techniques, which include iterations or symbolic manipulation to solve the implicit equations involoved.

A time invariant DAE can be written as

$$\tilde{f}(\dot{x}(t), x(t), u(t)) = 0 \tag{6.1}$$

The problem here with QSS and QSS2 is that the quantized integrators need the value of $\dot{x}$, which can be only approximately obtained with iterations.

One of the most important results in the area came from the idea of combining the ODE solver rules with the system implicit equations in order to solve them together (with iterations or symbolic manipulation). This idea, due to Gear [16], established the basis for all the modern DAE simulation methods.

Then, if the QSS or QSS2 solver rules are used here the state variables $x(t)$ should be replaced by their quantized versions $q(t)$. Thus, Equation (6.1) becomes

$$\tilde{f}(\dot{x}(t), q(t), u(t)) = 0 \tag{6.2}$$

Then, iteration rules –like Newton– can be used to obtain $\dot{x}$ from (6.2). In this work, it is assumed that the index is 1. If it is higher, the Pantelides' algorithm [52] can be applied in order to reduce it to 1.

Once the state derivatives are calculated, they can be sent to the quantized integrators which perform the rest of the job, i.e. calculating the quantized variable trajectories. Each time a quantized variable changes, a new iteration process should be performed in order to recalculate $\dot{x}$.

It was already explained that QSS and QSS2 methods exploit the system sparsity reducing the computational costs. Now, it will be shown that this is also true in DAE problems.

Equation (6.1) can be rewritten as

$$\dot{x}(t) = f(x(t), u(t), z(t)) \tag{6.3a}$$

$$0 = g(x_r(t), u_r, z(t)) \tag{6.3b}$$

where $z(t)$ is a vector of auxiliary variables whose dimension is equal or less than n. The vectors x_r and u_r are reduced versions of x and u respectively[1].

Equation (6.3b) expresses the fact that some state and input variables may not act directly on the algebraic loops.

[1] A strightforward way of going from (6.1) to (6.3) is defining $z(t) = \dot{x}(t)$, $x_r(t) = x(t)$, $u_r(t) = u(t)$, $g(x, u, z) = \tilde{f}(z, x, u)$ and $f(x, u, z) = z$.

Then, the use of the QSS or QSS2 method tranforms (6.3) into

$$\dot{x}(t) = f(q(t), u(t), z(t)) \quad (6.4a)$$
$$0 = g(q_r(t), u_r, z(t)) \quad (6.4b)$$

and now, the iterations should be only performed to solve Eq.(6.4b) when the components of q_r or u_r change. When the dimension of x_r is significantly less than the dimension of x, i.e. when there are several state variables which do not influence on the loop, this fact represents an important advantage.

When Equation (6.4b) defines the value of z, (6.4) defines a system which behaves like a QSS or a QSS2. In fact, it can be easily proven that the state and quantized variable trajectories correspond to QSS or QSS2. Moreover, the auxiliary variables z have piecewise constant and piecewise linear trajectories in QSS and QSS2 respectively.

Then, it is said that System (6.4) is an implicitely defined QSS or QSS2. What should be solved now is the way in which an implicitely defined QSS (QSS2) like this can be translated into a DEVS model.

It is clear that (6.4a) can be represented by quantized integrators and static functions as we did before. The only difference now is the presence of the auxiliary variables z which act as inputs like $u(t)$. However, while the components of $u(t)$ are known and they come from DEVS signal generators, the auxiliary variables should be calculated by solving the restriction (6.4b).

Thus a new DEVS model shuld be built. This DEVS model must receive events with the values of q_r and u_r and then it has to calculate z. Figure 6.1 shows the new coupling scheme with the addition of a new DEVS model which calculates z.

A DEVS model which solves a general implicit equation like

$$g(v, z) = g(v_1, \ldots, v_m, z_1, \ldots, z_k) = 0 \quad (6.5)$$

for the QSS case can be written as follows

$$\begin{aligned}
&M_{10} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \mathbb{R} \times \mathbb{N} \\
&Y = \mathbb{R}^k \times \mathbb{N} \\
&S = \mathbb{R}^{m+k} \times \mathbb{R}^+ \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(v, z, \sigma, e, x_v, p) = (\tilde{v}, h(\tilde{v}, z), 0) \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(v, z, \sigma) = (v, z, \infty) \\
&\lambda(s) = \lambda(v, z, \sigma) = (z, 1) \\
&ta(s) = ta(v, z, \sigma) = \sigma
\end{aligned}$$

where

$$\tilde{v} = (\tilde{v}_1, \ldots, \tilde{v}_m)^T; \;\; \tilde{v}_i = \begin{cases} x_v & \text{if} \quad p = i \\ v_i & \text{otherwise} \end{cases}$$

and the function $h(v, z)$ returns the result of applying Newton iteration or some other iteration rules to find the solution of (6.5) using an initial guess z.

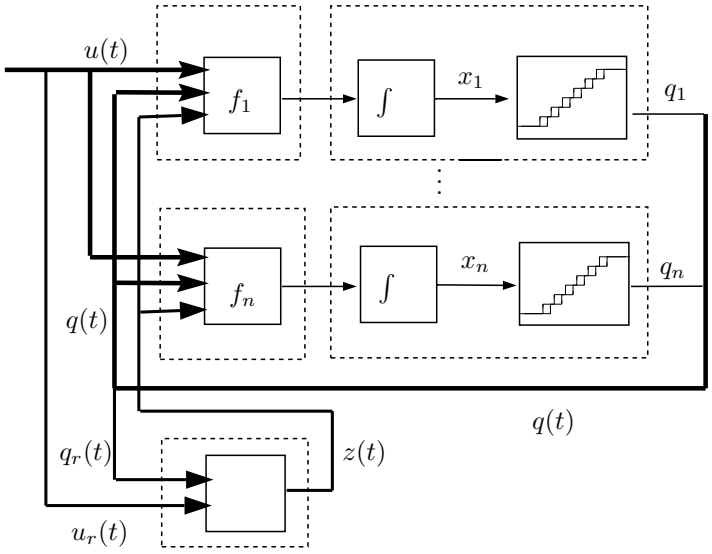

Figure 6.1: Coupling scheme for the QSS simulation of (6.3)

When the size of z (i.e. k) is greater than 1, the output events of model M_{10} contains a vector. Thus, they cannot be sent to static functions like M_2. Anyway, a DEVS model which demultiplexes a vectorial input value into scalar output values at different ports can be used in order to solve this difficulty.

The idea for the QSS2–method is similar, but now the implicit equation should be rewritten as:

$$g(v(t), z(t)) = g(v_0 + m_v \Delta t, z_0 + m_z \Delta t) = 0 \tag{6.6}$$

which can be splitted into

$$g(v_0, z_0) = 0 \tag{6.7a}$$

and a first order approximation

$$\left.\frac{\partial g}{\partial v}\right|_{(v_0,z_0)} m_v + \left.\frac{\partial g}{\partial z}\right|_{(v_0,z_0)} m_z = 0 \tag{6.7b}$$

Then, the algorithm should iterate using Eq.(6.7a) to obtain z_0 and then, it must use this value in (6.7b) to calculate m_z.

The calculation of m_z can be done using symbolic manipulation (i.e. matrix inversion) or a new iteration process which will finish after two steps (because this is a linear equation). If the partial derivatives are not available, they can be estimated using coefficients as we did in Equation (5.12).

When z and $g(v, z)$ are scalar, Equation (6.7b) can be easily solved. In this

case, we have

$$m_z = -\frac{\left.\frac{\partial g}{\partial v}\right|_{(v_0,z_0)} m_v}{\left.\frac{\partial g}{\partial z}\right|_{(v_0,z_0)}} \tag{6.8}$$

and then, a DEVS model can be built as follows

$M_{11} =< X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta >$, where:
$X = \mathbb{R}^2 \times \mathbb{N}$
$S = \mathbb{R}^{3(m+1)} \times \mathbb{R}_0^+ \infty$
$Y = \mathbb{R}^2 \times \mathbb{N}$
$\delta_{\text{int}}((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) =$
$= ((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \infty)$
$\delta_{\text{ext}}((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma, e, u, mu, p) =$
$= ((\tilde{v}_1, \tilde{m}_{v_1}, \tilde{c}_1), ..., (\tilde{v}_m, \tilde{m}_{v_m}, \tilde{c}_m), (\tilde{z}, \tilde{m}_z, \tilde{c}_z), 0)$
$\lambda((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) = (z, m_z, 1)$
$ta((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) = \sigma$

with

$$\tilde{v}_i = \begin{cases} u & \text{if} \quad p = i \\ v_i + m_{v_i} e & \text{otherwise} \end{cases}$$

and

$$\tilde{z} = h(\tilde{v}, z)$$

where h is the result of the iterations to solve Equation (6.7a) starting from the initial guess z.

The coefficients which estimate the partial derivatives can be recalculated as:

$$\tilde{c}_i = \begin{cases} \frac{g(v + m_v e, \tilde{z})}{v_i + m_{v_i} e - \tilde{v}_i} & \text{if} \quad p = i \wedge v_i + m_{v_i} e - \tilde{v}_i \neq 0 \\ c_i & \text{otherwise} \end{cases}$$

$$\tilde{c}_z = \begin{cases} \frac{g(\tilde{v}, z + m_z e)}{z + m_z e - \tilde{z}} & \text{if} \quad z + m_z e - \tilde{z} \neq 0 \\ c_z & \text{otherwise} \end{cases}$$

and finally, the new slopes are

$$\tilde{m}_{v_i} = \begin{cases} m_u & \text{if} \quad p = i \\ m_{v_i} & \text{otherwise} \end{cases}$$

and

$$\tilde{m}_z = -\frac{1}{\tilde{c}_z} \sum_{i=1}^{m} \tilde{m}_{v_i} \tilde{c}_i$$

If m is big, i.e. v has many components, the new output slope $\tilde{m}_z$ can be calculated with the equivalent formula:

$$\tilde{m}_z = \frac{c_z}{\tilde{c}_z} m_z + \frac{1}{\tilde{c}_z}(m_{v_p} c_p - \tilde{m}_{v_p} \tilde{c}_p)$$

This last DEVS model is just a possible alternative to solve an implicit restriction like (6.6) taking into account the slopes. There are many other possibilities. When function g is linear, the DEVS model produces the exact output values z and m_z. Otherwise, it gives only a first order approximation.

Example 6.1. *A Transmission Line with Surge Voltage Protection.*

Figure 6.2 shows the transmission line model of Figure 5.3, modified with the addition of a load.

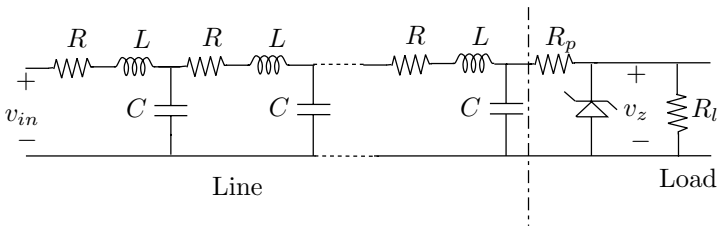

Figure 6.2: RLC Transmission Line with surge voltage protection

The load is composed by a resistor R_l –which may represent the gate of some electronic component– and a surge protection circuit formed by a zener diode and a resistor R_p. It will be considered that the zener diode satisfies the following nonlinear current–voltage function:

$$i_z = \frac{I_0}{1 - (v_z/v_{br})^m} \tag{6.9}$$

where m,v_{br} and I_0 are parameters which depend on different physical features.

If the transmission line is divided in five sections –as it was done before–, the following equations are obtained:

$$\begin{aligned}
\frac{di_1}{dt} &= \frac{1}{L}v_{in} - \frac{R}{L}i_1 - \frac{1}{L}u_1 \\
\frac{du_1}{dt} &= \frac{1}{C}i_1 - \frac{1}{C}i_2 \\
\frac{di_2}{dt} &= \frac{1}{L}u_1 - \frac{R}{L}i_2 - \frac{1}{L}u_2 \\
\frac{du_2}{dt} &= \frac{1}{C}i_2 - \frac{1}{C}i_3 \\
&\vdots \\
\frac{di_5}{dt} &= \frac{1}{L}u_4 - \frac{R}{L}i_5 - \frac{1}{L}u_5 \\
\frac{du_5}{dt} &= \frac{1}{C}i_5 - \frac{1}{R_pC}(u_5 - v_z)
\end{aligned}$$

Here, the state variables u_j and i_j represent the voltage and current in the capacitors and inductors respectively and the output voltage v_z is an algebraic

variable which should satisfy

$$\frac{1}{R_p}u_5 - \left(\frac{1}{R_p} + \frac{1}{R_l}\right)v_z - \frac{I_0}{1-(v_z/v_{br})^m} = 0 \tag{6.10}$$

Thus, there is a DAE here which cannot be converted into an ODE by symbolic manipulation and the simulation with any classic method should iterate at each step to solve the implicit equation (6.10).

However, as it can be deduced from (6.4)*, the QSS methods will only iterate when there are changes in the quantized version of* u_5*. In the rest of the steps –when the other 9 quantized variables change or when the input changes– no iteration has to be performed.*

Using the ideas expressed above, the simulation of page 73 was modified with the addition of a DEVS model like M_{11} *which solves* (6.10) *considering also the slopes.*

Taking $R_l = 100M\Omega$*,* $I_0 = 0.1\mu A$*,* $v_{br} = 2.5V$*,* $m = 4$ *and without modifying the remaining parameters, the results shown in Figure 6.3 were obtained.*

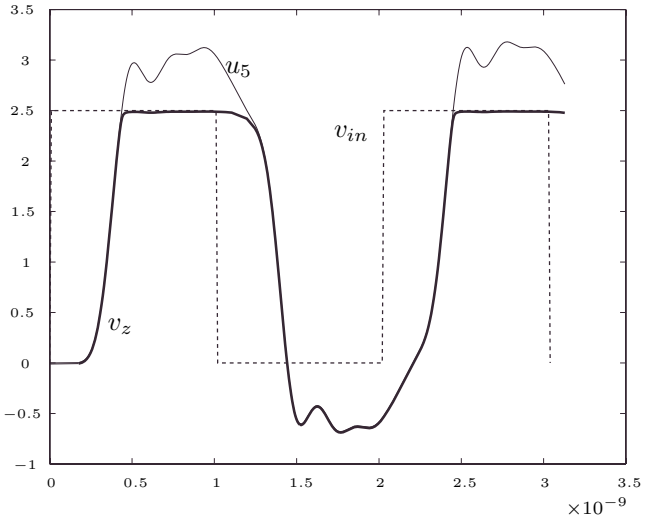

Figure 6.3: QSS2 simulation results in a RLC transmission line with surge protection

The first 3.2*ns of the simulation were completed after 2640 steps (between 200 and 316 steps at each integrator). The implicit model (*M_{11}*) performed a total of 485 iterations with the Secant–method. The reason for this is that the quantized integrator which calculates* u_5 *only performed 200 internal transitions, and then the implicit model received only 200 external events. The Secant–method needed between two and three iterations to find the solution of Eq.*(6.10) *with the required tolerance (it was taken* $tol = 1 \times 10^{-8}$*) which explains the fact that the total number of iteration was 485 (between 400 and 600).*

The advantages of the QSS2 method are evident in this example. In a discrete–time algorithm, the Secant–Method would have been invoked in all the steps while the QSS2 only called it after the changes in u_5 (about once every 13 steps). Thus, the presence of the implicit equation only adds a few calculations which do not affect significantly the total number of computations.

6.2 Block–Oriented DEVS Simulation of DAEs

DAE systems of index 1 are often represented by Block Diagrams containing algebraic loops.

The circuit of Figure 6.4, for instance, can be modelled by the block diagram of Figure 6.5. The bold lines in this block diagram indicate the presence of an algebraic loop.

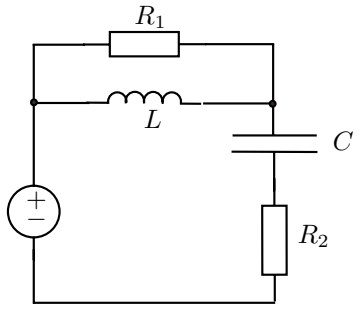

Figure 6.4: RLC circuit

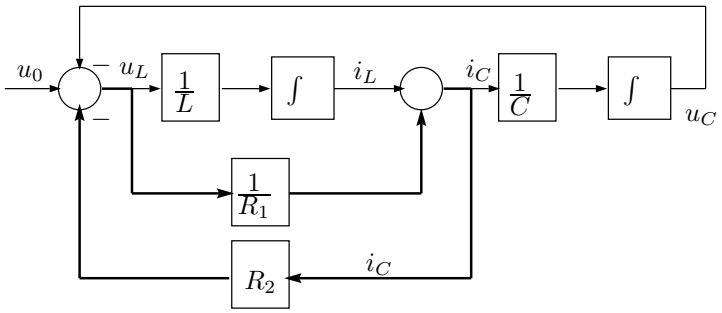

Figure 6.5: Block Diagram representation of the RLC circuit

The QSS–method can be implemented transforming the integrators into DEVS models like M_4 (quantized integrators) and the static functions into their DEVS representation (DEVS models like M_2). Then, these DEVS models can be coupled according to the coupling scheme of Figure 6.5.

In fact, that is what was done to convert System (3.2) into a DEVS representation of (3.3) (see Figure 3.2 in page 30).

Although the translation block by block from a continuous system to a DEVS model may result in an inefficient simulation (from the point of view of the computational costs), it is very simple and it does not require from any kind of symbolic manipulation.

Block Diagrams are a usual tool for representing differential equations and the available DEVS simulation programs do not offer tools to translate them into sets of equations like (3.2). Thus, if the translation by hand is not wanted and there are not another automatic tool to generate a set of equations like (3.2), the block by block translation is the only possibility to apply the QSS–method.

However, if this is done with the Block Diagram of Figure 6.5 a problem appears. Due to the algebraic loop, the DEVS model will result illegitimate and when an event comes into the loop, it will propagate forever through the static functions.

Evidently, it is necessary to add a new DEVS model into the loop so that it solves the implicit equation. This new model will be called loop–breaking DEVS.

In the example of Figure 6.5, that loop–breaking DEVS can be placed, for instance, before the gain R_2 as Figure 6.6 shows.

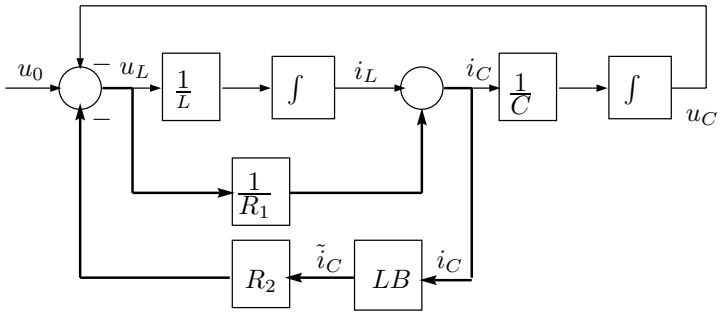

Figure 6.6: Addition of a Loop-Breaking model to the Block Diagram of Fig.6.5

Then, the loop–breaking model will receive the values of i_C and it should send $\tilde{i}_C$. Each time this DEVS model sends an event, it receives a new event with the value i_C calculated by the static functions belonging to the loop. When this value coincide with the value of $\tilde{i}_C$ that the model had previously sent it means that the implicit equation was solved and it is unnecessary to send a new $\tilde{i}_C$. In that way, the simulation continues outside the loop until a new value of i_C arrives to the breaking loop model and the process is repeated.

This idea can solve the problem. However, it was not said yet how the value of $\tilde{i}_C$ has to be calculated. Before giving an answer to this question the problem should be formulated in a more formal and general fashion.

Let us call z the variable sent by the loop–breaking model. Then, when it sends an event with value z_1 it immediately receives a new event with value $h(z_1)$ calculated by the static functions.

Thus, the model should calculate a new value for z, let us call it z_2, which

should satisfy

$$h(z_2) - z_2 \triangleq g(z_2) \approx 0 \tag{6.11}$$

If it is not verified, the process should be repeated sending a new value z_3.

It is clear that z_{i+1} must be calculated following some algorithm to find the solution of $g(z) = 0$. Taking into account that the derivative of $g(z)$ is not known, a good alternative to the Newton iteration is the use of the Secant–Method.

Using this method, the loop–breaking DEVS model can be represented as follows:

$$\begin{aligned}
&M_{12} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \mathbb{R} \times \mathbb{N} \\
&Y = \mathbb{R} \times \mathbb{N} \\
&S = \mathbb{R}^3 \times \mathbb{R}^+ \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(z_1, z_2, h_1, \sigma, e, x_v, p) = \tilde{s} \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(z_1, z_2, h_1, \sigma) = (z_1, z_2, h_1, \infty) \\
&\lambda(s) = \lambda(z_1, z_2, h_1, \sigma) = (z_2, 1) \\
&ta(s) = ta(z_1, z_2, h_1, \sigma) = \sigma
\end{aligned}$$

where

$$\tilde{s} = \begin{cases} (z_1, z_2, h_1, \infty) & \text{if} \quad |x_v - z_2| < tol \\ (z_2, \tilde{z}, x_v, 0) & \text{otherwise} \end{cases}$$

with

$$\tilde{z} = \frac{z_1 \cdot x_v - z_2 \cdot h_1}{x_v - h_1 + z_1 - z_2} \tag{6.12}$$

The parameter *tol* is the maximum error we allow between z and h. Equation (6.12) is the result of applying the secant–method to approximate $g(z) = 0$ where $g(z)$ is defined according to (6.11). The iteration algorithm can be then changed by modifying (6.12).

Example 6.2. *Block–Oriented Simulation of the RLC Circuit.*

For the circuit example, a coupled DEVS model was built according to Figure 6.6 and simulated until a final time of 30 seconds. The parameters used were $R_1 = R_2 = L = C = 1$ *and* u_0 *was chosen as a unit step. The quantum and hysteresis adopted were* 0.01 *in both state variables and the error tolerance tol was taken equal to* 0.001.

The simulation –which is shown in Figure 6.7– was completed after 118 and 72 internal transitions at each quantized integrator and a total of 377 iterations at the loop–breaking DEVS model.

In this case, due to the system linearity, during each step the secant–method arrives to the exact solution of $g(z) = 0$ *performing only two iterations. This explains the fact that the total number of iterations at the loop-breaking model was approximately twice the total number of steps at both quantized integrators.*

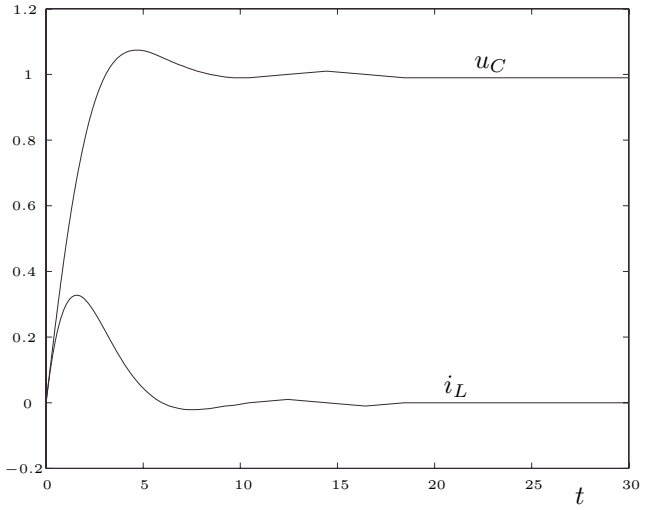

Figure 6.7: QSS simulation of the RLC circuit using a Loop–Breaking DEVS

An interesting observation is that the effect of the error in the calculation of i_C can be seen as an additional perturbation. If it can be ensured that this error is bounded, the perturbation is also bounded and it can be seen as something equivalent to having a bigger quantum which only affects the error bound but it does not modify the stability properties. It is clear that the same remark can be made with respect to the transmission line example.

Finally, it should be mentioned that a DEVS loop–breaking model model like M_{12} can be also obtained for the QSS2–method following the same ideas expressed above.

6.3 QSS and QSS2 Simulation of Hybrid Systems

The complexity of many technical systems yields models which often combine a continuous part (described by ODEs or DAEs) and discrete components. The interaction between these subsystems can produce sudden changes (discontinuities) in the continuous part which must be handled by the integration algorithms.

These discontinuities yield important difficulties in the context of discrete time classic methods. The problem is that numerical integration algorithms in use are incompatible with the notion of discontinuous functions [50] and an integration step which jumps along a discontinuity may produce an unacceptable error.

To avoid this, the methods should perform steps in the instants of time in which the discontinuities take place and then, the simulation of a hybrid system

should be provided of tools for detecting the discontinuities occurrence (what includes iterations and extra computational costs), for adapting the step size to hit those instants of time and of course, to represent and simulate the discrete part of the system (which can be quite complicated itself) in interaction with the continuous part.

Although there are several methods and software tools which simulates hybrid systems in a quite efficient way, none of them can escape from these problems.

The mentioned sudden changes are called *events* and two different cases can be distinguished according to the nature of their occurrence. The events which occur at a given time, independently of what happens in the continuous part are called *Time Events*. On the other hand, events which are produced when the continuous subsystem state reaches some condition are called *State Events*.

The integration along discontinuities without event detection techniques can cause severe inefficiency, and even simulation failures or incorrect event sequences to be generated, because the non–smoothness violates the theoretical assumptions on which solvers are founded [3]. Thus, time and state events must be detected in order to perform steps at their occurrence.

The incorporation of event detection techniques to numerical methods have been being studied since Cellier's Thesis [8] and many works can be found in the recent literature (see for instance [53, 62, 58, 13]).

Although these ideas work quite efficiently, the techniques do not say how to represent discrete parts and how to schedule the time events in general cases. Moreover, the state event detection requires performing some iterations to find the time of the event occurrence.

When it comes to QSS and QSS2, some improvements can be be expected due to their discrete event asynchronous nature.

Before starting to develop the ideas about this, it must be mentioned that there is not a unified representation of hybrid systems in the literature.

Anyway, the different approaches coincide in describing them as sets of ODEs or DAEs which are selected according to some variable which evolves in a discrete way (different examples of hybrid systems representation can be found in [61, 4, 6, 3]).

Here, it will be assumed that the continuous subsystem can be represented by

$$\dot{x}(t) = f(x(t), u(t), z(t), m(t)) \tag{6.13a}$$

$$0 = g(x_r(t), u_r(t), z(t), m(t)) \tag{6.13b}$$

being $m(t)$ a piecewise constant trajectory coming from the discrete part, which defines the different modes of the system. Thus, for each value of $m(t)$ there is a different DAE representing the system dynamics.

It will be also considered that the implicit equation (6.13b) has a solution for each value of $m(t)$ (which implies that System (6.13) has always index 1).

Independently of the way in which $m(t)$ is calculated, the simulation sub–model corresponding to the continuous part can be built considering that $m(t)$

acts as an input.

Then, the QSS and QSS2 methods applied to this part will transform (6.13) into:

$$\dot{x}(t) = f(q(t), u(t), z(t), m(t)) \quad (6.14a)$$
$$0 = g(q_r(t), u_r(t), z(t), m(t)) \quad (6.14b)$$

with the same definitions done in (6.4). Thus, the simulation scheme for the continuous part will be identical to the one shown in Figure 6.1, but now $m(t)$ must be included with the input.

One of the most important features of DEVS is its capability to represent all kind of discrete systems. Taking into account that the continuous part is being approximated by a DEVS model, it is natural representing also the discrete behavior by another DEVS model. Then, both DEVS models can be directly coupled to build a unique DEVS model which approximates the whole system.

In presence of only Time Events, the DEVS model representing the discrete part will be just an event generator like M_5 (page 36). Here, the output events will carry the successive values of $m(t)$

Then, the simulation of the complete hybrid system can be performed by coupling this time event generator with the continuous part.

Taking into account the asynchronous way in which the static functions and quantized integrators work, the events will be processed by the continuous part as soon as they come out from the generator without the need of modifying anything in the QSS or QSS2 methods. This efficient event treatment is just due to intrinsic behavior of the methods.

This fact makes a big difference with respect to discrete time methods which must be modified in order to hit the event times.

When it comes to state events, the discrete part is ruled not only by the time advance but also by some events which are produced when the input and state variables reach some condition.

Here, the QSS and QSS2 methods have a bigger advantage: The state trajectories are perfectly known for all time. Moreover, they are piecewise linear or piecewise parabolic functions which implies that detecting the event occurrence is straightforward.

The only thing which has to be done is to provide those trajectories to the discrete part so it can detect the event occurrence and it can calculate the trajectory $m(t)$. Since the state trajectories are only known inside the quantized integrators, these models could be modified in order to output not only the quantized variables but also the state trajectories.

However, this is not necessary. The discrete part can receive the state derivative trajectories and then integrate them. It is simple and it does not require computational effort since the derivative trajectories are piecewise constant or piecewise linear (in QSS2) and their integration only involves the manipulation of the polynomial coefficients.

Using these ideas, the simulation model for a hybrid system like (6.13) using the QSS or QSS2 method will be a coupled DEVS with the structure shown in

Figure 6.8.

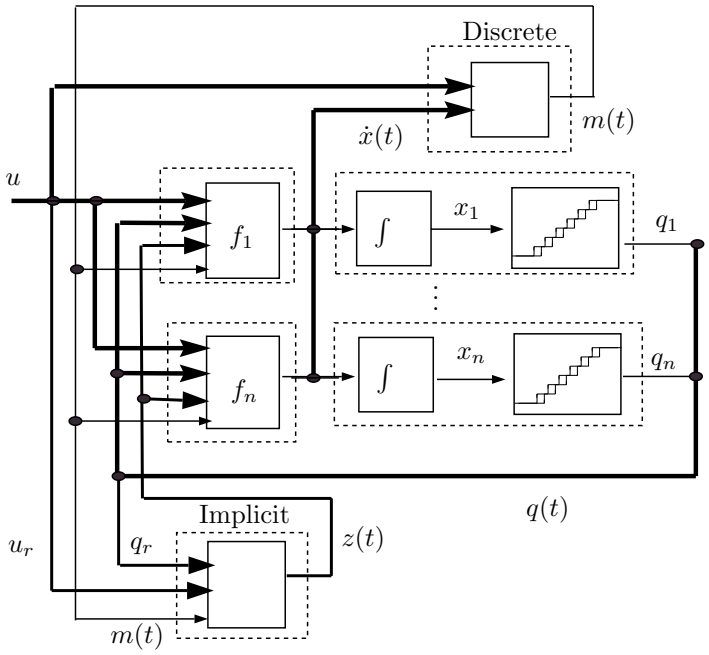

Figure 6.8: Coupling scheme for the QSS simulation of discontinuous systems

Here the discrete part is a DEVS model which receives the events representing changes in the state derivatives as well as changes in the input trajectories.

Since the discrete model receives and produce only a finite number of events in any finite interval of time (because of its definition as a discrete model), it can be ensured that a DEVS model can represent it no matter the complexity of its dynamic. Taking into account this, the scheme of Figure 6.8 can simulate any systems like (6.13) in interaction with any discrete model.

There are cases in which this scheme can be simplified. As we mentioned before, when only Time Events are considered, the DEVS model of the discrete part will not have inputs.

Usually, the event occurrence condition is related to a zero (or another fixed value) crossing of some state variable. In this case, if the simulation is performed with the QSS–method the event condition can be detected directly by the corresponding quantized integrator. This can be easily done provided that the quantization functions contain quantization levels at the given fixed crossing values.

Example 6.3. *DC-AC Inverter Circuit.*

The inverter circuit shown in Figure 6.9 can be used to feed different electrical machines.

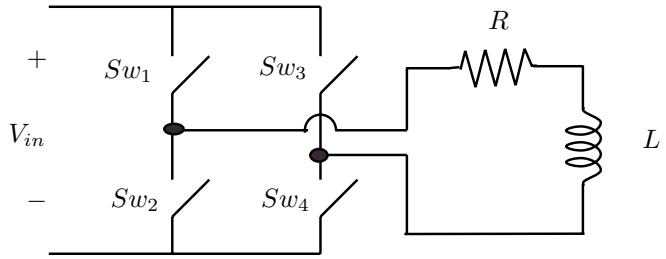

Figure 6.9: DC-AC Full Bridge Inverter

The set of switches can take two positions. In the first one the switches 1 and 4 are closed and the load receives a positive voltage. In the second position the switches 2 and 3 are closed and the load receives a negative voltage.

The system can be represented by the following differential equation:

$$\frac{d}{dt}i_L = -\frac{R}{L}\cdot i_L + s_w \cdot V_{in} \tag{6.15}$$

where s_w *is* 1 *or* -1 *according to the position of the switches.*

A typical way of controlling the switches in order to obtain a harmonic current at he load is using a pulse width modulation (PWM) strategy. The PWM signal is obtained by comparing a triangular wave (carrier) with a modulating sinusoidal reference. The sign of the voltage to be applied ($+V_{in}$ *or* $-V_{in}$*) and the corresponding position is given by the sign of the difference between those signals. Figure 6.10 shows this idea.*

In this case, the switches change their position independently of what happens in the circuit. Then, the events representing those changes are time events.

The system was simulated with the QSS2–method, using a scheme like the shown in Figure 6.8 but the discrete block was just an event generator producing events when the variable sw changes.

The event times where calculated for a carrier frequency of $1.6kHz$ *and a modulating sinusoidal signal of the same amplitude and a frquency of* $50Hz$*. Thus, the number of events per cycle was 64, which is enough to produce a quite smooth sinusoidal current.*

Using parameters $R = 0.6\Omega$*,* $L = 100mHy$ *and* $V_{in} = 300\,V$ *the simulation starting from* $i_L = 0$ *and taking a quantization* $\Delta i_L = 0.01A$ *gave the result shown in Figures 6.11–6.12.*

The final time of the simulation was 1 second and then the number of cycles was 50. This gives a total of 3200 changes in the position of the switches.

Despite this number of events, the simulation was completed after only 3100 internal transitions at the second order quantized integrator. Thus, the total number of steps was 6300.

In this case, since the commutations do not produce any structural change (they only affect the input voltage sign), the formula (4.50) *can be applied and*

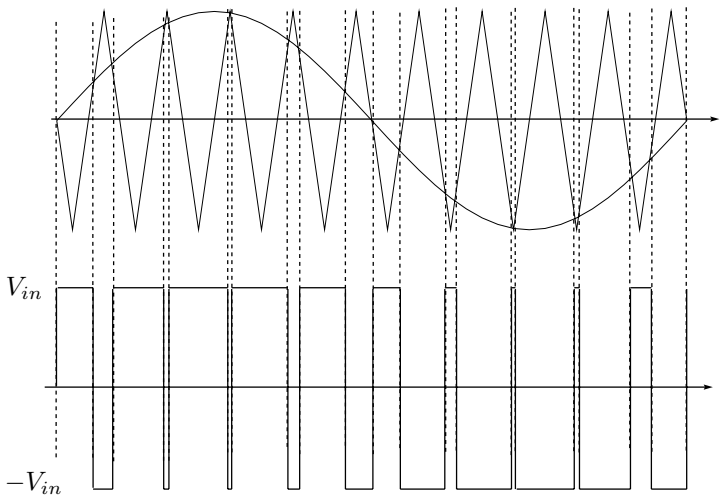

Figure 6.10: Pulse Width Modulation

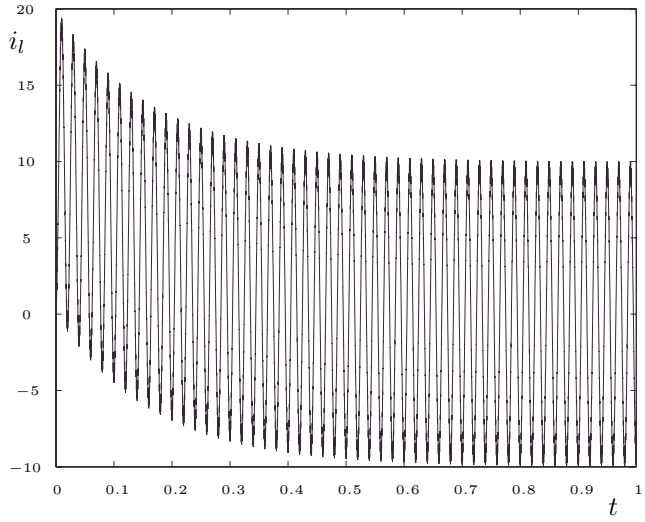

Figure 6.11: Load current with Pulse Width Modulation

it can be ensured that the error in the trajectory of i_L obtained is always less than $10mA$ (which is about the 0.1% of the oscillation amplitude) .

The same system was simulated with all the discrete time methods implemented in Simulink. The fixed step ode5 algorithm (5th order) needed more than 50000 steps to obtain an acceptable result. Of course, lower order fixed step methods required more steps.

Using variable step methods the result was even worse. Only the ode23s

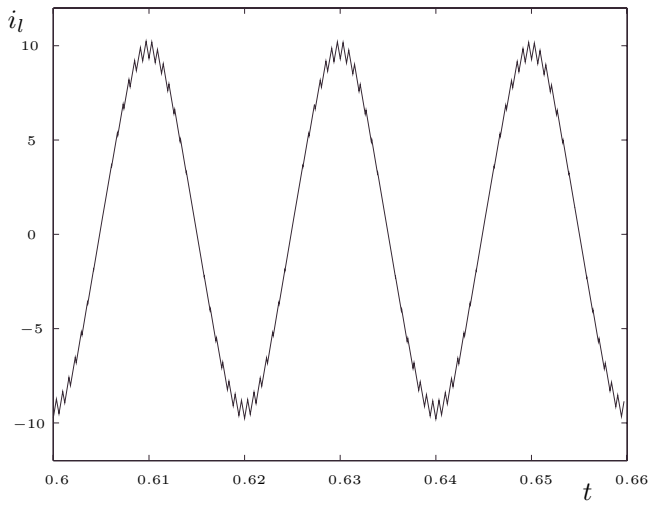

Figure 6.12: Detail of the permanent regime load current

methods gave acceptable results with about 100000 steps.

The simulations were repeated with variable step methods enforcing additional calculations at the event times. In this case they worked sensibly better. Anyway, using the tolerance obtained with QSS2, the ode23 (which now showed the best performance) needed more than 20000 steps to complete the simulation.

However, this trick –enforcing calculations at predetermined time instants– cannot be used in general cases since often the event times are not known before the simulation starts. In the PWM case it is usual to calculate them during the simulation since the frequency and amplitude of the modulating signal often change according to control strategies.

Example 6.4. *A Ball Bouncing Downstairs.*

A typical example of a discontinuous system is the bouncing ball. Here, it will be considered the case in which the ball moves in two dimensions (x and y) bouncing downstairs. Thus, the bouncing condition depends on both variables (x and y).

It will be assumed that the ball has a model when it is in the air –with the presence of friction– and a different model in the floor. Here, a spring–damper model will be consider.

According to this idea, the model can be written as

$$\begin{aligned}
\dot{x} &= v_x \\
\dot{v}_x &= -\frac{b_a}{m} \cdot v_x \\
\dot{y} &= v_y \\
\dot{v}_y &= -g - \frac{b_a}{m} \cdot v_y - s_w \cdot [\frac{b}{m} \cdot v_y + \frac{k}{m}(y - int(h + 1 - x))]
\end{aligned}$$

where s_w is equal to 1 in the floor and 0 in the air. The function $int(h+1-x)$ gives the height of the floor at a given position (h is the height of the first step). Note that we are considering steps of $1m$ by $1m$.

The state events *are produced when x and y verify the condition:*

$$y = int(h + 1 - x)$$

The simulation model structure results then similar to the one shown in Figure 6.8 but without the implicit block. The discrete model should receive the events with the derivatives of x and y and send events when the event condition is achieved (to calculate that, it just has to find the roots of a second degree polynomial).

The system was then simulated using parameters $m = 1$, $k = 100000$, $b = 30$, $ba = 0.1$, initial conditions $x(0) = 0.575$, $v_x(0) = 0.5$, $y(0) = 10.5$, $v_y = 0$ and a quantum of 0.001 in the horizontal position, 0.0001 in the vertical position and 0.01 in the speeds.

The first 10 seconds of simulation were completed after 2984 internal transitions at the integrators (39 at x, 5 at v_x, 2420 at y and 520 at v_y). The trajectories do not differ appreciably from what can be obtained with a fixed step high order method using a very small step size.

Figures 6.13 and 6.14 show the simulation results.

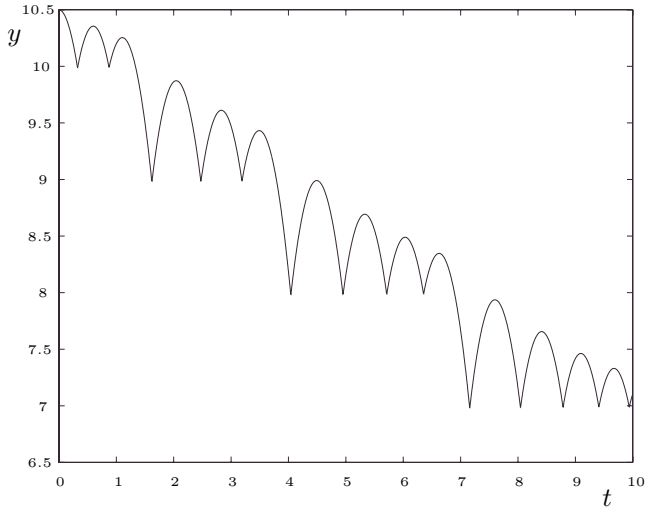

Figure 6.13: y vs. t in the bouncing ball example

It is important to remark that each step only involves very few calculations and the sparsity is well exploited. In fact, the internal transitions in x does not affect any other subsystem. The steps in v_x give events to itself, to the integrator which calculates x and to the discrete model which predicts the next event occurrence. Similarly, the internal events of y only provoke external events

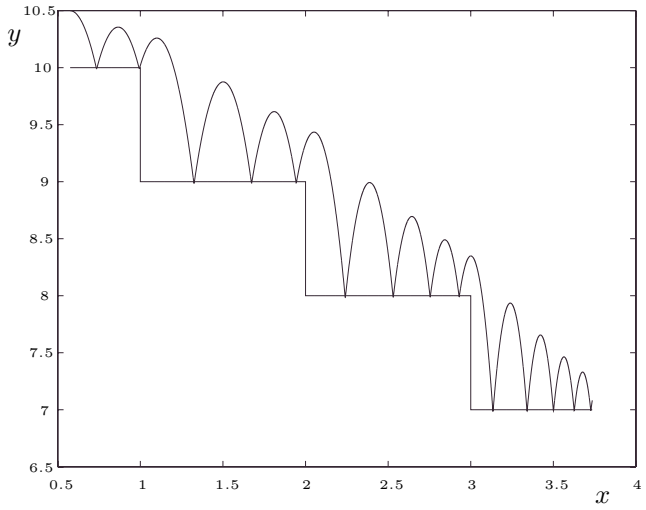

Figure 6.14: x vs. y in the bouncing ball example

to the integrator corresponding to v_y when the ball is in the floor and finally, the events produced in v_y are propagated to itself, to the integrator which calculates y and to the discrete model.

As a result, the discrete model receives 525 *events and it produces only* 26 *internal transitions (at the event occurrence times, it is, two events for each bounce).*

The same model was simulated with Simulink, using different fixed and variable step algorithms. Obtaining a similar result with a fifth order fixed step method requires more than 10000 steps (and here each step involves calculations in the whole system).

When it comes to variable step methods, the best result using Simulink was obtained with the ode23s, which could obtain a similar result with about 5000 steps.

In the bouncing ball case, the problem of discrete time methods is that when they increment the step size, they start skipping events as shown in Figure 6.15. An example of this problem was given in [13] where the authors gave a solution based on decreasing the step size as well as the system approximates the discontinuity condition.

The quantization-based approach does not modify anything. It just makes use of a discrete block which exactly predicts when the next event will occur and then produce an event at that time. The rest of the DEVS models (quantized integrators, static and implicit functions) work without taking into account the presence of discontinuities but they receive the events coming form the discrete part and treat them as if they were coming from an input generator or another quantized integrator. As a consequence, there are not extra calculations and

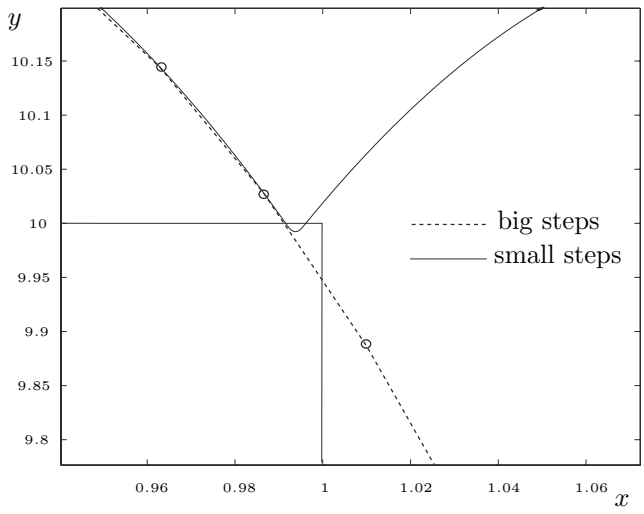

Figure 6.15: Event skipping in discrete time algorithms

there is no need of modifying anything.

6.4 Quantized Bond Graphs

Bond Graphs [56, 9] allow the graphical representation of complex physical systems. As other object–oriented modeling languages, the resulting mathematical models are often sets of DAEs instead of common ODEs.

The most efficient way –from the computational cost point of view– of implementing a QSS simulation of a Bond Graph model is to obtain the equivalent set of ODEs or DAEs and then to apply the results explained in the previous sections. In fact, there are several tools which automatically translate Bond Graph models into sets of equations (Dymola and Modelica for instance) or simple block diagrams (Power DynaMo [38] is an example of this).

However, as it was mentioned in Section 6.2, sometimes it is simpler to perform the simulation directly on the original model.

Bond Graph models are formed by static and dynamic components which relate the power variables e (*effort*) and f (*flow*). The basic static components are the resistors (**R**), sources (**Se** and **Sf**) and the structural elements: junctions (**0** and **1**), transformers (**TF**) and gyrators (**GY**).

The dynamic elements are capacitors **C** and inertias **I**, which establish relationships between efforts and flows with the presence of an integral (or derivative) operation.

The main difference with Block Diagrams is that all the mentioned relationships are not causally defined *a priori*. Anyway, a causalization of these relations can be proposed and represented in the Bond Graph .

In absence of structural singularities (i.e. in absence of coupled storage and resistor elements), a causal assignement can be found (following a standard algorithm) so that the relationships are equivalent to the obtained from a block diagram with integrators.

Taking into account that QSS modifies the original system by adding hysteretic quantizers at the output of its integrators, the only thing which should be modified to apply this method are the capacitors and inertias.

A capacitor defines the following relationships between the power variables and the energy variable (q or displacement):

$$e(t) - g(q(t)) = 0 \tag{6.16a}$$

$$\dot{q}(t) - f(t) = 0 \tag{6.16b}$$

In this element, the *integral causality* assignment provokes that $f(t)$ acts as input and $e(t)$ is the resulting output. The displacement $q(t)$ is the state variable (i.e. the output of the integrator).

The use of the QSS–method here transforms (6.16a) into

$$e(t) = g(q_q(t))$$

being $q_q(t)$ the quantized version of $q(t)$.

In this last equation, the quantization function can be composed together with function g and it can be rewriten:

$$e(t) = g_q(q(t))$$

where now function g_q is a *quantized function* (the composition of a quantization function with a continuous function). Figure 6.16 illustrates a nonlinear function and its quantized version.

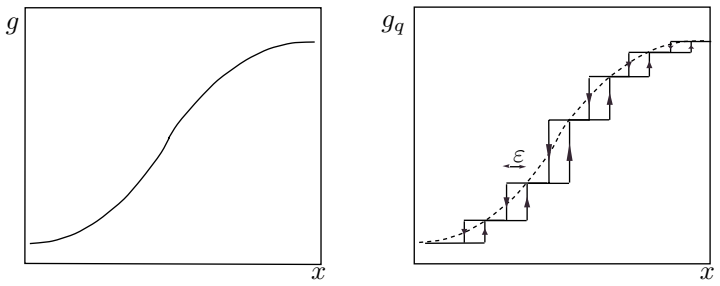

Figure 6.16: A function and its quantized version

The same concepts can be applied to the inertias and then the QSS–method can be applied in a Bond Graph by replacing the static functions of inertias and capacitors by their quantized versions.

The following definitions formalize the ideas expressed above:

Definition 6.1. *Hysteretic Quantized Function.*

Let $x(t)$ be a scalar continuous trajectory. We say that $y(t)$ is related with $x(t)$ by a quantized function if a function $g : \mathbb{R} \to \mathbb{R}$ exists so that

$$y(t) = g(z(t))$$

being $z(t)$ and $x(t)$ related by some hysteretic quantization function.

In that case we also say that $x(t)$ and $y(t)$ are related by the quantized version of function g.

Based on this definition, we can now define

Definition 6.2. *Quantized Capacitor (Inertia).*

A quantized capacitor (inertia) is a capacitor where the displacement (impulse) is related to the effort (flow) with a hysteretic quantized function.

and finally,

Definition 6.3. *Quantized Bond Graph (QBG).*

A quantized Bond Graph is a Bond Graph where the inertias and capacitors are quantized ones.

As it was mentioned before, in absense of structural singularities a causalized Bond Graph is equivalent to a Block Diagram and it defines a state equation system like (3.2) where the state variables are the corresponding energy variables (p and q) of inertias and capacitors.

If in that system the static functions which relate energy and power variables of inertias and capacitors are replaced by their quantized versions, a QBG is obtained. It can be easily seen that if the new equations are written, a system like (3.3) is obtained.

Thus, everything which was proved for the QSS–method is true in Quantized Bond Graph. When it comes to the trajectories, we have that,

Theorem 6.1. *Trajectories in QBG.*

Consider a Quantized Bond Graph without coupled storages, where all the passive and structural component are defined by continuous and bounded relations. Then the trajectories of all power variables are piecewise constant and the trajectories of all energy variables are piecewise linear.

Proof. Under the assumptions made, the application to the QBG of the standard procedure for the derivation of state equations [56], yields a QSS of the form (3.3).

There, because of the assumptions, function f is bounded and continuous in any compact domain. This property, along with Theorem 3.1, guarantees that the trajectories of the quantized energy variables are piecewise constant. The power variables are consequentely also piecewise constant, because they are computed from the former variables via static relationships.

Then, it follows that the energy variables are piecewise linear since they can be calculated as the integral of the power variables. □

Based on this result and taking into account that the Bond Graph elements are related between them by the power variables, an exact DEVS representation of each QBG component can be obtained.

The static element **R** calculates a piecewise constant effort or flow according to a piecewise constant flow or effort. Thus, they can be represented by a model like M_2.

Similar ideas can be applied to transformers and gyrators, but now they calculate two piecewise constant power variables and the DEVS model should be modified adding a new output port. A possible DEVS model for a transformer or gyrator (which relate efforts and flows with a function h) is the following one:

$$
\begin{aligned}
&M_{13} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where}\\
&X = Y = \mathbb{R} \times \mathbb{N}\\
&S = \mathbb{R}^2 \times \mathbb{N} \times \mathbb{R}_0^+\\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(u_1, u_2, m, \sigma) = (u_1, u_2, m, \infty)\\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}((u_1, u_2, m, \sigma), e, (x_v, p)) = (\tilde{u}_1, \tilde{u}_2, p, 0)\\
&\lambda(s) = \lambda(u_1, u_2, m, \sigma) = (h(u_m), m)\\
&ta(s) = ta(u_1, u_2, m, \sigma) = \sigma
\end{aligned}
$$

where

$$
\tilde{u}_i = \begin{cases} x_v & \text{if } i = p \\ u_i & \text{otherwise} \end{cases}
$$

The case of multiport junctions **0** and **1** can be treated in the same way (but now there are more than two inputs)

When it comes to the dynamic capacitors and inertias, the model will be the almost same than the quantized integrator. The only difference is that now the output is not the quantized variable but the power variable.

With this, the model is just the same as before (M_4 in page 33) with only a change in the output function, which now becomes:

$$\lambda(s) = \lambda(x, d_x, j, \sigma) = (g(Q_{j+\text{sgn}(d_x)}), 1)$$

being g the static function relating the corresponding energy and power variables.

The sources (**Se** and **Sf**), as in the QSS case, can be represented as DEVS generators.

Then, the QBG will be represented by a coupled DEVS model formed by the DEVS models corresponding to the Bond Graph elements.

The coupled DEVS structure will be the one shown by the Bond Graph model, where each arpoon is replaced by two connections between the corresponding elements. Those connection will carry the effort and flow values according to the causality assignement. Figure 6.17 illustrates this idea.

In absense of structural singularities a QBG defines a QSS. Thus, its theoretical properties –stability, convergence and error bound– will be the same shown by the corresponding QSS.

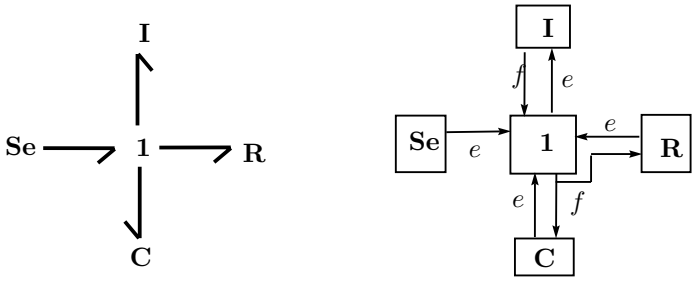

Figure 6.17: DEVS structure of a Quantized Bond Graph

Example 6.5. *QBG Simulation of a DC–Motor.*

The permanent magnet DC–motor of Figure 6.18 can be represented by the Bond Graph of Figure 6.19.

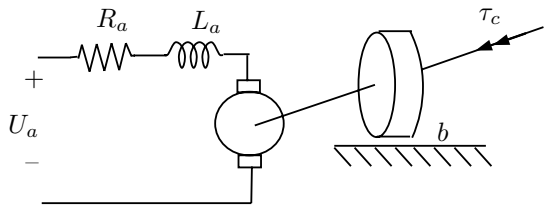

Figure 6.18: Permanent Magnet DC–Motor

Tha parameters adopted were $R_a = 0.01$, $L_a = 1 \times 10^{-5}$, $b = 1$, $J = 1$ *and* $k_m = 1$. *The simulation consists in the response of the initially relaxed system to an armature voltage step of* $U_a = 10$ *at* $t = 0$ *and a load torque step of* $\tau_c = 10$ *applied at* $t = 1$.

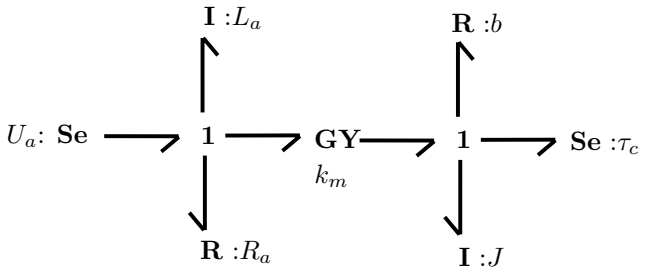

Figure 6.19: Bond Graph model of a DC–Motor

The QBG was obtained using a quantum equal to 1×10^{-5} *in the inductance and* 0.1 *in the mechanical inertia.*

The results are shown in Figures 6.20–6.22.

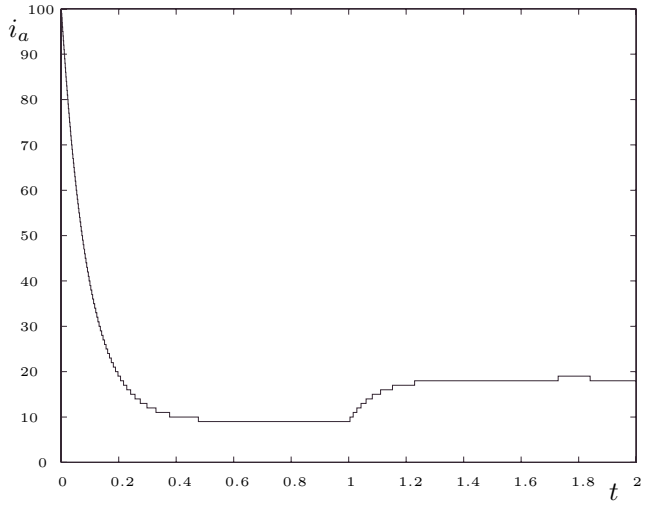

Figure 6.20: Armature current in the DC motor.

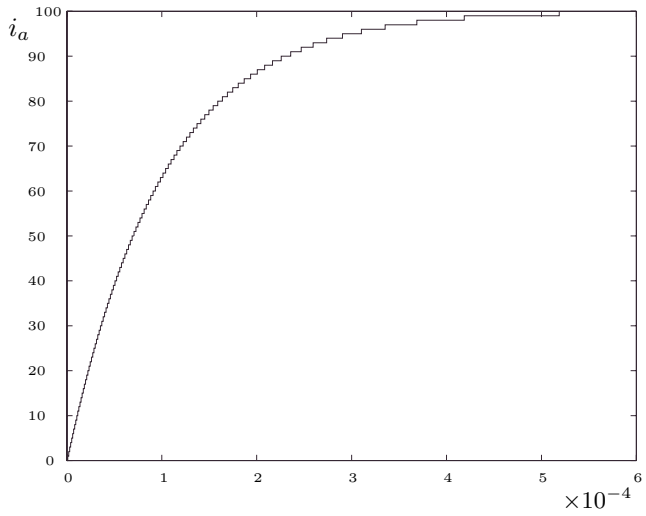

Figure 6.21: Start of the armature current.

The number of internal transitions performed by the quantized inertias during 2 seconds of simulation were 203 and 103 in the electrical and mechanical part respectively.

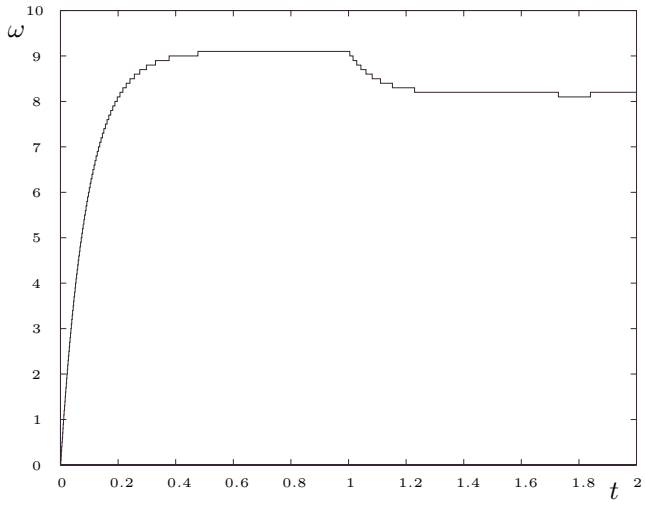

Figure 6.22: Angular speed in the DC motor.

The stiffness of this system can be easily seen comparing Figures 6.21 and 6.22. There, the startup of the armature current is very fast with respect to the speed evolution.

In fact, the evolution matrix is

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{J} \\ \frac{km}{L_a} & -\frac{b}{J} \end{bmatrix}$$

whose eigenvalues are located at -9990 *and* -11.011.

The use of (4.50) *here gives error bounds of* 3.006×10^{-5} *in the inductance flux and* 0.1004 *in the mechanical impulse (i.e.* 3.006 *in the current and* 0.1004 *in the angular speed).*

In this case, the QSS–method applied through the QBG approach worked incredibly well. It performed the complete simulation with only 306 steps while most classic method require much more than this. In fact, similar results with Euler's method requires at least 10000 steps, Runge Kutta 8000 and Runge-Kutta 4–5 (ode45 in Matlab) 6000. Only implicit variable–step methods can solve the problem with less than 100 steps, but the computational complexity of each step is considerably greater than our case.

Anyway, as it was mentioned before, QSS does not work in general stiff systems. This problem will be treated again later on.

6.5 QBG and Structural Singularities

The use of QBG in absense of structural singularities does not differ significantly from the simulation of simple Block Diagrams. The presence of these

singularities, however, changes everything.

When it comes to coupled resistors the resulting system has an algebraic loop which results in an index 1 DAE. Then, what was developped in Section 6.2 can be applied by adding some loop–breaking model between coupled resistors.

The case of coupled storage elements is as bit more complex since it results in derivative causalities which yield higher index DAEs. Thus, this case cannot be treated following what we already developped.

One possible way is to use Pantelides' algorithm in order to reduce the index. However, this solution implies dealing with the resulting equations and, in that way, we are coming back to simulation of DAEs without making use the Bond Graph structure.

In the higher index case, an alternative solution may result from the following remarks. Consider –for instance– a tank as the one shown in Figure 6.23.

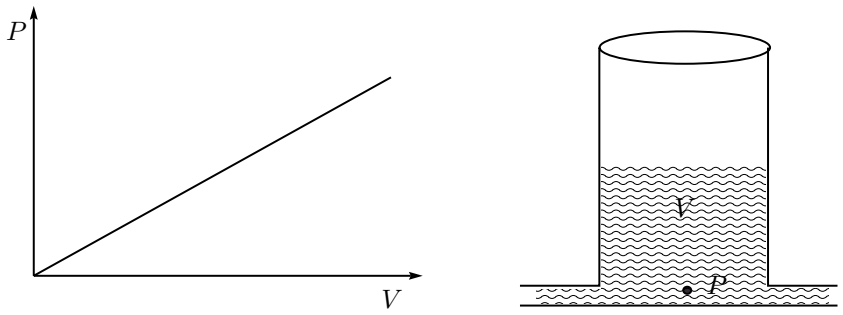

Figure 6.23: A tank with a linear relation P vs. V

This tank is represented by a capacitor in Bond Graph. The use of QBG results in the quantization of the linear characteristic P vs. V. Leaving aside the hysteresis, the quantized function can be physically interpreted as it is shown in Figure 6.24.

Here, it should be considered that the height of each compartment goes to zero while the area goes to infinite so that the volume remains the same.

Making use of this idea, the following example is introduced:

Example 6.6. *Coupled tanks.*

Consider the hydraulic system of Figure 6.25.

This system can be represented by the Bond Graph of Figure 6.26

In this example, the use of Quantized Bond Graph is equivalent to simulate the physically quantized system shown in Figure 6.27.

In a continuous system one capacitor should be in derivative causality, since it is impossible that both compute effort simultaneously.

Nevertheless, the system of Figure 6.27 works in a different way. Suppose feeding the tank system from zero initial conditions. The volume in the tank on the left begins to grow, without liquid flowing into the tank on the right (because the diameter of its first column is almost zero).

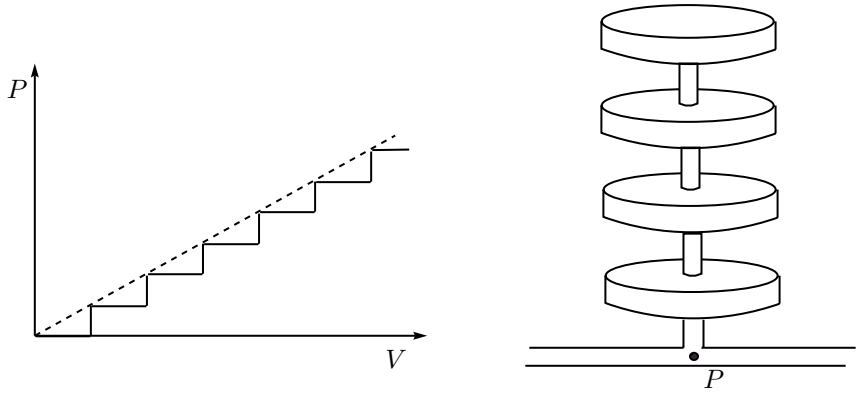

Figure 6.24: Physically quantized tank.

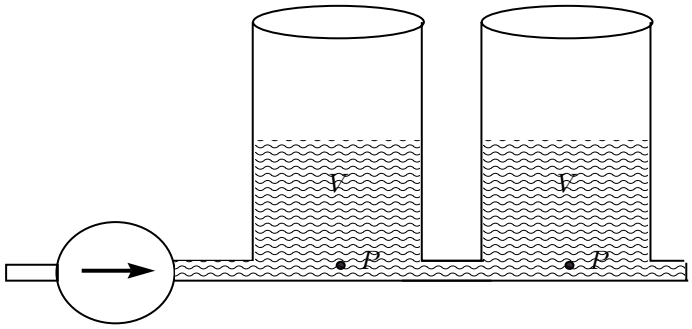

Figure 6.25: Two coupled tanks.

The situation reverses after the bottom recipient of the tank on the left is full: the volume in the second compartment on the right begins to grow, without liquid flowing now into the tank on the left.

The pressure at the bottom of the system is determined exclusively by the compartment being filled at each instant of time. The pressure changes discontinuously immediately after each compartment becomes full.

This behavior indicates that integral causality alternates in time from one to the other capacitor, what in turn reminds the behavior of Switched Bond Graphs [60], where the Switch impresses either zero flow or zero effort

The previous idea can also applied to the case of having coupling of many storages, including both, inertias and capacitors. This situation would result in a model with several alternative causal configurations, in dependence of the values of the energy variables.

The presence of hysteresis does not modify the concepts underlying the previous considerations.

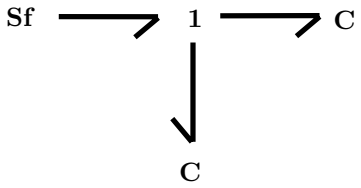

Figure 6.26: Bond Graph model of two coupled tanks

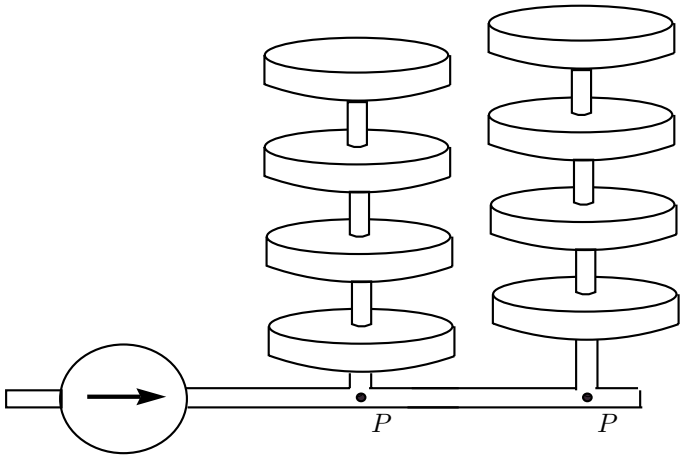

Figure 6.27: Two quantized coupled tanks.

An interesting consequence of this analysis is that we may be able of simulating higher order DAEs based on a switching behavior instead of index reduction and iterations.

However, this conjecture requires a deeper research and a generalization of these ideas for the QSS2–method is not evident.

Anyway, the application of QSS to Bond Graphs shows some interesting advantages related to the simplicity and the possibility of simulating directly on the Bond Graph without algebraic manipulation.

Chapter 7

Quantized–State Control

Practical implementations of most control systems require the use of digital devices. Due to the different nature of the signals present at the inputs and outputs of the digital controller and the continuous-time plant, the interconnection between them must be made through A/D and D/A converters.

The digital implementation of continuous controllers can be thought as a problem of ODE discretization, which is usually performed by Euler or similar approximating *discrete time* rules.

Consequently, the A/D conversion techniques perform synchronous sampling leaving the plant without control between successive sample instants. This fact usually provokes a lost of performance in the dynamic response and affects the regions of attraction in nonlinear systems.

Beside this problem, the A/D conversions use only a finite number of bits producing undesirable effects such as steady state errors and oscillations. Because of quantization problems attracting sets must be considered instead of equilibrium points, and *ultimate boundedness* of solutions instead of asymptotic stability.

Taking into account that the QSS–method avoids time discretization, it is natural to think that its use instead of Euler could provide a solution to the intersampling problem.

Of course, the time discretization avoidance at the digital controller should be complemented with *aynchronous* sampling at the A/D and D/A converters so the plant is always under control.

When it comes to the quantization effects, it is already known that QSS introduces final oscillations. However, as it was seen in the theoretical study, the ultimate bound can be limited with the choice of the quantization. This fact, translated into control, implies that the quantization effects can be bounded at the design stage.

These ideas –approximating the continuous controller with the QSS–method and using asynchronous sampling– will lead to the definition of a new digital *discrete event*–based control scheme.

The new method –which will be called Quantized State Control (QSC)– is

an alternative way to implement digital controllers which, in some cases, show a significant improvement in the dynamic response and quantization effects of the converters with respect to discrete time approximations.

From the theoretical point of view, the properties of QSC will result similar to the QSS ones, allowing to ensure stability, convergence and error bound of the implementation under different conditions.

7.1 Asynchronous Sampling

In all digital control schemes A/D and D/A conversions are performed in a synchronous way. As it was mentioned before, one of the goals is to avoid time discretization and then an asynchronous sampling scheme would be desirable.

D/A converters are devices of asynchronous nature but their digital inputs always change at a given rate. Anyway, they can be used with digital inputs which change asynchronously without any special consideration provided that the changes are not too fast with respect to the internal circuits dynamics.

Then, if there is a digital device producing output data in an asynchronous way, this digital output can be directly connected to a D/A converter and the corresponding analog signal will be obtained with a very small delay (the D/A conversion times are very short).

The case of A/D converters is more complex since the conversion period is usually much longer than the D/A case. In spite of the fact that a conversion can be ordered at any time, the digital result will not be immediatly available. Such kind of delays may result completely unacceptable under the goal of avoiding time discretization. Moreover, it is not clear when the asynchronous conversion should be made since the signal to be converted is the continuous output of the plant.

A different way of performing A/D sampling is proposed by Sayiner et al. in [57]. The idea there is using an A/D converter which only performs conversions when the difference between the analog input and the digital output is bigger than the given resolution (see Figure 7.1).

This kind of A/D conversion can easily implemented with a scheme as the shown in Figure 7.2

Here, the A/D converter is implemented with a counter, a D/A converter and a comparer. The output of the counter is connected to the D/A and the D/A output is compared with the signal to be sampled. When the difference is greater than some threshold (the quantization interval), the value in the counter is increased (or decreased according to the sign of the difference). Then, the value of this counter corresponds to the digital output of the asynchronous A/D converter.

The delay of this new scheme is insignificant compared with a standard A/D converter because the digital output is refreshed as soon as the comparer detects the threshold crossing.

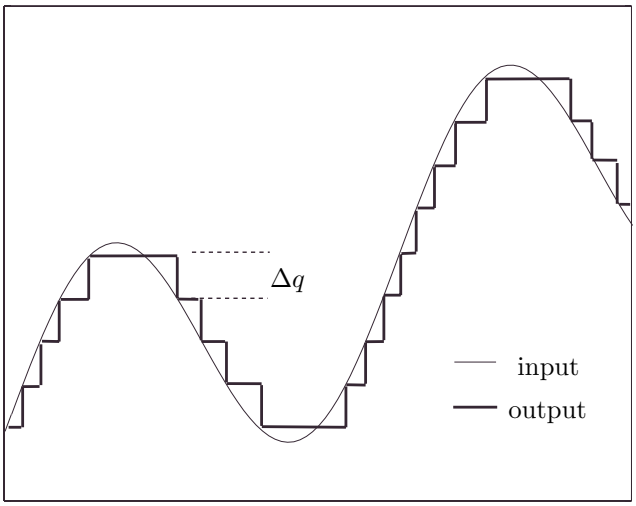

Figure 7.1: Asynchronous A/D conversion

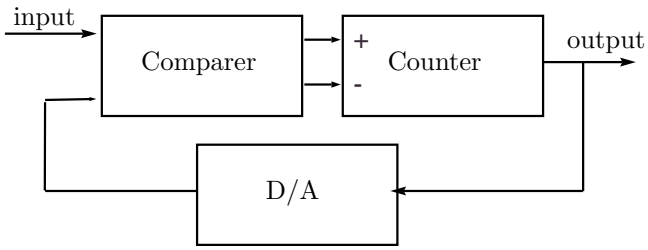

Figure 7.2: Asynchronous A/D Converter

7.2 The QSC Scheme

Quantized State Systems can be exactly represented by DEVS models, as it was seen along the previous chapters. Taking into account that DEVS models can be simulated in–real time[1] [65], a real–time approximations of ODEs based on the QSS–method can be implemented in a digital device.

With this, the main definitions of Quantized State Control can be introduced:

Consider the CCS consisting of plant and controller, Eqs.(7.1) and (7.2) respectively, and their (ideal) interconnection, Eq.(7.3).

$$\left\{ \begin{array}{lcl} \dot{x}_p(t) & = & f_p(x_p(t), u_p(t), t) \\ y_p(t) & = & g_p(x_p(t), t) \end{array} \right. \tag{7.1}$$

[1]DEVS representation of QSS is exact. However, real time simulation of DEVS has errors related to the temporal resolution and the round-off introduced by the digital device

$$\begin{cases} \dot{x}_c(t) &= f_c(x_c(t), u_c(t), u_r(t)) \\ y_c(t) &= g_c(x_c(t), u_c(t), u_r(t)) \end{cases} \tag{7.2}$$

$$u_p(t) = y_c(t), \;\; u_c(t) = y_p(t) \tag{7.3}$$

It is being considered here that the plant could be time varying. When it comes to the controller, it is assumed that it is stationary and it has an input reference $u_r(t)$.

Definition 7.1. *QSC Controller.*

A QSS associated to a continuous controller (7.2) *is called Quantized State Controller (QSC controller).*

Definition 7.2. *QSC System.*

A QSC system is defined as a control scheme composed by a continuous plant and a QSC controller connected through asynchronous A/D and D/A converters.

According to these definitions, a QSC implementation requires to design a continuous controller, to choose the quantization to be applied at each of its state variables to obtain the corresponding QSS and finally to choose the quantization at the A/D and D/A converters.

Figure 7.3 shows a block diagram representation of a QSC system.

The QSC implementation of the controller transforms (7.2) into the new set of equations:

$$\begin{cases} \dot{x}_c(t) &= f_c(q_c(t), u_c(t), u_r(t)) \\ y_c(t) &= g_c(q_c(t), u_c(t), u_r(t)) \end{cases} \tag{7.4}$$

Figure 7.1 also shows that the input and output of the asyncronous A/D converter are related in fact by a hysteretic quantization function where the hysteresis width ε is equal to the quantum Δq.

The D/A converters also introduce quantization (because they have a finite number of bits). However, now there is no hysteresis since they behave in a memoryless way.

As a consequence, the presence of the asynchronous converters transforms (7.3) into:

$$u_p(t) = y_{c_q}(t), \;\; u_c(t) = y_{p_q}(t) \tag{7.5}$$

where the variables $y_{p_q}(t)$ and $y_{c_q}(t)$ are the quantized versions of the plant and the controller output variables

7.3 QSC and Perturbations

The goal of QSC is to obtain a digital approximation to the originally continuous controller. Since the controller is originally designed to achieve a given goal, it should expected that the QSC controller behaves in a similar way.

Taking into account this, the first thing which should be ensured is that the QSC system conserves is some way the stability properties of the continuous control system.

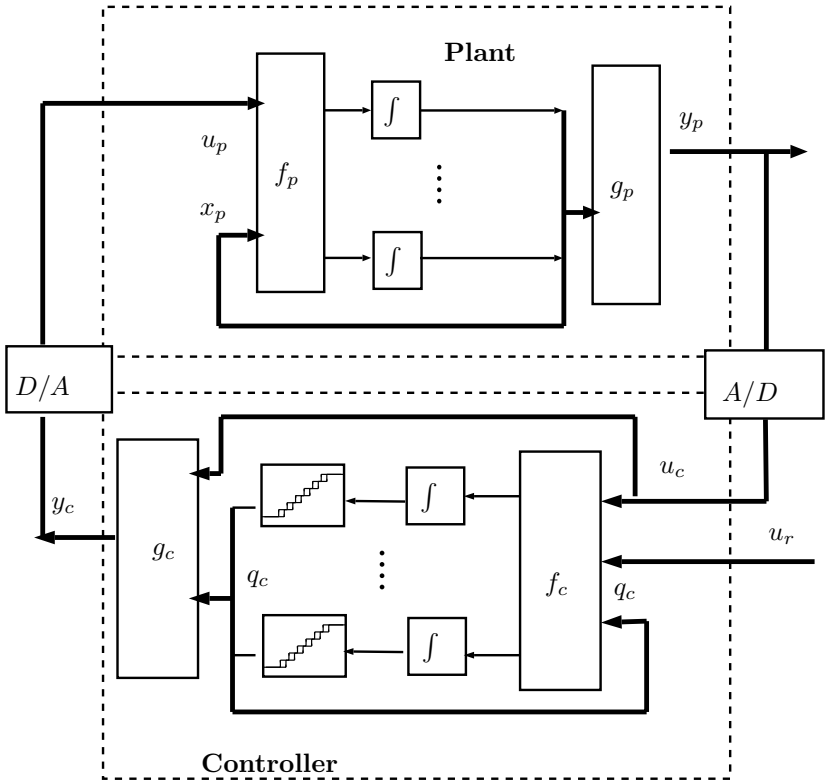

Figure 7.3: Block Diagram of the QSC system

In classical digital control there are some conditions that the sampling period must accomplish in order to conserve the system stability. For instance, in LTI systems, the sampling rate must be greater than the *Nyquist frequency*.

In the QSC case, it is natural to expect finding some conditions about the quantization in the controller and converters to ensure stability in the resulting scheme.

Let us then see first the most general case, i.e., a possibly nonlinear and time varying plant with a nonlinear controller.

Here, the CCS closed loop equations can be derived from Equations (7.1)–(7.3) arriving to

$$\begin{cases} \dot{x}_p &= f_p(x_p, g_c(x_c, g_p(x_p, t), u_r), t) \\ \dot{x}_c &= f_c(x_c, g_p(x_p, t), u_r) \end{cases} \tag{7.6}$$

Let us define

$$\Delta x_c(t) \triangleq q_c(t) - x_c(t) \tag{7.7a}$$
$$\Delta y_p(t) \triangleq y_{p_q}(t) - y_p(t) \tag{7.7b}$$
$$\Delta y_c(t) \triangleq y_{c_q}(t) - y_c(t) \tag{7.7c}$$

Thus, from these definitions and Equations (7.1), (7.4) and (7.5), the QSC closed loop equations can be written as:

$$\begin{cases} \dot{x}_p = f_p(x_p, g_c(x_c + \Delta x_c, g_p(x_p, t) + \Delta y_p, u_r) + \Delta y_c, t) \\ \dot{x}_c = f_c(x_c + \Delta x_c, g_p(x_p, t) + \Delta y_p, u_r) \end{cases} \tag{7.8}$$

Then, the QSC system (7.8) can be seen as a perturbed version of the original CCS (7.6). Moreover, taking into account what is already known about quantization functions, the perturbations in QSC are bounded componentwise by the corresponding quantum. This is true provided that the variables x_c, y_p and y_c do not reach the corresponding saturation bounds. From here to the end, it will be considered that the non–saturation region is enough large so that the trajectories do not leave it.

Based on this observation, the properties of the QSC implementation of a CCS can be studied by looking at the effects of bounded perturbations in the original closed loop system. This is completely analogous to what happened when the properties of the QSS approximation of ODEs were studied.

In that last case, the stability could not be ensured due to the presence of non–vanishing perturbations. Here is the same and all what can be expected is to achieve *ultimately boundedness* of solutions.

Moreover, it will be seen in the following Sections that all the properties studied in QSS will result very similar in QSC. However, the eventual dependence of f_p and g_p on t will add an extra complications to the analysis.

Thus, we shall start studying the properties of QSC in time invariant plants and then we shall come back to the general time varying case.

7.4 Stability of Time Invariant QSC Systems

The stability properties of a system similar to (7.8) related to the stability of (7.6) was already studied. In fact, Theorem 4.2 established conditions to ensure the ultimately boundedness of the Time Invariant (TI) version of (7.8) without the perturbation terms Δy_p and Δy_c.

Taking into account that result, this Section will try to extend and use it in the particular case of a QSC controller with a time invariant plant.

In this case –considering also that the reference u_r is zero or constant– the CCS equations (7.1) and (7.2) can be rewritten:

$$\begin{cases} \dot{x}_p(t) = f_p(x_p(t), u_p(t)) \\ y_p(t) = g_p(x_p(t)) \end{cases} \tag{7.9}$$

$$\begin{cases} \dot{x}_c(t) &= f_c(x_c(t), u_c(t)) \\ y_c(t) &= g_c(x_c(t), u_c(t)) \end{cases} \tag{7.10}$$

while the QSC controller can be now represented by

$$\begin{cases} \dot{x}_c(t) &= f_c(q_c(t), u_c(t)) \\ y_c(t) &= g_c(q_c(t), u_c(t)) \end{cases} \tag{7.11}$$

With this new set equations, the following theorem can be stated:

Theorem 7.1. *Stability of TI QSC.*

Consider that the origin of the closed loop CCS (7.9)–(7.10) *is a regionally stable equilibrium point. Suppose that the functions f_p, g_p, f_c and g_c are continuously differentiable. Further assume that a Lyapunov function V is known, defined in an open region D containing the origin. Then, a QSC system associated to the original CCS can be found, such that all initial conditions lying in an arbitrary interior region of D are attracted in finite time to another arbitrary interior region of the former one. Both interior regions must be limited by level surfaces of V.*

Proof. From Equations (7.9), (7.10) and (7.3) the following closed loop equations of the continuous system can be obtained:

$$\begin{cases} \dot{x}_p &= f_p(x_p, g_c(x_c, g_p(x_p))) \\ \dot{x}_c &= f_c(x_c, g_p(x_p)) \end{cases} \tag{7.12}$$

The implementation of the corresponding QSC system –Eqs.(7.11) and (7.5)– transforms (7.12) into:

$$\begin{cases} \dot{x}_p = f_p(x_p, g_c(x_c + \Delta x_c, g_p(x_p) + \Delta y_p) + \Delta y_c) \\ \dot{x}_c = f_c(x_c + \Delta x_c, g_p(x_p) + \Delta y_p) \end{cases} \tag{7.13}$$

where we used the perturbation notation introduced in (7.7a).

Define:

$$\begin{aligned} &\alpha(x, \Delta x_c, \Delta y_p, \Delta y_c) \triangleq \\ &\quad \frac{\partial V}{\partial x_p}(x) \cdot f_p(x_p, g_c(x_c + \Delta x_c, g_p(x_p) + \Delta y_p) + \Delta y_c) + \\ &\quad \frac{\partial V}{\partial x_c}(x) \cdot f_c(x_c + \Delta x_c, g_p(x_p) + \Delta y_p) \end{aligned} \tag{7.14}$$

where:

$$\begin{aligned} \frac{\partial V}{\partial x_p}(x) &= \left[\frac{\partial V}{\partial x_{p_1}} \cdots \frac{\partial V}{\partial x_{p_n}} \right](x) \\ \frac{\partial V}{\partial x_c}(x) &= \left[\frac{\partial V}{\partial x_{c_1}} \cdots \frac{\partial V}{\partial x_{c_k}} \right](x) \end{aligned}$$

with n and k being the order of the plant and the controller respectively and $V(x) = V(x_p, x_c)$ is the Lyapunov function of the closed loop system defined in (7.12). From Equation (7.14) it can be easily seen that:

$$\alpha(x, 0, 0, 0) = \dot{V}(x)\Big|_{(7.12)} \tag{7.15}$$

Let D_1 be an interior region of D ($D \subset \mathbb{R}^{n+k}$) limited by some level surface of V. Let D_2 be an interior region of D_1 also limited by a level surface of V. Let D_3 be the region defined by $D_3 = D_1 - D_2$.

Since $\dot{V}(x)$ is negative definite, it is possible to find a positive number s so that:

$$\dot{V}(x) < -s, \forall x \in D_3 \tag{7.16}$$

Let α_M be the function defined by:

$$\alpha_M(\Delta x_c, \Delta y_p, \Delta y_c) \triangleq \sup_{x \in D_3} (\alpha(x, \Delta x_c, \Delta y_p, \Delta y_c)) \tag{7.17}$$

From (7.15) and (7.16), it follows that:

$$\alpha_M(0, 0, 0) < -s \tag{7.18}$$

Since function α is continuous, function α_M results continuous. From this property and (7.18), given an arbitrary number s_1 ($s > s_1 > 0$), it is possible to find a positive number ρ so that the condition:

$$\|(\Delta x_c, \Delta y_p, \Delta y_c)\| < \rho \tag{7.19}$$

implies that:

$$\alpha_M(\Delta x_c, \Delta y_p, \Delta y_c) < -s_1 \tag{7.20}$$

Taking into account that the perturbations are bounded by the respective quanta, the condition given in (7.19) can be satisfied with the choice of an adequate quantization.[2]

Let $\phi(t)$ be a solution of Equation (7.13) for the initial condition $\phi(t = 0) = x_0 \in D_3$. Consider that the quantization was done in order to satisfy the condition given by (7.19). From (7.13) and (7.14) it follows that:

$$\begin{aligned} \alpha(\phi, \Delta x_c, \Delta y_p, \Delta y_c) &= \frac{\partial V}{\partial x_p}(\phi) \cdot \dot{\phi}_p + \frac{\partial V}{\partial x_c}(\phi) \cdot \dot{\phi}_c = \\ &= \frac{\partial V}{\partial x}(\phi) \cdot \dot{\phi} \end{aligned}$$

[2]For instance, considering the same uniform quantization for all the quantized variables the mentioned condition can be achieved by taking:

$$\Delta q < \frac{\rho}{\sqrt{k + m + p}}$$

where Δq are the quantization size (equal to the hysteresis width), k is the controller order (i.e. is the size of Δx_c), p is the number of output variables of the plant (size of Δy_p) and m is the number of input variables of the plant (size of Δy_c).

Using (7.17) and (7.20) in the equation above, we have:

$$\frac{\partial V}{\partial x}(\phi) \cdot \dot{\phi} < -s_1 \tag{7.21}$$

This condition will be satisfied at least during certain time while $\phi(t)$ remains inside D_3 (this is guaranteed by the continuity of $\phi(t)$). After integrating both sides of Inequality (7.21), we have:

$$\begin{aligned} \int_0^t \frac{\partial V}{\partial x}(\phi) \cdot \dot{\phi} \cdot dt &< \int_0^t -s_1 \cdot dt \\ V(\phi(t)) - V(\phi(0)) &< -s_1 \cdot t \\ V(\phi(t)) &< V(x_0) - s_1 \cdot t \end{aligned}$$

This implies that V evaluated along the solution is bounded by a strictly decreasing function while that solution remains inside D_3. Since the value $V(x_0)$ is smaller than the value that V takes in the bound of D_1, it is clear that the trajectory will never leave D_1.

Let V_1 be the value that V takes in the bound of region D_2. Then, it can be easily seen that the trajectory will reach the region D_2 in a finite time t_1 with:

$$t_1 \triangleq \frac{V(x_0) - V_1}{s_1} \tag{7.22}$$

which completes the proof. □

Corollary 7.1. *Semiglobal Stability of QSC.*

When the Lyapunov function derivative is negative definite in all the state space, the QSC implementation can assure semiglobal ultimately boundedness.

The proof of this corollary is straightforward. Achieving semiglobal stability requires enlarging the region D_1. Unfortunately, it also implies enlarging the saturation region and then, global stabilization cannot be assured in general cases.

Equation (7.19) gives the maximum perturbation allowed to ensure the achievement of the proposed goal (i.e. region of attraction D_1 and ultimate bound inside D_2). Since the maximum perturbation in each variable is given by the corresponding quantum, this equation should be used to choose the quantum at the different controller state variables and converters completing in that way the QSC design.

Observe that D_1 is also the estimation of the region of attraction of the CCS using the Lyapunov function V. Then, a QSC implementation can be found so that it conserves the estimated region of attraction.

The presence of quantization destroyes the asymptotic stability. However, ultimately boundedness of the solutions can be still ensured. Moreover, the ultimate region D_2 can be arbitrarily chosen. Anyway, if it is chosen to be too small then the quantum will also result too small and the computational costs will increase over what can be practically implemented since the rate of events in the controller is approximately proportional to the inverse of the quantum.

7.5 Design Procedure for TI QSC

Based on the previous ideas, the design of a QSC contoller can be divided in two steps. The first one is the design of the continuous controller, which can be done following any technique.

The second step is the choice of the quantization at each variable. The use of a very small quantum yields solutions whose ultimate bound can be reduced to arbitrary small values, as it was shown in Theorem 7.1.

However, as it was already mentioned, the use of a small quantum increases the number of events at the controller and the digital device can fail in its attempt to give the correct output values at the required time.

Therefore, there is always a trade–off between accuracy and practical considerations related to the computational costs. Then the idea is to exploit Theorem 7.1 in order to choose the quantization according to some essential features (region of attraction and ultimate bound). In that way, the quantization adopted will be just as small as necessary to ensure those properties and –provided that the CCS is not too fast– the digital device will be able to correctly implement the resulting QSC controller.

The translation of these ideas into a design algorithm for QSC can be written as follows:

1. Design a continuous controller and calculate the Lyapunov function $V(x)$ for the closed loop CCS.

2. Identify the region D where the where the derivative of the Lyapunov function is negative and choose the QSC region of attraction D_1 inside it and the ultimate region D_2 according to the control goals

3. Obtain the perturbed closed loop equations according to (7.13).

4. Obtain function α according to (7.14) and α_M according to (7.17).

5. Calculate constant s according to (7.16) and choose the positive constant $s_1 < s$. If the goal is just to ensure ultimately boundedness s_1 should be very small. Otherwise, if the speed of convergence is also important, it can be chosen taking into account (7.22).

6. Find the maximum value of ρ so that (7.19) implies (7.20).

7. Choose the quantization at the controller state variables and converters so that (7.19) is satisfied.

8. Choose the saturation bounds of the quantization functions outside the region D_1

It can be seen that this procedure leads to a QSC controller which ensures region of attraction D_1 and ultimate region D_2.

Example 7.1. *Stabilization of a TI Nonlinear Plant.*

Consider the plant:

$$\begin{cases} \dot{x}_p(t) &= x_p^2 + u_p \\ y_p(t) &= x_p(t) \end{cases} \tag{7.23}$$

It will be supposed that the goal is stabilizing the plant around the origin. The first step in the algorithm is the design of a continuous controller. For instance, the following controller can achieve the mentioned goal.

$$\begin{cases} \dot{x}_c(t) &= -x_c - u_c \\ y_c(t) &= x_c(t) - u_c(t) - u_c^2(t) \end{cases} \tag{7.24}$$

The resulting closed loop equations are:

$$\begin{cases} \dot{x}_p(t) &= -x_p + x_c \\ \dot{x}_c(t) &= -x_p - x_c \end{cases}$$

It can be easily verified that the origin is the equilibrium point, and it is asymptotically and globally stable. By taking the Lyapunov function

$$V(x) = \frac{1}{2}x_p^2 + \frac{1}{2}x_c^2 \tag{7.25}$$

it follows that:

$$\dot{V}(x) = -x_p^2 - x_c^2$$

Since the stability is global ($D = \mathbb{R}^2$) the definition of the region D_1 will be only neccessary for the choice of the saturation bounds (at least in this case). Suposse also that the goal is assuring the convergence of the trajectories to the region $D_2 = \{x/\|x\| < 1\}$.

It follows from (7.14), (7.23), (7.24) *and* (7.25) *that:*

$$\begin{aligned} &\alpha(x, \Delta x_c, \Delta y_p, \Delta y_c) = \\ &\quad -x_p^2 - x_c^2 + x_p(\Delta x_c - \Delta y_p - \Delta y_p^2 - 2x_p\Delta y_p + \\ &\quad +\Delta y_c) + x_c(-\Delta x_c - \Delta y_p) \end{aligned} \tag{7.26}$$

The calculation of α_M according to the definition in (7.17) *is quite difficult. However, a bound of this function can be easily obtained. It follows from* (7.26) *that:*

$$\begin{aligned} \alpha(x, \Delta x_c, \Delta y_p, \Delta y_c) \leq &-\|x\|^2 + \|x\|(|\Delta x_c| + |\Delta y_p|) + \\ &+\|x\|(|\Delta x_c| + |\Delta y_p| + |\Delta y_p^2| + 2\|x\|\ |\Delta y_p| + |\Delta y_c|) \end{aligned}$$

Then, it results from (7.17) *and the inequality above that:*

$$\begin{aligned} \alpha_M(\Delta x_c, \Delta y_p, \Delta y_c) \leq \sup_{\|x\| \geq 1} &[-\|x\|^2 + \\ &+\|x\|(2|\Delta x_c| + 2(\|x\| + 1)|\Delta y_p| + |\Delta y_p^2| + |\Delta y_c|)] \end{aligned}$$

Since outside the region D_2 the condition $\dot{V}(x) < -1$ is satisfied, the convergence speed s_1 (fifth step) can be chosen to be bounded between 0 and 1. Suppose that the choice is $s_1 = 0.5$. Then, the quantization must be chosen in order to satisfy $\alpha_M < -0.5$, condition than is verifyied using a quantum $\Delta q = \varepsilon = 0.07$ in all the variables.

If the restriction about the convergence speed is not taken into account and the goal is just assuring stability, that quantization interval of $\Delta q = 0.07$ is sufficiently small to guarantee convergence to the region given by $\|x\| < 0.4127$.

The resulting QSC system was simulated with the initial condition $x_p = 10$, $x_c = 0$. The results are shown in Figures 7.4 to 7.6. The number of conversions performed by the A/D converter was 178 for 40 seconds of simulation time. The minimun time between two successive conversions was 5.6 milliseconds (at the begining of the simulation) while the maximum was greater than 2 seconds.

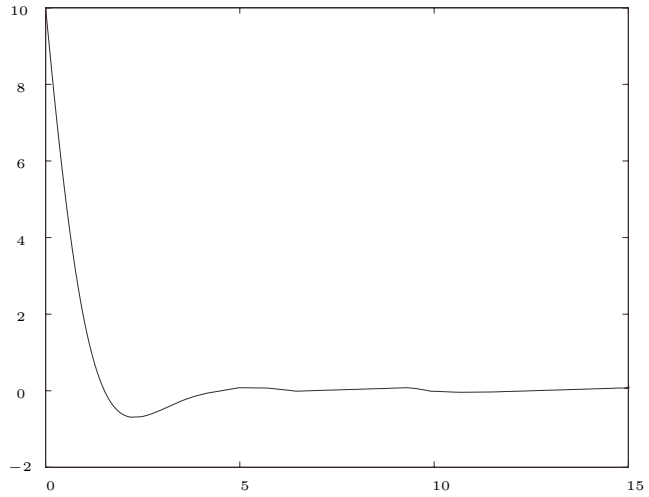

Figure 7.4: Plot of x_p for the plant with QSC

The trajectory of Fig.7.4 seems to be quite similar to the one obtained with the continuous controller except for the final oscillations in Figure 7.5. However, in Fig.7.6 the difference with the CCS behaviour is more evident. There, the phenomena of ultimately boundedness due to quantization and trajectory crossing due to hysteresis can be easily observed.

7.6 Stability of General QSC

Theorem 7.1 was based on the fact that the plant was time invariant. However, that was not the hypothesis made in the general definition of QSC (it was only assumed that the controller was time invariant since time varying ODEs cannot be approximated with the QSS method).

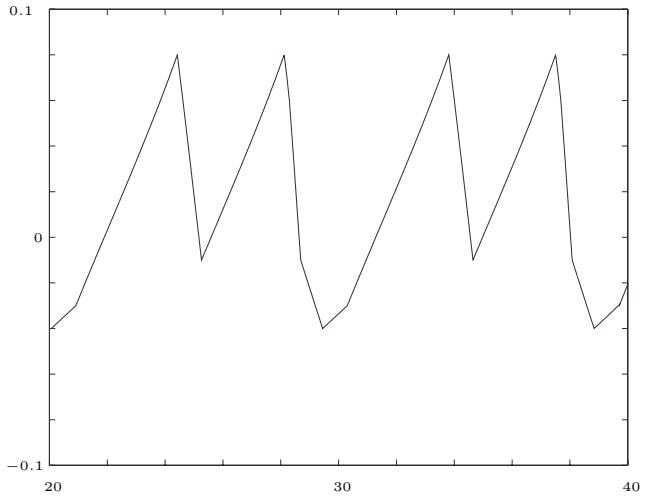

Figure 7.5: Final oscillations in x_p

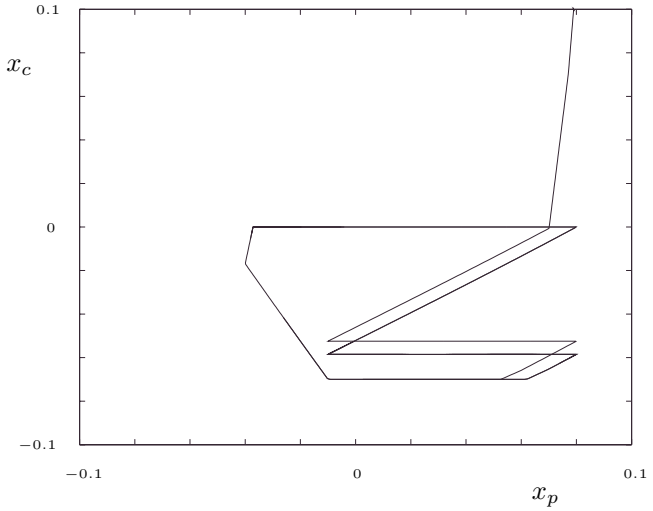

Figure 7.6: Final oscillations in the phase portrait

Thus, it is necessary to extend these results to the general time varying case. Here, when the reference trajectory $u_r(t)$ is zero (or constant), the QSC system (7.8) can be rewritten as

$$\dot{x} = f(x + \Delta x, \Delta y, t) \tag{7.27}$$

where $x \triangleq [x_p, x_c]^T$, $\Delta x \triangleq [0, \Delta x_c]^T$, $\Delta y \triangleq [\Delta y_p, \Delta y_c]^T$, and $f \triangleq [f_p, f_c]^T$.

With these definitions, the CCS (7.6) becomes

$$\dot{x} = f(x, 0, t) \triangleq \tilde{f}(x, t) \tag{7.28}$$

and the problem can be reduced to relate the stability of systems (7.27) and (7.28).

In order to study this problem, the following theorem can be stated:

Theorem 7.2. *Stability of QSC.*

Let the origin be an asymptotically stable equilibrium point of the closed loop CCS (7.28)*. Assume that function f is continuous and a continuously differentiable Lyapunov function $V(x,t)$ is known with*

$$W_1(x) \leq V(x,t) \leq W_2(x) \tag{7.29}$$

$$\frac{\partial V}{\partial x} \cdot \tilde{f}(x,t) + \frac{\partial V}{\partial t} \leq -W_3(x) \tag{7.30}$$

$\forall t \geq 0$, $\forall x \in D$ being D a compact set which contains the origin and W_i are continuous positive definite functions in D.

Let $\Omega_{2_a} = \{x | W_2(x) \leq a\}$ with a being an arbitrary positive constant which is enough small so that $\{x | W_1(x) \leq a\}$ is a closed region inside D.

Let $\Omega_{1_b} = \{x | W_1(x) \leq b\}$ being b an arbitrary positive constant ($b < a$) enough small so that $\Omega_{1_b} \subset \Omega_{2_a}$.

Then, a quantization can be found so that the QSC system trajectories starting in Ω_{2_a} finish inside Ω_{1_b}, reaching this region in finite time.

Note that the conditions that $V(x,t)$ must satisfy –(7.29) and (7.30)– are just the ones which are needed to ensure stability of the original CCS.

Proof. The derivative of $V(x,t)$ along the solutions of the QSC system (7.27) is

$$\begin{aligned} \dot{V}(x,t) &= \frac{\partial V}{\partial x} \cdot f(x + \Delta x, \Delta y, t) + \frac{\partial V}{\partial t} \\ &= \frac{\partial V}{\partial x} \cdot f(x, 0, t) + \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} \cdot (f(x + \Delta x, \Delta y, t) - f(x, 0, t)) \end{aligned}$$

Then, using (7.30) it results that

$$\dot{V}(x,t) \leq -W_3(x) + \frac{\partial V}{\partial x} \cdot (f(x + \Delta x, \Delta y, t) - f(x, 0, t))$$

Consider the sets $\Omega_{2_b} = \{x | W_2(x) \leq b\}$ and $\Omega_{1_a} = \{x | W_1(x) \leq a\}$. From the hypotesis made about a and b, it results that

$$\Omega_{2_b} \subset \Omega_{1_b} \subset \Omega_{2_a} \subset \Omega_{1_a} \subset D$$

Since $W_3(x)$ is positive definite in D, it is positive in $\Omega_{1,2} \triangleq \Omega_{1_a} - \Omega_{2_b}$. Moreover, it exists a positive constant s where

$$s \triangleq \min_{x \in \Omega_{1,2}} W_3(x) \tag{7.31}$$

Let us define the following function

$$\alpha(x, \Delta x, \Delta y, t) \triangleq -W_3(x) + \frac{\partial V}{\partial x} \cdot (f(x + \Delta x, \Delta y, t) - f(x, 0, t)) \tag{7.32}$$

The continuity in functions W_3 and f and the fact that V is continuously differentiable implies that α is continuous. From the definition of α and Eq.(7.31) it results that

$$\alpha(x, 0, 0, t) \leq -s \ \ \forall x \in \Omega_{1,2}, \forall t \geq 0$$

Let α_M be the function defined by

$$\alpha_M(\Delta x, \Delta y) \triangleq \sup_{x \in \Omega_{1,2}, t \geq 0} (\alpha(x, \Delta x, \Delta y, t)) \tag{7.33}$$

It can be easily seen that α_M is continuous and $\alpha_M(0, 0) \leq -s$. Then, for any positive number $s_1 < s$ a positive constant r can be found so that the condition

$$\|(\Delta x, \Delta y)\| \leq r \tag{7.34a}$$

implies that

$$\alpha_M(\Delta x, \Delta y) \leq -s_1$$

and then it results that

$$\dot{V}(x, t) \leq \alpha(x, \Delta x, \Delta y, t) \leq \alpha_M(\Delta x, \Delta y) \leq -s_1$$

in $\Omega_{1,2}$.

From here to the end, the proof follows Theorem 5.3 of [23].

Let $\Omega_{a,t} = \{x | V(x, t) \leq a\}$ and $\Omega_{b,t} = \{x | V(x, t) \leq b\}$ be time variable sets. From (7.29) it results that

$$\Omega_{2_b} \subset \Omega_{b,t} \subset \Omega_{1_b} \subset \Omega_{2_a} \subset \Omega_{a,t} \subset \Omega_{1_a} \subset D$$

The border of $\Omega_{a,t}$ is inside $\Omega_{1,2}$, where $\dot{V}(x, t)$ is negative. It means that the trajectories of the QSC system (7.27) cannot abandon $\Omega_{a,t}$.

Then, any trajectory starting in Ω_{2_a} cannot abandon Ω_{1_a}.

The border of $\Omega_{b,t}$ is also inside $\Omega_{1,2}$. Then the trajectories cannot abandon this time variable set.

To complete the proof, we need to ensure that the trajectories initiated in $\Omega_{2_a} \subset \Omega_{a,t}$ reach $\Omega_{b,t}$ in a finite time.

Let $\phi(t)$ be a solution of (7.27) starting in $\Omega_{a,t}$ (i.e. $V(\phi(0), 0) \leq a$) and let us suppose that

$$V(\phi(t), t) > b \ \ \forall t \tag{7.35}$$

then we have $\dot{V}(\phi(t), t) \leq -s_1$ and after

$$t_1 \triangleq \frac{a - b}{s_1} \tag{7.36}$$

it results that $V(\phi(t_1), t_1) \leq b$ which contradicts (7.35). Then, the region $\Omega_{b,t}$ must be reached before the finite time t_1.

Since $\Omega_{b,t} \subset \Omega_{1_b}$ the trajectory also reaches the set Ω_{1_b} before that time. □

As before, Equation (7.34a) gives the maximum perturbation allowed to ensure the achievement of the proposed goal (now the region of attraction is Ω_{2_a} and the ultimate bound is inside Ω_{1_b}). It is also true that Ω_{2_a} is the estimation of the region of attraction of the CCS using the Lyapunov function V. Then, we can still find a QSC implementation which conserves the estimated region of attraction.

7.7 General Procedure for QSC Implementation

Taking into account the differences between Theorem 7.1 and Theorem 7.2, the procedure introduced in Section 7.5 can be be modified as follows

1. Design a continuous controller and calculate the Lyapunov function $V(x,t)$ and the functions $W_i(x)$ according to (7.29)–(7.30) for the closed loop CCS.
2. Choose the QSC region of attraction Ω_{2_a} and the ultimate region Ω_{1_b} together with the constants a and b.
3. Obtain the perturbed closed loop function f according to (7.27).
4. Obtain function α according to (7.32) and α_M according to (7.33).
5. Calculate constant s according to (7.31) and choose the positive constant $s_1 < s$. If the goal is just to ensure ultimately boundedness s_1 should be very small. Otherwise, if the speed of convergence is also important, it can be chosen taking into account (7.36).
6. Compute the value of r according to (7.34).
7. Choose the quantization at the controller state variables and converters so that (7.34a) is satisfied.

As before, this procedure leads to a QSC controller which ensures region of attraction Ω_{2_a} and ultimate region Ω_{1_b}.

Example 7.2. *Quantized State Control of a Time Varying Plant.*
The unstable time varying plant

$$\left\{ \begin{array}{rcl} \dot{x}_p & = & x_p \cdot (1 + \sin t + \cos t) + u_p \\ y_p & = & (2 + \cos t) \cdot x_p \end{array} \right.$$

can be stabilized by the controller

$$\left\{ \begin{array}{rcl} \dot{x}_c & = & -x_c + u_c \\ y_c & = & -x_c - u_c \end{array} \right. \tag{7.37}$$

The resulting closed loop system can be written as

$$\left\{ \begin{array}{rcl} \dot{x}_p & = & -(1 - \sin t) \cdot x_p - x_c \\ \dot{x}_c & = & (2 + \cos t) \cdot x_p - x_c \end{array} \right.$$

Here, the Lyapunov candidate

$$V(x_p, x_c, t) = x_p^2 + \frac{1}{2}x_c^2 + \frac{1}{2}x_p^2 \cos t$$

verifies (7.29) *with*

$$W_1(x_p, x_c) = \frac{1}{2}x_p^2 + \frac{1}{2}x_c^2 \tag{7.38}$$
$$W_2(x_p, x_c) = \frac{3}{2}x_p^2 + \frac{1}{2}x_c^2$$

The orbital derivative is

$$\frac{\partial V}{\partial x} \cdot f(x,t) + \frac{\partial V}{\partial t} = (2 + \cos t) \cdot (\sin t - 1) \cdot x_p^2 - x_c^2 - \frac{1}{2}x_p^2 \sin t$$

which satisfies (7.30) *with*

$$W_3 = -\frac{1}{2}x_p^2 - x_c^2$$

Then, the closed loop CCS is asymptotically stable and the algorithm resulting from Theorem 7.2 can be used to design the QSC controller.

The first step for the QSC design consists in choosing the region of attraction and the ultimate bound. Since inequalities (7.29)–(7.30) *stand in* $\Re^2$ *(i.e. the CCS stability is global) it is not necessary to restrict the region of attraction except for chosing the saturation values. In this case, the choice of* Ω_{2_a} *does not affect the calculations.*

The ultimate bound Ω_{1_b} *will be chosen with* $b = 0.5$. *Then, taking into account* (7.38) *it results that* $\Omega_{1_b} = \{x \in \Re^2 | \|x\| \leq 1\}$.

The perturbed equations (7.27) *can be written as*

$$\left\{ \begin{array}{lcl} \dot{x}_p & = & -(1 - \sin t) \cdot x_p - x_c - \Delta x_c + \Delta y_c - \Delta y_p \\ \dot{x}_c & = & (2 + \cos t) \cdot x_p - x_c - \Delta x_c + \Delta y_p \end{array} \right.$$

and then, from (7.32) *function* $\alpha(x, \Delta x, \Delta y, t)$ *results*

$$\alpha = -x_c^2 - \frac{1}{2}x_p^2 + x_p \cdot (2 + \cos t) \cdot (-\Delta x_c + \Delta y_c - \Delta y_p) + x_c \cdot (-\Delta x_c + \Delta y_p)$$

Although the maximum α_M *in* (7.33) *cannot be easily obtained, it can be bounded as follows*

$$\begin{array}{rcl} \alpha & \leq & -\frac{1}{2}\|x\|^2 + \sqrt{9 \cdot (|\Delta x_c| + |\Delta y_c| + |\Delta y_p|)^2 + (|\Delta x_c| + |\Delta y_p|)^2} \cdot \|x\| \\ & \leq & \|x\| \cdot (-\frac{1}{2}\|x\| + \sqrt{9 \cdot (|\Delta x_c| + |\Delta y_c| + |\Delta y_p|)^2 + (|\Delta x_c| + |\Delta y_p|)^2}) \end{array}$$

and then, taking into account that $\|x\| > 1/3$ *in* $\Omega_{1,2}$, *it results that*

$$\alpha_M \leq -\frac{1}{18} + \sqrt{(|\Delta x_c| + |\Delta y_c| + |\Delta y_p|)^2 + \frac{1}{9}(|\Delta x_c| + |\Delta y_p|)^2}$$

Then, taking the quantum equal to 0.018 *in the controller and converters we can ensure that*

$$\alpha_M \leq -0.0093$$

in $\Omega_{1,2}$*, which implies that the trajectories finish inside* Ω_{1_b} *in finite time, with a minimum* speed $s_1 = 0.0093$*.*

Figures 7.7–7.12 show the simulation results for an initial condition $x_p = 5$*,* $x_c = 0$*.*

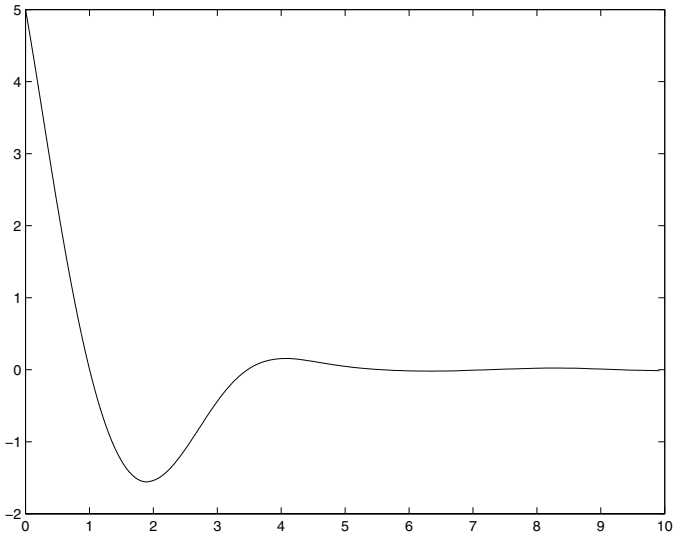

Figure 7.7: x_p vs. t

Figure 7.12 also shows that the design was very conservative. The ultimate bound observed in the simulation is less than 0.03 *(in norm 2) which is more than 30 times smaller than the calculated.*

7.8 Convergence of QSC

It was seen that the QSC implementation can approximate stability properties and convergence speed performance from the original continuous control design. However, it was not said yet anything about the QSC behavior during the transient period.

It will be introduced now the QSC version of Theorem 4.1, where it will be shown that the trajectories of the system with the QSC controller go to the trajectories of the system with the original continuous controller when the quantization go to zero.

It implies that any performance measure achieved by the original continuous controller can be approximately achieved by the QSC controller with the choice of sufficiently small quantization intervals.

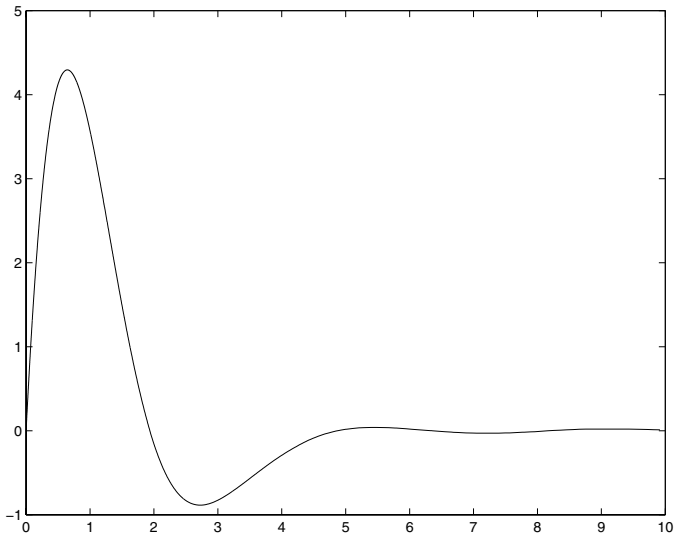

Figure 7.8: x_c vs. t

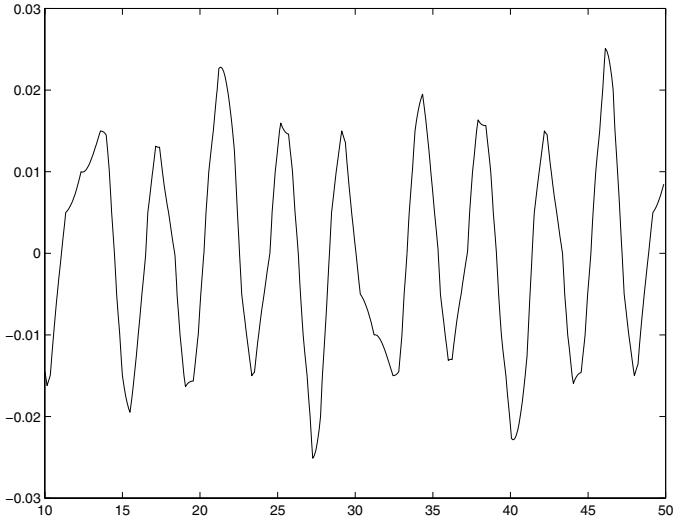

Figure 7.9: Final oscillations in x_p vs. t

Theorem 7.3. *Convergence of QSC.*

Consider the system corresponding to a continuous plant with a continuous controller given by (7.6) *and the associated QSC implementation* (7.8)*. Let D_{x_c}, D_{y_c} and D_{y_p} be the non-saturation regions of the QSC controller, D/A converters and A/D converters respectively (i.e. the regions where the corresponding variables do not reach the saturation bounds). Let D_{x_p} be a bounded region in $\mathbb{R}^n$ and let D be a region of the state space where there is not saturation of any*

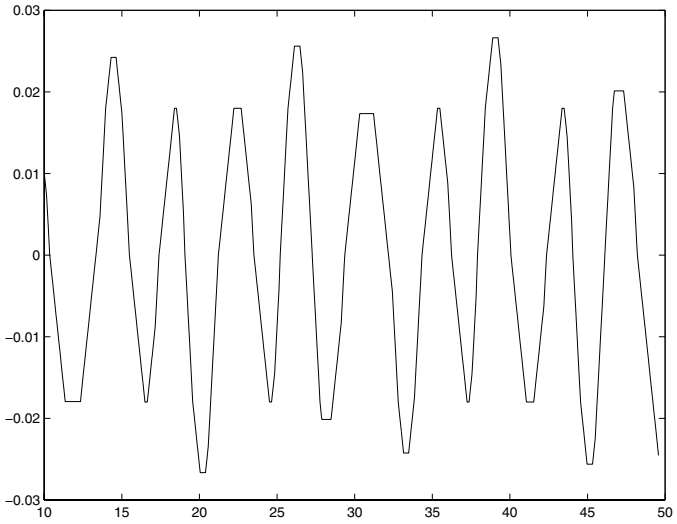

Figure 7.10: Final oscillations in x_c vs. t

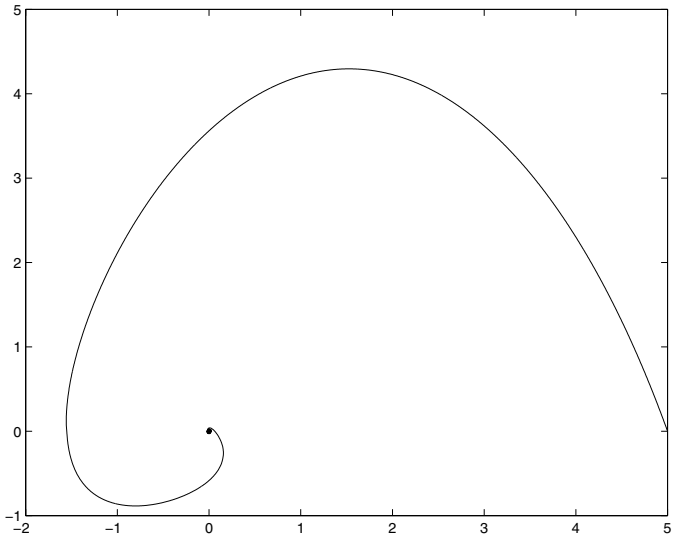

Figure 7.11: x_p vs. x_c

variable, i.e.

$$D \triangleq \{(x_p, x_c) | x_p \in D_{x_p}, x_c \in D_{x_c}, g_p(x_p, t) \in D_{y_p} \forall t, g_c(x_c, g_p(x_p)) \in D_{y_c}\}$$

Assume that functions f_c and g_c are Lipschitz on $D_{x_c} \times D_{y_p}$, function f_p is Lipschitz on $D_{x_p} \times D_{y_c}$ and function g_p is Lipschitz on D_{x_p}. Let $\phi(t)$ be the solution of (7.6) from the initial condition $x(0) = x_0$ and let $\phi_1(t)$ be a solution of (7.8) starting from the same initial condition x_0. Assume that $\phi(t) \in D_1$ $\forall t$ being

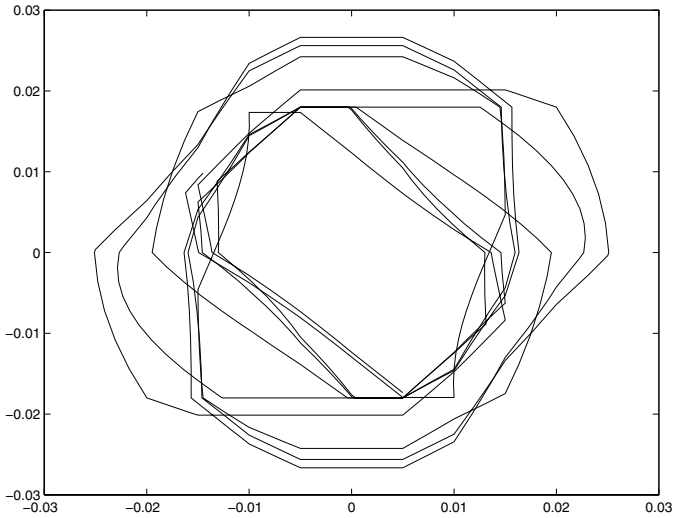

Figure 7.12: Final oscillations in x_p vs. x_c

D_1 a closed interior region of D. Then, $\phi_1(t) \to \phi(t)$ when the quantization intervals go to 0.

Proof. Let $S \triangleq \mathbb{R}^{n+k} - D$ and let

$$D_2 \triangleq D_{x_p} \times D_{x_c} \times D_{y_p} \times D_{y_c}$$

let F be

$$F \triangleq \sup_{(x_p,x_c,y_p,y_c)\in D_2} \left\| \begin{array}{c} f_p(x_p, y_c, t) \\ f_c(x_c, y_p) \end{array} \right\| \tag{7.39}$$

Let d be defined by

$$d \triangleq \inf_{x\in S}(\inf_{t\in[0,\infty]} \|\phi(t) - x\|) \tag{7.40}$$

Since d is positive and F is finite, a positive number t_1 can be taken satisfying

$$t_1 < \frac{d}{F} \tag{7.41}$$

From (7.39), (7.40) and (7.41) it can be seen that $\phi_1(t)$ does not leave the region D during the interval $[0, t_1]$.

In this time interval, the following conditions are fulfilled by System (7.8)

$$\|\Delta x_c\| \leq \Delta_{x_c} \tag{7.42a}$$
$$\|\Delta y_p\| \leq \Delta_{y_p} \tag{7.42b}$$
$$\|\Delta y_c\| \leq \Delta_{y_c} \tag{7.42c}$$

being $\Delta_{x_c}, \Delta_{y_p}$ and Δ_{y_c} constants defined by the quantization intervals in the QSC controller, A/D converters and D/A converters respectively.

It follows from (7.8), (7.6) and the fact that $\phi_1(0) = \phi(0)$ that:

$$\phi_1(t) - \phi(t) = \begin{bmatrix} \phi_{1_p}(t) - \phi_p(t) \\ \phi_{1_c}(t) - \phi_c(t) \end{bmatrix} = \int_0^t \begin{bmatrix} \psi_{1_p}(\tau) - \psi_p(\tau) \\ \psi_{1_c}(\tau) - \psi_c(\tau) \end{bmatrix} d\tau \tag{7.43}$$

where

$$\psi_{1_p}(\tau) \triangleq f_p(\phi_{1_p}, g_c(\phi_{1_c} + \Delta x_c, g_p(\phi_{1_p}, \tau) + \Delta y_p) + \Delta y_c, \tau) \tag{7.44a}$$
$$\psi_p(\tau) \triangleq f_p(\phi_p, g_c(\phi_c, g_p(\phi_p, \tau)), \tau) \tag{7.44b}$$
$$\psi_{1_c}(\tau) \triangleq f_c(\phi_{1_c} + \Delta x_c, g_p(\phi_{1_p}, \tau) + \Delta y_p) \tag{7.44c}$$
$$\psi_c(\tau) \triangleq f_c(\phi_c, g_p(\phi_p, \tau)) \tag{7.44d}$$

Then, applying the Euclidean norm to the first row in (7.43),

$$\begin{aligned} \|\phi_{1_p}(t) - \phi_p(t)\| &= \| \int_0^t \psi_{1_p}(\tau) - \psi_p(\tau) d\tau \| \\ &\leq \int_0^t \|\psi_{1_p}(\tau) - \psi_p(\tau)\| d\tau \end{aligned} \tag{7.45}$$

In a similar way, for the second row, we have

$$\|\phi_{1_c}(t) - \phi_c(t)\| \leq \int_0^t \|\psi_{1_c}(\tau) - \psi_c(\tau)\| d\tau \tag{7.46}$$

From (7.45) and (7.46) it follows that

$$\|\phi_{1_p} - \phi_p\| + \|\phi_{1_c} - \phi_c\| \leq \int_0^t (\|\psi_{1_p}(\tau) - \psi_p(\tau)\| + \|\psi_{1_c}(\tau) - \psi_c(\tau)\|) d\tau \tag{7.47}$$

Taking into account that $\phi(t)$ and $\phi_1(t)$ are inside region D, and taking Δx_c, Δy_c and Δy_p enough small we can assure that the arguments of functions f_p, f_c, g_c and g_p in (7.44a) and (7.44c) are inside the regions where these functions satisfy the Lipschitz conditions. Using these conditions and the definitions (7.44a) and (7.44b), we have

$$\begin{aligned} \|\psi_{1_p}(\tau) - \psi_p(\tau)\| &\leq M_{f_p} \|(\phi_{1_p}, g_c(\phi_{1_c} + \Delta x_c, g_p(\phi_{1_p}, \tau) + \Delta y_p) + \Delta y_c) \\ &\quad -(\phi_p, g_c(\phi_c, g_p(\phi_p, \tau)))\| \\ &\leq M_{f_p}(\|\phi_{1_p} - \phi_p\| + \|\Delta y_c\| + M_{g_c} \\ &\quad \|(\phi_{1_c} + \Delta x_c, g_p(\phi_{1_p}, \tau) + \Delta y_p) - (\phi_c, g_p(\phi_p), \tau)\|) \\ &\leq M_{f_p}(\|\phi_{1_p} - \phi_p\| + \|\Delta y_c\| + M_{g_c} \\ &\quad (\|\phi_{1_c} - \phi_c\| + \|\Delta x_c\| + \|\Delta y_p\| + M_{g_p}\|\phi_{1_p} - \phi_p\|)) \\ &\leq M_1(\|\phi_{1_p} - \phi_p\| + \|\phi_{1_c} - \phi_c\| + \Delta) \end{aligned} \tag{7.48}$$

where M_{f_p}, M_{g_c} and M_{g_p} are the Lipschitz constants of the corresponding functions. Here, we also defined $M_1 \triangleq \max(M_{f_p}(1 + M_{g_c} M_{g_p}), M_{f_p} M_{g_c})$ and $\Delta \triangleq \Delta_{x_c} + \Delta_{y_c} + \Delta_{y_p}$.

In a similar way, we can obtain that

$$\|\psi_{1_c}(\tau) - \psi_c(\tau)\| \leq M_2(\|\phi_{1_p} - \phi_p\| + \|\phi_{1_c} - \phi_c\| + \Delta) \tag{7.49}$$

with $M_2 \triangleq \max(M_{f_c}, M_{f_c} M_{g_p})$. Let $M \triangleq M_1 + M_2$. From (7.48) and (7.49) it follows that

$$\|\psi_{1_p}(\tau) - \psi_p(\tau)\| + \|\psi_{1_c}(\tau) - \psi_c(\tau)\| \leq M(\|\phi_{1_p} - \phi_p\| + \|\phi_{1_c} - \phi_c\| + \Delta) \tag{7.50}$$

Defining $\Delta_\phi(t) \triangleq \|\phi_{1_p} - \phi_p\| + \|\phi_{1_c} - \phi_c\|$ and using (7.50) into (7.47) we have

$$\Delta_\phi(t) \leq \int_0^t (M\Delta_\phi(\tau) + M\Delta) d\tau$$

and finally

$$\Delta_\phi(t) \leq \int_0^t M\Delta_\phi(\tau) d\tau + Mt\Delta$$

It can be seen that function Δ_ϕ is continuous. Being also M positive, it is possible to apply the *Gronwall-Bellman Inequality*, resulting in

$$\begin{aligned} \Delta_\phi(t) &\leq Mt\Delta + \int_0^t M^2 s\Delta e^{\int_s^t M d\tau} ds \\ \Rightarrow \Delta_\phi(t) &\leq (e^{Mt} - 1)\Delta \end{aligned}$$

Then, for any $t \in [0, t_1]$ it results that

$$\lim_{\Delta \to 0} \|\phi_{1_p}(t) - \phi_p(t)\| + |\phi_{1_c}(t) - \phi_c(t)\| = 0$$

and then

$$\lim_{\Delta \to 0} \|\phi_1(t) - \phi(t)\| = 0 \tag{7.51}$$

From (7.40) we have

$$d \leq \inf_{x \in S}(\|\phi(t_1) - x\|) \tag{7.52}$$

Taking into account (7.41), (7.51) and (7.52) an enough small quantization interval can be found such that

$$t_1 < \frac{\inf_{x \in S} \|\phi_1(t_1) - x\|}{F}$$

This inequality implies that ϕ_1 does not leave the region D during the interval $[t_1, 2t_1]$. Thus, all the analysis done from (7.42a) is also valid for the interval $[0, 2t_1]$. Repeating this argument the result of (7.51) holds for all t. □

The convergence property shows that the QSC system trajectories can be bounded to be arbitrary closed to the CCS trajectories. However, this fact only provides qualitative information. There is not a quantitative relationship between the quatum and the difference between the CCS and QSC trajectories.

Thus, the only performance measure which can be analyzed is the ultimate bound and the convergence speed of the QSC solution based on Theorem 7.2.

In order to arrive to stronger results, it is necessary to put stronger restrictions on the plant and controller.

The next chapter deals with particular cases of QSC arriving to simpler and stronger theoretical and practical results.

Chapter 8

Linear QSC Systems

In the previous chapter, the general definitions of QSC were introduced. There, theoretical properties like stability and convergence were also studied.

Although it was possible to deduce design algorithms from the stability theorems, they resulted quite complicated because of the presence of Lyapunov functions (which were time dependent in the time varying cases).

In many control applications the plant models are in fact linear time invariant (LTI). In these cases the deduced algorithms are still correct but they have useless complications which could be avoided.

As it was already seen, the use of Lyapunov functions produces conservative results which can be improved with the use of geometrical analysis like the one developped in Section 4.5.

Taking into acount these remarks, this chapter is concerned with the particularization of the QSC properties in order to arrive to simpler design conditions.

As it can be expected, the results will be completely analogous to the shown by QSS and QSS2 methods in the field of simulation. In consequence, it will be possible to find the quantization which allows to bound the error between the continuous control system and the digital one. As before, the digital implementation will result ultimately bounded for any quantization adopted.

From a more practical point of view, much simpler design rules will be found which can be applied based on a closed formula which replaces the design algorithms developed in the previous chapter.

Finally, the improvement of the dynamic response and computational costs with respect to classical digital control which was mentioned will become more evident in some of the applications that will be introduced.

8.1 QSC of LTI Systems

Consider the Continuous LTI plant (8.1) and controller (8.2).

$$\left\{ \begin{array}{lcl} \dot{x}_p(t) & = & A_p \cdot x_p(t) + B_p \cdot u_p(t) \\ y_p(t) & = & C_p \cdot x_p(t) \end{array} \right. \tag{8.1}$$

$$\left\{ \begin{array}{rcl} \dot{x}_c(t) & = & A_c \cdot x_c(t) + B_c \cdot u_c(t) + B_r \cdot u_r(t) \\ y_c(t) & = & C_c \cdot x_c(t) + D_c \cdot u_c(t) + D_r \cdot u_r(t) \end{array} \right. \tag{8.2}$$

With the interconnection given by (7.3), the following closed loop equation is obtained.

$$\dot{x}(t) = A \cdot x(t) + B \cdot u_r \tag{8.3}$$

where

$$x(t) = \left[\begin{array}{c} x_p(t) \\ x_c(t) \end{array} \right], \quad A = \left[\begin{array}{cc} A_p + B_p D_c C_p & B_p C_c \\ B_c C_p & A_c \end{array} \right]$$

and

$$B = \left[\begin{array}{c} B_p D_r \\ B_r \end{array} \right]$$

The QSC implementation of the controller modifies (8.2) which can be rewritten as

$$\left\{ \begin{array}{rcl} \dot{x}_c & = & A_c \cdot q_c + B_c \cdot u_c + B_r \cdot u_r \\ y_c & = & C_c \cdot q_c + D_c \cdot u_c + D_r \cdot u_r \end{array} \right. \tag{8.4}$$

As in the general case, the effects of the A/D and D/A asynchronous converters are given by (7.5). Thus, taking into account that equation, (8.1), (8.4) and the definition of the perturbation variables (7.7a), the use of the QSC scheme transforms Equation (8.3) into

$$\dot{x} = A(x + \Delta x) + F \cdot \Delta y + B \cdot u_r \tag{8.5}$$

where x, Δx and Δy have the same definition than in Eq.(7.27) and

$$F \triangleq \left[\begin{array}{cc} B_p D_c & B_p \\ B_c & 0 \end{array} \right] \tag{8.6}$$

8.2 Stability and Error in LTI QSC

If it is considered $u_r(t) = 0$ (without lost of generality), System (8.5) is just a perturbed LTI system, where the perturbations (i.e. the components of Δx and Δy) are bounded by the corresponding quanta.

Then, according to Theorem 4.4, the trajectories of the LTI QSC system will be globally ultimately bounded with ultimate bound[1]:

$$\mu = \left\| |V| \cdot (|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta q_x + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}F|\Delta q_y) \right\| \tag{8.7}$$

where Δq_x is the vector of quantum sizes in the plant and controller state variables (all the components corresponding to the plant are zero), Δq_y is the vector of quantum sizes in the plant and controller output variables (introduced by the A/D and D/A converters respectively), the matrix F is defined according to (8.6), Λ is a diagonal matrix of eigenvalues of A and V is a corresponding matrix of eigenvectors.

[1] The notation used here was introduced in page 50.

Moreover, according to the same theorem it exists a finite time $t_1 = t_1(c, x_0)$ so that for each positive constant c the solutions satisfy

$$|x(t)| \leq (1+c)|V| \cdot (|\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda| \cdot |V^{-1}|\Delta q_x + |\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}F|\Delta q_y)$$

Thus, for any quantization adopted the QSC system is globally ultimately bounded.

It is clear that the last inequality can be used for design purposes: given a maximum accepted final error in each variable, the quantum which ensures achieving that goal can be obtained.

As in the case of QSS and QSS2, it will be possible not only to calculate the ultimate bound estimation but also the maximum deviation from the CCS trajectories.

Let $x(t)$ and $\tilde{x}(t)$ and be solutions of (8.3) and (8.5) starting from the same initial condition $x(t_0) = \tilde{x}(t_0) = x_0$. Then the error

$$e(t) = \tilde{x}(t) - x(t) \tag{8.8}$$

can be seen as a bound for the lost of performance due to the QSC implementation.

From (8.3), (8.5), (8.8) and the definitions of $x(t)$ and $\tilde{x}(t)$ we have

$$\begin{aligned} \dot{e}(t) &= A \cdot [e(t) + \Delta x(t)] + F \cdot \Delta y(t) \\ e(t_0) &= 0 \end{aligned}$$

Taking into account that Δx and Δy are bounded and $e(t_0) = 0$ Theorem 4.3 can be used in order to quantify the error due to the QSC scheme. The result is expressed by the following theorem:

Theorem 8.1. *Error in LTI QSC.*

Let $x(t)$ and $\tilde{x}(t)$ be trajectories of a LTI CCS and its QSC implementation starting from the same initial condition and let A be the evolution matrix of the continuous closed loop system. If A is Hurwitz and diagonalizable, the difference between both trajectories is always bounded by

$$|\tilde{x}(t) - x(t)| \leq |V|(|\mathbb{R}e(\Lambda)^{-1}\Lambda||V^{-1}|\Delta q_x + |\mathbb{R}e(\Lambda)^{-1}V^{-1}F|\Delta q_y) \tag{8.9}$$

with the same definitions made in (8.7).

The proof is straightforward using the mentioned theorem.

Inequality (8.9) can be used as a design formula to obtain the quantization in the controller and converters according to the deviation allowed from the CCS trajectories.

A very important remark is that the mentioned bound does not depend neither on the initial condition nor in the reference trajectory ($u_r(t)$). The only restriction is that $u_r(t)$ must be piecewise constant in order to guarantee that the QSC can be exactly represented by a DEVS model and then implemented on a digital device.

Theorem 8.1 says that the difference between the closed loop CCS trajectories and the corresponding QSC are always bounded. Moreover, the bound stands for all t and this fact can produce an important improvement in the dynamic response with respect to discrete time approximations.

8.3 Procedure for LTI QSC Implementation

The closed form of Inequality (8.9) gives also a useful tool for the design of the quantization. The design procedure then consists just in choosing the maximum allowed error in each variable and then in finding the appropriate values of Δq_x and Δq_y to satisfy the inequality.

The only problem here is that for a given tolerance at each state variable there are many possible values of Δq_x and Δq_y which satisfy (8.9). In fact, the quantum can be enlarged in a given variable and reduced in the others so that the inequality is still verified.

From the theoretical point of view, any value which satisfies (8.9) is the same. However, taking into account practical considerations, it is not desirable that a variable changes very fast while the others remain without changes. Such a situation would result in very fast events which could be even faster than what the digital device can follow.

In consequence, a good solution would be the one which produces a *balanced* sequence of events at the different variables. However, it is not completely clear how this can be achieved.

Anyway, an empirical rule which usually works consists in choosing the quanta so that each term of Δq_x and Δq_y contributes to the whole error in a balanced way.

Example 8.1. *A PI controller example.*

Consider the plant (8.1) *with*

$$A_p = \begin{bmatrix} 0 & 1 \\ -1 & -4 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C_p = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

A PI controller with parameters K_P and K_I can be written in the form of (8.2) *with $A_c = 0$, $B_c = -1$, $B_r = 1$, $C_c = K_I$, $D_c = -K_P$ and $D_r = K_P$. Thus, using $K_P = K_I = 10$, the closed loop evolution matrix is*

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -11 & -4 & 10 \\ -1 & 0 & 0 \end{bmatrix}$$

A possible matrix of eigenvectors (calculated with Matlab) is

$$V = \begin{bmatrix} 0.0085 + j0.342 & 0.0085 - j0.342 & 0.5452 \\ -0.825 - j0.433 & -0.825 + j0.433 & -0.734 \\ -0.108 + j0.064 & -0.108 - j0.064 & 0.4049 \end{bmatrix}$$

Taking $\Delta q_c = \Delta y_p = 0.01$ *and* $\Delta y_c = 0.1$*, the error bound according to* (8.9) *is* 0.1443, 0.3383 *and* 0.0679 *in each state variable of the plant and controller respectively. Thus, the maximum error on the plant oputput is bounded by* 0.1443*. Figure 8.1 shows the evolution of the plant output with the continuous controller and the QSC controller when the reference is a step of amplitude* 10*. Figure 8.2 shows the difference between both trajectories. The maximum difference was* 0.1399 *which is near the theoretical bound.*

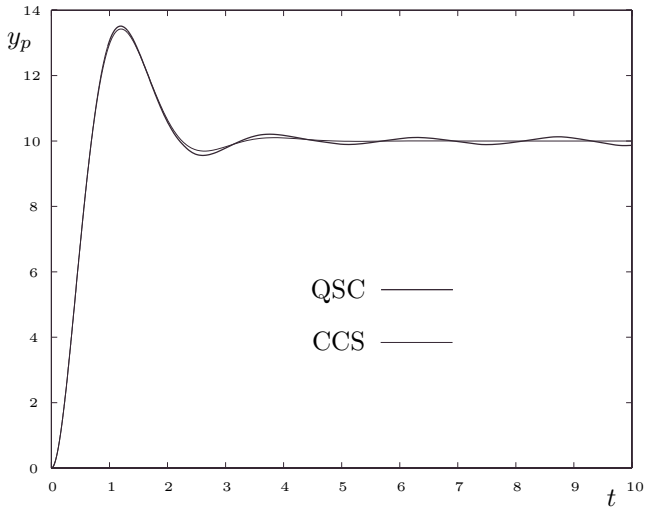

Figure 8.1: Plant output with CCS and QSC

The last example showed the use of Theorem 8.1 for the QSC implementation of a PI controller.

The next example follows an LQR design to control an inverted pendulum introduced by Messner and Tilbury in [46]. After simulating the continuous and the digital controller proposed in the reference, the results are compared with the obtained in the QSC implementation.

Example 8.2. *An inverted pendulum control.*

Consider the inverted pendulum of Figure 8.3 Using parameters $M = 0.5$*;* $m = 0.2$*;* $b = 0.1$ *(friction of the cart);* $J = 0.006$*;* $g = 9.8$*; and* $l = 0.3$ *(length to pendulum center of mass), a linearized model around* $\theta = \pi$ *is given by the equations:*

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Phi} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.181 & 2.672 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.454 & 31.18 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1.818 \\ 0 \\ 4.545 \end{bmatrix} u$$

where $\Phi \triangleq \theta - \pi$ *is the deviation angle from the vertical position.*

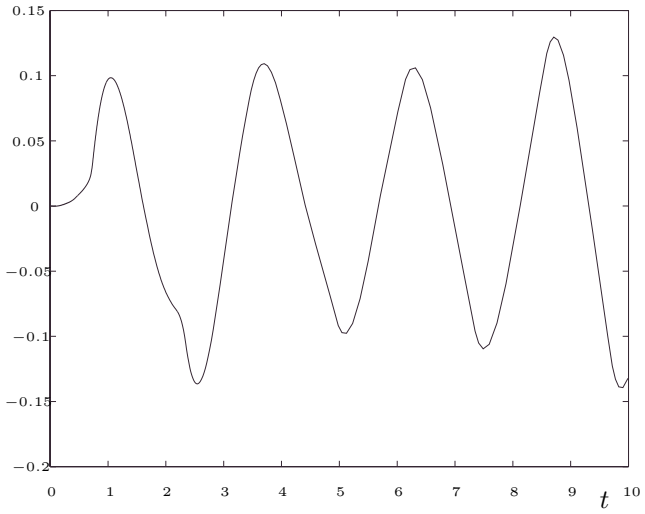

Figure 8.2: Difference in the plant output with CCS and QSC

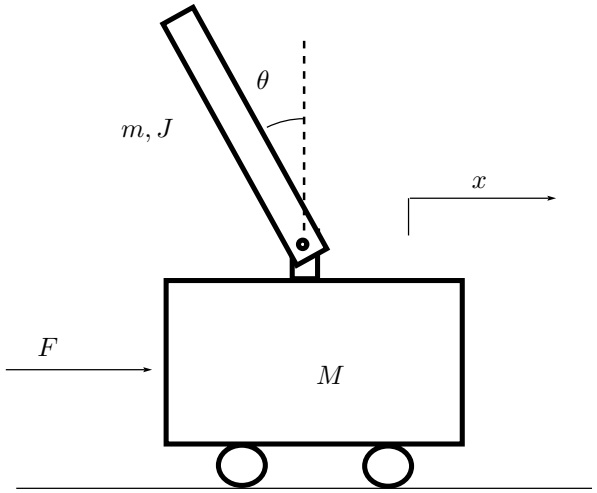

Figure 8.3: Inverted Pendulum scheme

The goal is to implement a control for the cart position. The variables mea-

sured are x and Φ. Thus, the plant output equations can be written as

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \Phi \\ \dot{\Phi} \end{bmatrix}$$

Following a LQR design with weights $w_x = 5000$ and $w_\Phi = 100$ and then using a full state observer with estimator poles placed at $p_1 = 40$, $p_2 = 41$, $p_3 = 42$ and $p_4 = 43$ the resulting controller can be written in the form of (8.2), where

$$A_c = \begin{bmatrix} -82.64 & 1 & 1.037 & 0 \\ -1570 & 68.61 & 148.9 & -38.04 \\ 1.385 & 0 & -83.18 & 1 \\ 397.6 & 171.5 & 2209 & -95.11 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 82.64 & -1.037 \\ 1699 & -40.22 \\ -1.385 & 83.18 \\ -76.18 & 1760 \end{bmatrix}; B_r = \begin{bmatrix} 0 \\ -128.6 \\ 0 \\ -3.214 \end{bmatrix}$$

$$Cc = \begin{bmatrix} 70.71 & 37.83 & -1055 & -20.92 \end{bmatrix}$$

and

$$Dc = \begin{bmatrix} 0 & 0 \end{bmatrix}; Dr = -70.71$$

For the QSC digital implementation of the controller, the following quantization sizes were adopted:

$$\Delta q_x = \begin{bmatrix} 0.001 \\ 0.015 \\ 0.002 \\ 0.04 \end{bmatrix}; \Delta q_y = \begin{bmatrix} 0.002 \\ 0.004 \\ 2 \end{bmatrix}$$

The QSC performance was compared with a classic discrete controller obtained from the discretization of the continuous controller. The sampling time of the classic digital controller was taken as $T_s = 0.01$. In this case, we also considered the presence of A/D and D/A converters with the same quantization than in the QSC control (i.e. 0.002 in the position, 0.004 in the angle and 2 in the controller output).

Figures 8.4-8.6 show the trajectories obtained with the continuous time controller (without quantization), with the QSC controller and the classic discrete time controller. The first observation is that the QSC shows a much better performance during the transient (see Figure 8.5) and it reduces the final oscillations.

The number of A/D conversions performed by the QSC converters in the 5 seconds of simulation was 211 and 185 in the position and angle respectively while the classic digital controller performs $5/0.01 = 500$ conversions in each

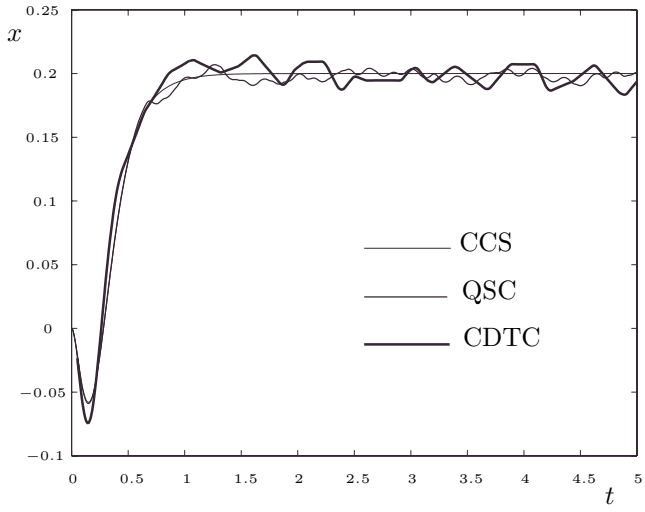

Figure 8.4: Cart position with QSC and Classic Digital Control.

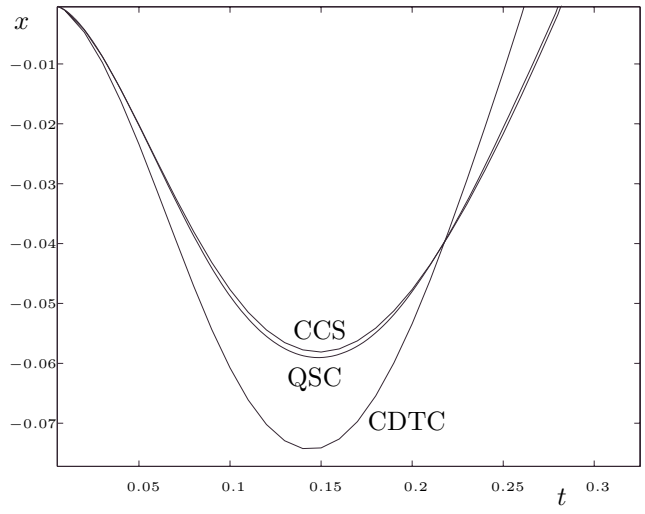

Figure 8.5: Cart position (start up) with QSC and Classic Digital Control.

variable. Similarly, the number of D/A conversions in the QSC control was 303*. The classic discrete time controller also performs* 500 *D/A conversions.*

Taking into account what happens from $t = 3$ *to* $t = 5$ *(during the* steady state*), it can be seen that the QSC only performs* 29 *A/D conversions in the postition,* 44 *A/D conversions in the angle and* 76 *D/A conversions while the discrete time controller performs* 200 *conversions in each variable. The reduction of the computational costs is evident.*

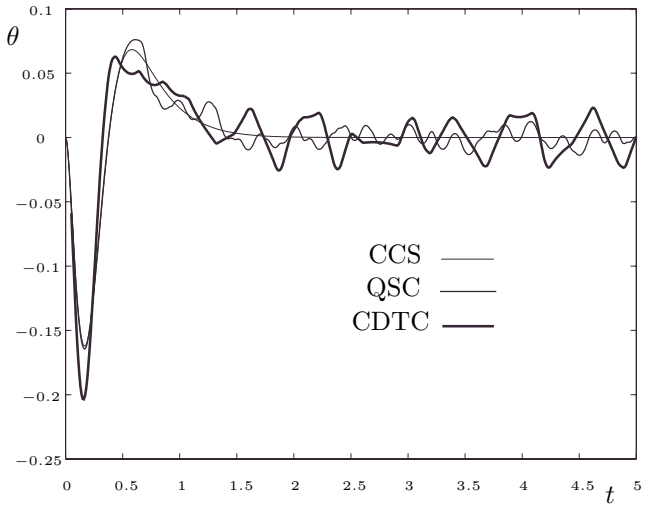

Figure 8.6: Pendulum angle with QSC and Classic Digital Control.

These advantages are paid with more calculations at the controller (the number of changes in the internal variables is 1600 *against the* 500 *in the discrete time controller). However, the time required to perform A/D and D/A conversions is always much bigger than the time used for a simple calculation.*

Anyway, the number of calculations at the controller can be also reduced using a different realization. An interesting alternative is to use a diagonal –or block-diagonal– evolution matrix A_c*. In this way, the QSS simulation becomes much more efficient. On one hand, the number of changes in the quantized variables is reduced. On the other hand, each transition involves calculations only in the state which is actually changing (one of the most important features of QSS simulation is that it exploits the sparsity properties). In this example, the use of a block diagonal controller reduces the number of calculations to* 547 *in* 5 *seconds obtaining a similar performance.*

It is also interesting to observe that the fastest pole of the closed-loop system is placed at 43*. Thus, the mean sampling frequencies in the QSC during the* steady state *are below the system bandwidth. Thus, the QSC scheme works violating the Nyquist frequency. It is well known that a sampled data system working below that frequency will be unstable. QSC can work using that frequency because the equilibrium point in the QSC scheme is in fact unstable (but the solutions are ultimately bounded).*

Finally, it should be mentioned that the use of Inequality (8.9) *gives a quite conservative bound in this case. The error bound in* $x(t)$ *is* 3.08*, which is about* 300 *times bigger than the error observed in Figure 8.4. Anyway, a Lyapunov analysis gives a bound of* 9.6×10^9*, which is completely useless for the design.*

8.4 Matching the Converters

The difference between the continuous designed control and the discrete event implementation is due to the effects of the quantization introduced by the state quantization and the presence of A/D and D/A converters. However, in some cases, the effects of quantization in the D/A converters can be eliminated.

The input of the D/A converters is given by

$$y_c = g_c(q_c, u_c)$$

Since q_c and u_c are quantized variables, g_c has a finite domain and then y_c can only take values in a finite set. This set is determined by the quantization levels in q_c and u_c and by function g_c. If that quantization values are chosen so that the mentioned set is a subset of the quantization values of the D/A converters, these converters will not introduce any error.

One of the cases where D/A errors can be avoided is when the output variables of the controller coincide with state variables of that system. In this case the mentioned error can be eliminated by choosing the same quantization in these state variables and in the corresponding D/A converters.

In the Example 7.2 a very conservative result was obtained. It is known from Chapter 4 that Lyapunov–based analysis usually arrive to conservative estimations.

However, in this case, the problem is also due to the fact that it was not taken into account that the D/A converter was perfectly matched. In fact, here the controller output function was

$$y_c(t) = -x_c(t) - u_c(t) \tag{8.10}$$

and the quanta were chosen equal to 0.018 in all the variables. Then, $q_c(t)$ (the quantized version of x_c), $u_c(t)$ and $u_p(t)$ can only take values in the set $\{\dots, -0.036, -0.018, 0, 0.018, 0.036, \dots\}$.

Taking into account (8.10), it results that y_c can only take values in the same set. Since $u_p(t)$ also takes those values, there is no quantization effect from y_c to u_p.

Despite these concepts could be applied to general control systems, they are only really useful when the controller output equation is linear. Otherwise, the converter matching would require the use of non–uniform quantization.

8.5 Computational Costs Reduction in QSC

In this chapter some theoretical advantages were shown. These advantages result in the improvement of the stability and the dynamic response of the QSC with respect to discrete time approximations. It was already mentioned and illustrated with examples that QSC also reduce the computational costs but this fact was not explained yet.

While traditional discrete time controllers perform calculations at regular intervals, QSC controllers only do it when a variable becomes greater (or less)

than some threshold. This fact can be seen as a first intuitive reason for that computational costs reduction.

For instance, in Example 7.1 (page 119), when the trajectory arrives near the origin the controller performs about one calculation per second. Any discrete time controller using that sampling rate (and even a rate 10 times faster) will diverge for the same initial condition of $x_p = 10$ due to problems of *finite escape time*[2].

Similar remarks can be done regarding the inverted pendulum control of Example 8.2 where the sampling rate near the equilibrium was below the Nyquist frequency.

The following example illustrates better the reduction of the computational costs in the QSC scheme.

Example 8.3. *Sampling rate reduction in QSC.*

Consider the first order system

$$\dot{x} = sgn(x) + u$$

Suppose that the controller can measure the variable x, but the A/D converter only produces even numbers (-2, 0, 2, 4, ...) giving the nearest to its analog input. When x is in the shaded region of Figure 8.7 (between -1 and 1) the control system sees the value 0.

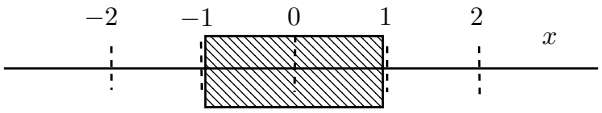

Figure 8.7: *Invisible* zone due to the quantization

Consider that the goal is keeping the value of x between -2 and 2. The time to go from 1 to 2 (or from -1 to -2) with $u = 0$ is $t_1 = 1$. This implies that it is impossible to find a discrete time controller achieving the proposed goal with a sampling period greater than t_1. The reason of this is that the controller cannot distinguish if the value of x is positive or negative when it is in the shaded region and then, the sign of u could be the same as the sign of x and the trajectory will abandon the desired region before the time t_1.

However, using QSC, the time between samplings can be done arbitrarily big. For instance, consider the following static control law

$$y_c = -u_c(1 + a)$$

where a is a positive constant. When x goes away of the shaded region the controller immediately detects the change and it inverts the sign of the derivative.

[2]The phrase "finite escape time" is used to describe the phenomenon that a trajectory escapes to infinity at a finite time [23]

The new speed on x is a. Then, x enters again the shaded region and the time to reach the origin is $1/a$. After that, x goes to the other bound of the shaded region, but with a new speed of $2 + a$. When x leaves the shaded region again the controller inverts the sign of the derivative and we obtain a cyclic behaviour where x oscillates between 1 and -1. The time between successive A/D conversions is

$$t_2 = \frac{1}{a} + \frac{1}{2+a}$$

It is clear that taking small values for the parameter a, t_2 can be done arbitrarily big. Then, with this implementation the number of calculations in the controller and the size of the oscillation can be considerably reduced.

Unfortunately, the QSC implementation is not exact. In fact, there are delays related to the presence of converters and the digital processor. In this last example, these delays must be smaller than the minimum sampling period $t_1 = 1$ in order to obtain the proposed goal of keeping x between 2 and -2.

One could think that if it is possible to implement a QSC achieving such minimum delay, then it would be possible to implement a discrete time controller using that sampling period. However, the delay in the QSC system is only the time required to detect the error and to perform the D/A conversion because the calculations can be done before the error is detected. This is possible because the QSC controller knows that its next input value can only adopt two different values.

In the example, if the last detected value of x was 0, then 2 and -2 are the only two values that the new input can adopt. Thus, the controller has two possible output values that could be calculated before the detection of the new input value. For a dynamic controller similar ideas can be applied.

A classic discrete time controller during each sampling period must perform the A/D conversion, calculate the next state and output of the controller and then perform the D/A conversion. The time required to finish all these tasks is much greater than the delay in the QSC scheme.

Chapter 9

Epilogue

As it was mentioned in the introduction, we see this Thesis as a contribution to the development of a theory about discrete event approximation of ordinary differential equations.

New theories –or paradigms, according to Khun– can provide better solutions to some *old* problems and open many *new* problems on one hand. On the other hand, they usually cannot give an answer to all the *old* problems involved in the area.

Thus, this Thesis should not be seen as an attempt to replace existing discrete time numerical methods.

Taking into account that classic numerical methods have been being studied for more than 330 years (Euler's method was first described in 1768 [18]), it is useless to expect to obtain the *magic* solution after only 3 years of work starting from a completely new phylosophy.

The best which can be expected is that the *old* problems as well as the *new* problems open by this work are progressively solved in the future so that the theory starts to give good solutions in more and more cases.

Considering these remarks, this last chapter will start ennumerating some of the *old* problems which are still unsolved by the new theory. Then, many of the new open problems will be mentioned together with some ideas to solve them. Finally, we shall present the general conclusions.

9.1 Unsolved Problems

- Stiff Systems. The most important unsolved problem is probably the one related to the simulation of stiff systems. In fact, this is one of the most difficult problems in the area of numerical integration.

  Let us introduce it with a simple example:

  Example 9.1. *QSS simulation of a stiff system.*

Consider the second order ODE given by

$$\begin{aligned} \dot{x}_1 &= 100 \cdot x_2 \\ \dot{x}_2 &= -100 \cdot x_1 - 10001 \cdot x_2 + U(t) \end{aligned} \tag{9.1}$$

The eigenvalues are -1 *and* -10000 *which means that the system, in spite of its simplicity, is stiff.*

The first 10 seconds of the system solution were simulated with the QSS–method using quantum and hysteresis sizes of 1×10^{-2} *and* 1×10^{-4} *in* x_1 *and* x_2 *respectively. According to* (4.50)*, this quantization ensures that the error in* x_1 *is bounded by* 0.01 *while the error in* x_2 *is bounded by* 0.0003*.*

The initial conditions were both zero and the input was a step $U(t) = 100$*. The simulation was completed after 100 and 200 internal transitions in the quantized integrators which calculate* x_1 *and* x_2 *respectively. The trajectory of* x_2 *is shown in Figures 9.1–9.2.*

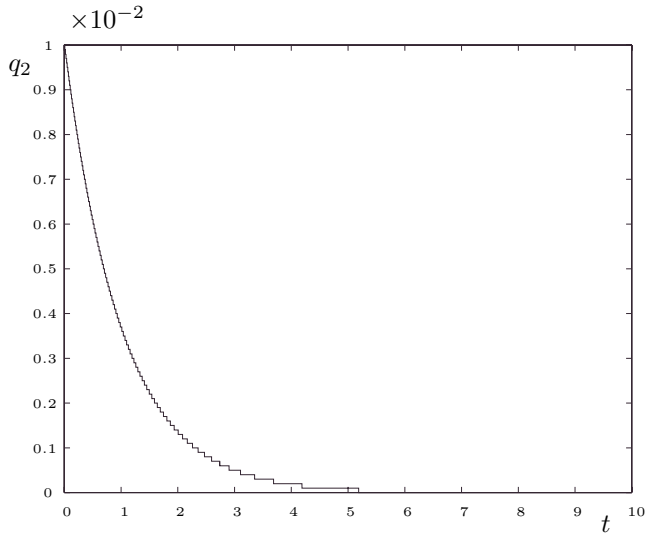

Figure 9.1: QSS simulation of System (9.1)

The stiffness of this systems is clear when the time scales of Figures 9.1 and 9.2 are compared. In this case, the behavior of the QSS–method is amazing. It just adapted the step size in a very natural way and performed the simulation in a few steps.

Notice that the number of transitions performed can be calculated here by dividing the trajectory amplitude by the quantum.

If we look at this result, we may think that the QSS–method is the best solution for stiff systems: it is an explicit method which allows simulating a stiff system much faster and more efficiently than what the most complex

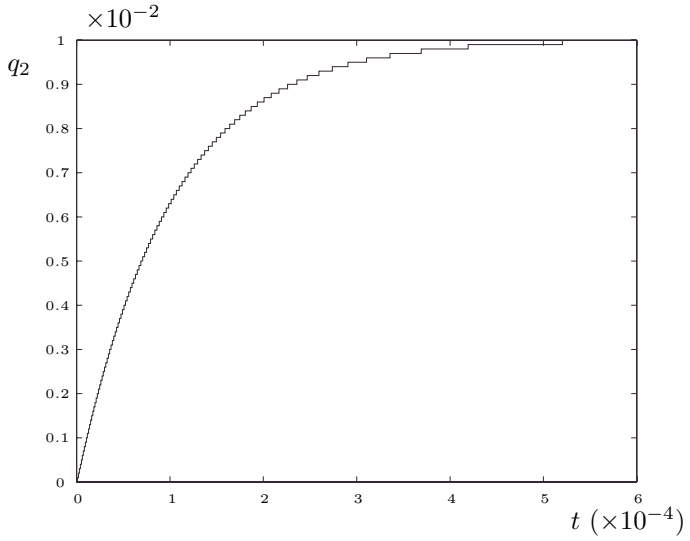

Figure 9.2: Start in the QSS simulation of System (9.1)

implicit variable–step algorithms can do. This idea seems to be coherent with the fact that the QSS–method is always "stable". Of course, if a simulation method were so good, it would probably change all the existing theory of numerical integration.

However, the meaning given to the word "stable" should not be forgotten. The QSS–method does not ensure stability but ultimately boundedness of the solutions and, unfortunately, this is the main problem it has with stiff systems.

Then, let us see what happen if a small modification in the input amplitude is introduced. When its value is changed from $U(t) = 100$ *to* $U(t) = 99.5$ *a problem appears.*

In the first 5 *seconds of simulation the quantized integrator which calculates* x_1 *performed* 100 *internal transitions (the same as before) but the other calculated a total of* 25057 *transitions. The trajectory of* q_2 *is shown in Figures 9.3–9.4.*

In this simulation, the result is almost the same than before. The error, in agreement with the theoretical prediction, is bounded, but the number of calculations is huge.

The reason of this is the appearance of very fast oscillations in x_2 *which ruined the wishes of having an explicit, efficient, stable and reliable method for stiff system. If those oscillations were not present, the number of transtions could have been calculated dividing the trajectory amplitude by the quantum. However, this is not possible here.*

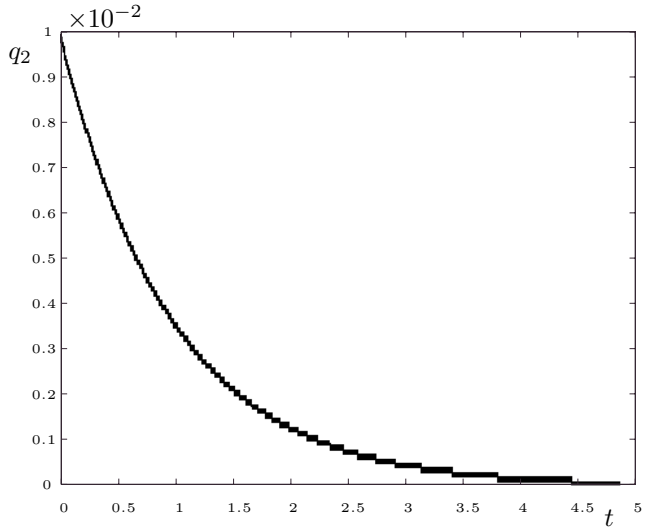

Figure 9.3: QSS simulation of System (9.1) with $U = 99.5$

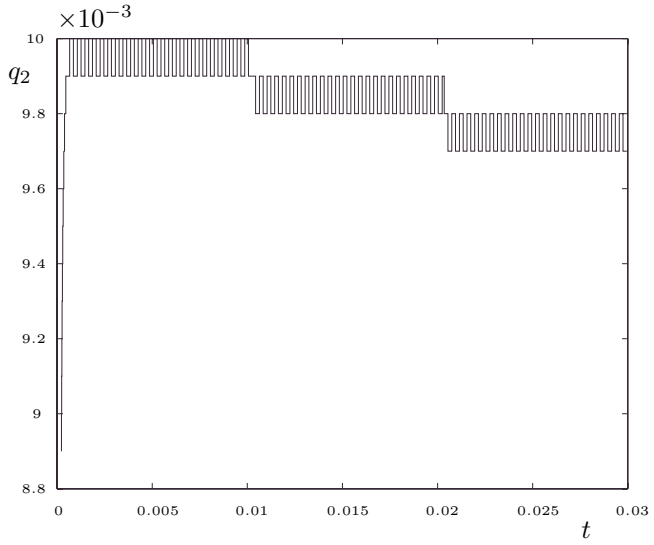

Figure 9.4: Detail of Fig.9.3

Although their presence may be surprising, the oscillations were alreay seen along the Thesis. Indeed, they are not different from the oscillations seen in Figure 4.1. However, the oscillations there were not a problem since their period was not much shorter than the system settling time.

Unfortunately, the oscillation frequency is always related to the system

speed –i.e. the location of the eigenvalues– which is a problem in the case of stiff systems. Here, there may be oscillations due to the fast mode which provoke several transitions before the slow modes reach their steady state.

When it comes to the solution of this problem, there is not a final answer yet.

An interesting modification which works in many cases is based on the use of backward integration concepts. The idea is looking at the future values of the quantized variables to calculate the derivatives which determine the time advances.

Taking into account that a future quantized variable value is known (if the actual value is Q_j, in the next transition it can only take two different values: Q_{j+1} or Q_{j-1}), it is not necessary to solve implicit equations. Here, backward integration does not imply using iteration rules.

Anyway, to determine if the next quantized value is the upper or the lower one, the sign of function f_i (which determines the derivative in both cases) must be evaluated. If f_i evaluated at Q_{j+1} is positive, then Q_{j+1} should be the next value. If f_i at Q_{j-1} is negative, then the next value is Q_{j-1}. If none of these cases are true, it can be ensured that between Q_{j-1} and Q_{j+1} there is a point in which $f_i = 0$ and then it can be directly put $\sigma = \infty$.

A remark here is that the quantized integrators have to be able to evaluate f_i, which means that the quantized integrators should not be separeted from the static functions which calculate the derivatives.

Although the resulting atomic models are more complex since they combine the features of a quantized integrator and a static function, the simulation becomes more efficient since the number of events is reduced to less than the half (the coupled model does not have to transmit events from the static functions to the quantized integrators and from the quantized integrators to the static functions which compute their own derivatives).

This idea was implemented and the simulation of the Example 9.1 was repeated with $U(t) = 99.5$. Now, the number of transitions was the same than using the QSS–method with $U(t) = 100$, i.e. without oscillations.

However, it was not proved yet that this idea constitute a general solution for stiff systems. In high order systems this *quantization in advance* is not straightforward because after each transition many sets of possible next values should be analyzed in order to determine which is the appropriate one.

Despite the fact that the quantization–based methods can give an appropriate and efficient solution in many stiff systems, the problem is still open.

- Quantum Choice. Another open problem in these methods has to do with the choice of the quantization. Although some algorithms and formulas

were provided allowing to choose the quantum according to desired error, this is not a completely satisfactory solution.

Except in applications were it is neccessary to ensure some error bound which justifies doing a precise analysis before performing the simulation, nobody wants to calculate a Lyapunov function or the matrices of eigenvectors and eigenvalues to determine the quantum to be used.

One possible solution may consist in providing the computer with the necessary tools to calculate such matrices and to choose automatically the quantum before simulating. However, a much more interesting idea would consist in doing something similar to what the variable step methods do. This would lead to something like *adaptive quantization*.

- Accuracy. Another problem of the methods is the accuracy. There are many applications in which the error given by the QSS2–method may be too big. Methods of order greater than 2 satisfying the same properties of QSS and QSS2 can be easily imagined (with higher order quantization fuctions).

  However, there is a problem here. The DEVS model of a higher order quantized integrator should calculate the time advance after each external transition. To do that, it has find the roots of a polinomial equation like (5.10). It can be seen that the degree of this equation is the same as the order of the method. Thus, the solution becomes more and more complicated as well as the order increases.

- Real–Time QSC. When it comes to QSC, there is a problem which was not mentioned yet. It was already said that the asynchronous converters work much faster than the common ones. It is also known that the computational costs are reduced in QSC and the calculations inside the controller are simpler.

  Anyway, the fact that the QSC imprementation is also a problem of real–time simulation should not be forgotten.

  The goal of real time simulation is not only to obtain a good approximation but also to get the result before the time expires. In order to achieve this, we must ensure that all the calculations involved in a single step do not take more time that what is allowed.

  In this context, the accuracy is often sacrified in order to obtain the results in the *allowed time* [15]. Thus, it is very important to have a bounded number of calculations at each step.

  This is not a problem for a QSS simulation model since each step only involves a couple of transitions and the number of calculation is approximately constant.

  However, the *allowed time* is what is not constant here and this fact can constitute a problem in some realistic applications.

Here, some remarks included in Section 8.5 may provide a partial solution but a final answer will also require from experimental implementations.

9.2 Open Problems and Future Research

Besides the unsolved problems already mentioned (the simulation of stiff systems, the quantum choice and the accuracy in QSS and QSS2 as well as some aspects of the QSC implementation) there are many *new* problems which were opened by the new methodologies.

One of the most interesting advantages shown by the QSS and QSS2 methods is the highly efficient sparsity exploitation. Taking into account that the *method of lines* as well as other methods for approximating partial differential equations (PDEs) result in sparse ODEs, the use of QSS and QSS2 in these cases may constitute a very interesting alternative.

In fact, this is what was done in the transmission line examples (Examples 5.1 and 6.1 in pages 71 and 84 resectively) where the second case was a partial differential algebraic equation (PDAE).

Although in the mentioned examples very good results were obtained, a deeper research is needed in this direction. In fact, the use of the approximating methods often arrives to stiff ODEs and then this problem should be solved first.

When it comes to DAEs, the higher index problems should be still taken into account. It was mentioned that the Pantelides'algorithm can be applied to reduce the index to 1 and then the systems can be simulated with the developed methods. However, a more efficient way might be found following the remarks of Section 6.5.

Although it was mentioned that stability is not a problem here and the errors in the calculation of implicit variables only increase the error bound, this fact should be formally proven. It is also important to find a relationship between the tolerance in the implicit equation solution and the increment in the error bound.

With respect to hybrid systems simulation, it comes to be necessary to extend the theoretical stability and error bound analysis to general discontinuous systems in order to establish conditions which ensure the correctness of the simulation. What was done in the DC-AC inverter example (page 92) –where the error bound was estimated according to Eq.(4.50)– might constitute a first step in this direction which could be extended to general systems with only time events.

In the approach presented, only hybrid systems whose continuous part does not change its order were considered. It would be interesting to consider also more general cases including variable order ones.

Another interesting problem is the parallelization of the methods. The parallel implementation of QSS and QSS2 in ODE and DAE seems to be straightforward since the division into subsystems is clear. If we consider each pair composed by a quantized integrator and the corresponding static function as a subsystem, the traffic of messages between different subsystems is not big.

Moreover, in QSS the value carried by a message differ from the previous in a quantum. Then, it can be transmitted using only one bit saying if the quantized variable increased or decreased. This fact may constitute a very important advantage in real–time simulation (in off–line simulation the message should also carry information about the time).

Coming back to the QSC method, it is known that the real–time implementation does not exactly simulate the QSS model (because of the already mentioned problems of delays and round–off). Although the round–off problems do not result important, the delays do modify significantly the QSC system behavior.

Thus, it is important to study their effect on the stability and error bound. That study should also include a characterization of the delays which would require some experimental work in real applications.

When it comes to theoretical aspects of QSC, the most important properties were proven for general nonlinear time varying systems and for LTI cases.

There is an intermediate case which should be taken into account which corresponds to Linear Parameter–Varying (LPV) plants. The Example 7.2 (page 124) in fact corresponds to that category and although a Lyapunov analysis could be done for that case, the result was very conservative. If the geometrical analysis for LTI systems in Section 8.2 were extended for LPV plants, less conservative results might be obtained.

It is also important to study the way in which QSC affects not only the stability, region of attraction and error bound but also other performance measures (mean square error, overshoot, etc.).

Finally, it should be taken into account that QSS are just a small class of the wide family of DEVS models. With the use of more general DEVS models the study of optimal conditions on asynchronous controllers could be eventually performed.

9.3 General Conclusions

This Thesis introduced a wide variety of contributions to the development of a new theory about discrete event approximations to differential equations.

Two approximating methods were provided –that were called QSS and QSS2– which, based on quantization ideas, map continuous systems into DEVS models. Here, not only ODEs but also DAEs, Hybrid Systems and Bond Graphs were considered.

The first order method –QSS– was also applied to approximate continuous controllers for digital control applications. There, also making use of asynchronous sampling, a new digital asynchronous control scheme was obtained. This scheme –which was called QSC– has the remarkable feature of avoiding the time discretization.

Besides formulating the methods, mathematical tools –based on perturbations theory– were developed in order to analyze theoretical properties of the approximations.

In the more general cases (nonlinear and time varying), those tools were just the adaptation of classic Lyaunov analysis. In the case of LTI systems, in order to avoid conservative results, a new way to analyze the effects of perturbations was proposed. These new tools can also be used for much more general purposes than the presented in the context of this work.

The new methodologies show many advantages (and also disadvantages) with respect to the classic discrete time approximations as it can be expected from the fact that they are based on completely different principles.

From a theoretical point of view, the most remarkable advantages concern the stability and error bound properties. On one hand, the simulation methods (QSS and QSS2) are always *stable*[1] in LTI systems. This is important considering that the methods are explicit and non–adaptive.

On the other hand, Inequality (4.50) gives a closed formula for the global error bound in those methods. This is another advantage since discrete time algorithms do not have such kind of expressions.

All these facts have their correspondence in QSC. There, it is also very important the conservation of the region of attraction in nonlinear and time varying systems.

Another advantage which can be found in this case is that the effects of the A/D and D/A converters can be taken into account and bounded at the design stage.

When it comes to practical advantages, one of the most importants is the way in which the QSS and QSS2 methods exploit sparsity due to the fact that each integrator runs independently.

In DAE integration, the iterations to solve the implicit equations do not have to be performed in every step. In this way, the methods avoid a big number of calculations. Taking into account this feature and the fact that each step only involves calculations in some state variables, we conclude that the quantization–based approximations may constitute a powerful tool to solve sparse DAE systems.

Here, it is also remarkable the simplicity in which block diagrams containing algebraic loops can be simulated. Although the computational efficiency of this *block oriented* methodology is not optimal, it is a systematic methodology with straightforward implementation in any DEVS simulation environment which still exploits sparsity and the other advantages of quantization–based methods.

As it could be expected of a discrete–event approximation, the most important reduction of computational costs was observed in hybrid systems. The facilities to deal with discontinuities constitutes one of the most important advantages of the methodologies with respect to classic discrete time algorithms.

In the examples analyzed, the methods showed a performance clearly superior to all the complex implicit, high order and variable step methods implemented in Simulink. Taking into account that QSS as well as QSS2 are very simple, low order and explicit algorithms with fixed quantization size; it is quite natural to think that future more complex discrete event methods may offer an

[1] in fact they have ultimately bounded solutions

unexpected high performance.

With respect to the practical advantages of QSC, it was already mentioned the reduction of the computational costs as a result of the asynchronous sampling scheme. Here, the converters only take samples when it is necessary (i.e. when they differ from the previous value in a given quantity). Similarly, the controller internal states are recalculated under the same condition. Thus, QSC can be thought as a control strategy which only acts when it has to.

Although the principles of QSC implementation seems to be quite difficult (at least for people who are not familiar with the DEVS formalism) its programming is rather simple. In fact, the only thing which has to be done is to implement the routines explained in Section 2.4 and to execute them in real time. Moreover, there are tools like PowerDEVS [51] which do it automatically. With this tool, for instance, the problem is reduced to build a block diagram model of the controller and then to execute the simulation with the real–time settings.

Another advantage is that for QSC implementation, classic design techniques for continuous controllers can be applied (for instance, in the Example 7.1 in page 119, the original continuous controller was designed via *exact linearization* [23]). Moreover, if Lyapunov–based design techniques are used, then the algorithms described in Sections 7.5 and 7.7 can be easily applied.

Another fact which should be remarked is related to the information used by the QSC controller. Once the controller gets its first input value, the asynchronous A/D converters can transmit the following values using only one bit per conversion with the sign of the change. In a similar way –provided that there is some matching between the internal quantizers and converters– the controller outputs can be transmitted with only one bit each time.

This is a very important advantage in distributed control systems, where the information has to be transmitted between sensors, controllers and actuators. Although there are some results based on quantization to reduce the amount of information [12], the use of only one bit was never achieved with other non–trivial control schemes.

Finally, and leaving aside the advantages, disadvantages and unsolved problems, the essence of this Thesis is simply a new and formal way of conceiving the approximation of differential equations, in which we broke the principle that the discrete states must change simultaneously in a synchronous way.

Bibliography

[1] François Baccelli, Guy Cohen, Geert Olsder, and Jean Pierre Quadrat. *Synchronization and Linearity*. Wiley, 1992.

[2] J.S. Balduc and H. Vangheluwe. Expressing ODE models as DEVS: Quantization approaches. In *Proceedings of AIS'2002*, pages 163–169, Lisbon, Portugal, 2002.

[3] P.I. Barton. Modeling, Simulation, and Sensitivity Analysis of Hybrid Systems: Mathematical Foundations, Numerical Solutions, and Sofware Implementations. In *Proc.of the IEEE International Symposium on Computer Aided Control System Design*, pages 117–122, Anchorage, Alaska, 2000.

[4] M. Branicky. Stability of Switched and Hybrid Systems. In *Proc. 33rd IEEE Conf. Decision Control*, pages 3498–3503, Lake Buena Vista, FL, 1994.

[5] R. Brockett and D. Liberzon. Quantized feedback stabilization of linear systems. *IEEE Trans. Automat. Contr.*, 45:1279–1289, 2000.

[6] J.F. Broenink and P.B. Weustink. A Combined–System Simulator for Mechatronic Systems. In *Proceedings of ESM96*, pages 225–229, Budapest, Hungary, 1996.

[7] François Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, New York, 2003. In preparation.

[8] François Cellier. *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.* PhD thesis, Swiss Federal Institute of Technology, 1979.

[9] François Cellier. *Continuous System Modeling*. Springer, New York, 1991.

[10] François Cellier. Object-Oriented Modeling: Means for Dealing With System Complexity. In *Proc. 15th Benelux Meeting on Systems and Control*, pages 53–64, Mierlo, The Netherlands, 1996.

[11] D. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE Trans. Automat. Contr.*, 35:916–924, 1990.

[12] N. Elias and S. Mitter. Stabilization of Linear Systems with Limited Information. *IEEE Trans. Automat. Contr.*, 46(9):1384–1400, 2001.

[13] J.M. Esposito, V. Kumar, and G.J. Pappas. Accurate Event Detection for Simulating Hybrid Systems. In *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 204–217. Springer, 2001.

[14] J. Farrel and A. Michel. Estimates of asymptotic trajectory bounds in digital implementations of linear feedback control systems. *IEEE Trans. Automat. Contr.*, 34:1319–1324, 1989.

[15] Javier Garcia de Jalón and Eduardo Bayo. *Kinematic and Dynamic Simulation of Multibody Systems –The Real-Time Challenge–*. Wiley, 1994.

[16] C.W. Gear. The Simulataneous Numerical Solution of Differential–Algebraic Equations. *IEEE Trans. Circuit Theory*, TC–18(1):89–95, 1971.

[17] Norbert Giambiasi, Bruno Escude, and Sumit Ghosh. GDEVS: A generalized Discrete Event specification for accurate modeling of dynamic systems. *Transactions of SCS*, 17(3):120–134, 2000.

[18] Ernst Hairer, Syvert Norsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer, 2nd edition, 1993.

[19] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Problems*. Springer, 1st edition, 1991.

[20] L. Hou, A. Michel, and H. Ye. Some qualitative properties of sampled-data control systems. *IEEE Trans. Automat. Contr.*, 42:1721–1725, 1997.

[21] L. Hu and A. Michel. Some qualitative properties of multirate digital control systems. *IEEE Trans. Automat. Contr.*, 44:765–770, 1999.

[22] Y. Ismail, E. Friedman, and J. Neves. Figures of merit to characterize the importance of On-Chip inductance. *IEEE Trans. on VLSI Systems*, 7(4):442–449, 1999.

[23] Hassan Khalil. *Nonlinear Systems*. Prentice-Hall, New Jersey, 2nd edition, 1996.

[24] Kihyung Kim, Wonseok Kang, and Hyungon Seo. Efficient Distributed Simulation of Hierarchical DEVS Models: Transforming Model Structure into a Non-Hierarchical One. In *Proceedings of Anual Simulation Symposium*, 2000.

[25] Tag Gon Kim. *DEVSim++ User's Manual. C++ Based Simulation with Hierarchical Modular DEVS Models*. Korea Advance Institute of Science and Technology, 1994. Available at http://www.acims.arizona.edu/.

[26] Ernesto Kofman. Quantized Bond Graph. Una aproximación para la simulación de sistemas físicos por eventos discretos. In *Proceedings of AADECA 2000*, pages 419–424, Buenos Aires, 2000.

[27] Ernesto Kofman. Simulación de sistemas dinámicos por cuantificación de estados. In *Proceedings of AADECA 2000*, pages 425–430, 2000.

[28] Ernesto Kofman. Algunas Propiedades de la Simulación de ODE's por Cuantificación de Estados. In *Proceedings of RPIC'01*, pages 532–537, 2001.

[29] Ernesto Kofman. Quantized-State Control. A Method for Discrete Event Control of Continuous Systems. Part I. In *Proceedings of RPIC'01*, pages 103–108, 2001.

[30] Ernesto Kofman. Quantized-State Control. A Method for Discrete Event Control of Continuous Systems. Part II. In *Proceedings of RPIC'01*, pages 109–114, 2001.

[31] Ernesto Kofman. A Second Order Approximation for DEVS Simulation of Continuous Systems. *Simulation*, 78(2):76–89, 2002.

[32] Ernesto Kofman. Discrete Event Simulation of Hybrid Systems. Technical Report LSD0205, LSD, UNR, 2002. Submitted to *SIAM Journal on Scientific Computing*. Available at www.fceia.unr.edu.ar/∼kofman/.

[33] Ernesto Kofman. Non Conservative Ultimate Bound Estimation in LTI Perturbed Systems. In *Proceedings of AADECA 2002*, Buenos Aires, Argentina, September 2002.

[34] Ernesto Kofman. Quantization–Based Simulation of Differential Algebraic Equation Systems. Technical Report LSD0204, LSD, UNR, 2002. To appear in *Simulation*.

[35] Ernesto Kofman. Quantized-State Control of Linear Systems. In *Proceedings of AADECA 2002*, Buenos Aires, Argentina, September 2002.

[36] Ernesto Kofman. Discrete Event Approximation of Continuous Controllers. Technical Report LSD0301, LSD, UNR, 2003. Submitted to *Automatica*. Available at www.fceia.unr.edu.ar/∼kofman/.

[37] Ernesto Kofman. Quantized-State Control. A Method for Discrete Event Control of Continuous Systems. *Latin American Applied Research*, 33(4):399–406, 2003.

[38] Ernesto Kofman and Sergio Junco. Un ambiente computacional para la modelización de sistemas dinámicos no lineales con Bond Graphs. In *Proceedings of RPIC99*, volume 3, 16–10, 1999. In Spanish.

[39] Ernesto Kofman and Sergio Junco. Quantized State Systems. A DEVS Approach for Continuous System Simulation. *Transactions of SCS*, 18(3):123–132, 2001.

[40] Ernesto Kofman and Sergio Junco. Quantized Bond Graphs: An Approach for Discrete Event Simulation of Physical Systems. In *Proceedings of ICBGM'01*, pages 369–374, Phoenix, 2001.

[41] Ernesto Kofman, J.S. Lee, and Bernard Zeigler. DEVS Representation of Differential Equation Systems. Review of Recent Advances. In *Proceedings of ESS'01*, 2001.

[42] A.J. Koskouej and A.S. Zinober. Sliding Mode Controller-Observer Design for Multivariable Linear Systems with Unmatched Uncertainty. *Cybernetics*, 36(1):95–115, 2000.

[43] J.S. Lee, H. Shim, J. Byun, and J.H. Seo. Robust Stabilization of a class of MIMO nonlinear systems: Sliding mode control & Passification Approach. In *Proceedings of ACC98*, 1998.

[44] D.G. Luenberger. *Linear and Nonlinear Programming, 2nd Ed.* Addison-Wesley, 1989.

[45] J. Lunze, B. Nixdorf, and J. Schrder. Deterministic Discrete Event Representation of Linear Continuous–Variable Systems. *Automatica*, 35(3):395–406, 1999.

[46] William Messner and Dawn Tilbury. *Control Tutorials For MATLAB And Simulink: A Web-Based Approach.* Addison-Wesley, 1998.

[47] R. Miller, A. Michel, and J. Farrel. Quantizer effects on steady-state error specifications of digital feedback control systems. *IEEE Trans. Automat. Contr.*, 34:651–654, 1989.

[48] R. Miller, M. Mousa, and A. Michel. Quantization and overflow effects in digital implementation of linear dynamic controllers. *IEEE Trans. Automat. Contr.*, 33:698–704, 1988.

[49] A. Naamane, A. Damiba, and N. Giambiasi. Bond Graphs and GDEVS. In *Proceedings of ICBGM'01*, pages 375–380, Phoenix, 2001.

[50] Martin Otter and François Cellier. *The Control Handbook*, chapter Software for Modeling and Simulating Control Systems, pages 415–428. CRC Press, Boca Raton, FL, 1996.

[51] Esteban Pagliero and Marcelo Lapadula. Herramienta Integrada de Modelado y Simulación de Sistemas de Eventos Discretos. Diploma Work. FCEIA, UNR, Argentina, September 2002.

[52] C. Pantelides. The Consistent Initialization of of Differential–Algebraic Systems. *SIAM Journal of Scientific and Statistical Computing*, 9(2):213–231, 1988.

[53] T. Park and P.I. Barton. State Event Location in Differential-Algebraic Models. *ACM Trans. Mod. Comput. Sim.*,, 6(2):137–165, 1996.

[54] Herbert Praehofer. *System Theoretic foundations for combined Discrete - Continuous System simulation*. PhD thesis, J. Kepler University of Linz, 1991.

[55] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes*. Cambridge: Cambridge University Press, 1986.

[56] R. Rosenberg and D. Karnopp. *Introduction to Physical System Dynamics*. McGraw Hill, 1983.

[57] N. Sayiner, H. Sorensen, and T. Viswanathan. A non-uniform sampling technique for A/D conversion. In *Proceedings of ISCAS'93*, volume 2, pages 1220–1223, 1993.

[58] T. Schlegl, M. Buss, and G. Schmidt. Development of Numerical Integration Methods for Hybrid (Discrete-Continuous) Dynamical Systems. In *Proceedings of Advanced Intelligent Mechatronics*, Tokio, Japan, 1997.

[59] L. Shampine and Reichelt M. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[60] J.E. Strömberg. *A Mode Switching Modeling Philosophie*. PhD thesis, Linköpings Universitet, 1994.

[61] J.H. Taylor. Toward a Modeling Language Standard for Hybrid Dynamical Systems. In *Proc. 32nd IEEE Conference on Decision and Control*, pages 2317–2322, San Antonio, TX, 1993.

[62] J.H. Taylor and D. Kebede. Modeling and Simulation of Hybrid Systems in Matlab. In *Proc. IFAC World Congress*, pages 275–280, San Francisco, CA, July 1996.

[63] G. Walsh, O. Beldiman, and L. Bushnell. Error Encoding Algorithms for Networked Control Systems. *Automatica*, 2001.

[64] Bernard Zeigler, George Ball, Hyup Cho, J.S. Lee, and Hessam Sarjoughian. Bandwidth Utilization/Fidelity Tradeoffs in Predictive Filtering. In *Proceedings of 1999 Spring SIW*. SISO, 1999.

[65] Bernard Zeigler and J. Kim. Extending the DEVS-Scheme knowledge-based simulation environment for real time event-based control. *IEEE Trans. on Robotics and Automation*, 3(9):351–356, 1993.

[66] Bernard Zeigler, Tag Gon Kim, and Herbert Praehofer. *Theory of Modeling and Simulation. Second edition.* Academic Press, New York, 2000.

[67] Bernard Zeigler and J.S. Lee. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. In *SPIE Proceedings*, pages 49–58, 1998.

[68] Bernard Zeigler and Hessam Sarjoughian. *Introduction to DEVS Modeling and Simulation with JAVA: A Simplified Approach to HLA-Compliant Distributed Simulations.* Arizona Center for Integrative Modeling and Simulation. Available at http://www.acims.arizona.edu/.

[69] Bernard Zeigler. *Theory of Modeling and Simulation.* John Wiley & Sons, New York, 1976.

Appendix A

List of Abbreviations

- BG – Bond Graph
- CCS – Continuous Control System
- DAE – Differential Algebraic Equation
- DEVS – Discrete EVent System Specification
- F.O.Q. – First Order Quantizer
- LTI – Linear Time Invariant
- PDE – Partial Differential Equation
- QBG – Quantized Bond Graph
- QS – Quantized System
- QSC – Quantized State Control
- QSS – Quantized State System
- QSS2 – Second Order Quantized State System
- ODE – Ordinary Differential Equation
- TI – Time Invariant

Appendix B

Pseudo–Codes for DEVS Simulation

The following pseudo-code corresponds to a simulator associated to a generic atomic model.

DEVS-simulator
  variables:
    tl // time of last event
    tn // time of next event
    s // state of the DEVS atomic model
    e // elapsed time in the actual state
    $y = (y.value, y.port)$ // current output of the DEVS atomic model
  when receive i-message (i, t) at time t
    $tl = t - e$
    $tn = tl + ta(s)$
  when receive *-message (*, t) at time t
    $y = \lambda(s)$
    send y-message (y, t) to parent coordinator
    $s = \delta_{\text{int}}(s)$
    $tl = t$
    $tn = t + ta(s)$
  when receive x-message (x, t) at time t
    $e = t - tl$
    $s = \delta_{\text{ext}}(s, e, x)$
    $tl = t$
    $tn = t + ta(s)$
end DEVS-simulator

The routine corresponding to a coordinator can be writen as follows:

DEVS-coordinator

variables:
tl // time of last event
tn // time of next event
$y = (y.value, y.port)$ // current output of the DEVS coordinator
D // list of children
IC // list of connections of the form $[(d_i, port_1), (d_j, port_2)]$
EIC // list of connections of the form $[(N, port_1), (d_j, port_2)]$
EOC // list of connections of the form $[(d_i, port_1), (N, port_2)]$
when receive i-message (i, t) at time t
send i-message (i, t) to all the children
when receive *-message (*, t) at time t
send *-message (*, t) to d^*
$d^* = \arg[\min_{d \in D}(d.tn)]$
$tl = t$
$tn = t + d^*.tn$
when receive x-message (($x.value, x.port$), t) at time t
$(v, p) = (x.value, x.port)$
for each connection $[(N, p), (d, q)]$
send x-message ((v, q), t) to child d
$d^* = \arg[\min_{d \in D}(d.tn)]$
$tl = t$
$tn = t + d^*.tn$
when receive y-message (($y.value, y.port$), t) from d^*
if a connection $[(d^*, y.port), (N, q)]$ exists
send y-message (($y.value, q$), t) to parent coordinator
for each connection $[(d^*, p), (d, q)]$
send x-message (($y.value, q$), t) to child d
end DEVS-coordinator

Finally, the root coordinator executes the following routine:

DEVS-root-coordinator
variables:
t // global simulation time
d // child (coordinator or simulator)
$t = t_0$
send i-message (i,t) to d
$t = d.tn$
loop
send *-message (*,t) to d
$s = \delta_{\text{int}}(s)$
$t = d.tn$
until end of simulation
end DEVS-root-coordinator

Index