On the design of event- and self-triggered controllers for certain classes of dynamical systems

Fairouz Zobiri

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques Appliquées

Arrêté ministériel : 25 mai 2016

Présentée par

Fairouz ZOBIRI

Thèse dirigée par **Brigitte BIDEGARAY-FESQUET** , CR, CNRS
et codirigée par **Nacim MESLEM**

préparée au sein du **Laboratoire Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Conception de contrôleurs événementiels pour certaines classes de systèmes dynamiques

On the design of event- and self-triggered controllers for certain classes of dynamical systems

Thèse soutenue publiquement le **15 février 2019**,
devant le jury composé de :

Madame BRIGITTE BIDEGARAY-FESQUET
CHARGE DE RECHERCHE, CNRS DELEGATION ALPES, Directeur de thèse
Monsieur NACIM MESLEM
MAITRE DE CONFERENCES, GRENOBLE INP, Examinateur
Monsieur MOHAMED DJEMAI
PROFESSEUR, UNIVERSITE DE VALENCIENNES - UVHC, Examinateur
Monsieur NICOLAS MARCHAND
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Président
Monsieur CHRISTOPHE PRIEUR
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Examinateur
Monsieur LIONEL ROSIER
PROFESSEUR, MINES PARISTECH, Rapporteur
Monsieur ALEXANDRE SEURET
DIRECTEUR DE RECHERCHE, CNRS DELEGATION OCCITANIE OUEST, Rapporteur
Madame FATIHA ALABAU
PROFESSEUR, UNIVERSITE DE LORRAINE, Examinateur

Acknowledgment

Because no thesis is an individual effort, and because what remains at the end of every journey is the memory of the people who cross one's path, I would like to thank all those who contributed to this achievement.

I would like to thank my supervisors, Brigitte Bidégaray-Fesquet and Nacim Meslem, first for entrusting me with this research topic, and for their help and support throughout the time we have spent together. Thank you for believing in me even when I doubted myself.

I would also like to thank Alexandre Seuret and Lionel Rosier for agreeing to review my work, and all the members of the jury, for their insightful remarks, challenging questions and precious advice.

Even if I can never thank them enough, I would like to thank my parents, my brother and all the other members of my family, including those who have left us, for their unconditional love. For there is no joy they cannot magnify, no pain they cannot quell and and no situation they cannot draw laughter from.

I would also like to thank all the personnel of the Laboratoire Jean Kuntzmann and Gipsa-lab for their assistance, the administrative staff, the IT teams, the PhD students, the teachers. Thank you for making life easier for me on a daily basis.

Finally, thank you to my wonderful friends, those I left, those I found and those who found me, and to my amazing office mates for offering me help and support before I even asked. Life would be tasteless without all of you.

Contents

List of Figures

List of Tables

Abstract

Event-triggered control offers a promising alternative to the classical, resource-consuming, periodic control. It suggests to replace the periodic, high frequency sampling used in the continuous-to-discrete transformations of control signals with aperiodic sampling. A new value of the event-triggered control law is computed only when the system's response is unsatisfactory. The control value is kept constant otherwise. In this thesis, we explore ways to induce fewer updates, and to have longer intervals between two samples. We also seek to make the algorithms that we design more detailed, by describing how to choose or compute the optimal parameters.

In the linear case, we present a stabilizing algorithm in which we relax the stability conditions on the system's Lyapunov function to produce fewer, sparser updates of the control law. Stability is ensured by maintaining the Lyapunov function below a certain decreasing threshold. The optimal threshold function is derived by solving a maximum generalized eigenvalue problem. This approach is then extended to switched linear systems. We also present a self-triggered version of this algorithm using Newton methods for optimization and root-finding. The reference tracking problem is treated in the event-triggered control framework as well. Finally, in the nonlinear case, due to the difficulty of finding a Lyapunov function, we explore the use of contraction analysis. This approach allows us to describe the nonlinear event-triggered control algorithm more thoroughly than if we had used Lyapunov techniques.

Résumé

La commande événementielle est une nouvelle alternative à la commande périodique classique jugée peu économe en ressources. Dans la commande événementielle, l'échantillonnage périodique à haute fréquence effectué lors du passage d'un temps continu à un temps discret, est remplacé par un échantillonnage apériodique. Dans cette approche, une nouvelle valeur de la loi de commande n'est calculée que lorsque la réponse du système enfreint des mesures de performances prédéfinies. Dans le cas contraire, la loi de commande est maintenue constante. Dans cette thèse, nous explorons des méthodes qui permettent de générer moins de mises à jour de la commande, et d'obtenir des intervalles plus longs entre deux échantillons. Nous nous efforçons aussi de concevoir des algorithmes plus détaillés, avec des procédures précises pour le calcul et le choix des paramètres.

Dans le cas des systèmes linéaires, nous présentons un algorithme de commande événementielle dans lequel nous relaxons les conditions de stabilité qui gouvernent la fonction de Lyapunov d'un système, afin de mettre à jour la commande moins souvent. La stabilité est garantie en maintenant la fonction de Lyapunov en dessous d'une borne supérieure décroissante. Nous définissons la borne supérieure optimale par la solution d'un problème de valeur propre maximale généralisée. Cette méthode est ensuite étendue aux systèmes linéaires à commutation. Nous présentons également une version self-triggered, ou auto-déclenchée, de cette méthode, dans laquelle le temps de mise à jour est prédit à l'avance à travers une combinaison d'algorithmes d'optimisation et de recherche de zéros. Nous traitons également le problème de suivi de trajectoire par commande événementielle.

Dans le cas des systèmes non-linéaires, en raison de la difficulté de trouver une fonction de Lyapunov, nous utilisons une autre définition de la stabilité, l'analyse de contraction. Cette approche nous permet de décrire un algorithme de commande événementielle avec plus de détails que si nous avions utilisé la méthode de Lyapunov.

List of Symbols and Abbreviations

$\mathbb{N}$	set of positive integers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^*$	set of non-zero real numbers
$\mathbb{R}^+$	set of positive real numbers
$\mathbb{R}^-$	set of negative real numbers
$\mathbb{R}^n$	n-dimensional vector space over the field of real numbers
$\mathbb{R}^{n\times m}$	set of $n \times m$ real matrices
0_n	zero vector in $\mathbb{R}^n$
$0_{n\times n}$	zero matrix in $\mathbb{R}^{n\times n}$
M^T	transpose of matrix M
M^{-1}	inverse of matrix M
$\mid . \mid$	the absolute value of a real scalar
$\| . \|$	both the Euclidean vector norm and the equivalent matrix norm
$\lambda_{\min}(N)$	minimum eigenvalue of the positive definite matrix N
$\lambda_{\max}(N)$	maximum eigenvalue of the positive definite matrix N
$\lambda_{\max}(M, N)$	maximum generalized eigenvalue of the pair (M, N) where N is a positive definite matrix
L^k	k^{th} generalized Lie derivative
$\nabla_{\mathrm{t}} f$	gradient of function f with respect to variable t
$\nabla^2_{\mathrm{t}} f$	Hessian of function f with respect to t
PLF	Pseudo-Lyapunov Function
LTI	Linear Time-Invariant
LMI	Linear Matrix Inequality
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
ISS	Input-to-State Stable
CLF	Control Lyapunov Function
CQLF	Common Quadratic Lyapunov Function
LQR	Linear Quadratic Regulator

Introduction

Optimizing the performance of control systems has been the topic of several research efforts throughout the history of the discipline. Researchers have always sought ways to improve the control task to take into account several aspects of the engineering design. Therefore, control tasks have been optimized to reduce energy consumption or financial costs, to meet actuator constraints, to speed up the convergence to a desired behavior or to follow an optimal trajectory.

However, one aspect of the control design that remained unchanged for quite some time is the way a control law is transformed from a continuous-time signal to discrete form in order to be eventually implemented on a digital platform. This transformation is often necessary, as one of the most widespread methods of control design is emulation. In emulation, the control law is first designed in continuous-time to meet a set of stability requirements and performance criteria. Then, the continuous-time control is transformed into a digital signal by first going through the process of sampling. The continuous-time signal is discretized at a fixed, and generally high, frequency. The final result is a discrete-time signal with a large number of samples evenly distributed in time.

Sampling at a high frequency is necessary to ensure a faithful representation that captures all the important features of the original continuous-time signal. Additionally, the selected sampling frequency has to satisfy the Shannon-Nyquist theorem, which states that the sampling frequency of a given signal has to be higher than twice the largest frequency present in the signal. If not, we run the risk of creating an aliasing problem. Aliasing occurs when a series of samples can represent more than one signal. These signals are called aliases of the original signal. To meet these requirements, we need to sample at high frequency even at times when a signal is not undergoing much change.

So the question that arises naturally is whether the sampling rate is a parameter that can be optimized as well. An optimal sampling would take into account the control needs of the system, and induce fast sampling in times of high demand, and slower sampling when the system does not need much attention. Inspired from Lebesgue sampling, asynchronous sampling appeared in signal processing as an amplitude-based sampling instead of a time-based sampling. In its control theory counterpart, event-triggered control, the control law is only updated when the system infringes predefined requirements imposed on it. Otherwise, the control is kept constant.

If we no longer need to recompute the control law so frequently, we can manage to reduce our energy consumption by being less solicitous of the CPU and the actuators. The CPU could also make use of the free time to turn its attention to the other tasks in the network. Additionally, if the system does not need a new control value to be sent with every ticking of the clock, then less data will have to travel through the network, avoiding the congestion of the communication channels and eliminating the risk of fatal packet losses. In some forms of event-triggered control, it is the sensors that are less solicited, thus decreasing the risk of wearing out these expensive pieces of equipment.

In fact, it is even possible to go one step further and design self-triggered control methods. In self-triggered control, the event-triggering conditions do not need to be monitored continuously. Instead, we use the system model to predict the time at which its response will breach the performance measures. The time of the next event is then determined in advance and no computations are required in between two updates of the control law. The sampling instants can be determined either offline where the entire sampling pattern is predicted over some time horizon. Or online, and in this case, at each update of the control, only the next update instant is computed.

Self-triggered control has been introduced to solve some of the issues encountered in event-triggered control. Having to check the event-triggering conditions at all instants can be computationally demanding, depending on the complexity of the conditions. It can also be demanding in terms of hardware, as extra equipment is sometimes necessary to monitor the event-triggering conditions. Consequently, self-triggered control can alleviate the demand on the software and the hardware. It can also further reduce the communications, as the output does not have to be measured all the time to evaluate the performance.

The benefits reaped from a reduction of the sampling rate cover a large array of applications. The most obvious example is the case of networked control systems where the different components of a control system (controller, plant, sensors, ...) exchange data through communication channels, that are also often wireless and shared among several other activities. A reduction of the sampling rate relieves the load on the communication channels, and frees them for the more urgent tasks. We can also mention the case of embedded systems, where resources are quite limited, and where every saved memory byte counts and every unit of energy matters. There is also the example of systems with actuators with a strong inertia where a frequent change in the control value can be energy consuming and damaging to the actuators.

In this thesis, we focus on the problem of reducing the communications between the controller and the plant. Such a reduction is achieved either by increasing the time span between two updates of the control law, or by reducing the number of updates, or both. To reach this goal, we build the event-triggering conditions around Lyapunov functions, and we relax the stability requirements that the system has to respect. We use this approach both for stabilizing and for reference tracking of linear time-invariant systems. In the case of nonlinear systems, since finding a Lyapunov function can be challenging, we build the event-triggering conditions using a different concept of stability called contraction analysis. With these approaches, we show that we can obtain fewer updates of the control law during a system's operation time.

But first, in this chapter, we introduce the terminology related to event-triggered control, we give some of the alternate terms found in the literature and establish the ones that will recur throughout this manuscript. We also present a brief history and non exhaustive literature review of the topic. Finally, we describe the layout of the rest of the manuscript.

Terminology

We call event-triggered control the strategy of organizing the control task where the value of the control is recomputed only when a set of predefined stability or performance criteria is violated. Otherwise, when the system's behavior is satisfactory, a zero-order hold (ZOH) maintains the value of the control law constant. This type of control has appeared in the literature under the designations of aperiodic, asynchronous, event-based or event-driven control. Lately, however, the control community seems to have a preference

for the term event-triggered control, and to largely settle for it. In this work, we also prefer this designation, even though event-based control can appear in some rare instances.

Conversely, classical control, or occasionally periodic control, is the name we give to the type of control where the control law is discretized at a fixed rate. The sampling frequency should satisfy the Shannon-Nyquist theorem. Moreover, it is common practice in control to demand even faster sampling, where the sampling frequency of a linear control law is chosen in a range going from six to twenty-five times the highest frequency. In the case of nonlinear control, however, no general condition on the sampling frequency exists, making it difficult to choose one. This type of control has been privileged for a long time because a strong and complete theory exists to support it, and because it can easily be transferred to the frequency domain via the z-transform. But as explained earlier, this method is very wasteful of resources.

Self-triggered control is sometimes viewed as a sub-category of event-triggered control, when considered by opposition to time-triggered control. However, self-triggered control refers to the type of control where the time at which an event will occur is determined in advance. In self-triggered control, we still determine a set of performance requirements, but we do not check if they are violated all the time, but instead predict these violations using the system's model and available measurements.

The stability and performance criteria mentioned before are referred to as the event-triggering conditions. These conditions are related to the behavior of the system. They can be built around the state, the output of the system or the control law itself. They can encompass stability criteria, errors with respect to set-points, or other design specifications. The events can be tied to other criteria as well, not directly related to the behavior of the system, such as a maximum time elapsed since the previous event, or a user-programmed event such as the mode changes of a switched system.

We also want to emphasize the separation between an event and the time at which an event occurs. An event is defined as the act of violating the stability or performance criteria, and the event-triggering conditions becoming true. The time at which an event is detected is referred to, most of the time, as the update instant, as it is the time instant at which the value of the control law is recomputed and updated. It can also be referred to as the sampling instant, because if we look at it from an emulation point of view, the control law is first designed in continuous-time and in closed

form (e.g. $u(t) = -Kx(t)$), then we choose to evaluate it, or sample it, only when an event occurs. The sampling instants are denoted by t_k, where $k \in \mathbb{N}$.

The inter-event time is the time lapse between two consecutive events. The validity of any event-triggered control strategy depends on the existence of a minimum inter-event time. If no such time interval can be proven to exist, we run the risk of creating Zeno behavior. The Zeno phenomenon occurs when an infinite number of events takes place in a finite interval of time. Most applications nowadays run on digital platforms with internal clocks, thus, consecutive updates of the control law are separated by one period of the clock signal, at least. Therefore, there is no real danger of creating Zeno behavior. However, with an event-triggered control strategy we are interested in longer inter-event times than the machine sampling time, and we still need to establish the existence of a minimum inter-event time.

Finally, the event generator designates the block, hardware or software, responsible for monitoring the event-triggering conditions and issuing the order to update the control when necessary. Even if we make few mentions of the event generator, it should be implicitly understood that event-triggering control requires this extra block. We also suppose that whether it is the state or the output, the quantity used in the event-triggering conditions is always available, either measured or observed, and sent to the event generator. Therefore, since we want to reduce the load on the communication channels, the event generator should be implemented in direct contact with the sensors or observers.

Figure 1 shows the different blocks of an event-triggered control mechanism. The diagram illustrates how the response is first read by the sensors, which communicate directly with an event generator. The response can designate the state or output, or any other value that quantifies the behavior of the system. At every time instant (or in practice according to a clock signal), the event generator uses the value of the response to evaluate the event-triggering conditions. If the event-triggering conditions turn true, the event generator gives an order to send the current value of the response, and any other necessary variable, to the controller that computes a new control value. If there is a set-point to track, it is sent to the controller too. In Figure 1, the full lines represent signals read and transmitted continuously, while the dashed lines represent signals transmitted only when an event occurs. The new value of the control is then sent to the plant. The new control is also stored (the new control becomes the current control in the diagram) and will be held constant as long as the event generator does not issue the

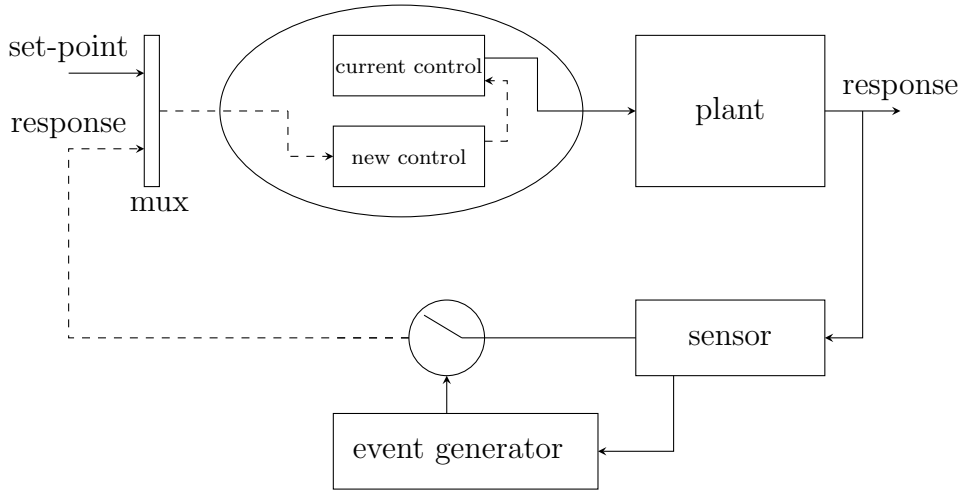


Figure 1: Block diagram of the event-triggered control process.

order to update it.

Brief History and Literature Review

Even though the realization that periodic sampling might not be the most efficient form of sampling came upon the signal processing community in the 60's and 70's, the control community stood by the classical digital control longer. Apart from brief mentions, as in [1] and [2] in 1959 and 1960, respectively, the name aperiodic control remained discreet. For a long time, this type of control was restrained to inherently aperiodic controllers, such as ON-OFF controllers [3].

More recently, the year 1999 saw the publication of three pioneering works in event-triggered control. First, inspired by Lebesgue integrals, the authors of [3] compare the conventional Riemann sampling with the event-based Lebesgue sampling, and conclude on the worthiness of event-based sampling for control purposes. In [4], the presence of low resolution encoders while experimenting with motor synchronization, forces the authors to successfully consider asynchronous control. Almost concurrently, the author of [5] developed and tested an event-based PID controller, managing to achieve large reductions of CPU utilization.

In addition to [5], the years 2000 saw the emergence of a large event-triggered control community, and a myriad of works that span a wide range

of control applications. In [6], event-triggered control is applied to systems in their state-space representation, and the event-triggered control problem is formulated as an input-to-state stability problem. The presence of disturbances is dealt with in [7] where the control is updated when the disturbance damages the response beyond a certain level. Since in practice, the event-triggering conditions are only checked periodically according to a clock signal, the authors of [8] integrate this periodic nature into the event-triggering conditions.

Event-triggered control then stretched to reach several other disciplines of control theory. For example, event-triggered methods for $\mathcal{H}_\infty$ control have appeared in [9] and [10]. Also, event-triggered model predictive control can be found in [11], for instance. Event-triggered control found adepts within the infinite-dimensional control community as well, among which we can cite [12] and [13]. With time, event-triggered control has also evolved to become a control design method on its own with the development of co-design methods. In this form of control, the control law and the event-triggering conditions are designed simultaneously, generally by solving a system of linear matrix inequalities. Among the works that describe these methods, we can mention [14], [15], and [16].

As to self-triggered control, the difficulty to predict the behavior of a dynamic system resulted in the lack of a complete theory. In most cases, the next execution times can only be stated implicitly. Besides, this task is made even more difficult in the presence of unknown disturbances. For these reasons, in most works, the next sampling instant is found by approximation or through optimization. Among the works on self-triggered control, we can cite [17] and [18].

For the event-triggered techniques designed for linear systems, we get the inspiration from [19]. In [19], the event-triggering conditions consist in comparing the system's behavior to the behavior of auxiliary systems, by comparing the evolution of their respective Lyapunov functions. In this thesis, in the case of linear systems, the auxiliary systems are replaced by an exponentially decreasing scalar threshold function. A similar idea can be found in [20], but the difference is that we introduce a way to determine the optimal parameters of threshold function. We also give a self-triggered version of this method and an extension to switched systems.

Objectives

The main objective of this work is to develop new event-triggered control methods for large classes (linear, nonlinear, switched) of systems that aim at reducing the number of communications between the CPU and the plant. To reach this objective, we have to develop event-triggering conditions that relax some of constraints on system performance. Such flexibility decreases the number of updates of the control law, and creates large inter-event intervals.

The second objective is to describe the algorithms with as many details as possible. We want to give algorithms that are easy to use, with procedures to find optimal decay rates for the exponential threshold that we use, Lyapunov matrices for the Lyapunov-like functions, common Lyapunov functions for switched systems, procedures to identify contraction regions for nonlinear systems and detailed optimization algorithms for self-triggered control.

Organization of the Manuscript

This manuscript is divided into five chapters. The first four chapters deal with event-triggered control for stabilization and reference tracking of linear systems, and stabilization of switched and nonlinear systems. The fifth and last chapter introduces a self-triggered stabilizing control algorithm for linear systems. A brief description of the chapters is given below.

- **Chapter 1 :** In this chapter, we present an event-triggered stabilizing control algorithm. The system performance is quantified by a Lyapunov-like function. At every time instant, the event-generator checks the evolution of this function. The event generator sends the order to update the control law if it reaches an exponentially decreasing threshold function. We formulate the problem of finding the Lyapunov-like function and the optimal parameters for the threshold function as a single optimization problem. The chapter ends with a comparison between this method and other event-triggered control methods found in the literature.

- **Chapter 2 :** The event-triggered control algorithm introduced in the first chapter is extended to the stabilization of switched linear systems. A switched system is composed of several subsystems, each subsystem gets activated according to a switching rule. The switching can be

time-based or state-based or both. The problem of finding a Lyapunov function common to all the subsystems and the optimal threshold function parameters is again formulated as a unique optimization problem.

- **Chapter 3 :** In control, we also need to be able to drive the output of a system to track a reference trajectory. We transform the tracking problem into a stabilization problem in the event-triggered control algorithm. We first select a reference system that produces the reference trajectory. Then, the difference between the trajectory of the event-triggered system and the reference trajectory is forced to tend to zero. This formulation is equivalent to expressing a stability problem on the error between the two trajectories. The pseudo-Lyapunov function associated with this error is compared, at all instants, to a constant upper threshold. An event is generated when the function exceeds this upper limit.

- **Chapter 4 :** In this chapter, we describe an event-triggered control algorithm for the stabilization of nonlinear systems. We rely on a different approach to stability known as contraction analysis. In this form of stability, all the trajectories of a system are made to converge towards a reference trajectory, by making the virtual displacement between any given trajectory and the reference trajectory tend to zero. The region of the state-space where this is realized is called a contraction region. Thus, the event generator issues the order to update the control when the system's trajectory leaves the contraction region.

- **Chapter 5 :** In this chapter, a self-triggered version of the event-triggered control algorithm of the first chapter is developed. This means that we try to predict the times at which the Lyapunov-like function meets the threshold function. We notice that the pseudo-Lyapunov function goes through a local minimum in every sampling interval, before reaching the threshold, and therefore, we can use an optimization algorithm to identify the time at which this minimum is attained. As this time is closer to the next execution time, it is used as a starting value in a root-finding algorithm to predict the time at which the next control update should be carried out.

Introduction (français)

L'optimisation des performances des systèmes de contrôle-commande a ouvert la voie à plusieurs disciplines sous-jacentes qui visent à améliorer les performances de la commande classique. La recherche dans ces domaines vise à améliorer les lois de commande pour inclure bien des aspects de la mise en œuvre finale par l'ingénieur. Pour cette raison, les méthodes de contrôle-commande actuelles prennent en compte la consommation d'énergie, le coût de l'implémentation, les limites des actionneurs, la rapidité de la convergence et le suivi d'une trajectoire optimale.

Cependant, l'un des aspects qui est resté longtemps sans se développer, est la conversion de loi de commande depuis un signal continu dans le temps, en un signal discret, afin de transférer la commande vers des plateformes numériques. Cette conversion est souvent primordiale, les contrôleurs étant dans la majorité des cas des supports numériques, et l'émulation étant la méthode de synthèse de lois de commande la plus répandue. L'émulation consiste à d'abord concevoir la commande en temps continu de sorte à satisfaire les critères de stabilité et de performance désirés. Ensuite le signal continu est transformé en signal discret par un procédé d'échantillonnage. L'échantillonnage se fait à une fréquence constante et généralement élevée. Le résultat de ce procédé est un signal discret possédant un grand nombre d'échantillons, uniformément distribués dans le temps.

L'échantillonnage à haute fréquence est nécessaire pour obtenir une représentation fidèle à l'original, et qui capture toutes les propriétés importantes du signal en temps continu. De plus, la fréquence d'échantillonnage choisie doit satisfaire le théorème de Shannon-Nyquist qui stipule que la fréquence d'échantillonnage d'un signal doit être au moins deux fois plus élevée que la plus haute fréquence présente dans ce signal. En automatique, cette condition est encore plus contraignante car on requiert d'échantillonner à des fréquences égales à 6 à 25 fois la plus haute fréquence. Si cette condition n'est pas satisfaite, un repliement du spectre se produit. Lors d'un repliement du

spectre, le signal discret produit peut représenter plusieurs signaux continus différents et pas uniquement le signal d'origine. De ce fait, nous devons échantillonner à haute fréquence même lorsque le signal est de nature sporadique.

La question qui se pose naturellement dans ce cas est de savoir si la fréquence d'échantillonnage est un paramètre qui peut être optimisé aussi. Un échantillonnage optimal prendrait en compte les besoins en contrôle du système en donnant lieu à un échantillonnage rapide dans les zones de fonctionnement où la demande en contrôle est élevée, mais à un échantillonnage à basse fréquence quand le système ne requiert pas beaucoup d'attention. Inspiré de l'échantillonnage de Lebesgue, l'échantillonnage asynchrone est apparu en traitement du signal comme une forme d'échantillonnage basée sur l'amplitude et non sur le temps. En contrôle-commande, cette nouvelle approche a donné naissance à la commande événementielle dans laquelle la loi de commande n'est recalculée que si le système enfreint des mesures de performance prédéfinis, et sinon la loi de commande est maintenue constante.

Le fait de ne plus avoir besoin de recalculer la loi de commande aussi fréquemment nous permet de réduire la consommation d'énergie du système, en sollicitant de moins en moins le CPU et les actionneurs. Le CPU pourrait alors utiliser ce gain de temps libre pour se tourner vers d'autres tâches du réseau, qui pourraient en avoir plus besoin. De plus, si le système n'a plus besoin qu'on lui envoie une nouvelle valeur de la loi de commande à chaque cycle d'horloge, alors moins de données devront être transmises dans le réseau, évitant ainsi la congestion des canaux de communication, et les pertes de paquets. Dans d'autres variantes de la commande événementielle, ce sont les capteurs qui sont moins sollicités, ce qui permet d'éviter d'user ces composants coûteux.

Récemment, la commande événementielle a connu une nouvelle mutation sous la forme de la commande self-triggered ou auto-déclenchée. Dans la commande self-triggered, les critères de performance – ou conditions de garde – du système n'ont plus besoin d'être surveillés constamment, mais le modèle du système est utilisé pour prédire le temps auquel la réponse du système va enfreindre ces critères de performance. L'instant de la prochaine mise à jour est déterminé à l'avance et donc aucun calcul n'est effectué entre deux temps de mise à jour. Ces instants sont déterminés soit hors ligne, et alors la séquence de commande est prédite en totalité pour un horizon de temps défini, soit en ligne, et l'instant suivant est calculé à chaque instant de mise à jour.

La commande self-triggered a été introduite pour résoudre certains problèmes rencontrés par la commande événementielle. Surveiller les conditions de garde à tout instant peut être coûteux suivant la complexité des calculs à effectuer. Il sera nécessaire de recourir à des installations supplémentaires pour surveiller les conditions de garde. Par conséquent, la commande self-triggered peut alléger les demandes en calcul et en équipement. Ce type de commande permet également de réduire les communications entre le contrôleur et le système, mais aussi entre le système et les capteurs, car si nous n'avons pas besoin de surveiller la réponse du système, nous n'avons pas besoin de la mesurer.

Les avantages de la commande événementielle peuvent bénéficier à un large spectre d'applications. L'exemple le plus évident est le cas des systèmes déployés en réseau et où les différentes composantes (contrôleur, système, capteurs, ...) échangent des informations à travers des canaux de communication à bandes passantes limitées, souvent sans fil et partagés entre plusieurs activités. Nous pouvons aussi mentionner les systèmes embarqués où les ressources sont limitées, et où toutes les économies d'énergie ou d'espace de stockage sont les bienvenues. Il y a aussi les systèmes où les actionneurs ont une grande inertie. Dans ces systèmes, le changement trop fréquent de la valeur de la loi de commande entraîne une importante consommation d'énergie et comporte le risque d'endommager les équipements.

Dans cette thèse, nous nous concentrons sur la question de comment réduire les communications entre le contrôleur et le système. Cela passe par la réduction du nombre de mises à jour de la loi de commande et l'augmentation du temps écoulé entre deux mises à jour consécutives de la commande. Pour atteindre ces objectifs, nous construisons les conditions de garde autour des fonctions de Lyapunov du système, et nous relaxons les critères de stabilité que ces dernières doivent respecter. Nous utilisons cette approche pour la stabilisation et le suivi de trajectoire pour les systèmes linéaires à temps invariant. Dans le cas des systèmes non-linéaires, trouver des fonctions de Lyapunov peut s'avérer difficile pour certains systèmes, pour cela nous définissons les conditions de garde en utilisant une forme différente de stabilité, l'analyse de contraction. Avec ces méthodes, nous démontrons que nous pouvons induire moins de mises à jour de la loi de commande et de plus grands intervalles de temps entre les événements.

Dans ce qui suit, nous définissons clairement les objectifs de ce travail et décrivons l'organisation de ce manuscrit.

Objectifs

L'objectif de ce travail est de développer de nouvelles stratégies de commande événementielle qui ont pour but de réduire le nombre de communications entre le CPU et le système. Pour atteindre cet objectif, nous devons formuler des conditions de mise à jour qui relaxent certains critères de performance qui peuvent être contraignants, mais qui sont partie intégrante du contrôle-commande classique. Cette flexibilité de la commande événementielle permet de moins solliciter le contrôleur.

Le deuxième objectif est de décrire les algorithmes que nous introduisons avec le plus de détail possible. Nous voulons fournir des algorithmes faciles à utiliser et qui intègrent de manière fluide comment calculer ou choisir les paramètres auxquels ils font appel. Nous montrerons comment calculer la constante de décroissance optimale pour les fonctions seuils que nous utilisons, les matrices de Lyapunov qui définissent les pseudo-fonctions de Lyapunov, les fonctions de Lyapunov communes pour les systèmes à commutation. Nous donnons également des procédures pour définir les régions de contraction d'un système non-linéaire. Nous détaillons également les algorithmes d'optimisation pour la commande self-triggered.

Organisation du manuscrit

Le manuscrit est divisé en cinq chapitres. Les quatre premiers chapitres traitent de la commande événementielle pour la stabilisation et le suivi de trajectoire des systèmes linéaires à temps invariant (LTI), et la stabilisation des systèmes à commutation et des systèmes non-linéaires. Le cinquième et dernier chapitre introduit un algorithme de commande self-triggered pour systèmes linéaires. Une brève description des chapitres est donnée dans ce qui suit.

- **Chapitre 1**: nous présentons une méthode de commande événementielle pour stabiliser un système LTI. Les performances du systèmes sont quantifiées par une pseudo-fonction de Lyapunov. A chaque instant, un générateur d'événements surveille l'évolution de cette fonction. Lorsque cette dernière atteint une borne supérieure donnée par une fonction exponentiellement décroissante, le générateur d'événements donne l'ordre

de mettre à jour la loi de commande. Nous formulons les problèmes de trouver une pseudo-fonction de Lyapunov et de calculer les paramètres de la borne supérieure en un seul problème d'optimisation. Nous terminons le chapitre par une comparaison avec d'autres méthodes de commande événementielle.

- **Chapitre 2**: la stratégie de commande événementielle présentée au premier chapitre est étendue aux systèmes à commutation. Un système à commutation est composé de plusieurs sous-systèmes, chaque sous-système devient actif ou inactif suivant une loi de commutation. Le problème de trouver une fonction de Lyapunov commune à tous les sous-systèmes et les paramètres de la fonction seuil est ici aussi présenté comme un seul problème d'optimisation.

- **Chapitre 3**: dans cette partie, c'est le problème de suivi de trajectoire qui est traité. Nous transformons le problème de suivi de trajectoire en un problème de stabilité. Nous définissons d'abord un système de référence qui produit la trajectoire souhaitée. Ensuite la différence entre la trajectoire du système à commande événementielle et celle du système de référence est contrôlée pour tendre vers zéro. La pseudo-fonction de Lyapunov associée à l'erreur entre les deux trajectoires est comparée à tout instant à une borne supérieure constante. La loi de commande est mise à jour lorsque cette borne est atteinte.

- **Chapitre 4**: nous présentons un algorithme de contrôle événementiel des systèmes non-linéaires. Pour cela, nous nous appuyons sur une approche différente de la stabilité, l'analyse de contraction. Dans cette forme de stabilité, toutes les trajectoires d'un système ont besoin de converger vers une même trajectoire nominale, en faisant en sorte que le déplacement virtuel entre deux trajectoires adjacentes tende vers zéro. Une région de l'espace d'états dans laquelle cette condition est satisfaite est nommée région de contraction. Dans notre stratégie de commande événementielle, le générateur d'événement donne l'ordre de mettre à jour la commande lorsque la trajectoire du système quitte la région de contraction.

- **Chapitre 5**: ici une version self-triggered de l'algorithme de commande événementielle du Chapitre 1 est développée. Ceci veut dire, que nous souhaitons prédire les instants auxquels la pseudo-fonction de Lyapunov atteint la borne supérieure. Nous remarquons que, entre deux événements, la pseudo-fonction de Lyapunov passe par un minimum local avant d'atteindre la borne supérieure. Par conséquent,

nous pouvons utiliser un algorithme de minimisation pour identifier le temps auquel ce minimum est atteint. Nous utilisons ensuite ce temps pour initialiser un algorithme de localisation de zéro qui va identifier le temps auquel la fonction de Lyapunov croise la borne supérieure et donc le temps de la prochaine mise à jour de la commande.

Chapter 1

Event-Triggered Stabilizing Controller for Linear Systems

Contents

1.1 Introduction

In this chapter we introduce a stabilizing event-triggered control strategy for Linear Time-Invariant systems. This strategy aims at simplifying and improving the method developed in [19]. In this method the authors propose to bound the system's Lyapunov function-like between the Lyapunov functions of a slower and a faster system. This creates the need to define auxiliary systems and thus expands the complexity of the problem. In our method though, we replace the bounding Lyapunov functions by a simple scalar function. The method is applied over two stages by splitting the operation time of the system to be controlled into two parts. The first part is referred to as the *transient region* and the second part is the *steady-state region*.

In the transient region the event-triggered strategy relies on monitoring a pseudo-Lyapunov function associated to the system to be stabilized. The pseudo-Lyapunov function, also referred to as a Lyapunov-like function, is defined in the same way as a classical Lyapunov function, the only difference is that the pseudo-Lyapunov function does not have to be strictly decreasing in time along the trajectories of the system. It is allowed to increase locally, provided that it remains bounded from above and that it ultimately converges to zero. The control law is updated to prevent the pseudo-Lyapunov function from taking values above a certain decreasing upper bound.

In the steady-state region we no longer allow the pseudo-Lyapunov function to increase locally. Instead, the control law is updated as soon as the pseudo-Lyapunov function stops decreasing, that is when its first time derivative along the trajectories of the system vanishes.

In what follows we give proper definitions of the quantities described above. We provide the system model, the form of the pseudo-Lyapunov function and how to schedule the control law. The problem of finding both a suitable pseudo-Lyapunov function and the optimal upper bound is formulated as a unique optimization problem. We also explain why and how we delimit the operation time into transient and steady-state regions. Subsequently, we test the developed control strategy on two numerical examples, first on a single-input single-output (SISO) system, then on a multiple-input

multiple-output (MIMO) system. Finally, we compare the results obtained through this method to other methods from the literature. We carry out the comparisons using the SISO system.

1.2 Problem Definition

1.2.1 System Description

We consider the following Linear-Time Invariant (LTI) system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \\ x(t_0) &= x_0, \end{aligned} \tag{1.1}$$

where

- $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input, and $y(t) \in \mathbb{R}^p$ is the output,
- $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{p\times q}$ are constant matrices,
- t_0 is the initial time and x_0 is the initial state.

We suppose that the pair (A, B) is controllable and we want to stabilize the system, i.e. to drive the state $x(t)$ to the equilibrium point. We consider, without loss of generality the equilibrium point to be located at the origin. For this, we design a stabilizing state-feedback control law of the form

$$u(t) = -Kx(t). \tag{1.2}$$

The feedback gain K renders the matrix $(A - BK)$ Hurwitz, by placing its eigenvalues in the open left half of the complex plane. The resulting closed-loop system,

$$\dot{x}(t) = (A - BK)x(t),$$

is globally asymptotically stable.

1.2.2 Event-Triggered Control Law

In event-triggered control, the control law is a piecewise-constant signal. The value of the control is kept constant when the system provides a satisfactory

performance. If, on the contrary, the behavior of the system is deemed unacceptable, the control law is updated to a new value.

The performance of the system is expressed as a condition on the state or output. This condition is called an *event-triggering condition*. The violation of the event-triggering condition is called an *event*. The event-triggering conditions are monitored by an event generator that issues orders to the controller to produce a new control value or to remain idle. Let t_k, $k \in \mathbb{N}$, be the times at which an event occurs in the system. The control law is then scheduled as follows

- at time t_k when an event occurs

$$u(t_k) = -Kx(t_k), \tag{1.3}$$

- for all $t \in (t_k, t_{k+1})$

$$u(t) = u(t_k). \tag{1.4}$$

For each case, the corresponding system model is

- at time t_k when an event occurs

$$\dot{x}(t) = (A - BK)x(t), \tag{1.5}$$

- for all $t \in (t_k, t_{k+1})$

$$\dot{x}(t) = (A - BK)x(t) - BK\Delta_k x(t), \tag{1.6}$$

  where $\Delta_k x(t) = x(t_k) - x(t)$.

Then in the interval $[t_k, t_{k+1})$, for all k, System (1.1) has a unique solution

$$x(t) = e^{(A-BK)(t-t_k)}x(t_k) - \int_{t_k}^{t} e^{(A-BK)(t-s)}BK\Delta_k x(s)ds. \tag{1.7}$$

An important property of this solution is its Lipschitz continuity, with constant L_x, namely

$$\|\Delta_k x(t)\| \leq L_x(t - t_k), \quad \forall k. \tag{1.8}$$

1.2.3 Lyapunov Stability

Since the system in its closed-loop form is asymptotically stable, there exists a quadratic positive definite function $V : \mathbb{R}^n \to \mathbb{R}^+$,

$$V(x(t)) = x(t)^T P x(t), \tag{1.9}$$

(where $P \in \mathbb{R}^{n \times n}$ is a symmetric positive-definite matrix) that decreases along the trajectories of the system with rate

$$\frac{dV(x(t))}{dt} = -x(t)^T Q x(t), \tag{1.10}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive-definite matrix that satisfies the Lyapunov equation

$$(A - BK)^T P + P(A - BK) = -Q. \tag{1.11}$$

In event-triggered control $V(x(t))$ is referred to as a pseudo-Lyapunov function or PLF, as it keeps its form (1.9), but its derivative along the trajectories of the system is no longer described by Equation (1.10). The PLF is described in more details in Section 1.2.4.

1.2.4 Event-Triggering Conditions

To determine the event-triggering conditions, we assume that the state vector can be fully measured or reconstructed, so as to evaluate a Lyapunov function for the system. The function serves to gauge the performance of the system, as it is an indicator of the stability of the system.

In classical control, the Lyapunov function decreases along the trajectories of the system at all times. In contrast, in our event-based control method, we define a pseudo-Lyapunov function (PLF) that is allowed to increase locally, as long as it remains bounded from above and that it ultimately tends toward zero. The control law is updated when the PLF reaches a predefined upper bound or threshold. Figure 1.1 illustrates the evolution of the pseudo-Lyapunov function.

When the control law is updated, the time derivative of the PLF along the trajectories of the system is given by

$$\frac{dV(x(t))}{dt}|_{t=t_k^+} = -x(t_k)^T Q x(t_k). \tag{1.12}$$

which is the same as the derivative of a classical Lyapunov function given by Equation (1.10). However, between two events, the derivative of the PLF differs from its classical counterpart and is given by

$$\frac{dV(x(t))}{dt} = -x(t)^T Q x(t) - 2\Delta_k x(t)^T K^T B^T P x(t), \quad t \in (t_k, t_{k+1}). \quad (1.13)$$

In Figure 1.1, we have represented the threshold as a positive, strictly decreasing function of time, that bounds the PLF from above. We later show that an exponentially decreasing threshold is a natural choice that ensures that the updates of the control law are sparse in time, while the PLF converges to zero as time tends to infinity. Even though such a threshold has been used in previous works such as [20], in the present work we give an explicit method to simultaneously find the PLF and the optimal upper threshold.

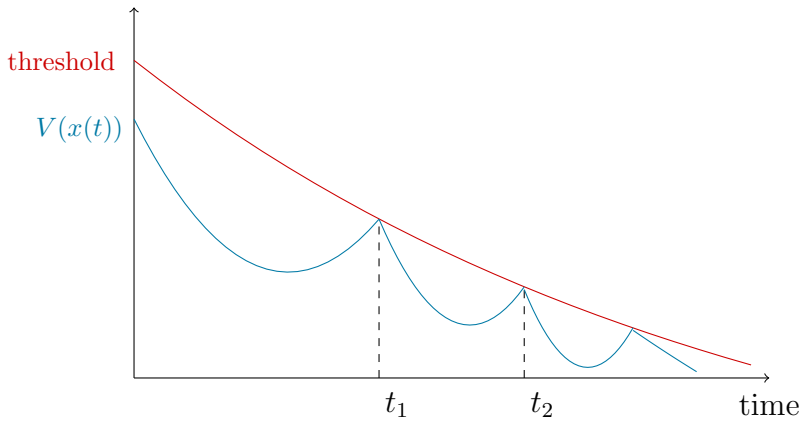


Figure 1.1: Behavior of the pseudo-Lyapunov function.

The approach described above is the one we use in transient time. When the state of the system enters in a neighborhood of the equilibrium point, we want it to converge as fast as possible to equilibrium and to remain there. However, increases in the PLF leave the state free to wander off the equilibrium point. Because of this, at this point, we force the PLF to decrease, by forcing its time derivative to remain negative. The control law is then updated when the time derivative of the PLF becomes zero. Switching to this mode of operation has the advantage of speeding up the convergence to the equilibrium point. Figure 1.2 illustrates the transient and steady-state operation.

We cannot predict whether this change in event-triggering conditions will increase or decrease the number of updates of the control law. In theory, asking for the time derivative of the PLF to be negative all the time demands more control effort than keeping the PLF below a certain threshold. However, we will show in Section 1.4, when we test this approach on a SISO and a MIMO system, that switching to this condition can increase or decrease the number of updates depending on the application.

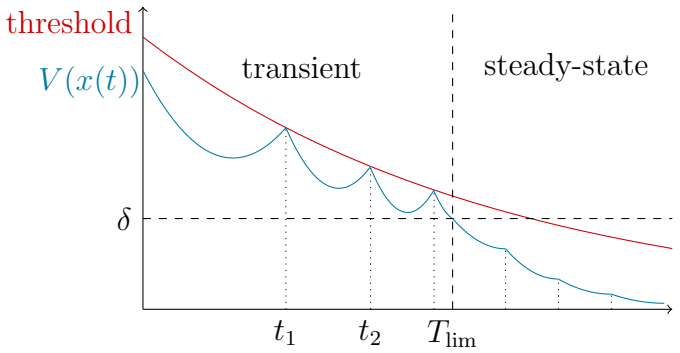


Figure 1.2: Behavior of the Lyapunov-like function in transient and steady state regions.

We later show that it is preferable to define the limits of the transient and steady-state regions in terms of the threshold function that we introduce below. For now, we denote by T_{lim} the time instant separating the transient and steady-state regions. Thus we define the transient region as all time instants t such that $t < T_{\text{lim}}$ ($T_{\text{lim}} > 0$). The steady state consists thus of all time instants t, such that $t \geq T_{\text{lim}}$.

1.3 Event-Triggering Conditions

In this section, we define the event-triggering conditions according to which the control law is updated. At first, for the transient time, we establish the properties of the upper threshold. We then demonstrate how to obtain the threshold function that satisfies these properties. An optimization problem is then derived and solved for both the PLF and the optimal threshold. We also explain why such threshold is optimal.
In the second part, the steady-state event-triggering conditions are given in terms of the time derivative of the PLF.
In both parts, we prove the stability of the system under the event-triggered

control law. We also give proof of the existence of a minimum inter-event time.

1.3.1 Triggering Conditions in the Transient Region

1.3.1.1 Decreasing Threshold Function

We look for a scalar function $W : \mathbb{R}^+ \to \mathbb{R}^+$, such that for all t

$$\frac{dW(t)}{dt} < 0, \tag{1.14}$$

and

$$V(x(t)) \ \leq \ W(t). \tag{1.15}$$

To find a function corresponding to the above criteria, we first look for a scalar $\lambda < 0$ such that the following constraint is satisfied

$$\frac{dV(x)}{dt}|_{t=t_k^+} \leq \lambda V(x(t_k)), \quad \forall k. \tag{1.16}$$

This is equivalent to

$$-x(t_k)^T Q x(t_k) \leq \lambda x(t_k)^T P x(t_k). \tag{1.17}$$

We are interested in ensuring the fastest possible convergence, by making $|\lambda|$ as large as possible. Equation (1.17) tells us that λ can be chosen as the maximum generalized eigenvalue of the pair $(-Q, P)$, denoted by $\lambda_{\max}(-Q, P)$ and defined as (see [21])

$$\lambda_{\max}(-Q, P) \equiv \inf\{\lambda \in \mathbb{R}^- | \lambda P + Q > 0\}, \quad \lambda_{\max}(-Q, P) < 0. \tag{1.18}$$

From now on, we drop the argument $(-Q, P)$ and refer to the maximum generalized eigenvalue simply as $\lambda_{\max}$.

Extrapolating equation (1.16) for all t and for $\lambda = \lambda_{\max}$, we obtain the first order linear differential inequality,

$$\frac{dV(x(t))}{dt} \leq \lambda_{\max} V(x(t)), \tag{1.19}$$

which admits a solution

$$\frac{dV(x(t))}{dt} \leq V(x_0) e^{\lambda_{\max}(t-t_0)}. \tag{1.20}$$

Thus, we define the threshold function as

$$W(t) = W_0 e^{-\alpha(t-t_0)}, \tag{1.21}$$

where $W_0 \geq V(x_0)$ guarantees that at $t = t_0$, $V(x(t))$ is below $W(t)$. The coefficient α is selected such that $0 < \alpha < |\lambda_{\max}|$ in order to enforce the condition given by equation (1.15).

1.3.1.2 Stability in the Transient Region

We can now describe the event-triggered strategy, and prove its stability.

Definition 1. *Let $V(x)$ be the function given by equation* (1.9)*, and $W(t)$ be the exponential function given by equation* (1.21)*. The time instant t_{k+1} ($k \in \mathbb{N}$) at which the control $u(t)$ is updated is defined as*

$$t_{k+1} = \inf\{t > t_k, \quad V(x(t)) = W(t)\}.$$

Additionally, the first update time is t_0.

Theorem 1. *The control law defined by equations* (1.3) *and* (1.4) *and scheduled by the event-triggering condition given by definition 1 renders system* (1.1) *asymptotically stable.*

Proof. Since $W(t)$ is exponentially decreasing toward zero, to prove that the equilibrium point is stable, it is sufficient to show that $V(x(t))$ is always lower than $W(t)$.
We already know that at t_0, $V(x_0) < W_0$, by definition, and we need to show that for all $t > t_0$, whenever $V(x(t)) = W(t)$, $V(x(t))$ is pushed back below $W(t)$, i.e. that $dV(x(t))/dt < dW(t)/dt$, at $t = t_k^+$.

At $t = t_k$ and from equation (1.16)

$$-x(t_k)Qx(t_k) \leq \lambda_{\max}\, V(x(t_k)). \tag{1.22}$$

But, since at instant $t = t_k$, $V(x(t_k)) = W(t_k)$, we can re-write the last equation as

$$\begin{aligned} -x(t_k)Qx(t_k) &\leq \lambda_{\max}\, W(t_k) \\ &< -\alpha W(t_k). \end{aligned}$$

In other words, at $t = t_k^+$

$$\frac{dV(x(t))}{dt}\Big|_{t=t_k^+} < \frac{dW(t)}{dt}\Big|_{t=t_k^+},$$

Then, for all $t_k < t < t_{k+1}$,

$$V(x(t)) < W(t),$$

thus completing the proof.

□

1.3.1.3 Computing the Decay Rate α

We say of a threshold $W(t)$ that it is optimal if it decreases to zero as fast as possible, while ensuring the best trade-off between the number of updates and the smoothness of the system's response. We have already established that the maximum possible value for α is the maximum generalized eigenvalue of the pair $(-Q, P)$. The problem of finding the maximum generalized eigenvalue is discussed in this section.

Equation (1.16) can be re-written as

$$x(t_k)^T((A-BK)^TP + P(A-BK))x(t_k) \leq \lambda_{\max}\, x(t_k)^T P x(t_k). \quad (1.23)$$

$\lambda_{\max}$ is the solution of the following optimization problem

$$\begin{aligned} &\text{minimize} \;\; \lambda \\ &\text{subject to} \\ &\quad (A-BK)^TP + P(A-BK) \leq \lambda P \\ &\quad P > 0 \end{aligned} \quad (1.24)$$

This problem is quasiconvex [21] because the function $\lambda_{\max}(-Q, P)$ is defined on $\mathbb{R}^-$ which is convex and it satisfies Jensen's inequality for quasiconvex functions, i.e. for any symmetric matrices $P_1 > 0$, $P_2 > 0$, $-Q_1$, $-Q_2$ and real $0 \leq \theta \leq 1$

$$\begin{aligned} \lambda_{\max}(\theta(-Q_1) + (1-\theta)(-Q_2),\, &\theta P_1 + (1-\theta)P_2) \\ &\leq \max\{\lambda_{\max}(-Q_1, P_1), \lambda_{\max}(-Q_2, P_2)\}. \end{aligned}$$

There exist many works devoted to the solution of the problem (1.24) in the literature, among which we can cite [21] and [22], as well as several ready-for-use software solutions, such as the 'gevp' command of MATLAB, and the YALMIP toolbox.

Once we have $\lambda_{\max}$, we select α such that

$$0 < \alpha < |\lambda_{\max}|. \quad (1.25)$$

We have stated before that the larger α, the better, but α can be chosen anywhere inside the interval $(0, \lambda_{\max})$. So, what happens as α takes on different values?

Choosing α closer to $\lambda_{\max}$ means a larger value of α and a threshold that decreases faster to zero. Such a choice of α leads to a faster response and a quicker convergence to steady-state. It also induces less events as the systems reaches equilibrium more quickly.

On the other hand, choosing α far from $|\lambda_{\max}|$ has the advantage of increasing the inter-event time. Indeed, the average time between successive events is higher when α is far from $|\lambda_{\max}|$ as the threshold does not decrease as steeply as in the previous case. This leads to larger increases of the pseudo-Lyapunov function, and therefore, more time elapses between events. However, this choice of α leads to a slower convergence and a higher number of events. In addition, allowing the pseudo-Lyapunov function to increase further has a negative effect on the quality of the response. The increases of the Lyapunov function induce oscillations in the response, and larger increases lead to more pronounced oscillations.

1.3.2 Steady-State Triggering Conditions

We demand a stronger control effort in the neighborhood of steady-state. In this case, we require the time derivative of $V(x(t))$ to remain negative. This will drive the state to the equilibrium position, from which it is harder to deviate, thus spreading the controls further in time.

Definition 2. *Let $V(x)$ be the function given by equation* (1.9)*. The time instant t_{k+1} ($k \in \mathbb{N}$) at which the control $u(t)$ is updated is defined as (see [23])*

$$t_{k+1} = \inf \left\{ t > t_k, \quad \frac{dV(x(t))}{dt} \geq 0 \right\}.$$

Theorem 2. *The control law defined by equations* (1.3) *and* (1.4) *and scheduled by the event-triggering condition given by definition 2, renders system* (1.1) *globally asymptotically stable.*

Since the time derivative of the Lyapunov function remains negative for all t, the system is globally asymptotically stable.

1.3.3 Defining T_{lim}

We define T_{lim} as the time when $W(t)$ reaches a predefined distance (a small enough distance $\delta > 0$) from the equilibrium point. T_{lim} is then defined as

$$T_{\text{lim}} = \min\{t \mid W(t) \leq \delta\}, \tag{1.26}$$

Since, $W(t)$ is a known function, we can give an exact value for T_{lim} as

$$T_{\text{lim}} = \min\{t \mid W(t) = \delta\} \quad = \frac{1}{\alpha} \ln(\frac{W_0}{\delta}). \tag{1.27}$$

1.3.4 Minimum Inter-sample Time

To validate an event-triggered control algorithm, it is not enough to prove the stability of the system to be controlled. It is also necessary to prove that any two consecutive events are separated by a minimum interval of time. We denote by $\tau_{\min} > 0$, the minimum inter event time.

The accumulation of events is undesirable as it creates a situation where an infinite number of updates in a finite interval of time. In the framework of hybrid systems, this situation is known as the Zeno behavior and is even considered as a form of instability [24].

However, for our event-triggered control algorithm, we can prove the existence of such $\tau_{\min}$, and state the following theorem.

Theorem 3. *There exists a minimum time delay $\tau_{\min} > 0$ independent of k, such that for all $k \in \mathbb{N}$, $t_{k+1} - t_k > \tau_{\min}$.*

Proof. We give the proofs for the existence of $\tau_{\min}$ in the transient time and in stead-state separately.

1. *Transient Region*:

   To prove the existence of a minimum time delay between any two consecutive events, we need to show that $V(x(t))$ decreases faster than $W(t)$ for some time after an update of the control law, so that no other intersection is possible in that amount of time. For this, we first find lower and upper bounds on $\|x(t_k)\|$. Afterward, we use these bounds to show that $dV(x(t))/dt < dW/dt$, in some interval $[t_k, t_k + \tau)$.

   - Lower and upper bounds on $\|x(t_k)\|$:
     At $t = t_k$, $V(x(t_k))$ admits the following lower and upper bounds

     $$\lambda_{\min}(P)\|x(t_k)\|^2 \leq V(x(t_k)) \leq \lambda_{\max}(P)\|x(t_k)\|^2. \tag{1.28}$$

Since $V(x(t_k)) = W(t_k)$, and $W(t)$ is an exponentially decreasing function, such that, in transient time, $\delta < W(t_k) < W_0$, we can write

$$\sqrt{\frac{\delta}{\lambda_{\max}(P)}} \leq \|x(t_k)\| \leq \sqrt{\frac{W_0}{\lambda_{\min}(P)}}. \tag{1.29}$$

We denote $M = \sqrt{W_0/\lambda_{\min}(P)}$.

- Proving that $dV/dt < dW/dt$ for $t \in [t_k, t_k + \tau)$:
  When $t \in (t_k, t_{k+1})$, the time derivative of $V(x(t))$ is given by Equation (1.13), which can be re-written in terms of $x(t_k)$ and $\Delta_k x(t)$ as

$$\begin{aligned}\frac{dV(x(t))}{dt} =& - x(t_k)^T Q x(t_k) + x(t_k)^T Q \Delta_k x(t) \\ &+ \Delta_k x(t)^T (Q - 2K^T B^T P) x(t_k) \\ &+ \Delta_k x(t)^T (2K^T B^T P - Q) \Delta_k x(t).\end{aligned}$$

  We replace $V(x(t_k))$ by $W(t_k)$ in Equation (1.22), and we use Equation (1.29), along with the Lipschitz continuity of $x(t)$ with the Lipschitz constant given by Equation (1.8), to find an upper bound on $dV(x(t))/dt$,

$$\begin{aligned}\frac{dV(x(t))}{dt} \leq& \lambda_{\max} W_0 e^{-\alpha(t_k - t_0)} + M \lambda_{\max}(Q) L_x (t - t_k) \\ &+ L_x M \|Q - 2K^T B^T P\| (t - t_k) \\ &+ \|2K^T B^T P - Q\| L_x^2 (t - t_k)^2.\end{aligned} \tag{1.30}$$

  Equation (1.30) is of the form

$$\frac{dV(x(t))}{dt} \leq \lambda_{\max} W_0 e^{-\alpha(t_k - t_0)} + C_1 (t - t_k) + C_2 (t - t_k)^2.$$

  Re-writing the derivative of $W(t)$ as

$$\frac{dW(t)}{dt} = -\alpha W_0 e^{-\alpha(t_k - t_0)} e^{-\alpha(t - t_k)},$$

  it is sufficient to show that for $t \in (t_k, t_k + \tau)$,

$$\begin{aligned}\lambda_{\max} W_0 e^{-\alpha(t_k - t_0)} + C_1 (t - t_k) + C_2 (t - t_k)^2 \\ < -\alpha W_0 e^{-\alpha(t_k - t_0)} e^{-\alpha(t - t_k)}.\end{aligned} \tag{1.31}$$

Dividing both sides of Equation (1.31) by the quantity $-\alpha W_0 e^{-\alpha(t_k-t_0)}$ yields

$$\frac{|\lambda_{\max}|}{\alpha} > \frac{C_1(t-t_k)}{\alpha W_0 e^{-\alpha(t_k-t_0)}} + \frac{C_2(t-t_k)^2}{\alpha W_0 e^{-\alpha(t_k-t_0)}} + e^{-\alpha(t-t_k)} =: f_k(t).$$

This equation is satisfied at $t = t_k$, as $f_k(t_k) = 1$ and $|\lambda_{\max}|/\alpha > 1$, and the function $f_k(t)$ is Lipschitz continuous uniformly in k. Therefore, in a sufficiently small interval $[t_k, t_k + \tau)$, and since in the transient region, $t_k \leq T_{\lim}$, for all k, we can guarantee that

$$\frac{|\lambda_{\max}|}{\alpha} > f_k(t),$$

with τ as a uniform lower bound on $\tau_{\min}$ for all k.

2. *Steady-State Region*:
In practice, an event-triggered algorithm is implemented on a digital platform, where the event-triggering conditions are evaluated periodically according to a clock signal. A digital implementation imposes the period of the clock signal as a lower bound for the inter-event time, and the Zeno phenomenon can never occur in a physical system. However, in continuous-time, using the event-triggering condition given in Definition 2 can lead to Zeno behavior. This is clear when $x(t) = 0$, the event-triggering condition is activated. However, an update of the control law leads to $\frac{dV}{dt}|_{t_k^+} = 0$, which reactivates the event-triggering condition at t_k^+.

As $x(t)$ approaches zero, it becomes hard to give an estimate of τ. We first consider $\|x(t)\| \neq 0$, and re-write dV/dt as

$$\frac{dV(x(t))}{dt} = -x(t)^T(2K^TB^TP - Q)x(t) - 2x(t_k)^TK^TB^TPx(t).$$

At $t = t_k$, $dV/dt = 0$, but as $\|x(t)\| \neq 0$, we can set

$$\frac{dV(x(t))}{dt} = -x(t_k^+)^TQx(t_k^+) =: -\beta,$$

where $\beta > 0$. Let us suppose that $dV(x(t))/dt$ does not decrease further and $-\beta$ is the minimum value it reaches on the interval (t_k, t_{k+1}), as shown on Figure 1.3. If we consider that dV/dt increases linearly with the largest possible rate, we can estimate the time it takes to reach zero again. The rate of change of dV/dt is

$$\frac{d^2V(x(t))}{dt^2} = x(t)^T\Omega_1x(t) + x(t_k)^T\Omega_2x(t) + x(t_k)\Omega_3x(t_k).$$

where $\Omega_1 = A^T(2K^TB^TP-Q)+(2K^TB^TP-Q)A$, $\Omega_2 = -2K^TB^T(2K^TB^TP-Q-PA)$, $\Omega_3 = 2K^TB^TPBK$.

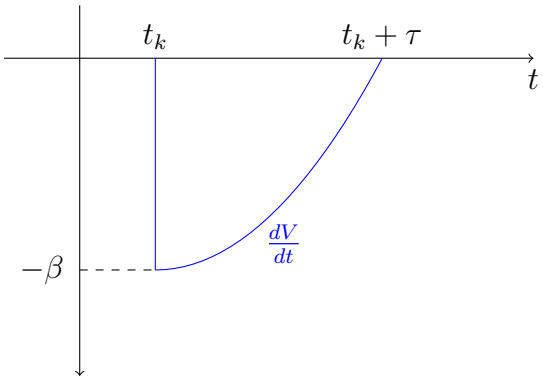


Figure 1.3: Derivative of the PLF in the interval $[t_k, t_{k+1})$.

Since in the steady-state region, $V(x(t))$ can no longer increase, we have

$$V(x(t)) < W(T_{\text{lim}}) < \delta.$$

Therefore,

$$\|x(t)\| \leq \sqrt{\frac{\delta}{\lambda_{\min}(P)}} =: \eta. \tag{1.32}$$

Likewise $\|x(t_k)\| \leq \eta$. The largest possible rate of growth of dV/dt is

$$\left|\frac{d^2V(x(t))}{dt^2}\right| \leq (\|\Omega_1\| + \|\Omega_2\| + \|\Omega_3\|)\eta^2 =: \psi.$$

Therefore, a lower bound on the time it takes for dV/dt to reach zero from $-\beta$ is given by

$$\tau \geq \frac{\beta}{\psi}.$$

However, as $\|x(t)\|$ tends to zero, this lower bound becomes undefined. As explained earlier, this is not a problem in reality, as the clock period is always a minimum inter-event time. Moreover, the ultimate objective $x(t) = 0$ is never reached, prevented by measurement noise, small perturbations and round-off errors.

□

1.4 Simulation Results

In this section we test the validity of the approach on numerical examples. The first example is a SISO system whereas the second example is a MIMO system.

Remark 1. *In continuous time, an event is detected when $V(x(t)) = W(t)$. However, as the simulation is run in discrete time, this equality is impossible to detect. Instead we detect the instants at which $V(x(t)) \geq W(t)$.*
In that case, there is no guarantee that an update of $u(t)$ will send $V(x(t))$ below $W(t)$. As a solution to this problem, we propose the following update at $t = t_k$

$$W(t_k) = V(x(t_k)).$$

Therefore, for $t \in [t_0, t_1)$, $W(t) = W_0 e^{-\alpha(t-t_0)}$; whereas for $k \geq 1$ and $t \geq t_k$, $W(t) = V(x(t_k))e^{-\alpha(t-t_k)}$. It is possible to do so, as $W(t)$ is a fictitious function and does not correspond to any actual data. Additionally, this update corresponds to continuous-time behavior.
This way, we also render the system robust to impulsive disturbances.

Remark 2. *At $t = 0$, we choose to set $W_0 > V(x_0)$, as a safety measure. This is not really necessary as we have proved that the case when $W_0 = V(x_0)$ is well handled by the algorithm.*

1.4.1 SISO System

1.4.1.1 Example Description

We consider the following second order LTI system, with one input

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \tag{1.33}$$

where $x(t) = \begin{bmatrix} x_1(t) & x_2(t) \end{bmatrix}^T$.
The initial state x_0 is given as

$$x_0 = \begin{bmatrix} 1 & -1 \end{bmatrix}.$$

The linear feedback control law is chosen (for a convergence time of 3 seconds) as

$$u(t) = \begin{bmatrix} -1.7329 & -5.6667 \end{bmatrix} x(t).$$

We associate to the system a quadratic Lyapunov-like function, as described by (1.9).

To find the matrices P and Q, and the decay rate α, we solve the maximum generalized eigenvalue problem (1.24) using the 'gevp' command of Matlab, which is based on the algorithms in [22]. This command relies on the formulation of problem (1.24) as linear matrix inequalities (LMI) using the LMI declaration tools of the Robust Control Toolbox.
We obtain as a result

$$P = \begin{bmatrix} 12.1917 & 4.3548 \\ 4.3548 & 3.2660 \end{bmatrix}, \; Q = \begin{bmatrix} 32.5123 & 11.6128 \\ 11.6128 & 8.7089 \end{bmatrix}.$$

The solver gives a value of $\lambda_{\max} = -2.66$, so we pick $\alpha = 2.3$ s^{-1}. We also set $\delta = 0.01$ close enough to the equilibrium point, and $W(0) = 9.72$ so that $W(0) > V(x_0)$.

1.4.1.2 Results

We have simulated the behavior of the system for 8 seconds (both transient and steady-state included), and we have chosen a sampling period of 10^{-4} s, resulting in 8×10^4 sampling times. The choice to sample at such high frequency is motivated by the fact that detecting the exact moment at which $V(x(t))$ intersects with $W(t)$ is impossible in a discrete-time simulation, and thus we try to emulate continuous-time behavior as closely as possible.

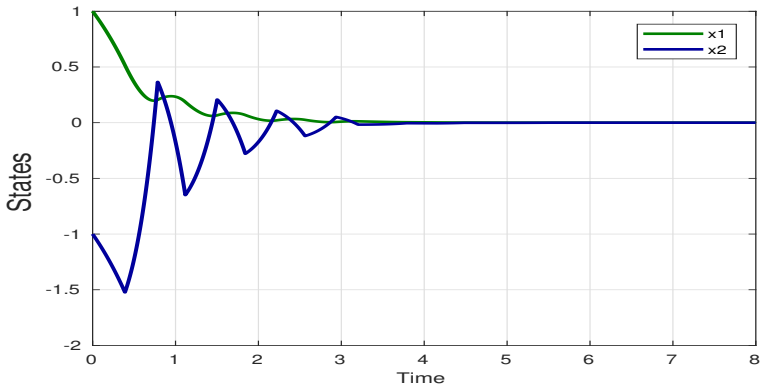


Figure 1.4: Time evolution of the states of the event-based system.

Figure 1.4 represents the time evolution of the states, $x_1(t)$ and $x_2(t)$. It shows that even though the response is not quite smooth in transient

time, the oscillations are not exaggerated and die out quickly. We define the convergence time t_c as the time at which the norm $\|x(t)\|$ settles below a certain value, i.e.

$$\|x(t)\| \leq \varepsilon_c, \quad \forall t \geq t_c.$$

If we take $\varepsilon_c = 0.02$, then the convergence time is $t_c = 3.29$ s.

Figure 1.5 represents the event-based control law. The control signal is updated only 17 times for $80,000$ sampling instants, i.e, we have reduced the number of samples by a factor of $1/5000$ compared to a periodic implementation. This result is illustrated better in Figure 1.6, where the intersections between the pseudo-Lyapunov function and the threshold function $W(t)$ represent the events.

In Figure 1.5, we notice that even if the updates of the control are distant in time, they occur, on average, at a regular pace. One possible reason for this could be the linearity and time-invariance of the system, which make for a predictable behavior.

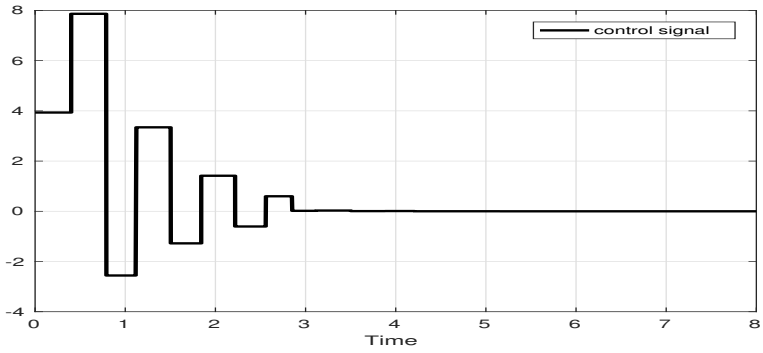


Figure 1.5: The piecewise constant event-based control law.

In control engineering, to convert a continuous control law to a digital signal, the recommended sampling frequency is contained in the range $[6\omega_b,\ 25\omega_b]$, where ω_b is the system's bandwidth. The bandwidth of System (1.33) being $\omega_b = 1.98$ rad/s, the recommended sampling period would be between 0.13 s and 0.53 s. The average sampling time obtained through the event-triggered control algorithm is 3.36 s, reducing the sampling frequency by nearly 6 to 25 times.

Figure 1.6 depicts the Lyapunov-like function and the decreasing threshold. As proven earlier, the PLF remains below the threshold. Every time

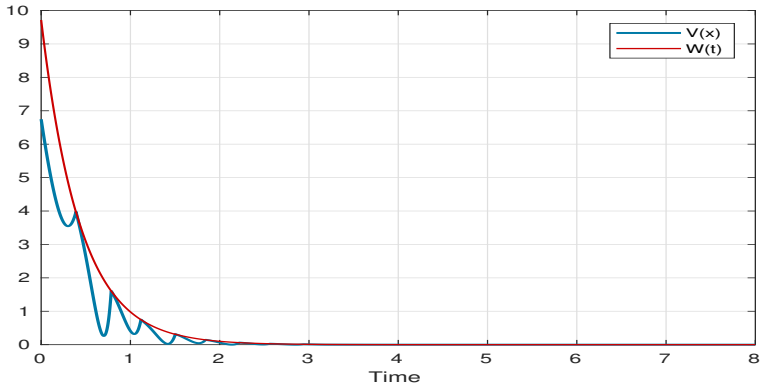


Figure 1.6: Time evolution of the Lyapunov-like function along with the exponentially decaying upper threshold function.

the two functions intersect, the PLF is pushed back below the threshold by a control update.

As the norm of the state vector drops to a very small value after 3.29 s, it would be nearly impossible to see the events on the curve of the derivative of the Lyapunov function. For this reason we propose a graph of the distribution of the events for the entire simulation interval, as given by Figure 1.7. It shows that in the steady state region, we do in fact obtain less updates than if we had kept the same triggering conditions for the entire simulation.

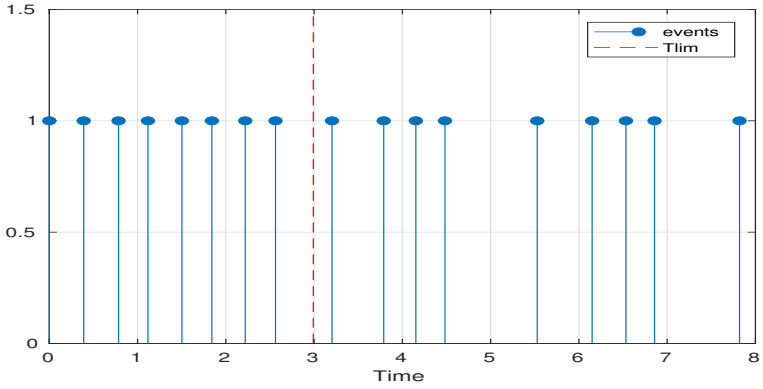


Figure 1.7: Time distribution of the events in transient and steady-state regions.

1.4.1.3 Varying the Settling Time

Since $V(x(t))$ and $W(t)$ are determined by solving an optimization problem, the only degree of freedom that we are left with is the choice of where to place the poles of the closed-loop system.

We usually place the eigenvalues in order to satisfy a set of closed-loop performance criteria, such as the speed and shape of the response [25]. The location of the poles particularly affects the speed with which the system reaches a steady-state, as poles closer to the imaginary axis yield a slow response, while poles further in the left half plane induce a faster response. The approximate time it takes to reach a steady-state is known as the settling time, and is defined as the time it takes for the system's step response to reach 2% (or sometimes 5%) of its final value.

The example that we have given above is a second order system for which the location of the poles can be easily determined by the settling time and the percent overshoot. The percent overshoot is defined as the percentage by which the peak value of the step response differs from its final value. Since we are not interested in this parameter, we keep it constant at 5%, and we vary the settling time. We should also mention that the system's control is event-triggered, and for this, the response may not exhibit the percent overshoot and settling time chosen for it. However, we use these quantities as a systematic way of changing the location of the poles. The value of $\lambda_{\max}$ changes for every location of the poles, and we take $\alpha = \rho_r|\lambda_{\max}|$, where $0 < \rho_r < 1$.

We start by increasing the simulation window to 20 s. Then, we vary the settling time T_{set} between $T_{\text{set}} = 2$ s and $T_{\text{set}} = 9.5$ s. We stop when $T_{\lim}$ exceeds 20 s. We plot the variations of the number of updates of the control law versus the settling time, shown in the graph of Figure 1.8.

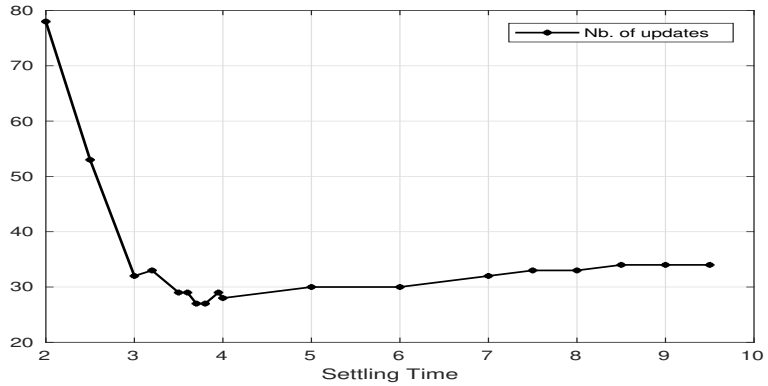


Figure 1.8: The variation of the number of events with respect to the settling time.

We can see that if we desire a small settling time, the number of events is going to be very high, 78 events for $T_{\text{set}} = 2$ s. As we increase T_{set}, the number of events drops rapidly to 29 at $T_{\text{set}} = 3.5$ s. At this stage, we notice that the number of updates oscillates around this value (between a minimum of 27 and 29). After $T_{\text{set}} = 4$ s, the number of events increases again, but at a slower rate this time. For example, it does not exceed 34 events at $T_{\text{set}} = 9.5$ s. This phenomenon is then nonlinear and deserves further investigation.

1.4.2 MIMO Systems

1.4.2.1 Example Description

An example of a MIMO system is the 4×4 linearized model of an aircraft under cruise control flight [26]. The system has two inputs and two outputs. The inputs consist in the rudder deflection and the aileron deflection. The outputs are the bank angle measured in radians and the yaw rate, which is an angular velocity measured in radians per second. Here are matrices A, B, and C which describe the system

$$A = \begin{bmatrix} -0.0558 & -0.9968 & 0.0802 & 0.0415 \\ 0.5980 & -1150 & -0.0318 & 0 \\ -3.0500 & 0.3880 & -0.4650 & 0 \\ 0 & 0.0805 & 1.0000 & 0 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.0729 & 0.0000 \\ -4.7500 & 0.00775 \\ 0.15300 & 0.1430 \\ 0 & 0 \end{bmatrix}, \; C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.34}$$

with

$$x_0 = \begin{bmatrix} -1 & 0.2 & 1 & 0.4 \end{bmatrix}^T.$$

We place the closed-loop eigenvalues at $-0.2896+0.6447i$, $-0.2896-0.6447i$, -0.3292, and -0.9814 by selecting the following state feedback gain

$$K = \begin{bmatrix} -0.0859 & -0.1088 & 0.0258 & -0.0466 \\ -20.7490 & -0.3931 & 5.1933 & 2.7084 \end{bmatrix}.$$

Solving the generalized eigenvalue problem problem yields $\lambda_{\max} = -0.5783$ and

$$P = \begin{bmatrix} 0.4202 & -0.3735 & 0.4389 & 0.1686 \\ -0.3735 & 1.0195 & -0.5305 & -0.2327 \\ 0.4389 & -0.5305 & 1.6434 & 0.7611 \\ 0.1686 & -0.2327 & 0.7611 & 0.6197 \end{bmatrix}.$$

Matrix P has a minimum and a maximum eigenvalue $\lambda_{\min}(P) = 0.1752$ and $\lambda_{\max}(P) = 2.4724$, respectively.
We set $\alpha = 0.3783$ s^{-1}, $W(0) = 1.9208$ and we simulate the behavior of the system for 30 seconds with a sampling period of $t_s = 10^{-4}$ s. We select $\delta = 0.002$, which yields $T_{\lim} = 18.1515$ s.

1.4.2.2 Simulation Results

Figure 1.9 shows the outputs of the system, the yaw rate designated by y_1, and the bank angle y_2. The quality of the response is good for this example. This can be explained by the fact that more events have been generated as we see from Figure 1.10, which shows the two control inputs of the system. We can see that the control law has been updated more frequently compared to the previous example, as it required 49 updates for the entire simulation window. The quality of the response can also be explained by the fact that the open-loop system is marginally stable and not completely unstable as the

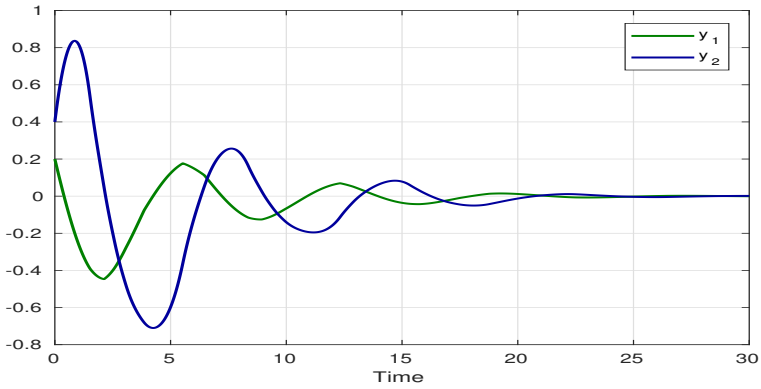


Figure 1.9: Time evolution of the response, the yaw rate y_1 and the bank angle y_2.

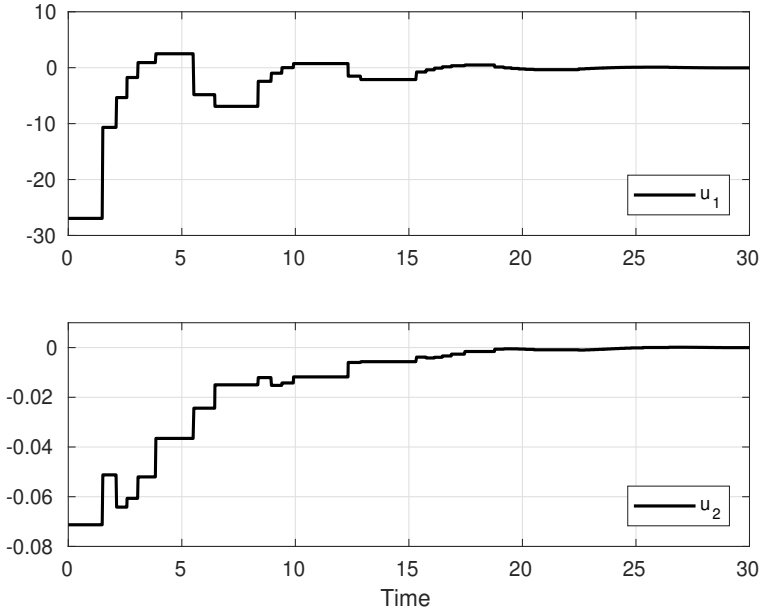


Figure 1.10: The two control inputs, rudder (u_1) and aileron (u_2) deflections.

SISO system treated before.

Figure 1.11 depicts the evolution of the pseudo-Lyapunov function. For this system, we notice that the events occur in clusters, as there are intervals where the updates are frequent separated by long time intervals where no update is carried out. This phenomenon is seen more clearly in the distribution of events in Figure 1.12. This can also be explained by the marginal stability of the system, as it can operate autonomously for longer periods of

time. Then, frequent updates are taken to steer the system back to the right trajectory.

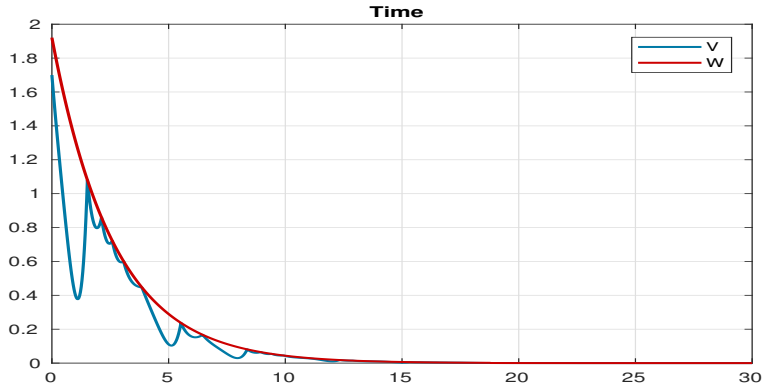


Figure 1.11: Time evolution of the pseudo-Lyapunov function and the exponential threshold.

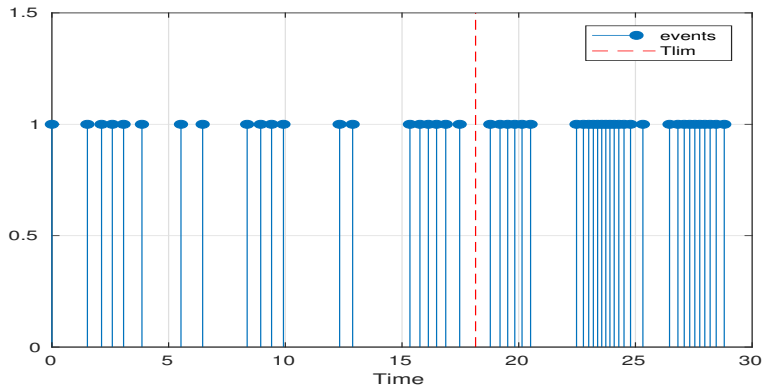


Figure 1.12: Distribution of the events in transient and steady-state regions.

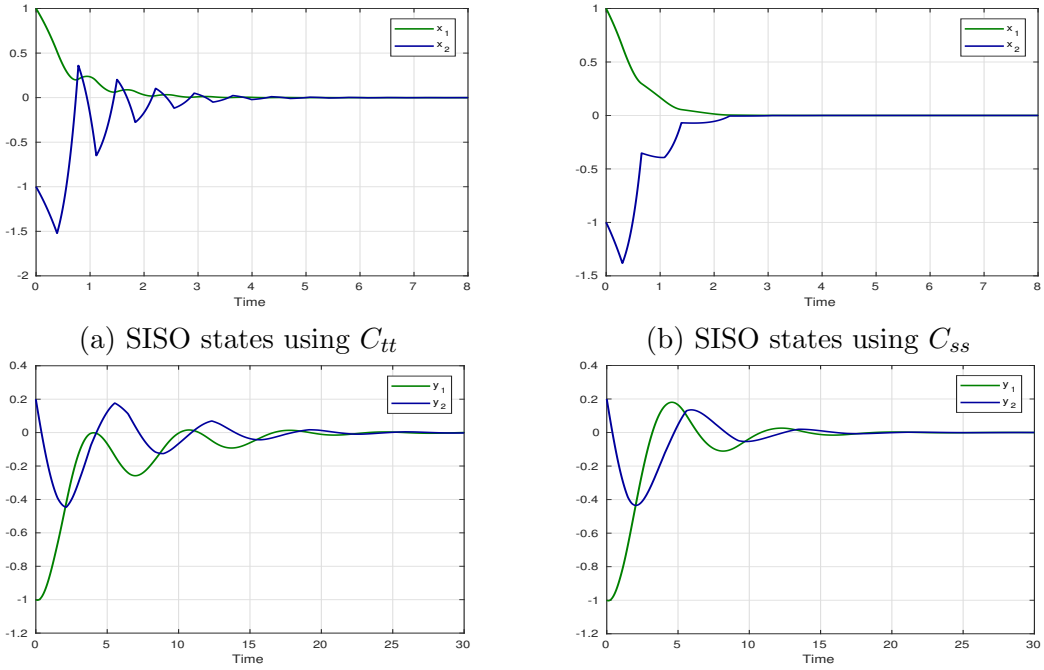
From the distribution of the events in Figure 1.12, we notice that the events are sparser in transient-time and denser in steady-state. Conversely, in the case of the SISO system, the events are sparser in steady-state. This may be due to the fact that for the MIMO system, more control effort is required to maintain the system at the equilibrium point.

1.4.3 Comments on Using Dual Conditions

We explain our choice to use dual event-triggering conditions by the fact that it yields a faster convergence in steady-state. In this section, we want to

examine the behavior of SISO System (1.33) and MIMO System (1.34) when we use only one condition, either C_{tt} or C_{ss}, for the entire operation time. For simplicity, we denote C_{tt} the transient-time condition of Definition 1, and C_{ss} the steady-state condition of Definition 2.

We first simulate the SISO system with a control law updated using condition C_{tt} alone, then condition C_{tt} alone. We repeat this procedure for the MIMO system. Then, we compare the consequences of using C_{tt} or C_{ss} in terms the convergence time and the number of updates for each system.



(a) SISO states using C_{tt}

(b) SISO states using C_{ss}

(c) MIMO states using C_{tt}

(d) MIMO states using C_{ss}

Figure 1.13: Comparing the condition C_{tt} with C_{ss} for the SISO and MIMO systems.

Figure 1.13a shows the time evolution of the states of SISO System (1.33) with event-triggering condition C_{tt} only. It takes the SISO system 4.03 s to converge with this condition. We recall that using the dual approach leads to a convergence time of 3.29 s. Figure 1.13b shows the states of the SISO system under condition C_{ss}, the convergence time is 2.23 s. The shape of the response is not much improved, even though the oscillations are less pronounced than when using C_{tt} alone. The number of updates drops to 14 with C_{ss} and rises to 23 with C_{tt}.

Figure 1.13c shows the response of MIMO System (1.34) under condition C_{tt} alone. The convergence time is 22.77 s which is slightly worse than what the dual condition yielded, namely 20.98 s. Figure 1.13d shows the response of condition C_{ss} alone which yields a convergence time of 15.54 s. However, the benefit of the dual condition is apparent when comparing the number of updates of the control law. The C_{tt} condition alone yields 32 updates, and the C_{ss} condition alone yields 80 updates, whereas the dual condition leads to 49 updates.

In both cases, the results of this experiment comfort the idea that using the dual condition speeds up the convergence. It also confirms that a decrease in the number of updates is not guaranteed by the dual condition. When treating each system individually, it is clear that condition C_{ss} alone works better for the SISO system in terms of number of updates and convergence time. The response of the MIMO system, on the other hand, is always deteriorated in some aspect when either condition is used alone. In this case, the dual condition offers a good trade-off, a faster convergence and less updates.

1.5 Comparison with Other Methods

In this section we compare the performances of our method with other event-based control methods. We choose three methods from the literature and we compare them in terms of the number of updates of the control law and in terms of the quality of the performance. We test all three methods on the SISO system (1.33).

The first method that we explore was developed by Tabuada in [6]. Since this method has been primarily developed for nonlinear systems and is based on the existence of an Input-to-State Stable (ISS) Lyapunov function, we refer to it as the ISS method.

The second method introduced by Marchand, Durand and Guerrero Castellanos [27] relies on an extension of Sontag's stabilization formula for nonlinear systems and requires the existence of a Control Lyapunov Function (CLF). For this reason, we refer to it as the CLF method.

Finally, the method introduced by Meslem and Prieur [19] is based on reachability analysis and therefore is referred to as the rechability method.

Whenever necessary, we introduce new notations that are specific to each method, but otherwise keep the notations used until now (P, K, Q, etc).

1.5.1 The ISS Method

The ISS method relies on the existence of an ISS Lyapunov function. However, for linear systems, this property can be satisfied as long as the plant is controllable.

For the linear case, the event-triggered control algorithm updates the control law when

$$\|x(t) - x(t_k)\| \leq \sigma \|x(t)\|, \ \forall t \in [t_k, t_{k+1}), \tag{1.35}$$

where $0 < \sigma < 1$. If this condition is satisfied for all t, $dV(x(t))/dt < 0$, thus guaranteeing stability. The control is updated when $\|x(t) - x(t_k)\| = \sigma \|x(t)\|$.

In order to perform the comparison, we have reproduced the simulation conditions of section 1.4.1.

The parameter σ is chosen such that

$$\lambda_{\min}(Q) > \sigma \|K^T B^T P + PBK\|. \tag{1.36}$$

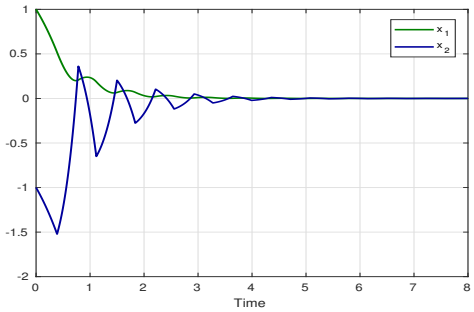
Condition (1.36) yields $\sigma = 0.05$.

Figure 1.14b represents the evolution of the states of system (1.33). We can notice that the response is of good quality and does not exhibit any oscillations, unlike the response produced by our method. However, such good quality comes at a price, as this implementation requires 357 updates of the control law for $80,000$ sampling instants. This is a relatively high number compared to our 17 updates for the same simulation duration.

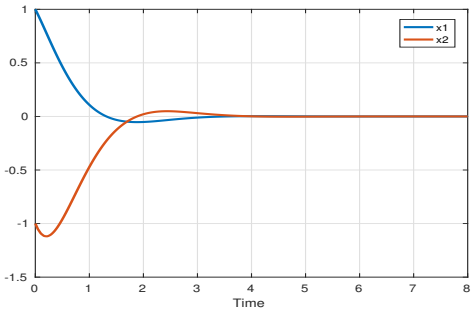
If we use the control law developed by the author of this method in [6]

$$u(t) = \begin{bmatrix} -1 & 4 \end{bmatrix} x(t),$$

the number of updates is reduced to 177, but is still very large compared to our results.



(a) SISO states using our method



(b) Evolution of the states with the ISS method.

Figure 1.14: Comparing the response of the SISO system using our method and the ISS method

1.5.2 The CLF method

The CLF method proves that it is possible to extend Sontag's stabilizing formula to event-based systems provided that a CLF exists. Again the existence of a CLF is guaranteed in the linear case by the controllability of the plant. The event-based algorithm detects the time instants when the Lyapunov function is not decreasing enough. Some smoothness considerations are also added to the triggering conditions.

In the linear case, this method translates into an optimal control problem, for which we need to introduce the following weighting matrices on the state and the control respectively

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \qquad R = \frac{1}{12}.$$

To find P, a Riccati equation of the form $PA + A^T P - R^{-1}PBB^T P = -Q$ is solved. We obtain

$$P = \begin{bmatrix} 3.2743 & 0.2743 \\ 0.2743 & 0.7743 \end{bmatrix}.$$

Figure 1.15 represents the evolution of the states of plant (1.33). It shows that the quality of the response is relatively good. For a simulation interval of 8 seconds, it requires 23 updates of the control law.

Even though our approach has required less samples, for a simulation of 8 seconds, the difference is not significant. The advantage of our method

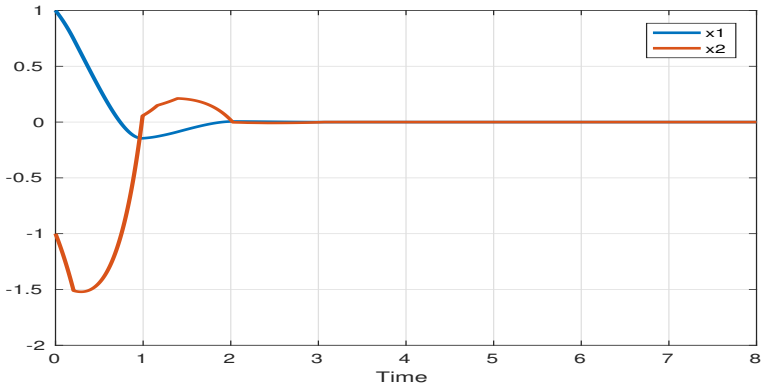


Figure 1.15: Time evolution of the states with the CLF method.

appears in the long run. As we run the simulation for 30 seconds, our method requires 46 samples, while the CLF method needs about the double (86 samples).

1.5.3 The Reachability Method

In the reachability method, a Lyapunov-like function, similar to the one we describe, is associated to the plant. This function is then forced to remain framed between the decaying Lyapunov functions of two auxiliary systems, a slow and a fast system. An event is generated when the pseudo-Lyapunov function intersects with either the upper or lower threshold.

Since the faster system does not play a part in the stability of the event-based implementation, we have decided to omit it from our analysis. The slow system is given by

$$\dot{x}_s(t) = \beta_s(A - BK)x_s(t), \quad \forall t$$

where $x_s(t) \in \mathbb{R}^n$ is the state of the slow system, and $0 < \beta_s < 1$.
By sweeping the interval of definition of β_s, we notice that the method yields better results as β_s gets closer to 1. For this reason, we select $\beta_s = 0.95$.

The response is shown in Fig. 1.16. As expected, the quality of the response is similar to ours, as the two methods work on the same principle. The reachability method needed 23 samples for an 8 second simulation.
The reachability method also requires the definition of extra systems, thus increasing the complexity of the implementation as the order of the plant increases. In contrast, our method requires the introduction of a simpler scalar function, regardless of the order of the system.

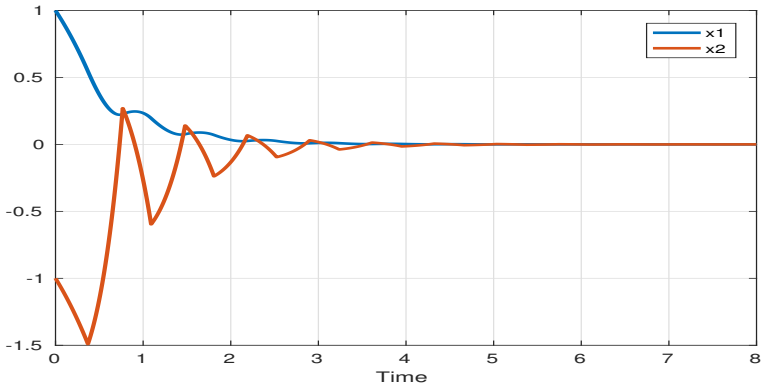


Figure 1.16: Time evolution of the states with the reachability method.

1.6 Conclusion

We have presented a two-part event-triggered control strategy for Linear Time-Invariant systems.

In the first part, when the system is in a transient region, the event generator monitors a pseudo-Lyapunov function associated to the system. The event generators sends orders to update the control law when the pseudo-Lyapunov function hits an upper threshold. We have shown that by solving a maximum generalized eigenvalue problem, we can obtain the PLF and the threshold at the same time. Such threshold is optimal, as it allows a fast convergence of the system to equilibrium while maintaining an acceptable quality for the shape of the response and requiring very few updates of the control law.

In the steady-state region, the time derivative of the Lyapunov function takes over. At this stage, an event is generated when the Lyapunov function no longer decreases in time, or equivalently when its time derivative along the trajectories of the system vanishes.

We have shown the advantages and shortcomings of the method, and we have offered a discussion of the choice of parameters for a faster and smoother response with fewer updates. We have also provided numerical implementations and comparisons with other methods.

Chapter 2

Event-Triggered Stabilizing Controllers of Switched Linear Systems

Contents

2.1 Introduction

A switched system is a dynamical system composed of a finite set of subsystems with continuous dynamics, and a switching rule that determines which

subsystem is active at a given time. The coexistence of continuous and discrete-time dynamics makes switched systems a subclass of hybrid systems. The difference between hybrid and switched systems though, is that the analysis of switched systems focuses on the continuous-time behavior, with discrete transitions between subsystems treated as isolated events [28].

Many physical systems can be modeled as switched systems. A non-exhaustive list of switched systems includes electronic circuits containing a switching device [29], [30], systems driven by several controllers, or systems with dynamics changing due to a damaged component [31].

Despite the modeling advantages that they offer, the stability analysis of switched systems is a challenging task. The reason is that the stability of each individual subsystem does not imply the stability of the entire system, under arbitrary switching. It has been proved in [32] and [33] that the existence of a quadratic Lyapunov function common to all subsystems (CQLF) guarantees the asymptotic stability of the switched linear system under an arbitrary switching rule. Even if this property is conservative, for a switched system can be stable when no common Lyapunov function exists, the existence of a CQLF can be directly verified by solving a set of Linear Matrix Inequalities (LMI) [34].

In this part, we adapt to switched systems the event-triggered stabilizing control algorithm introduced in Chapter 1. The application of event-based control to the case of switched systems knows a growing interest among the control community. In [35], the authors synthesize an event-triggered dynamic controller for switched systems with time delays, based on the periodic event-triggered approach described in [8] and the dynamic event-triggering mechanism of [36]. In [37], the event-triggered control algorithm presented in [6] is applied to switched linear systems with model uncertainties, and multiple Lyapunov functions are used to prove stability. In [38], the authors propose an output-based event-triggered approach to the control of continuous-time switched systems. The event-triggering conditions rely on the squared error between the current and the most-recently sampled state.

In order to extend the method of Chapter 1 to switched systems, we choose the CQLF to be the pseudo-Lyapunov function (PLF). Since the CQLF is common to all the subsystems, we can re-write the generalized eigenvalue problem of Chapter 1 as a unified optimization problem, that we solve for the CQLF and the optimal upper threshold. However, in this part, we no longer split the system's operation time into transient and steady-state

regimes, and keep instead the same set of event-triggering conditions for the entire operation. Therefore, the control is updated when the PLF reaches the upper threshold, or when a switch in the active subsystem occurs.

This chapter is organized as follows. In Section 2.2, we provide a brief reminder on switched linear systems, and discuss their stability in Section 2.3. Then, we formally define the problem that we solve in Section 2.4. In Section 2.5, we give a detailed description of the event-triggered control algorithm, along with a proof that it asymptotically stabilizes a switched system under an arbitrary switching sequence. The second part of Section 2.5 is dedicated to proving that a minimum time separates any two events. The final section provides a numerical example to show the performance of the algorithm, how it stabilizes a test system through a small number of control updates and with no Zeno behavior. We give an example with time switching and another with a state-dependent switching rule.

2.2 Overview of Switched Linear Systems

Consider the switched linear system modeled as

$$\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}u(t), \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state vector, and $u \in \mathbb{R}^m$ is the control signal. System (2.1) is also defined by the mapping $\sigma : [t_0, \infty) \to \mathcal{I}$, that we call the switching rule. The set $\mathcal{I} = \{1, 2, ..., I\}$ is a finite set, called the index set, such that every subsystem of the form

$$\dot{x}(t) = A_i x(t) + B_i u(t) \qquad \forall i \in \mathcal{I}, \tag{2.2}$$

is a realization of the switched system (2.1). The matrices $A_i \in \mathbb{R}^{n\times n}$ and $B_i \in \mathbb{R}^{n\times m}$ are constant. Let x_0 be the initial state $x(t_0)$.

In what follows, we consider the switching rule to be arbitrary and either time-dependent or state-dependent, or both. Therefore, the notation $\sigma(t)$ does not imply that σ is only time dependent but refers to the realization of the rule σ at time t. The switching rule σ is a piecewise continuous signal, with discontinuities that signal a change in the active subsystem. We refer to these discontinuities as jumps of the switching rule.

We also suppose that the jumps of the switching rule are separated by a minimum time $\tau_d > 0$. The time duration τ_d is called a minimum dwell time,

and its existence means that each subsystem remains active for at least a period of time τ_d. The dwell time is essential in order to avoid Zeno behavior.

In the case of state-dependent switching, imposing a dwell time can be hard, as the switching generally occurs when the state crosses a set of surfaces. Instead of crossing, the state may slide along the surface, creating instantaneous switches. Fortunately, there exist several solutions to this problem. For example, in [39], a minimum dwell time is imposed by coupling state-switching and time-switching. Also, in Chapters 1 and 6 of [28], a dwell time is obtained through hysteresis switching, in which switching occurs when the state crosses a strip instead of a surface.

The pairs (A_i, B_i) are taken to be controllable for all $i \in \mathcal{I}$. Thus, there exist feedback gains $K_i \in \mathbb{R}^{m\times n}$, such that each individual matrix $A_i - B_i K_i$, for all $i \in \mathcal{I}$ can be rendered Hurwitz, and each individual subsystem is stable. However, the stability of each individual subsystem does not guarantee the stability of the switched system, under an arbitrary switching rule. The stability of switched linear systems is discussed in more details in Section 2.3.

2.3 Stability of Switched Linear Systems

A sufficient condition for the stability of a switched linear system under arbitrary switching is the existence of a CQLF. The function $V : \mathbb{R}^n \to \mathbb{R}^+$ of the form

$$V(x(t)) = x(t)^T P x(t), \tag{2.3}$$

is a CQLF for the system (2.1) if and only if there exists a single positive-definite matrix P such that

$$(A_i - B_i K_i)^T P + P(A_i - B_i K_i) = -Q_i, \quad \forall i \in \mathcal{I}, \tag{2.4}$$

where Q_i are symmetric positive definite matrices.

Even when each individual matrix $A_i - B_i K_i$ is Hurwitz for all $i \in \mathcal{I}$, a CQLF is not guaranteed to exist. In the literature, conditions for the existence of a CQLF were established for some particular classes of systems, such as the case when $A_i - B_i K_i$ are triangular matrices or when they commute. However, if the subsystems have a general structure, there is no algebraic condition on $A_i - B_i K_i$ to establish beforehand whether a CQLF exists [34].

There also exist techniques to determine the location of the closed-loop poles, if any, for which a CQLF exists. In [40], the authors propose to solve a set of linear matrix inequalities to simultaneously design a stabilizing state feedback control law and deduce a CQLF. In its simplest form, this approach states that if we can find a positive definite matrix $M \in \mathbb{R}^{n\times n}$ and matrices $Z_i \in \mathbb{R}^{m\times n}$ such that for all $i \in \mathcal{I}$

$$A_i M + M A_i^T + B_i Z_i + Z_i^T B_i^T < 0, \tag{2.5}$$

then the feedback control laws with $K_i = Z_i M^{-1}$ render the switched system stable, and additionally ensure the existence of a CQLF with $P = M^{-1}$. The existence of such methods shows that requiring a CQLF is not too restrictive.

2.4 Problem Definition

Our objective is to stabilize the switched system (2.1) via an event-triggered control law. In the case of switched systems, there are two types of events at which the control law is updated

- the state of the system no longer satisfies some predefined performance criteria,
- there is a jump in the switching rule, and a new subsystem becomes active.

We denote by t_k, $k \in \mathbb{N}$, the time instants at which the events occur. The control law $u(t)$ is a state-feedback control and is scheduled as follows

$$u(t) = -K_i x(t_k), \quad t \in [t_k, t_{k+1}), \tag{2.6}$$

where the ith subsystem ($i \in \mathcal{I}$) is active at time t_k. Then in the interval $[t_k, t_{k+1})$ and for all k, System (2.1) admits the solution

$$x(t) = e^{(A_i - B_i K_i)(t-t_k)} x(t_k) - \int_{t_k}^{t} e^{(A_i - B_i K_i)(t-s)} B_i K_i \Delta_k x(s) ds. \tag{2.7}$$

where $\Delta_k x(t) = x(t_k) - x(t)$. In the framework of switched systems with a finite number of subsystems, despite the jumps that the system undergoes, $x(t)$ is still Lipschitz continuous, with Lipschitz constant L_x.

Problem statement. Consider the switched linear system (2.1), composed of I subsystems that switch according to an arbitrary switching rule σ. Each

subsystem is stabilizable through a state-feedback control law. The control law is updated when an event is triggered, and otherwise kept constant. Therefore, we want to design an event-triggering condition, or a set of event triggering conditions that

- detects the system's failure to satisfy the performance criterion that we introduce in Section 2.5,
- detects a change in the active subsystem, so that the right feedback K_i is applied when subsystem $i = \sigma(t)$ is active, maintaining the Hurwitz property of $A_i - B_i K_i$.

Under an arbitrary switching rule, the control law is updated such that

- the switched linear system (2.1) is asymptotically stable,
- for any two consecutive events at t_k and t_{k+1}, there exists a minimum duration $\tau > 0$, such that $t_k - t_{k+1} \geq \tau$, for all $k \in \mathbb{N}$.

In the next section, we show how we develop the event-triggering conditions around the CQLF and the switching rule.

2.5 Event-triggered Control Algorithm

2.5.1 Algorithm Description

As explained in the previous chapter, we assess the performance of the system through the behavior of its pseudo-Lyapunov function. The case of switched linear systems does not differ considerably, as the CQLF is taken as PLF. The difference lies in the necessity to also monitor jumps in the switching rule in the case of switched linear systems. Moreover, for switched systems, we do not split the operation into steady-state and transient time. Instead, for the entire operation time, the event-generator monitors intersections between the PLF and the threshold, as well as the jumps in the switching rule.

The behavior of the PLF can thus be described as follows. The PLF $V(x)$ decreases for some time after the update of the control law. Then, if no jump occurs and when the control ceases to be effective, $V(x)$ increases until it reaches an upper threshold. At that moment the control law is updated, and the $V(x)$ decreases again. If, on the other hand, a new subsystem becomes active, the control has to be updated, and whether $V(x)$ is in a decreasing or in an increasing phase prior to the update, it decreases again

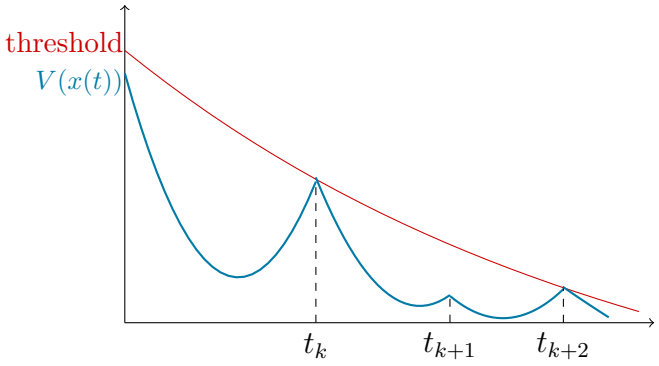


Figure 2.1: The Lyapunov-like function $V(x(t))$ in blue and the upper threshold in red.

following the update.

Fig. 2.1 illustrates the behavior of the PLF. It shows that at times t_k and t_{k+2} an event is generated as the PLF hits the upper bound, and is pushed back below by control update, whereas the event at $t = t_{k+1}$ corresponds to a jump in the switching rule.

In Chapter 1, we have shown that in the case of linear time-invariant systems, if we take a positive, exponentially decreasing function as the upper threshold for the PLF, we can guarantee that the PLF decreases globally, despite being locally increasing. In the case of switched systems, if a CQLF exists, then a similar exponentially decreasing threshold can be found. In a similar way, we look for a scalar $\lambda < 0$ such that the following constraint is satisfied

$$\frac{dV(x(t))}{dt}|_{t=t_k^+} \leq \lambda V(x(t_k)), \quad \forall k. \tag{2.8}$$

Yet, at $t = t_k^+$, after control update,

$$\frac{dV(x(t))}{dt}|_{t_k^+} = -x^T(t_k)Q_i x(t_k), \tag{2.9}$$

therefore, from equations (2.8) and (2.9), we want

$$-x^T(t_k)Q_i x(t_k) \leq \lambda x^T(t_k)Px(t_k) \quad \forall k, \; i = \sigma(t_k). \tag{2.10}$$

To ensure the fastest possible decay rate in the case of switched systems, λ has to be the maximum generalized eigenvalue of all the pairs $(-Q_i, P)$,

which is defined as [21]

$$\lambda_{\max} \equiv \inf\{\lambda \in \mathbb{R} \mid -Q_i < \lambda P, \forall i = 1, ..., \mathcal{I}\}. \tag{2.11}$$

This value can therefore be found by solving the following optimization problem

$$\begin{aligned} &\text{minimize } \lambda \\ &\text{subject to the LMI constraints} \\ &\quad (A_i - B_i K_i)^T P + P(A_i - B_i K_i) \leq \lambda P, \quad \forall i \in \mathcal{I} \\ &\quad P > 0, \ \lambda < 0. \end{aligned} \tag{2.12}$$

We solve the above optimization problem for the maximum generalized eigenvalue $\lambda_{\max}$. Then, we can define the upper threshold function $W(t)$,

$$W(t) = W_0 e^{-\alpha(t-t_0)}, \tag{2.13}$$

where $W_0 \geq V(x_0)$ and $0 < \alpha \leq |\lambda_{\max}|$.

We can then define the execution times of the control law.

Definition 3. *We define the time instants t_k, $k \in \mathbb{N}$, at which the control signal $u(t)$ is updated, as*

$$t_k = \inf\{t > t_{k-1} \mid V(x(t)) \geq W(t) \text{ or } \sigma(t) \neq \sigma(t_{k-1})\}, \tag{2.14}$$

where $V(x(t))$ and $W(t)$ are given by Equations (2.3) *and* (2.13)*, respectively.*

2.5.2 Stability Results

In this section, we discuss the stability of a switched linear system under the event-triggered control strategy described above.

Theorem 4. *The control law, defined by Equation* (2.6) *and scheduled by the event-triggering condition given by Definition 3, renders System* (2.1) *asymptotically stable under arbitrary switching.*

Proof. We show that the evolution of the PLF resulting from the control algorithm described above remains upper bounded by $W(t)$, for all t. And since $W(t)$ tends to zero as time tends toward infinity, so does $V(x(t))$, which in turn means that $x(t)$ converges to the zero equilibrium.

For $t = t_0$, $V(x_0) \leq W(t_0)$, by definition. Then, when $t > t_0$, we identify three cases:

1. Case $\sigma(t_k^-) = \sigma(t_k^+)$ (no jump) and $V(x(t_k)) = W(t_k)$
   When $t_{k-1} < t < t_k$, $V(x(t)) < W(t)$, by Definition 3.
   When $t = t_k$, $V(x(t_k)) = W(t_k)$, and after updating the control,

$$\frac{dV(x)}{dt}|_{t=t_k^+} = -x^T(t_k)Q_i x(t_k) \leq \lambda_{\max} V(x(t_k)). \tag{2.15}$$

   Since we select $\alpha < |\lambda_{\max}|$, then at time t_k, $\lambda_{\max} V(x(t_k)) < -\alpha W(t_k)$, and equation (2.15) becomes

$$\frac{dV(x)}{dt}|_{t=t_k^+} \leq -\alpha W(x(t_k)) = \frac{dW(t)}{dt}|_{t=t_k} < 0. \tag{2.16}$$

   This proves that at $t = t_k^+$, $V(x(t))$ decreases faster than $W(t)$, and therefore remains below $W(t)$ for a certain time.

2. Case $\sigma(t_k^-) \neq \sigma(t_k^+)$ (jump) and $V(x(t_k)) < W(t_k)$
   Since $V(x)$ is a continuous function, an update of the control law at $t = t_k$ does not change the fact that $V(x(t)) < W(t)$.

3. Case $\sigma(t_k^-) \neq \sigma(t_k^+)$ and $V(x(t_k)) = W(t_k)$
   Since $V(x)$ is a common Lyapunov function for all the subsystems, this case is not different from Case 1 when $V(x(t_k)) = W(t_k)$ with no jump of the switching rule.

Therefore, in all three cases, we guarantee that $V(x(t)) \leq W(t)$, for all t. □

2.5.3 Minimum Inter-Event Time

The event-triggered control algorithm also needs to guarantee a minimum time lapse between any two successive events. If no such time exists, we could end up with an infinite number of updates in a finite interval of time, a situation known as the Zeno phenomenon.

Theorem 5. *Let $T > t_0$ arbitrarily large, t_k and t_{k+1} two consecutive time instants in $[t_0, T]$ given by Definition 3. Then there exists a minimum time $\tau > 0$ such that $t_{k+1} - t_k \geq \tau$, on the finite interval $[t_0, T]$.*

Remark 3. *The parameter T, that can be chosen arbitrarily large, allows us to prove the existence of an inter-event time as it offers many advantages, which are used in the following proof*

- *We avoid the risk of obtaining events due to the switching rule that are arbitrarily close to the events due to the PLF for large times. As a result, there always exists a minimum time τ, either between two intersections or between a jump and an intersection.*
- *It allows us to determine a lower bound on the exponential threshold, that we can use in the proof.*

Proof. The proof of Theorem 5 depends on the nature of each event in every pair of consecutive events. We identify the following cases

- t_k and t_{k+1} are due to an intersection between the PLF and the threshold: this possibility is covered in Case 1, below.
- t_k is due to a jump in the switching rule, while t_{k+1} is due to an intersection: this is covered in Case 2.
- t_k and t_{k+1} are both due to a jump: in this case the minimum inter-event time is τ_d, the dwell time.
- t_k is the result of an intersection and t_{k+1} is due to a jump: since there is a finite number of jumps, this can occur only a finite number of times. Therefore, the minimum of these finite delays is nonzero, and there exists a minimum inter-event time. However, we cannot give an estimation of this inter-event time, unlike in the three other cases.

1. Case $V(x(t_k)) = W(t_k)$
   To prove the existence of τ, we need to show that $V(x(t))$ decreases faster than $W(t)$ for some time after an update of the control law, such that no other intersection is possible. For this, we follow the same procedure as in Chapter 1. When $t \in [t_k, t_{k+1})$,

$$\frac{dV(x(t))}{dt} = -x(t)^T Q_i x(t) - 2\Delta_k x(t)^T K_i^T B_i^T P x(t). \tag{2.17}$$

   We can re-write $dV(x(t))/dt$ in terms of $x(t_k)$ and $\Delta_k x(t)$ as

$$\begin{aligned}\frac{dV(x(t))}{dt} =& - x(t_k)^T Q_i x(t_k) + x(t_k)^T Q_i \Delta_k x(t) \\ &+ \Delta_k x(t)^T (Q_i - 2K_i^T B_i^T P) x(t_k) \\ &+ \Delta_k x(t)^T (2K_i^T B_i^T P - Q_i) \Delta_k x(t).\end{aligned}$$

   We recall that at $t = t_k$

$$\|x(t_k)\| \leq M,$$

where $M = \sqrt{W_0/\lambda_{\min}(P)}$.
We use this upper bound on $\|x(t_k)\|$, Equations (2.15), and the Lipschitz continuity of $x(t)$, to find an upper bound on $dV(x(t))/dt$

$$\begin{aligned}\frac{dV(x(t))}{dt} \leq \lambda_{\max} W_0 e^{-\alpha(t_k - t_0)} + M\lambda_{\max}(Q_i)L_x(t - t_k) \\ + L_x M\|Q_i - 2K_i^T B_i^T P\|(t - t_k) \\ + \|2K_i^T B_i^T P - Q_i\|L_x^2(t - t_k)^2.\end{aligned} \tag{2.18}$$

Equation (2.18) is of the form

$$\frac{dV(x(t))}{dt} \leq \lambda_{\max} W_0 e^{-\alpha(t_k - t_0)} + C_1(t - t_k) + C_2(t - t_k)^2,$$

Similarly to the proof in Chapter 1, we arrive at the inequality

$$\frac{|\lambda_{\max}|}{\alpha} > \frac{C_1(t - t_k)}{\alpha W_0 e^{-\alpha(t_k - t_0)}} + \frac{C_2(t - t_k)^2}{\alpha W_0 e^{-\alpha(t_k - t_0)}} + e^{-\alpha(t - t_k)} =: f_k(t),$$

This inequality is satisfied at $t = t_k$, as $f_k(t_k) = 1$ and the quantity $|\lambda_{\max}|/\alpha > 1$. From the expression of $f_k(t_k)$, this function is uniformly Lipschitz continuous, and there exists a constant $L_f > 0$, independent of k such that

$$|f_k(t) - f_k(t_k)| \leq L_f|t - t_k|. \tag{2.19}$$

This implies that in a sufficiently small interval $[t_k, t_k + \tau)$, we can guarantee that

$$\frac{|\lambda_{\max}|}{\alpha} > f_k(t).$$

Therefore, τ is a uniform lower bound on the minimum inter-event time for all k.

2. Case $\sigma(t_k^-) \neq \sigma(t_k)$ and $V(x(t_k)) < W(t_k)$
For this type of events, there is nothing we can say about the rate of decay of $V(x(t))$ with respect to the decay rate of $W(t)$. However, depending on how far $V(x(t_k))$ is from $W(t_k)$, we can show that some time has to pass before their next intersection. We can analyze this time lapse based on the difference between $V(x(t_k))$ and the quantity $W(T)/2$, which falls into two categories. The PLF $V(x(t_k))$ is either above or below $W(T)/2$, as shown on Figure 2.2.

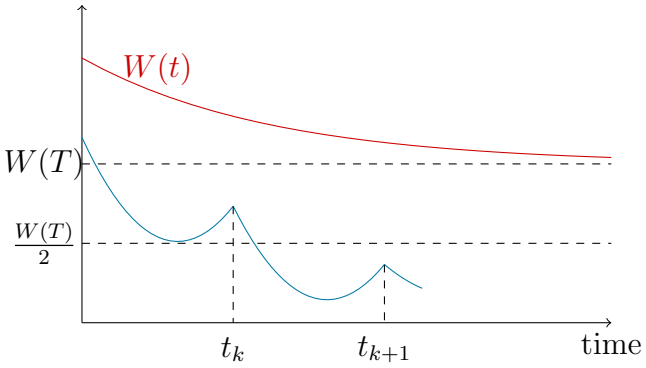


Figure 2.2: The two positions of the PLF during a jump due to switching.

- Case $V(x(t_k)) \geq W(T)/2$.
  Since $V(x(t_k)) < W(t_k)$, we show that the $V(x(t))$ changes slowly enough, so that no intersection between the PLF and the threshold is possible until some time passes. For this we show that the derivative of $V(x(t))$ remains bounded from above for some amount of time.

  In the case when $V(x(t_k)) \geq W(T)/2$, we have

  $$\frac{W(T)}{2} \leq V(x(t_k)) \leq \lambda_{\max}(P)\|x(t_k)\|^2,$$

  which yields

  $$\|x(t_k)\| \geq \sqrt{\frac{W(T)}{2\lambda_{\max}(P)}}.$$

  We denote $\mu = \sqrt{W(T)/2\lambda_{\max}(P)}$.
  When $t_k < t < t_{k+1}$, $x(t)$ is given by the integral equation (2.7). Also, in the interval $[t_0, T]$ and due to the Hurwitz property of $A_i - B_iK_i$, there exist two positive constants γ and Γ, such that

  $$\gamma\|x(t_k)\| \leq \|e^{(A_i - B_iK_i)(t-t_k)}x(t_k)\| \leq \Gamma\|x(t_k)\|.$$

  Besides, since $x(t)$ is Lipschitz continuous, we can deduce the following lower bound on $\|x(t)\|$,

  $$\|x(t)\| \geq \gamma\ \mu - \Gamma\|B_iK_i\|\frac{L_x}{2}(t - t_k)^2.$$

  So, if we select a sufficiently small τ_1, when $t \in [t_k, t_k + \tau_1)$, there exists $\varepsilon > 0$, such that

  $$\|x(t)\| \geq \varepsilon.$$

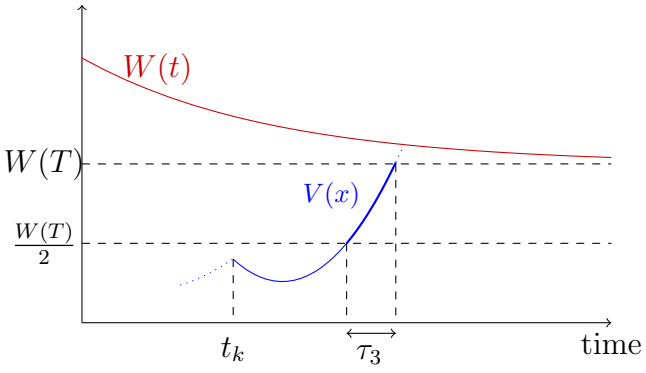


Figure 2.3: Illustration of the existence of an inter-event time when $V(x(t_k)) < W(T)/2$.

Using this lower bound yields an upper bound on the first term of Equation (2.17)

$$-x(t)^T Q_i x(t) \leq \lambda_{\min}(Q_i)\varepsilon^2 =: -2\beta.$$

The second term admits an upper bound

$$\| - 2\Delta_k x(t)^T K_i B_i P x(t)\| \leq 2L_x(t - t_k)\|K_i B_i P\| M,$$

which can be rendered smaller than β by choosing t from an interval $[t_k, t_k + \tau_2)$ with $\tau_2 \leq \tau_1$ and τ_2 small enough. Therefore, for $t \in [t_k, t_k + \tau_2)$, $dV(x(t))/dt \leq -\beta$, and $V(x(t))$ is strictly decreasing during this interval.

- Case $V(x(t_k)) < W(T)/2$.
  In this case, we cannot find a lower bound on $\|x(t)\|$, and thus we cannot estimate the time during which $dV(x(t))/dt$ decreases. However, we know that

  $$V(x(t_k)) < W(T)/2 < W(T) < W(t_k).$$

  Due to the large gap between $V(x(t_k))$ and $W(t_k)$, and to the Lipschitz continuity of $V(x(t))$, an intersection between $V(x(t))$ and $W(t)$ is not possible until some time τ has elapsed, as shown on Figure 2.3.

  Additionally, before an intersection with $W(t)$ is possible, $V(x(t))$ has to go first through the value $W(T)/2$ and then $W(T)$ (see Figure 2.3). If we assume that $V(x(t))$ increases linearly with the maximum possible rate, we can estimate the time it takes for $V(x(t))$ to

go from $W(T)/2$ to $W(T)$, as a lower bound on τ. Let τ_3 be this lower bound.

To determine the maximum possible rate, we need to find an upper bound on $|dV(x(t))/dt|$. We first re-write Equation (2.17) in the following form

$$\frac{dV(x(t))}{dt} = x(t)^T(2K_i^T B_i^T P - Q_i)x(t) - 2x(t_k)^T K_i^T B_i^T P x(t).$$

When $W(T)/2 \leq V(x(t)) \leq W(T)$

$$\|x(t)\| \leq \sqrt{\frac{W(T)}{\lambda_{\min}(P)}} =: \eta.$$

Then, we recall that as $V(x(t_k)) < W(T)/2$

$$\|x(t_k)\| < \sqrt{\frac{W(T)}{2\lambda_{\min}(P)}} = \frac{\sqrt{2}}{2}\eta.$$

The maximum possible rate is then

$$\left|\frac{dV(x(t))}{dt}\right| \leq (\|2K_i^T B_i^T P - Q_i\| + \sqrt{2}\|K_i^T B_i^T P\|)\frac{W(T)}{\lambda_{\min}(P)} =: \psi W(T).$$

Therefore, the time it takes $V(x(t))$ to reach $W(T)$ from $W(T)/2$ is given by the equation

$$W(T)\psi\ \tau_3 \approx \frac{W(T)}{2}.$$

A lower bound on τ is then

$$\tau \geq \frac{1}{2\psi},$$

where

$$\psi = \frac{\|2K_i^T B_i^T P - Q_i\| + \sqrt{2}\|K_i^T B_i^T P\|}{\lambda_{\min}(P)},$$

is a constant and is independent of t, t_k and T.

□

2.6 Numerical Example

2.6.1 Time-Dependent Switching

Consider the following second order switched system with three subsystems [41]

$$A_1 = \begin{bmatrix} 0.13 & -0.25 \\ 0.39 & -1.17 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 0.35 & -0.42 \\ -0.43 & 0.01 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -3.925 \\ -2.11 \end{bmatrix},$$
$$A_3 = \begin{bmatrix} -1.58 & 0.01 \\ -0.91 & 0.71 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 0.02 \\ -0.08 \end{bmatrix},$$

with $x_0 = [1 \quad -0.2]^T$. By using the following feedback gains

$$K_1 = \begin{bmatrix} 0.499 & -0.0074 \end{bmatrix},$$
$$K_2 = \begin{bmatrix} -1.0146 & 1.0493 \end{bmatrix},$$
$$K_3 = \begin{bmatrix} 1.7845 & -16.1789 \end{bmatrix},$$

we can make the three subsystems individually stable. They also allow us to find a CQLF with

$$P = \begin{bmatrix} 3.7698 & -3.7031 \\ -3.7031 & 4.4162 \end{bmatrix},$$

and decay rate $\lambda_{\max} = -0.5701$. We have obtained P and $\lambda_{\max}$ using the 'gevp' function of MATLAB. We choose $\alpha = 0.52$ s^{-1} and $W(t_0) = 5.928$. We have simulated the system for 30 seconds with a sampling time of 0.001 s. We have chosen a small sampling time to mimic the continuous-time behavior of the system. In addition, the active subsystem is chosen at random every $T_\sigma = 1.5$ s. The switching sequence is shown in Figure 2.4.

Figure 2.5 shows the time evolution of the two states $x_1(t)$ and $x_2(t)$. The two states converge to equilibrium around $t = 10$ s.

Figure 2.6 shows the event-triggered control law $u(t)$. The control has been updated 27 times, 14 times due to an intersection between $V(x(t))$ and $W(t)$, and 13 times due to a jump in the switching rule. Considering that the total number of simulation steps is $30,000$, we have decreased the communications between the controller and the plant by a factor of $1/1000$.

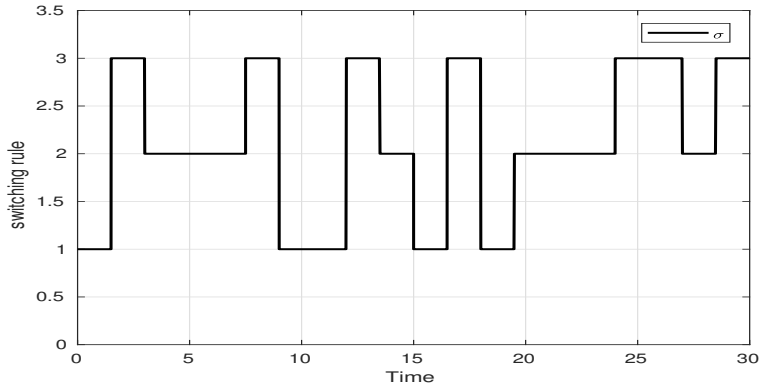


Figure 2.4: The switching sequence.

Figure 2.7 represents the PLF $V(x(t))$ and the upper threshold $W(t)$. We display the first 11 seconds only, for beyond that time, $W(t)$ and $V(x(t))$ approach zero, and it becomes harder to spot the events. We can notice the increases and decreases of the PLF, and the global convergence to zero. We can see the updates due to a jump in the switching rule, for example at $t = 1.5$ s. Examples of updates due to an intersection between the PLF and the threshold occur at $t = 2.83$ s and $t = 3.64$ s.

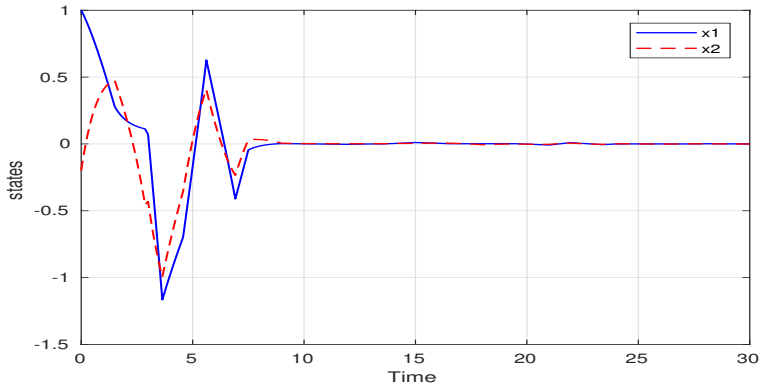


Figure 2.5: The time evolution of the states of the switched system.

To see the events for the entire simulation window, we display the distribution of events in Figure 2.8. It shows events that are unevenly distributed in time. Successive events are generally far apart, but can also be clustered together. This distribution further emphasizes the philosophy of event-triggered control to give attention to a system when most needed.

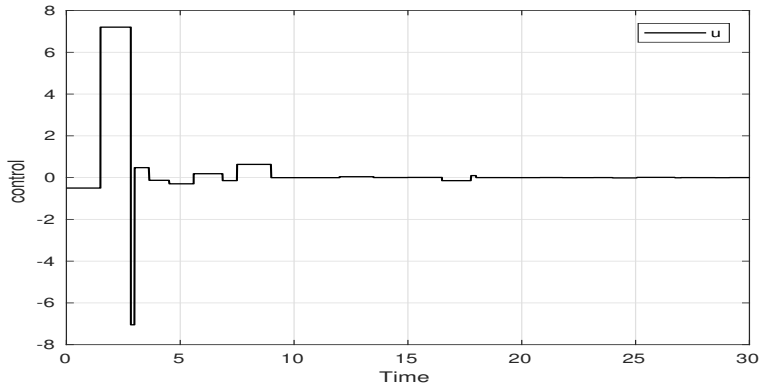


Figure 2.6: The event-based control signal.

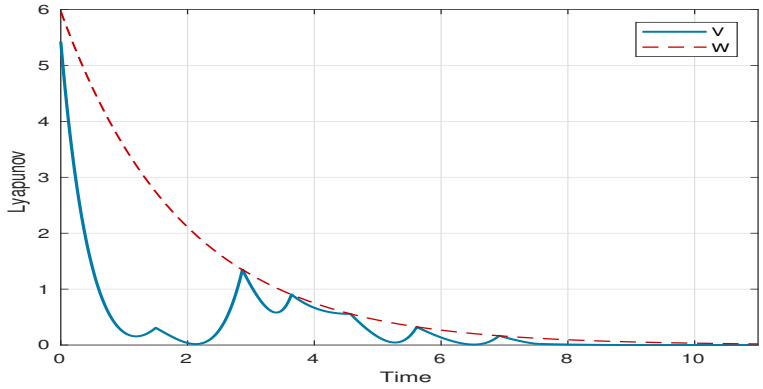


Figure 2.7: The Lyapunov function (in blue) and the exponential threshold (in red).

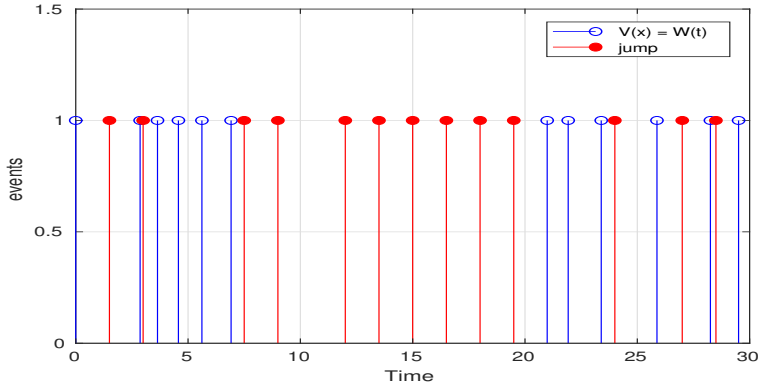


Figure 2.8: The events due to intersections (in blue) and to jumps (in red).

2.6.2 Event-Based Switching

We consider the previous second-order system with three subsystems. We keep the same experimental parameters as in the previous example and we

change the rate of decay to $\alpha = 0.47\ \mathrm{s}^{-1}$. This time, we consider a switching rule tied to the intersections between the PLF and the threshold. In this switching rule, the active subsystem changes when an intersection between the PLF and the threshold occurs. The switching sequence follows an ordered pattern of 1, 2, 3, 1, and so on. We chose this pattern because it represents one of the worst case scenarios, as there is a jump in the switching rule at every period T_σ. The switching signal is shown on Figure 2.9b.

Figure 2.9a shows the time evolution of the states. We can see that the states exhibit more oscillations and take a longer time to converge (more than 20 s compared to less than 10 s for the previous example). The number of updates of the control law remains comparable, with 31 updates.

Figure 2.9c and Figure 2.9d show that the largest inter-event times are obtained when subsystem 1 is active. This is due to the fact that susbsystem 1 has the least unstable mode in open-loop of all the subsystems. It has an unstable open-loop eigenvalue at 0.05, whereas subsystem 2 has an unstable eigenvalue at 0.64 and subsystem 3 possesses an open-loop eigenvalue at 0.71.

We have also experimented with various values of the rate of decay α, as increasing this value should speed up the convergence of the states to equilibrium. A maximum value of $\alpha = 0.569$ decreases the convergence time to within 17 s, which still represents a large value compared to the previous example.

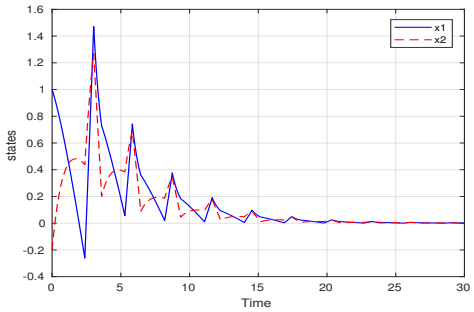
2.6.3 State-Dependent Switching

In this example, we examine the event-triggered control strategy when the switching rule is state-dependent. For this, we consider the following second order switched linear system, with two subsystems [42],

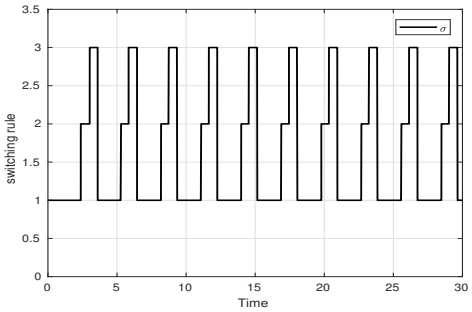
$$A_1 = \begin{bmatrix} -0.5 & 0 \\ 0.1 & 0.4 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 0.2 & 1 \\ 0 & 0.3 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.2 \\ 1 \end{bmatrix},$$

with $x_0 = [0.15 \quad -0.25]^T$.
For the feedback gains

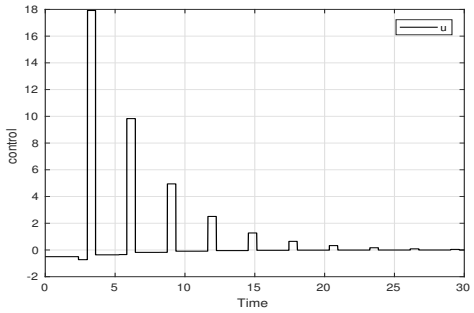
$$K_1 = \begin{bmatrix} 0.0737 & -0.6632 \end{bmatrix},$$
$$K_2 = \begin{bmatrix} 1.7797 & 1.5441 \end{bmatrix},$$



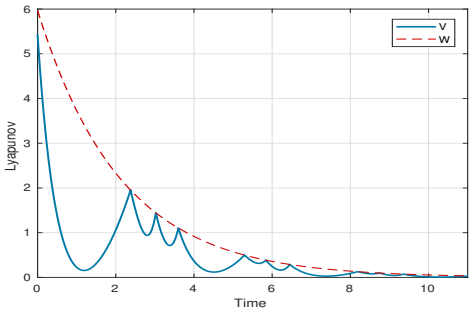
(a) States



(b) Switwhing rule



(c) Control



(d) Zoom on the PLF

Figure 2.9: Simulation results for state-dependent switching.

there exists a CQLF with

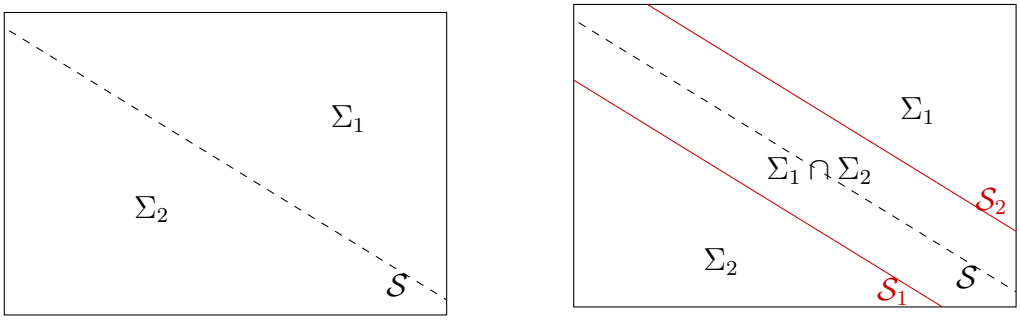
$$P = \begin{bmatrix} 0.6659 & 0.1481 \\ 0.1481 & 0.4937 \end{bmatrix}, \quad \lambda_{\max} = -0.514.$$

Hence, we select $\alpha = 0.513\ \mathrm{s}^{-1}$ and $W(0) = 1.1V(x_0) = 0.0382$. We also use the same simulation time and sampling period as the previous example.

To construct a state-dependent switching rule, we divide the state space into two regions, Σ_1 and Σ_2, separated by the surface $\mathcal{S}$, as shown in Figure 2.10a. Thus, when the state is in region Σ_1, subsystem 1 is active, whereas in region Σ_2, subsystem 2 is active.

When the surface $\mathcal{S}$ is a sliding surface, the state slides along $\mathcal{S}$ towards the origin. However, in a discrete-time simulation, the state keeps crossing to one side of $\mathcal{S}$ or the other during the sliding mode, creating switches at every instant. Such a situation contradicts our assumption that each subsystem must remain active for at least some duration τ_d. To solve this problem, we use the strategy in [28] called hysteresis switching.

In hysteresis switching, the surface $\mathcal{S}$ is off-set to the right and to the left, to define two new surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. This results in a strip (see Figure 2.10b) between the two new surfaces, intersecting both regions Σ_1 and Σ_2. This way, no switching occurs when either surface is crossed until the state leaves the common region $\Sigma_1 \cap \Sigma_2$.



(a) Regions of the state-space

(b) Off-setting the switching surface

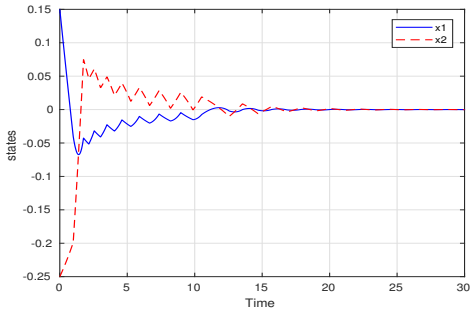
Figure 2.10: State-dependent and hysteresis switching.

In our example, we choose

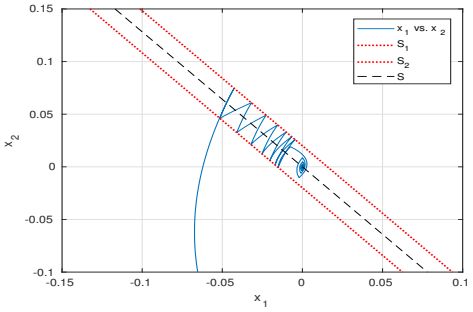
$$
\begin{aligned}
(\mathcal{S}) &: \; x_2 = -1.2825 x_1, \\
(\mathcal{S}_1) &: x_2 = -1.2825 x_1 - 0.02, \\
(\mathcal{S}_2) &: x_2 = -1.2825 x_1 + 0.02.
\end{aligned}
$$

Figure 2.11a shows the evolution of the states of the system with time. We see that the states eventually tend to equilibrium, proving the effectiveness of our approach. However, the states undergo an oscillation phase in transient-time. This is due to the fact that in the transient regime, the system experiences many switches as shown in the phase portrait of Figure 2.11b. From Figure 2.11b, we also verify the effects of hysteresis switching as the state bounces between $\mathcal{S}_1$ and $\mathcal{S}_2$, thus allowing for a dwell time. When the state reaches a vicinity of the equilibrium, it remains inside a ball centered at the origin and switching stops.

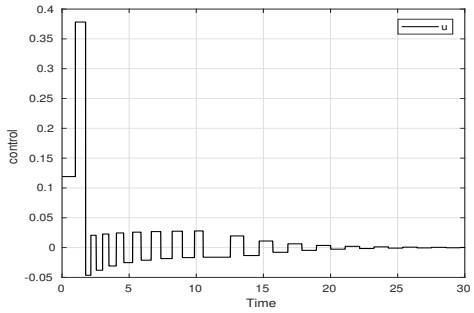
Figure 2.11c shows the event-triggered control law, which has been updated 34 times, 20 times due to an intersection between the PLF and threshold and 14 times due to a jump in the switching rule. Figure 2.11d shows the PLF between $t = 6$ s and $t = 15$ s. Figure 2.11d reflects what is seen on the phase portrait of Figure 2.11b, as the events in transient-time are mostly



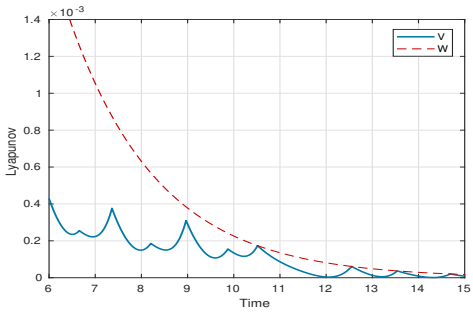
(a) States



(b) Phase Portrait



(c) Control



(d) Zoom on the PLF

Figure 2.11: Simulation results for state-dependent switching.

due to a jump in the switching rule, whereas events in stead-state are due to an intersection between the PLF and the threshold.

2.7 Conclusion

We have extended the event-triggered stabilizing control algorithm of Chapter 1 to switched systems. We have also shown how to extend the generalized eigenvalue problem to find a common quadratic Lyapunov function for all the subsystems of the switched system. The solution of this problem, as in the case of a single system, also allows us to find the maximum possible decay rate for the exponential threshold.

This approach allows us to decrease the number of updates of the control law. However, we have decreased the number of samples at the expense of the quality of the response, as the state undergoes sharp oscillations. This makes our method more suitable for applications where the shape of the response is not detrimental to the control objective. Additionally, in applications where

the users have control over the switching rule, they can smooth out the response by decreasing the dwell time of the switching rule.

Chapter 3

Event-Triggered Reference Tracking for Linear Systems

Contents

3.1 Introduction

So far we have only treated the problem of stability and stabilization of linear systems. In this chapter, we tackle the issue of reference tracking for linear

systems. Given the output of a system that consists in the entire state vector or part of it, we want to drive the output to follow a given reference trajectory.

A few works on event-triggered control for reference tracking exist throughout the literature. For instance, in [43] an event-based LQR controller is implemented, while the reference tracking is achieved through integral action. In [44], [45], [46], an additional fictitious system that generates the desired trajectory is defined. The event-triggering conditions are then established according to the error between the state of the reference system and that of the actual system.

In the approaches where a reference system is used, this system is chosen to be an appropriate external system. However, in our work, we propose to select the reference system as the continuously-controlled version of the event-triggered system being considered. It is indeed reasonable to assume that the behavior of the continuously-controlled system should serve as a model for the performance of the event-triggered algorithm. As the two systems share similar dynamics, the error between the two would be the result of the event-triggered implementation only, and not of the difference in behavior, reducing thus the number of events.

Our contribution also consists in how we maintain the error between the event-triggered state and the reference state below a certain threshold. In the case of linear systems, the error signal is treated as the state of a new virtual system. If the virtual system is stable, the error signal ultimately decays to zero, thus converting the tracking problem to a stability problem, that we know how to solve in the framework of event-triggered control. The stability requirement is again achieved by associating a pseudo-Lyapunov function function to the error system. However, in this chapter, we require only practical stability and not asymptotic stability. For this reason, we change the exponentially decreasing threshold to a constant one.

This chapter is organized as follows: in the next section we expose the mathematical framework of the problem at hand by explaining the event-based scheme. In Section 3.3, we define the event-triggering conditions that ensure the boundedness of the error. Some simulation results are then presented in Section 3.4.

3.2 Problem Statement

3.2.1 System Description

Let us re-consider the previous LTI system, with the output $y(t)$

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \\ x(t_0) &= x_0. \end{aligned} \tag{3.1}$$

The output $y(t) \in \mathbb{R}^p$ can be the entire state vector or a linear combination of the state variables. Since most of the time, only a portion of the state vector is accessible for measurement, the output vector is generally a linear combination of the state vector. However, we consider in this work that the entire state vector is available, either through measurement or observation. For this reason, in addition to the controllability of the pair (A, B), we require the pair (A, C) to be observable.

We wish to drive the output of system (3.1) to follow a trajectory determined by an exogenous signal $r(t)$, called the reference input. For this, we design an event-triggered control law of the form

$$u(t) = -Kx(t_k) + Gr(t_k), \quad \forall t \in [t_k, t_{k+1}), \tag{3.2}$$

where $K \in \mathbb{R}^{m\times n}$ is the feedback gain and $G \in \mathbb{R}^{m\times p}$ is the calibration $G = (-C(A - BK)^{-1}B)^{-1}$.

As a result, we obtain the following closed-loop system

$$\begin{aligned} \dot{x}(t) &= (A - BK)x(t) - BK\Delta_k x(t) + BGr(t_k), \\ y(t) &= Cx(t). \end{aligned} \tag{3.3}$$

This control law has to satisfy the following requirements

1. $(A - BK)$ is Hurwitz with desired eigenvalues,
2. $\lim_{t\to\infty}\|y(t) - r(t)\| \leq \varepsilon$,

where $\varepsilon > 0$ is a user-defined parameter.

3.2.2 Reference System

Previously, we have demonstrated how to stabilize an LTI system by keeping its Lyapunov-like function under a certain threshold. In this case though, the presence of the reference signal $r(t)$ adds an extra difficulty, as the pseudo-Lyapunov function does not give much information on the tracking process. To solve this problem, we define a reference system, as a continuously controlled version of System (3.1), with state $x_r(t)$, and driven by the control law $u_r(t) = -Kx_r(t) + Gr(t)$, for all t. The closed-loop reference system is given by

$$\begin{aligned} \dot{x}_r(t) &= (A - BK)x_r(t) + BGr(t), \\ y_r(t) &= Cx_r(t), \\ x_r(t_0) &= x_0. \end{aligned} \tag{3.4}$$

The output signal satisfies

$$\lim_{t\to\infty} \|y_r(t) - r(t)\| = 0. \tag{3.5}$$

The control law $u_r(t)$ is applied continuously to the reference system. From classical control theory, we know that achieving the objective of equation (3.5) is always possible when the pair (A, B) is controllable.

At each time instant, the behavior of the event-based system will be compared to that of the reference system. From this comparison, we will determine whether the behavior of System (3.1) is acceptable or whether the control should be updated.

Remark 4. *The principle of a reference system in the event-based scheme has also been used in references [44], [45], [46]. Our approach differs in the fact that the reference system is provided within the method, whereas in [44] and [45] the choice of the reference system is left for the user to make. The work presented in [46] deals with a stability problem since the states are driven to zero.*

Since the reference system achieves $\lim_{t\to\infty}\|y_r(t) - r(t)\| = 0$, it is sufficient to guarantee that $\lim_{t\to\infty}\|y(t) - y_r(t)\| = 0$, to ensure that the output $y(t)$ of the event-triggered system follows the trajectory of $r(t)$.

In this work, however, we consider the substitute problem of driving the state error $\|x(t) - x_r(t)\|$ to zero as time t tends to infinity. This principle

is depicted in Figure 3.1. From the relationship between $x(t)$ and $y(t)$, we deduce that $\lim_{t\to\infty}\|x(t) - x_r(t)\| = 0$ implies $\lim_{t\to\infty}\|y(t) - y_r(t)\| = 0$.

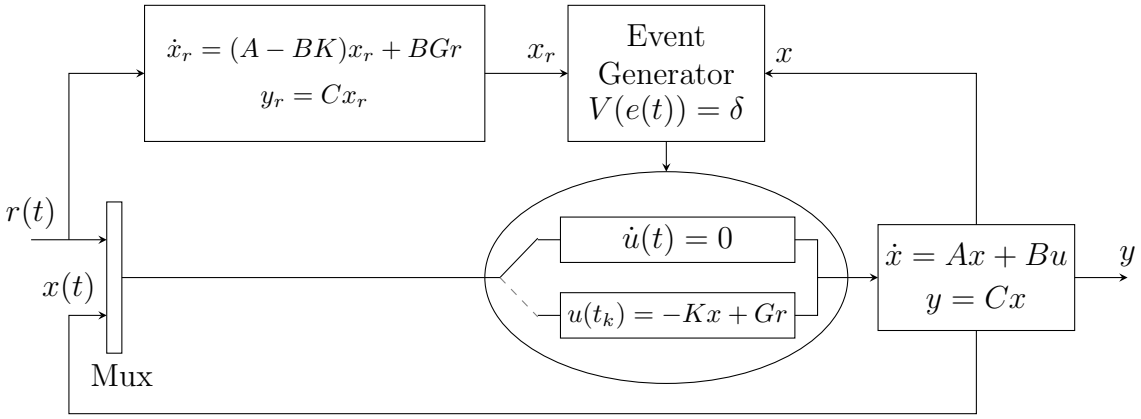


Figure 3.1: Schematic of the proposed event-based tracking controller.

For the plants where the states are not available for measurement, techniques for obtaining an estimation of these states (such as observers) can be used alongside our approach with only a small modification.

3.3 Event-Triggering Conditions

3.3.1 Defining the Event-Triggering Conditions

In this section, we define a new system based on the reference and the event-based systems. The state of this new system, the error between the real state $x(t)$ and the reference state $x_r(t)$,

$$e(t) = x(t) - x_r(t), \tag{3.6}$$

serves as a basis for updating the control $u(t)$. The dynamics of the error can be described as follows

1. when $t \in (t_k, t_{k+1})$

$$\dot{e}(t) = (A - BK)e(t) - BK\Delta_k x(t) + BG\Delta_k r(t), \tag{3.7a}$$

where $\Delta_k r(t) = r(t_k) - r(t)$, and $\Delta_k x(t) = x(t_k) - x(t)$;

2. when $t = t_k$, $k = 0, 1, 2, \ldots$

$$\dot{e}(t) = (A - BK)e(t). \tag{3.7b}$$

In order to satisfy the condition $\lim_{t\to\infty}\|x(t) - x_r(t)\| = 0$, we need to make $\lim_{t\to\infty}\|e(t)\| = 0$. From the point of view of the tracking error, this is a stabilization problem, that we can solve by associating a PLF to the virtual error system.

Since we defined the tracking problem as keeping the output $y(t)$ within a value ε of $r(t)$, we do not try to drive the tracking error to zero. Instead, we relax this condition and only require the error norm to remain within a ball of radius ε.

Remark 5. *We allow the exogenous signal $r(t)$ to be only piecewise Lipschitz, and we authorize it to contain jumps, provided that there exists a minimum interval of time between two successive jumps where $r(t)$ is continuous. Moreover, we require $r(t)$ to be uniformly bounded, i.e. $\|r(t)\| \leq B_r$, $B_r > 0$.*

Remark 6. *Note also that despite the jumps in $r(t)$, the states $x_r(t)$, $x(t)$ and $e(t)$ remain Lipschitz. However, their first derivatives do exhibit jumps. Indeed, jumps in $\dot{x}(t)$ occur when an event happens and jumps in $\dot{x}_r(t)$ are due to jumps in $r(t)$. Therefore $\dot{e}(t)$ exhibits jumps both when either $\dot{x}(t)$ or $\dot{x}_r(t)$ exhibit discontinuities.*

Based on equation (3.7b), we can associate to the system (3.7) a Lyapunov-like function of the following form

$$V(e(t)) = e(t)^T P e(t). \tag{3.8}$$

Since our objective is not to drive either the error $e(t)$ or the PLF to zero, and we are only interested in a practical form of stability, we can use a constant threshold, as shown on Figure 3.2. For this reason, P is chosen such that it satisfies the Lyapunov equation, with Q as a user-defined parameter, and does not have to be a solution of an optimization problem.

In what follows, we show how to choose the constant threshold in order to fulfill the requirement $\|e(t)\| \leq \varepsilon$.

Let $\delta > 0$ be the upper limit that we impose on $V(e)$,

$$0 \leq V(e) \leq \delta. \tag{3.9}$$

The PLF is naturally bounded as follows

$$\lambda_{\min}(P)\ \|e(t)\|^2 \leq V(e) \leq \lambda_{\max}(P)\ \|e(t)\|^2. \tag{3.10}$$

Therefore, if we keep $V(e(t))$ confined to the region delimited by the constant threshold δ, we can guarantee the following upper bound on $e(t)$,

$$\|e(t)\| \leq \sqrt{\frac{\delta}{\lambda_{\min}(P)}} =: \varepsilon. \tag{3.11}$$

Therefore,

$$\delta = \lambda_{\min}(P)\varepsilon^2. \tag{3.12}$$

Consequently we can define our triggering conditions based on this premise.

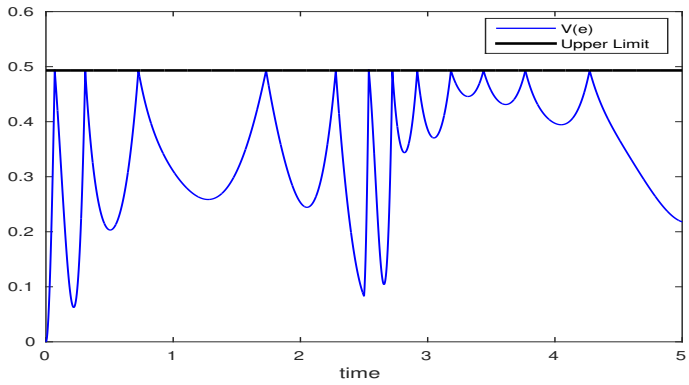


Figure 3.2: Evolution of the Lyapunov function with constant threshold.

Definition 4. *We define the time-instant t_{k+1} ($k \in \mathbb{N}$) at which the control-law $u(t)$ is updated as the minimum time instant $t > t_k$ for which $V(e(t)) = \delta$:*

$$t_{k+1} = \inf\{t > t_k, V(e(t)) = \delta\}. \tag{3.13}$$

3.3.2 Practical Stability Results

Theorem 6. *The event-triggered control law defined by Equation (3.2), scheduled by the event-triggering condition given by Definition 4, and driving System (3.1) with $e(t_0) = 0$, keeps the tracking error $e(t)$ confined in the ball of radius ε, denoted $\mathcal{B}(\varepsilon)$, i.e.*

$$\|e(t)\| \leq \varepsilon, \tag{3.14}$$

where ε is defined in Equation (3.11).

Proof. At $t = t_0$, $\|e(t_0)\| = 0 < \varepsilon$, by construction.
Then, at time $t = t_k$, when $V(e(t)) = \delta$

$$\frac{dV(e(t))}{dt}|_{t=t_k^+} = -e(t_k)^T Q e(t_k) < 0.$$

So, $V(e(t))$ decreases at time $t = t_k^+$ and is pushed back to the region comprised between the lines $V(e) = 0$ and $V(e) = \delta$. We have already shown that in this region, $\|e(t)\| \leq \varepsilon$.
Since Q is a positive-definite matrix, its largest and smallest eigenvalues, $\lambda_{\max}(Q)$ and $\lambda_{\min}(Q)$, respectively, are real and positive. Then, we can set an upper and lower limit on the descent of $V(e(t))$,

$$-\lambda_{\max}(Q)\|e(t_k)\|^2 \leq \frac{dV(e(t))}{dt}|_{t=t_k^+} \leq -\lambda_{\min}(Q)\|e(t_k)\|^2. \tag{3.15}$$

□

3.3.3 Minimum Inter-Event Time

Theorem 7. *If $r(t)$ is a Lipschitz continuous signal, there exists a minimum time $\tau > 0$, independent of k, such that*

$$\forall k \in \mathbb{N}, \quad t_{k+1} - t_k > \tau.$$

Remark 7. *Assuming that $r(t)$ is Lipschitz is not a very strong assumption, since in practice, a low-pass filter is used on the reference input to handle the abrupt changes that it may contain and avoid actuator saturation, eventually rendering it Lipschitz.*

Proof. We show that $V(e(t))$ decreases for some time τ_1, after an update of the control law.

When $t_k \leq t \leq t_{k+1}$,

$$\begin{aligned}\frac{dV(e(t))}{dt} &= -e(t)^T Q e(t) - 2\Delta_k x(t)^T K^T B^T P e(t) + 2\Delta_k r(t)^T G^T B^T P e(t) \\ &\equiv -e(t)^T Q e(t) + R_k(t). \end{aligned} \tag{3.16}$$

We first show that $\|e(t)\|$ remains large enough for some time after an update of the control law. We then use this fact to show that dV/dt remains strictly negative for a duration τ_1 after an event.

In the interval $[t_k, t_{k+1})$, $\dot{e}(t)$ is given by Equation (3.7a), where we denote

$$F_k(t) = BK\Delta_k x(t) - BG\Delta_k r(t).$$

The solution of Equation (3.7a) is

$$e(t) = e(t_k)e^{(A-BK)(t-t_k)} - \int_{t_k}^{t} e^{(A-BK)(t-s)}F_k(s)ds.$$

As established in the previous chapters, $x(t)$ is Lipschitz continuous with constant L_x, and Theorem 7 requires the signal $r(t)$ to be Lipschitz, with Lipschitz constant L_r. The term $\|F_k(t)\|$ can then be bounded as follows

$$\|F_k(t)\| \leq \|BK\|L_x(t-t_k) + \|BG\|L_r(t-t_k) \leq c_1(t-t_k).$$

For t sufficiently close to t_k, there exist two positive constants γ and Γ such that

$$\gamma\|e(t_k)\| \leq \|e(t_k)e^{(A-BK)(t-t_k)}\| \leq \Gamma\|e(t_k)\|.$$

Therefore,

$$\|e(t)\| \geq \gamma\|e(t_k)\| - \frac{1}{2}\Gamma c_1\tau_1^2.$$

From (3.10), we have $\|e(t_k)\| \geq \sqrt{\delta/\lambda_{\max}(P)}$. Therefore, for a sufficiently small τ_1 independent of k, we can ensure that

$$\|e(t)\| \geq \frac{\gamma}{2}\sqrt{\frac{\delta}{\lambda_{\max}(P)}}.$$

This yields an upper bound on the first term of Equation (3.16)

$$-e(t)Qe(t) < -\beta,$$

where $\beta = \lambda_{\min}(Q)\gamma^2\delta/2\lambda_{\max}(P)$.

The second term of Equation (3.16) can be bounded from above as follows

$$|R_k(t)| \leq 2\|PBK\|\|e(t)\|\|\Delta_k x(t)\| + 2\|PBG\|\|e(t)\|\|\Delta_k r(t)\|.$$

We recall that

$$\|e(t)\| \leq \sqrt{\frac{\delta}{\lambda_{\min}(P)}}.$$

Then, in an interval $[t_k, t_k + \tau_2)$,

$$|R_k(t)| \leq \sqrt{\frac{\delta}{\lambda_{\min}(P)}}(2\|PBK\|L_x + 2\|PBG\|L_r)\tau_2.$$

By choosing $\tau_2 < \tau_1$ small enough, we can make $|R_k(t)| \leq -\beta/2$, which in turns yields

$$\frac{dV(e(t))}{dt} \leq -\frac{\beta}{2}.$$

Therefore, there exists a minimum time interval during which the derivative of $V(e(t))$ is strictly negative and $V(e(t))$ is decreasing. The time duration for which $V(e(t))$ is a lower bound on the minimum inter-event time

$$\tau \geq \tau_2.$$

□

Remark 8. *If $r(t)$ experiences jumps as is the case in the numerical examples below, it is no longer possible to give a lower bound of the delay between two samples. Indeed, although when $r(t)$ is bounded, the term $R_k(t)$ can be bounded by a constant on the time interval $[t_k, t_k + \tau_2)$, we do not know the times at which jumps in $r(t)$ take place. In practice, $r(t)$ has a finite number of jumps, and so there exists a minimum inter-event time between updates, but it cannot be estimated in terms of the parameters of the system.*

3.4 Simulation Results

3.4.1 Yaw damper example

Let us reconsider the yaw damper example from Chapter 1 with

$$K = \begin{pmatrix} 0.3235 & -0.6325 & 0.1891 & 0.2561 \\ -12.5446 & -0.7213 & 25.9652 & 27.2314 \end{pmatrix}$$

and

$$G = \begin{pmatrix} -11.7069 & 0.7115 \\ -202.0659 & 35.3351 \end{pmatrix}.$$

We choose the positive-definite matrix P as

$$P = \begin{pmatrix} 4.6408 & -0.8446 & -0.1942 & -0.1980 \\ -0.8446 & 0.9479 & 0.2071 & 0.3299 \\ -0.1942 & 0.2071 & 0.6457 & 0.5917 \\ -0.1980 & 0.3299 & 0.5917 & 4.6188 \end{pmatrix}.$$

For a maximum tolerance on the error of $\varepsilon = 0.2$, we find $\delta = 0.02$. We set the sampling time to $t_s = 10^{-4}$.

We want the outputs to track a reference signal that consists in a combination of step signals. Let $h(t)$ be the unit-step function, then

$$r_1(t) = 0.2h(t) + 0.2h(t-15),$$
$$r_2(t) = 4h(t) - 2h(t-15).$$

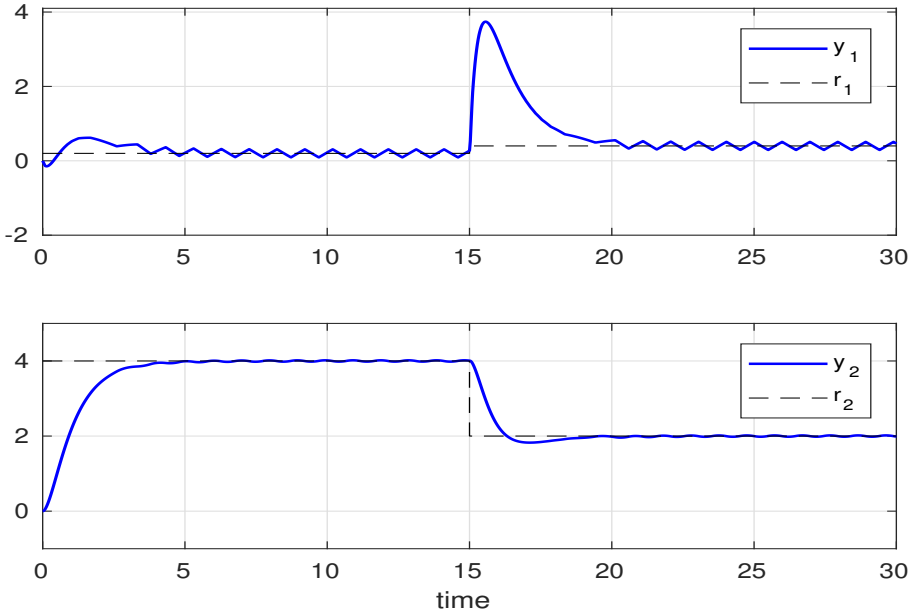


Figure 3.3: Time evolution of the two output signals with their respective reference signals.

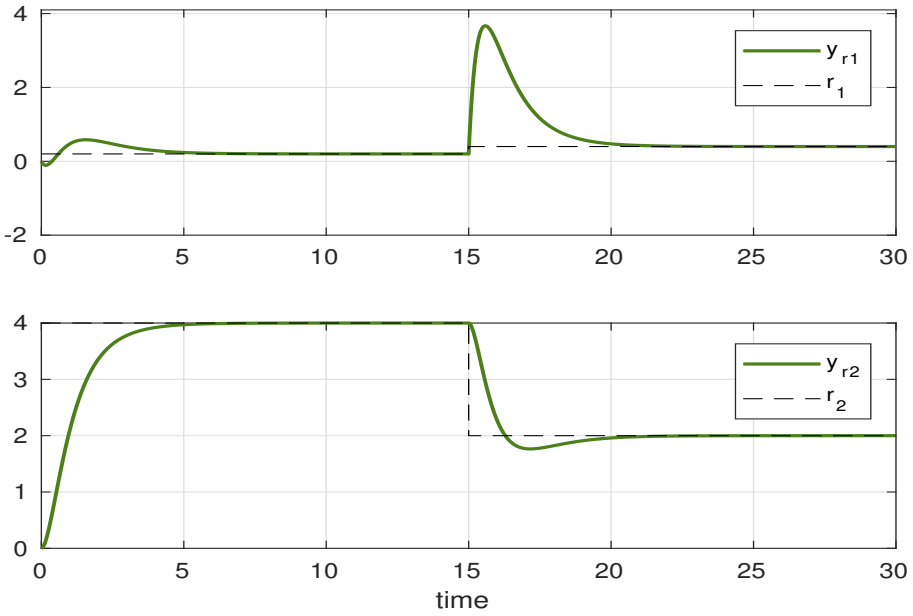


Figure 3.4: Outputs of the reference system

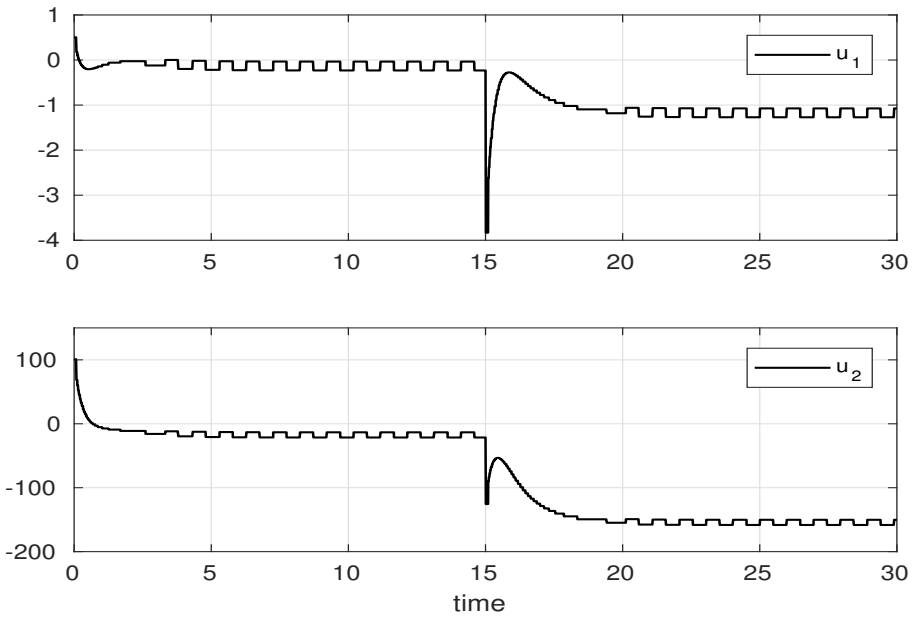


Figure 3.5: The event-based control signals.

Figure 3.3 shows the evolution of the two outputs with respect to time, with each output represented with the reference signal it is supposed to track. We can see that for a relatively large value of ε, the system is able to track

the reference. The system outputs are similar to the reference outputs shown on Figure 3.4, even though the first output y_1 exhibits some ripples. This is due to the large value of δ which results in updates of the control law that are further spaced in time as shown in Figure 3.5.

From Figure 3.5 we can see that the control updates are unevenly spaced in time. Indeed, during the transient period or when there is an abrupt change in the reference input, the control is updated more often, whereas when the reference settles to a constant value, the updates become less frequent and rather regularly spaced.

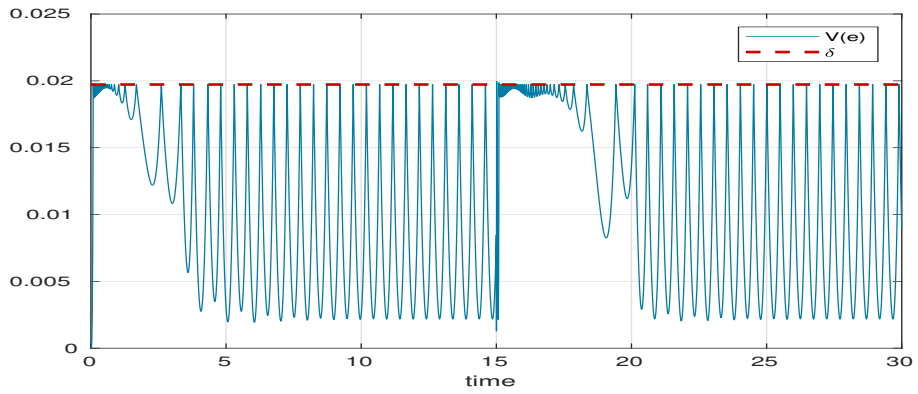


Figure 3.6: Time evolution of the pseudo-Lyapunov function.

Figure 3.6 shows the evolution of the PLF $V(e(t))$ with respect to time. We can notice that it remains enclosed within the region bounded from above by δ.

Around $t = 0$ and $t = 15$ s on Figure 3.6, when $r(t)$ changes values instantaneously, the events are also separated by a minimum time span, as seen on the zoom around $t = 15$ s on Figure 3.7. We notice that $V(e(t))$ goes above δ in some instants. This is due to the fact that computer simulations run in discrete-time rather than in continuous-time, and therefore, detecting the time at which $V(e(t))$ reaches precisely the value δ is not possible. This phenomenon is analyzed in more details when we discuss the discrete-time implementation in Section 3.5.1.

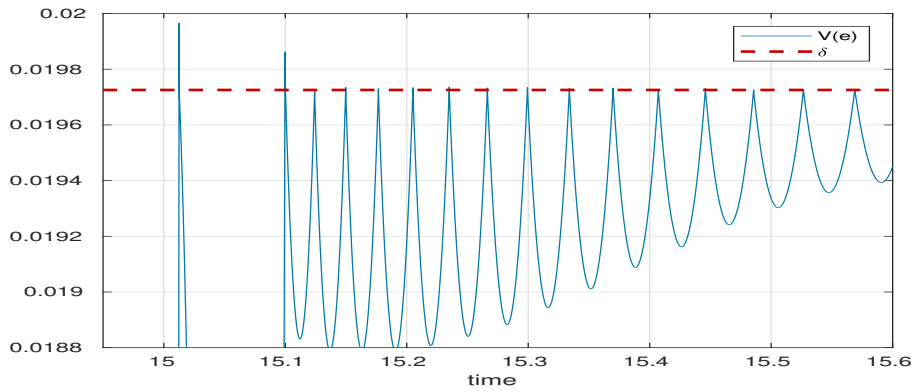


Figure 3.7: Zoom on the PLF at $t = 15$ s.

3.5 Effects of the Parameter ε

We have seen in the previous section that a large tolerance on the variations of the tracking error produces a response that oscillates around the desired trajectory, while requiring fewer updates of the control law.

We can further illustrate this property by increasing the value of the tolerance ε to $\varepsilon = 0.5$. We then observe the shape of the response and record the number of updates. Figure 3.8 shows the response of the system for this value of ε, with $\delta = 0.1233$. As expected, the quality of the response deteriorates, especially for the yaw rate, and exhibits oscillations of larger amplitudes around the desired trajectory. As to the number of updates of the control law, it drops to 70, for $300,000$ simulation instants, a factor of $1/4,200$.

Inversely, if we decrease the tolerance to $\varepsilon = 0.05$, we obtain a much smoother response as shown on Figure 3.9. The system requires 336 updates of the control law to achieve an output signal of such high quality. The further we decrease the tolerance, the closer the response is to the reference output shown on Figure 3.4.

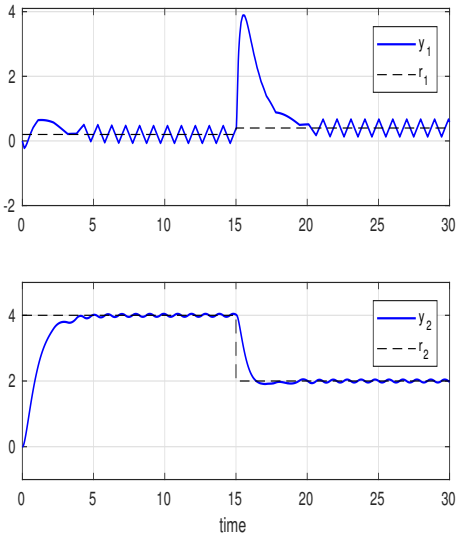


Figure 3.8: The response of the event-based system for $\varepsilon = 0.5$.

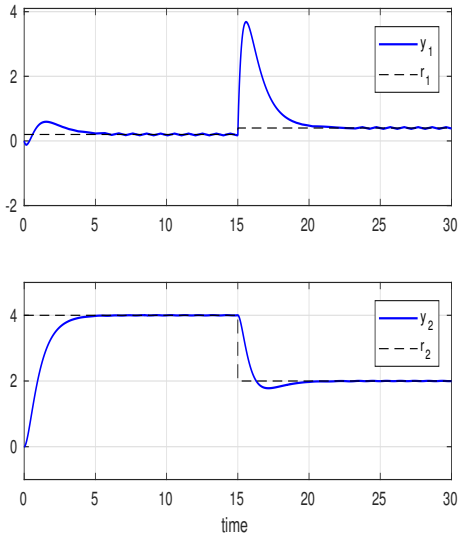


Figure 3.9: The response of the event-based system for $\varepsilon = 0.05$.

Table 3.1 lists the number of updates for several values of ε and δ.

ε	δ	Number of Updates
0.8	0.316	66
0.5	0.1233	71
0.05	0.0012	334
0.001	4.93×10^{-7}	10088

Table 3.1: Number of updates of the control signal for different values of ε and δ

3.5.1 Discrete-Time Implementation

The implementation of the event-based algorithm, whether in simulation or on a physical system, always contains a discrete-time part. In computer simulation, a simulation period has to be selected as computers are digital platforms. Physical systems are no different, because the computation of the control law and the monitoring of the event-triggering condition are carried out on digital processors. In either case, the event-triggering condition is checked periodically, at multiples of the simulation period or the clock signal.

In the event-triggered control algorithm that we propose, we assume that the event-triggering condition is checked continuously. However, since it is in reality discrete, it is impossible for us to detect the precise moment at which the equality $V(x(t)) = \delta$ is verified. Instead, we can only detect the instant t_k at multiples of the sampling time, when $V(e(t)) \geq \delta$.

Figure 3.7 zooms in on the region around δ and shows that the Lyapunov-like function goes above what should be the upper limit δ, as the event $V(e(t)) = \delta$ always occurs in between two sampling instants of the digital simulation.

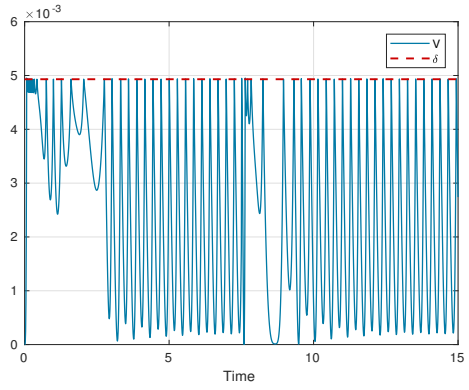
For this example, this phenomenon is not very inconvenient, as $V(e(t))$ decreases below δ within one sampling instant. But, this is not always the case. Let us reconsider the SISO system from Chapter 1 with

$$K = \begin{bmatrix} 3.0 & 7.5 \end{bmatrix}, \quad G = 5.$$

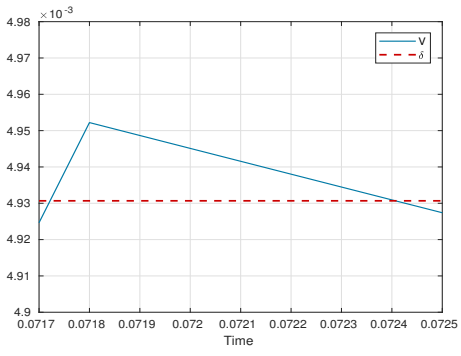
The Lyapunov-like function $V(e(t))$ has specifications

$$P = \begin{bmatrix} 1.1167 & 0.1000 \\ 0.1000 & 0.1333 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $\lambda_{\min}(P) = 0.1233$ and $\lambda_{\max}(P) = 1.1267$.



(a) The PLF of the SISO system and the threshold δ.



(b) Zoom on the region around $t = 0.0718$ s.

Figure 3.10: The PLF of the SISO system.

We choose $t_s = 10^{-4}s$, $\varepsilon = 0.2$, leading to $\delta = 0.0049$. Figure 3.10a shows the time evolution of $V(e(t))$ and the threshold δ. Figure 3.10b, which is a

zoom on the event at $t = 0.0718$ s, shows that the Lyapunov-like function does not fall below δ within one sampling instant as before, but takes 6 sampling instants to do so. The event is detected at $t = 0.0718$ s and we have to wait until $t = 0.0724$ s to recover $V(e(t)) < \delta$. This means that if we simulate the system for 15 s, out of the 104 events recorded for this case, a large portion is due not to the system's needs in control effort, but to the discrete-time implementation.

Even though this behavior is undesirable, and we suggest a few modifications to the algorithm in the next section to fix it, it is different from the continuous-time Zeno phenomenon. If the Zeno phenomenon induces an infinite number of updates in a finite interval of time, the discrete-time implementation guarantees that the updates are separated by a minimum time $\tau_{\min} = t_s$, in the worst case.

In addition, we can find an upper bound for this transgression. For the sake of simplicity let us assume that we use a simple explicit Euler scheme. Let t^n, $n \in \mathbb{N}$ be the uniform discrete times. Then $e(t^n)$ is approximated by e^n and

$$\frac{e^{n+1} - e^n}{t_s} = (A - BK)e^n - BK\Delta_k x^n + BG\Delta_k r(t^n), \quad t^n \in (t_k, t_{k+1}).$$

Hence, using the Hurwitz property of $A - BK$, and the fact that in the worst case $\|e^n\| = \sqrt{\delta/\lambda_{\max}(P)}$,

$$\|e^{n+1}\| \leq (\Gamma t_s + 1)\sqrt{\delta/\lambda_{\max}(P)} + t_s\|BK\|L_x(t^n - t_k) + t_s\|BG\|B_r,$$

where $r(t) \leq B_r$, for all t.
This yields an upper bound on the worst value of $V(e^{n+1}) \leq \lambda_{\max}(P)\|e^{n+1}\|^2$, which can be bigger than δ. We cannot avoid such overflows with explicit numerical schemes. When this happens, t^{n+1} is chosen as t_{k+1}. Then

$$V(e^{n+2}) = V(e^{n+1}) - t_s {e^{n+1}}^T Q e^{n+1} + t_s^2 V((A - BK)e^{n+1}),$$

and

$$\|V(e^{n+2})\| \leq (\lambda_{\max}(P) + t_s\lambda_{\max}(Q) + t_s^2\lambda_{\max}(P)\Gamma^2)\|e^{n+1}\|^2.$$

which can still be above δ. A few iterations can be necessary to recover $V(e^n) < \delta$.

3.5.2 Solutions in Discrete Time

3.5.2.1 Adaptive Threshold

One way to avoid the successive updates of the control to recover $V(e) < \delta$ in discrete time is to move the threshold temporarily to the value $V(e(t)) > \delta$. Then, when $V(e(t))$ passes below δ, the threshold is moved back to its original position at δ.

Figure 3.11 shows the same zoom around $t = 0.0718$ s as on Figure 3.10b, but this time we move the threshold to the value of $V(e(t))$ at that instant. We notice that it still takes six sampling instants for $V(e(t))$ to descend below δ. However, this time the event generator does not detect events at these instants and the control law is not updated anymore during this time interval. This time, for a 15 s simulation interval, we recorded 59 updates only, against 104 without the correction.

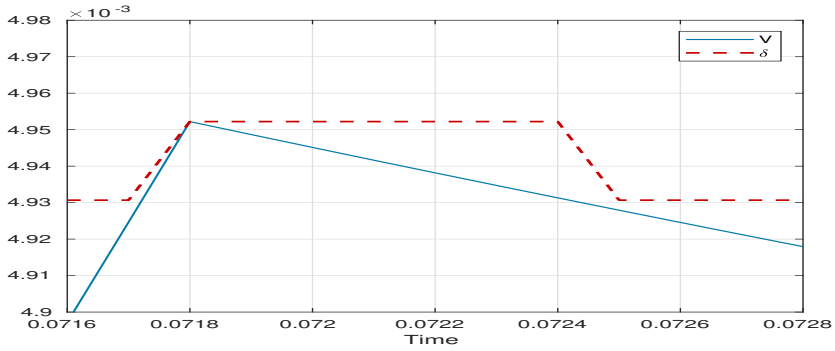


Figure 3.11: Lifting the threshold at $t = 0.0718$ s.

There is, however, a danger of destabilizing the system by using this method. If we imagine that during the time interval when the threshold is increased, instead of falling below the original δ, the PLF stops decreasing and increases again. As we are in discrete-time, it will go above the new threshold, and the threshold will be moved upwards once more. If this behavior repeats infinitely, the value of the threshold would grow to infinity, destabilizing the system. We can set an upper limit at which the threshold cannot increase anymore. This upper limit is determined by the maximum error that can be tolerated.

3.5.2.2 Updating the Reference State

The only virtual variable in our algorithm is the state of the reference system, which we introduce and which has no counterpart in the physical world. Therefore, it is the only variable that we can modify to correct the value of $V(e(t))$ at $t = t_k^+$. However, finding the modification to introduce on $x_r(t_k^+)$ is not a trivial task, because we want to make $V(e(t_k^+)) = \delta$, $V(e(t))$ being a pondered norm of $x(t) - x_r(t)$. The norm of a vector does not tell us much about its individual components, and there is an infinite number of ways to move the coordinates of a vector so that it has a particular norm.

One idea is to solve an optimization problem with an equality constraint. We look for a vector $\tilde{x}_r$, such that the L_1 distance between $\tilde{x}_r$ and x_r is minimal and

$$(x - \tilde{x}_r)^T P(x - \tilde{x}_r) = \delta.$$

This optimization problem can be written as

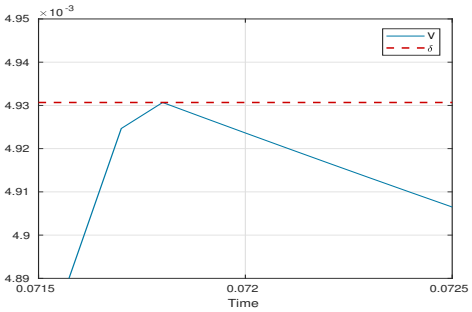
$$\begin{aligned} &\textit{minimize} \quad \sum_{i=1}^{n} |x_{ri} - \tilde{x_{ri}}| \\ &s.t. \quad (x - \tilde{x}_r)^T P(x - \tilde{x}_r) = \delta. \end{aligned} \tag{3.17}$$

Once we find this vector we can carry out the update

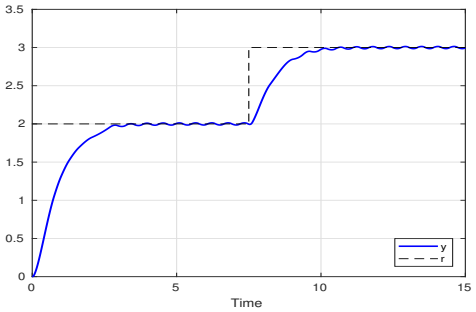
$$x_r(t_k^+) = \tilde{x}_r.$$

Figure 3.12a illustrates the result of applying this method on the SISO system. We use MATLAB's *fmincon* function to solve Problem (3.17). The zoom on the graph $V(e(t))$ at $t = 0.0718$ s shows that updating x_r with this method corrects the value of the PLF at $t = 0.0718$ s to $V(e(0.0718)) = \delta$, leaving the PLF below δ at the following instant. The response of the event-based system, where $y = x_1$, and the reference signal are shown on Figure 3.12b. This implementation led to 60 updates of the control law, a number also consistent with the previous correction method. The slight difference is due to the way we interfere with the reference trajectory in this method.

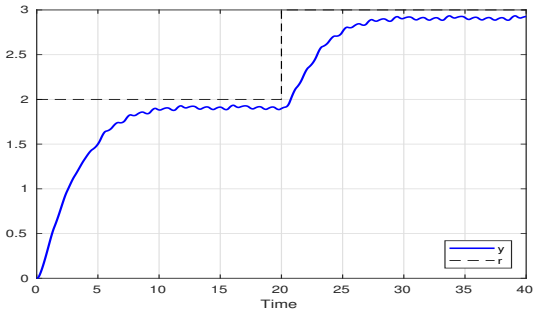
This approach appears to be too costly in time and too complex to carry out online. However, modern processors can solve much more difficult problems in a very short time, much shorter than the simulation time that we are using. As to the complexity of the problem and the computation effort that must be allocated to it, we recall that in this event-triggered control



(a) The PLF after correcting $x_r(t_k)$



(b) SISO response with $x_r(t_k)$ update



(c) SISO response for weaker control

Figure 3.12: Behavior of the SISO system when modifying x_r.

method, the number of updates is reduced considerably compared to the operation time of the system, and few optimization problems need to be solved. For example for the SISO system above, for $\varepsilon = 0.2$, and the operation time divided into 10^4 simulation instants, the system has required only 72 updates.

On the other hand, changing repeatedly the state of the reference system is equivalent to introducing impulsive disturbances every $t = t_k$. Therefore, in order not to derail the trajectory of the reference system, the sensitivity to state disturbances has to be low. For example, if we change the location of the poles of the SISO system by moving them closer to the imaginary axis, thus increasing the system's sensitivity to disturbances, we obtain the response shown on Figure 3.12c, which shows the trajectory of the reference system. We can see that there is a large tracking error, as the reference system fails to recover from the disturbances that we introduce every $t = t_k$.

3.6 Conclusion

In this chapter, we have introduced an event-triggered algorithm to solve an output tracking problem. By taking the error between the state of the system and the state of a reference system, and comparing the pseudo-Lyapunov function of this error signal to a constant threshold, we manage to drive the system to track a given reference signal, while ensuring a minimum inter-sample time.

This approach also offers a tremendous reduction in the communications between the controller and the plant as well as in the frequency at which we request the use of the CPU. This is true even if the reference signal is not Lipschitz continuous and experiences jumps, as has been demonstrated in the numerical example. The number of updates depends also on the tolerance imposed on the tracking error, the larger the tolerance, the less updates we need.

The use of a constant instead of an exponentially decreasing threshold gives rise to new challenges in the discrete-time framework. We have proposed two solutions to remedy these issues, by exposing the advantages and the drawbacks of each. The first method is simpler to implement but can cause instabilities if the tolerance on the tracking error is very low. The second method is harder to implement, as it requires to solve a constrained optimization problem, and to have a low sensitivity to disturbances, but offers good results.

Chapter 4

Event-Triggered Nonlinear Controller

Contents

4.1 Introduction

The event-triggered control algorithms introduced so far rely heavily on Lyapunov-like functions. When dealing with linear systems, if the system is stable in closed-loop, it is always possible to find a quadratic Lyapunov function. However, when the system is nonlinear, finding a Lyapunov function can be a long and arduous task. Additionally, in the linear case, we gave a detailed description of the event-triggered control algorithm, with a method for finding optimal parameters. Since no standard form for the Lyapunov function exists in the nonlinear case, we cannot give a similar level of

details if we want to introduce an event-triggered control technique.

For this reason, instead of using Lyapunov's theory, we experiment the use of an alternative approach to nonlinear analysis, known as contraction analysis [47]. Unlike Lyapunov theory, contraction analysis does not consider stability with respect to an equilibrium point. Instead, it defines stability as the ability of a system to dissipate initial conditions. If it is the case, all the neighboring trajectories converge to one another and to a nominal trajectory. To see whether a given trajectory converges to the nominal trajectory, we examine the virtual displacement between the two. If the virtual displacement decays to zero, then the system trajectory converges to the nominal trajectory and the system is stable.

The idea of a reference trajectory is similar to the reference system that we use in the event-triggered control algorithm of Chapter 3. Therefore, in the event-triggered algorithm that we propose, we take the nominal trajectory to be the reference trajectory, and the virtual displacement as the error between the event-triggered and the reference state. We use this approach for both stability and tracking problems. The stability condition on the virtual displacement is written as a generalization of the linear Lyapunov equation. This condition is used to construct the event-triggering condition for updating the control law.

This chapter is divided as follows. Section 4.2 gives an overview of contraction analysis, describes all the quantities involved and introduces the stability criteria. Section 4.3 describes the event-triggered control algorithm. Section 4.4 provides a numerical example to show the performance the approach. Finally, Section 4.5 offers comments on the limitations of this approach.

4.2 Overview of Contraction Analysis

4.2.1 Basic Principles

Let System (4.1) be a closed-loop nonlinear system

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

where f is an $n \times 1$ smooth, locally Lipschitz and continuously differentiable function. Let δx be the virtual displacement (infinitesimal displacement at

fixed time) between two neighboring trajectories, described as

$$\delta\dot{x}(t) \approx \frac{\partial f(x)}{\partial x}\ \delta x.$$

In [47], a contraction region is defined as a region of the state space where the symmetric form of the Jacobian $\frac{\partial f}{\partial x}$ is negative definite, i.e.

$$\frac{1}{2}\left(\frac{\partial f(x)}{\partial x}^T + \frac{\partial f(x)}{\partial x}\right) < -\beta I, \quad \forall x, \tag{4.2}$$

where $\beta > 0$. A Jacobian matrix with this property is referred to as uniformly negative definite.

Any trajectory which starts within a contraction region, remains in this region and converges to a nominal trajectory. The system is then exponentially stable, and globally exponentially stable if the entire state space is a contraction region.

However, even if we can design a stabilizing control law such that the closed-loop form (4.1) is stable, we cannot guarantee that its Jacobian is uniformly negative definite for all x. The authors of [47] suggest then a coordinate change that transforms the virtual displacement which becomes

$$\delta z = \Theta(x)\ \delta x,$$

where $\Theta(x)$ is a full rank non-zero matrix. In the new coordinate system, the time derivative of δz is

$$\frac{d}{dt}\delta z = F\delta z,$$

with the generalized Jacobian $F = \left(\dot{\Theta} + \Theta\ \frac{\partial f}{\partial x}\right)\Theta^{-1}$.
Consider then the time derivative of squared length

$$\frac{d}{dt}(\delta z^T \delta z) = \delta z^T (F^T + F)\ \delta z. \tag{4.3}$$

The counterpart of Equation (4.2) in the new coordinate system is Equation (4.3). Equivalently, if the symmetric form of F is negative definite in some region, the virtual displacement δz tends to zero. As $\Theta(x)$ is a full rank non-zero matrix, then if δz tends to zero, then so does δx, meaning that the neighboring trajectories that start in this region converge to one another. Moreover, the region in question is a contraction region, and any trajectory

that starts in this region remains inside this region.

The idea is then to select F as a constant negative definite matrix, and look for a coordinate transformation Θ that transforms δx to δz. Stability is then studied in this new coordinate system where $F + F^T$ is negative definite for all x (because constant), and where the entire state space is a contraction region. We later deal with the subject of how to find $\Theta(x)$ to ensure these properties.

In the old coordinate system, the squared length $\delta z^T \delta z$ is

$$\delta z^T \delta z = \delta x^T M(x) \delta x =: \Phi(\delta x),$$

where $M(x) = \Theta(x)^T \Theta(x)$ is a positive definite, initially bounded, and continuously differentiable metric. The time derivative of $\Phi(\delta x)$ is given by

$$\frac{d}{dt}\Phi(\delta x) = \delta x^T \left(\frac{\partial f(x)}{\partial x}^T M(x) + M(x)\frac{\partial f(x)}{\partial x} + \frac{d}{dt}M(x) \right) \delta x.$$

Therefore, the regions $F + F^T$ is negative definite correspond to

$$\frac{\partial f(x)}{\partial x}^T M(x) + M(x)\frac{\partial f(x)}{\partial x} + \frac{d}{dt}M(x) \leq -\beta_m M(x), \quad \beta_m > 0. \tag{4.4}$$

and a regions of the state space where Equation (4.4) correspond to contraction regions. Thus trajectories of the system converge exponentially to a nominal trajectory.

In this general framework, Theorem 2 in [47] states the following:

Theorem 8. *Give the system* (4.1)*, any trajectory, which starts in a ball of constant radius with respect to the metric $M(x)$, centered at a given trajectory and contained at all times in a contraction region with respect to $M(x)$, remains in that ball and converges exponentially to this trajectory. Furthermore, global exponential convergence to the given trajectory is guaranteed if the whole state space is a contraction region with respect to the metric $M(x)$.*

4.2.2 Coordinate Transformation and Control

To find Θ we use a slightly modified version of the procedure described in [48]. We consider the following nonlinear system affine in the control input u,

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ x(t_0) &= x_0. \end{aligned} \tag{4.5}$$

The method we present here is valid for more general nonlinear systems of the form $\dot{x} = f(x, u)$. But, in order to introduce the method in simpler terms, we choose to do so with the class of nonlinear systems affine in the control.

We assume that System (4.5) possesses a single equilibrium point. Let $x(t)$ be the trajectory of this system starting at x_0, and $x_r(t)$ be the nominal trajectory starting at x_{r0}. The dynamics of the virtual displacement between the two trajectories is given by

$$\delta\dot{x} \approx \frac{\partial f}{\partial x}\delta x + g(x)\delta u.$$

Let $\frac{\partial f}{\partial x} = J(x)$ and $g(x) = H(x)$, and the differential control $\delta u = -K(x)\delta x$. The matrix F is given by

$$F = \left(\dot{\Theta} + \Theta\ (J - HK)\right)\Theta^{-1}. \tag{4.6}$$

In [48] the authors propose to set F to a constant Hurwitz matrix in controllable canonical form, so that the rows of Θ can be computed recursively from Equation (4.6). The first $n-1$ equations of the system of equations resulting from (4.6) are rendered independent of K, by constraining the generalized Lie derivatives as follows

$$\begin{aligned}
&\theta_1 H = \theta_1 L^0 H = 0,\\
&\theta_2 H = \theta_1 L^1 H = (\frac{d}{dt}\theta_1 + \theta_1 J)H - \frac{d}{dt}(\theta_1 H) = 0,\\
&\quad\vdots\\
&\theta_{n-1} H = \theta_1 L^{n-2} = (\frac{d}{dt}\theta_1 + \theta_1 J)L^{n-3}H - \frac{d}{dt}(\theta_1 L^{n-3}H) = 0,
\end{aligned}$$

where θ_1, ..., θ_n are the row vectors composing the matrix Θ. Then, the only link between Equation (4.6) and K is ensured by the following equation

$$\theta_n H = \theta_1 L^{n-1} H = 1.$$

This system of n equations allows us to compute the entries of the row θ_1. In [48], the rest of the row vectors are computed recursively by using θ_1 and the sparse structure offered by the controllable canonical form of F.

However, it is easy to verify that setting F to a Hurwitz matrix is not enough to ensure that its symmetric part $\frac{1}{2}(F^T + F)$ is negative definite, nor

that Equation (4.4) is verified. So, in our approach we construct F such that it is negative definite and yet sparse enough to allow for a similar recursive computation. Therefore, we maintain the constraints on the generalized Lie derivatives above, and we set

$$F = \begin{bmatrix} -a_1 & 1 & 0 & 0 & \dots & 0 \\ 1 & -a_2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -a_3 & 1 & \dots & 0 \\ & \vdots & & & & \\ 0 & 0 & \dots & 1 & -a_{n-1} & 1 \\ 0 & 0 & \dots & 0 & 1 & -a_n \end{bmatrix}.$$

such that $a_i > 1$ for all $i = 1, \dots, n$. As before, θ_1 is computed from the constraints on the Lie derivatives. The remaining rows are then computed as follows

$$\begin{aligned} \theta_2 &= \dot{\theta}_1 + \theta_1 J + a_1 \theta_1, \\ \theta_3 &= \dot{\theta}_2 + \theta_2 J + a_2 \theta_1 - \theta_1, \\ &\vdots \\ \theta_n &= \dot{\theta}_{n-1} + \theta_{n-1} J + a_{n-1} \theta_{n-1} - \theta_{n-2}. \end{aligned}$$

Then, the feedback gain K is

$$K = \dot{\theta}_n + \theta_n J + a_n \theta_n - \theta_{n-1}.$$

The control input $u(t)$ can be taken as

$$u = -\int_{x_r}^{x} K(\chi) d\chi, \tag{4.7}$$

where $\chi \in \mathbb{R}^n$ is an integration variable.

The above integral is a path integral computed along the forward image of the initial line segment connecting x_0 and x_{r0}. However, if x_0 and x_{r0} are close enough, we can approximate this path by the line segment connecting $x(t)$ and $x_r(t)$ at every time instant.

We note also that the method for finding Θ is not the only one. We have chosen this one because it is the most straightforward, and the one that we can detail the most.

Remark 9. *We have written the feedback K as a function of x only, but in the more general case, K can be a function of u, $\dot{u}$, ..., $u^{(2n-2)}$. If it is the case, u can only be computed numerically. Notice however that K does not depend on δu [49], which implies that u can be computed explicitly.*

4.3 Event-triggered Algorithm

4.3.1 Algorithm Description

Consider the nonlinear system (4.5). We want to control this system through an event-triggered control law of the form

$$u(x) = \nu(x(t_k)), \quad \forall t \in [t_k, t_{k+1}). \tag{4.8}$$

where ν is a nonlinear function of the state.

We define the following system

$$\dot{x}_r = f(x_r) + g(x_r)u_r, \; x_r(t_0) = x_{r0}, \tag{4.9}$$

as the reference system that produces the trajectory that the event-triggered system has to follow. This reference system can be used to solve either a regulation or a tracking problem. Note that the control u_r exists only to generate the reference trajectory.

Let the virtual displacement between the two trajectories be

$$\delta x = x - x_r. \tag{4.10}$$

When the two trajectories are close enough, the dynamics of δx can be approximated using the calculus of variations as

$$\delta\dot{x} \approx J(x)\delta x + H(x)\delta u, \tag{4.11}$$

where $\delta u = -K(x)\delta x$. The system trajectory tends to the reference trajectory if δx tends to zero. The global exponential stability of the error system is analyzed by shifting to a new coordinate system through the transformation Θ. We use the procedure described in Section 4.2.2 to find $K(x)$, the control u and the transformation Θ.

Since Equation (4.4) sets the condition for the exponential stability of the system, we want to make sure that it is verified for all t. However, since $M(x)$ varies in time, it would be very hard to find a parameter β_m that satisfies condition (4.4) for all t. For this reason, we want to relax the condition of uniform negative definiteness on the left hand term of Inequality (4.4) by requiring this term to be only strictly negative definite. As a consequence, we lose the exponential stability property, but can still ensure global asymptotic stability. Then, we can define our event-triggering conditions as follows

Definition 5. *We define the time instant t_k at which the control law $u(t)$ is updated as*

$$t_{k+1} = \inf\{t > t_k \mid \frac{d}{dt}\Phi(\delta x) \geq 0\}. \tag{4.12}$$

To give an idea of how $\Phi(\delta x)$ behaves when the control is not updated, we present here its rate of change when $t \in [t_k, t_{k+1})$,

$$\frac{d}{dt}\Phi(\delta x) = \delta x^T(J^T M + MJ + \frac{d}{dt}M)\delta x - 2\delta x_k^T\ K^T H^T M\ \delta x,$$

where $\delta x_k = \delta x|_{t_k}$. However, when the control law is updated, the rate of change of $\Phi(\delta x)$ becomes

$$\frac{d}{dt}\Phi(\delta x)|_{t_k^+} = \delta x^T((J - HK)^T M + M(J - HK) + \frac{d}{dt}M)\delta x.$$

4.3.2 Stability Results

Theorem 9. *The event-triggered control law given by Equation* (4.8) *and scheduled by the event-triggering condition described in Definition 5 renders the system* (4.5) *exponentially stable. Moreover, if the entire state space is a contraction region, the system is globally exponentially stable.*

The proof of this theorem follows automatically from the fact that the relationship (4.4) is maintained for all t. Then, from Theorem 2 in [47], all the trajectories that start within a ball of constant radius with respect to the metric M and centered around a nominal trajectory, remain in this ball, and converge to the nominal trajectory. Since the theorem states that condition (4.4) is a necessary and sufficient conditions, we can conclude that when it is satisfied, the system trajectory converges to the desired trajectory.

4.4 Numerical Simulation

We consider the following two-state, control affine nonlinear system, with one input and one output [50]

$$\begin{aligned} \dot{x}_1 &= x_2 + \sin x_1, \\ \dot{x}_2 &= x_1^2 + u, \\ y &= x_1, \\ x_0 &= \begin{bmatrix} 1.2 & 1 \end{bmatrix}^T. \end{aligned} \tag{4.13}$$

We consider the problem of stabilizing the system (4.13). We select the following stabilizing control law to produce the desired trajectory

$$u_r(t) = -x_1^2 - (x_2 + \sin x_1)\cos x_1 - k_1 x_1 - k_2(x_2 + \sin x_1),$$

where $k_1 = 6,\ k_2 = 5$ and $x_{r0} = [1.25\ 1.05]^T$. The control $u_r(t)$ renders the closed-loop reference system asymptotically stable.

The matrices J and H describing the dynamics of the virtual displacement are

$$J = \begin{bmatrix} \cos x_1 & 1 \\ 2x_1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We look for a feedback gain $K(x)$ and a coordinate transformation Θ such that

$$\dot{\Theta} + \Theta(J - HK) = F\Theta,$$

where

$$F = \begin{bmatrix} -2 & 1 \\ 1 & -3 \end{bmatrix}, \quad \Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}.$$

The constraints on the Lie derivatives introduced in Section 4.2.2 are written as

$$\begin{aligned} \theta_1 H &= 0, \\ \theta_1 J H &= 1. \end{aligned} \tag{4.14}$$

Solving System (4.14) of equations for θ_1, we get

$$\theta_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

We can then deduce θ_2 from the equation

$$\dot{\theta_1} + \theta_1 J = -2\theta_1 + \theta_2,$$

so that we obtain

$$\Theta = \begin{bmatrix} 1 & 0 \\ \cos x_1 + 2 & 1 \end{bmatrix},\ M = \begin{bmatrix} (\cos x_1 + 2)^2 + 1 & \cos x_1 + 2 \\ \cos x_1 + 2 & 1 \end{bmatrix}.$$

The metric M is positive definite for all x as it is required to be.
The feedback gain of the virtual system $K(x)$ is

$$K = -\begin{bmatrix} -2x_1 + x_2 \sin x_1 - 2\cos^2 x_1 - 3\cos x_1, & -3 - \cos x_1 \end{bmatrix}.$$

The stabilizing event-triggered control law u is

$$\begin{aligned} u = &- x_1^2 - (x_2 + \sin x_1)\cos x_1 - x_1 - 3\sin x_1 - (3 - \cos x_1)x_2 \\ &+ x_{r_1}^2 + (x_{r_2} + \sin x_{r_1})\cos x_{r_1} + x_{r_1} + 3\sin x_{r_1} + (3 - \cos x_{r_1})x_{r_2}. \end{aligned}$$

To verify that condition (4.4) is verified for all x, we compute

$$(J - HK)^T M + M(J - HK) + \dot{M} = \begin{bmatrix} -2(\cos x_1 + 1)^2 - 2 & -2\cos x_1 - 2 \\ -2\cos x_1 - 2 & -2 \end{bmatrix}.$$

The determinant of this matrix is 4 and the entry $-2(\cos x_1 + 1)^2 - 2$ is negative for all x, meaning that the matrix $(J - HK)^T M + M(J - HK) + \dot{M}$ is negative definite for all x and that Condition (4.4) is always satisfied.

The virtual displacement is defined as

$$\delta x = x(t) - x_r(t).$$

The reference trajectory can be simulated either online or offline. Then, the control law u is updated when the condition

$$\delta x^T (J(x)^T M(x) + M(x)J(x) + \frac{d}{dt}M)\delta x - 2\delta x_k^T \; K(x)^T H^T M(x) \; \delta x = 0$$

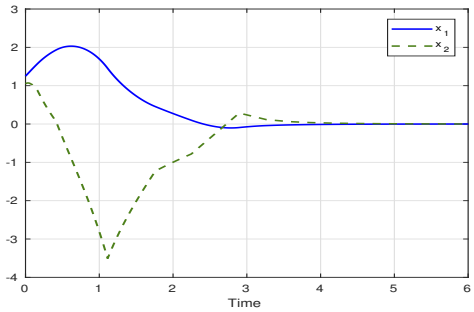
becomes true.

We have simulated the system for 20 s, with a sampling period of 10^{-3} s. Figure 4.1a shows the time evolution of the states of the system. We see that the system which is originally unstable is stabilized by the control law u, and the states converge to the equilibrium point. The response exhibits a few sharp edges but is overall smooth. Figure 4.1b shows that the virtual displacement ultimately tends toward zero, which means that the system trajectory and the reference trajectory converge toward each other.

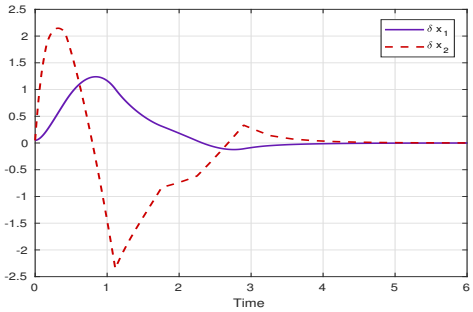
Figure 4.1d depicts the time evolution of $d\Phi/dt$. We notice that this function remains always negative. We can clearly see the updates of the control law when $d\Phi/dt$ reaches zero, and the function becoming negative again. The control law is depicted in Figure 4.1c. The control law has been updated 53 times in $20,000$ simulation instants.

4.5 Existence of Θ

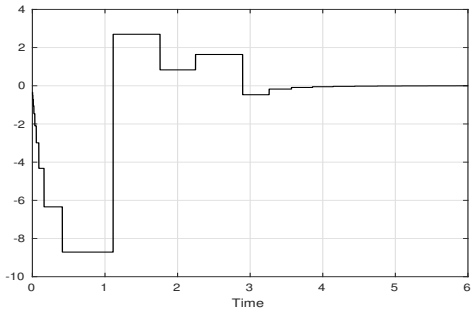
When introducing System (4.5), we have specified the condition that it should possess a single equilibrium point. The reason for this, as mentioned in [49],



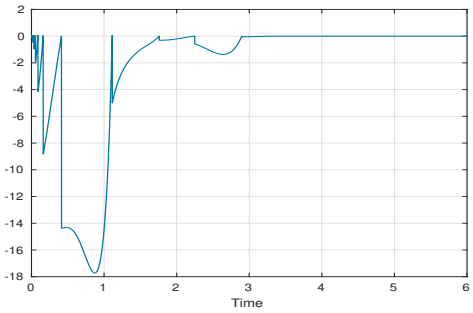
(a) States of the system



(b) Virtual displacement



(c) Control signal



(d) The function $d\Phi/dt$

Figure 4.1: Simulation results for nonlinear stabilization.

is that when a system possesses more than one equilibrium point, the coordinate transformation Θ will have singularities. These singularities divide the state-space into separate regions. Such singularities might not be problematic when using this method to prove the stability of a system, but in our event-triggered control algorithm we need explicit values of Θ and its inverse in order to evaluate the event-triggering conditions. This is impossible when Θ has singularities.

To illustrate this fact consider the following system [50]

$$\begin{aligned} \dot{x}_1 &= x_1 x_2, \\ \dot{x}_2 &= x_1 + u. \end{aligned} \tag{4.15}$$

The system has an infinite number of equilibria, consisting of the straight line $x_1 = 0$. The procedure described above gives a matrix Θ

$$\Theta = \begin{bmatrix} \frac{1}{x_1} & 0 \\ \frac{2}{x_1} & 1 \end{bmatrix},$$

for the same generalized Jacobian F as the one used in the numerical example.

The transformation Θ has a singularity at $x_1 = 0$, which shows how the equilibrium points on the line $x_1 = 0$ divide the state-space into two partitions. Therefore, when the system trajectory approaches this line, it becomes impossible to evaluate the event-triggering conditions.

Moreover, the virtual system (4.11) becomes uncontrollable at $x_1 = 0$ through linear feedback, as shown by the controllability matrix

$$\mathcal{C} = \begin{bmatrix} 0 & x_1 \\ 1 & 0 \end{bmatrix}.$$

When $x_1 = 0$, the rank of $\mathcal{C}$ drops to 1 and the virtual system becomes uncontrollable and no feedback gain $K(x)$ can be found.

However, consider the following nonlinear system with multiple equilibria [50]

$$\begin{aligned} \dot{x}_1 &= x_2 + x_1 \sin x_1, \\ \dot{x}_2 &= x_1 x_2 + u. \end{aligned} \tag{4.16}$$

System (4.16) possesses equilibrium points of with coordinates of the form $(j\pi,\ 0),\ j \in \mathbb{Z}$. The equilibrium points are scattered and do not divide the state-space into separate regions.

For the same F, we find

$$\Theta = \begin{bmatrix} 1 & 0 \\ \sin x_1 + x_1 \cos x_1 + 2 & 1 \end{bmatrix}.$$

We notice that in this case despite the existence of several equilibria, Θ has no singularities. This could be due to the fact that the equilibrium points are isolated.

4.6 Conclusion

In this chapter we propose an event-triggered control algorithm for nonlinear systems using contraction analysis. This approach takes the virtual displacement between the system trajectory and the desired trajectory. To ensure the decay of the virtual displacement to zero, we move to a new coordinate system where the generalized Jacobian of the system is uniformly negative definite. A generalized version of the linear Lyapunov equation is obtained.

Based on this equation, we construct event-triggering conditions that detects the instants at which the system trajectory exits the contraction region.

A lot of improvements can still be made to this approach. First, this approach has been described from the perspective of stabilization, but it can be generalized to reference tracking as well. Then, there is the problem of the existence of the transformation Θ. We know that when the system has a single equilibrium, Θ exists and does not have singularities. The author of [49] states that Θ has singularities when the system has multiple equilibrium points. We have seen that this was the case when the equilibrium points divide the state-space into separate regions. However, we have produced an example of a system with isolated equilibrium points and for which a transformation Θ with no singularities can be found.

Another point which we have not risen throughout this chapter is the existence of a minimum inter-event time. The event-triggering conditions that we use look similar to the ones used in Chapter 1 over the steady-state regime. Hence, we believe that a proof for the existence of a minimum inter-sample time should be similar to the one presented therein.

Finally, we want to be able to use any control law that we want to stabilize a system. For this, we need to break free from the procedure described in [48] to find Θ. This could be done by solving the equation $\dot{\Theta} + \Theta J = F\Theta$ or equivalently $\frac{\partial \Theta}{\partial x} f(x) + \Theta J = F\Theta$, numerically along system trajectories. Currently, we cannot carry out such computations because we lack knowledge about initial and boundary conditions. The only piece of information that we have is that the eigenvalues of the generalized Jacobian are coordinate invariant around the equilibrium point. Nonetheless, this information can only be used if we know the system trajectory in advance, which is not the case in event-triggered control.

Chapter 5

Self-Triggered Stabilizing Controller for Linear Systems

Contents

5.1 Introduction

The main advantage of the event-triggered control methods that we have introduced so far is to reduce the communications between the CPU and the controlled system, thus freeing the communications channels for the more urgent tasks. The drawback of this type of control is the necessity to monitor the event-triggering conditions constantly. This operation, depending on each individual method, could involve heavy computations, or require extra

circuitry. Such additions are not always possible, and in some cases, the location, age or design of the plant make them impossible to implement.

To counter these issues, we want to suppress the need for monitoring the event-triggering conditions at every instant. So we turn to methods that involve predicting the next execution time, and that are called self-triggered control methods. In self-triggered control, the system model is used to determine the time of the next event in advance. Thus, at each execution time, we update the control law, and we compute the next execution instant. This way, for the rest of the sampling interval, the CPU is free to run other tasks, and no additional wiring is needed to supervise the system.

Even in the case of linear systems, it is difficult to predict the time instants at which an event occurs in the plant, due to the complexity of the equations involved. For this reason, most of the works up to now have dealt with the issue in the framework of discrete-time systems. This is the case for [51], [52], [53]. In [18], the event-triggering conditions are developed in continuous-time, whereas the next execution time is found by setting a time horizon that is divided in sub-intervals. An event is then determined by checking the event-triggering conditions in each sub-interval.

Continuous-time systems have been studied in [54], where the problem is treated as an optimal control problem, with the next sampling instant as an unknown. The result is a non-convex quadratic programming problem which is then approximated by a convex problem. In [55] and [56] the authors suggest a self-triggered control method that preserves the $\mathcal{L}_2$ stability of the system in the presence of disturbances. In [55] the disturbance is bounded by a linear function of the norm of the state, while this condition is relaxed in [56]. Furthermore, self-triggered control schemes have often been coupled with model predictive control, as both use the model to project the behavior of the system up to some future time. Among these works we can mention [57] and [58].

In this chapter, we propose a self-triggered version of the event-triggered control algorithm introduced in Chapter 1. We have shown that this method relaxes the decay requirement on the Lyapunov function, by only requiring a Lyapunov-like function to remain under a decreasing threshold. This way, the events can be spread further apart in time. Using the properties of the functions involved, we formulate the self-triggered control problem as a combination of an optimization problem and a root-finding problem, in a continuous-time framework.

This chapter is divided as follows. Section 5.2 sets the notations used throughout the chapter. Section 5.3 is divided into two parts; the first part introduces the minimization algorithm, while the second part describes the root-finding algorithm. Finally, Section 5.4 offers numerical examples that validate our approach and that cover all the cases that we mention.

5.2 Event-Triggered Algorithm

In this section, we give a brief recap of the event-triggered control algorithm introduced in Chapter 1 and introduce some new notations. The event-triggering conditions consist in comparing the PLF to an upper threshold function. This condition is the only one that we use in this chapter, unlike in Chapter 1, where we used this condition for the transient regime only, and used another event-triggering conditions for the steady-state regime.
We reconsider the LTI system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ x(t_0) &= x_0. \end{aligned} \tag{5.1}$$

The control signal $u(t)$ is an event-triggered linear state-feedback control law which takes the following values

$$u(t) = \begin{cases} -Kx(t_k), & t = t_k \\ u(t_k), & t \in [t_k, t_{k+1}). \end{cases} \tag{5.2}$$

We can then re-write System (5.1) as

$$\dot{x}(t) = (A - BK)x(t) + BKe_k(t), \tag{5.3}$$

where $e_k(t) = x(t) - x(t_k)$, and define the augmented system with state $\xi_k(t) = [x(t),\ e_k(t)]^T \in \mathbb{R}^{2n}$ in $[t_k, t_{k+1})$, and dynamics (these notations are borrowed from [18])

$$\begin{aligned} \dot{\xi}_k(t) &= \begin{bmatrix} A - BK & BK \\ A - BK & BK \end{bmatrix} \xi_k(t) = \Psi\ \xi_k(t), \\ \xi_k(t_0) &= \begin{bmatrix} x_0 & 0_n^T \end{bmatrix}^T =: \xi_0, \end{aligned} \tag{5.4}$$

where 0_n is the vector of zeros in $\mathbb{R}^n$. The system of equations (5.4) admits a unique solution

$$\xi_k(t) = e^{\Psi(t-t_k)}\xi_k(t_k), \tag{5.5}$$

where $\xi_k(t_k) = \begin{bmatrix} x(t_k) & 0_n^T \end{bmatrix}^T$.
Then, for all t, the state of the augmented system is given by

$$\xi(t) = \sum_k \xi_k(t)\ I_k(t), \tag{5.6}$$

where $I_k(t)$ is the indicator function that takes values

$$I_k(t) = \begin{cases} 1, & t \in [t_k, t_{k+1}), \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

In what follows, we designate $\xi_k(t)$ as $\xi(t)$ when the two can be distinguished from the context.

Remark 10. *Instead of defining the augmented system* (5.4)*, we could have worked with the simpler system* (5.3)*. However, re-writing Equation* (5.3) *as* $\dot{x}(t) = Ax(t) - BKx(t_k)$ *yields a solution* $x(t) = (e^{A(t-t_k)} - A^{-1}(e^{A(t-t_k)} - I)BK)x(t_k)$*, which requires* A *to be non-singular. This requirement would force us to exclude a certain class of systems, unlike the System* (5.4)*, which admits a solution for all* A*.*

We associate to System (5.4) the pseudo-Lyapunov function

$$V(\xi(t)) = \xi(t)^T \begin{bmatrix} P & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix} \xi(t) \equiv \xi(t)^T\ \mathcal{P}\ \xi(t), \tag{5.8}$$

where P is defined in Chapter 1 and $0_{n\times n}$ is the $n \times n$ matrix of zeros.
The time instants t_k, $k \in \mathbb{N}$ are defined as

$$t_k = \{t > t_{k-1} \mid V(\xi(t)) \geq W(t)\}, \tag{5.9}$$

where $W(t)$ is a decreasing threshold function expressed as

$$W(t) = \begin{cases} W_0 e^{-\alpha(t-t_0)}, & t \in [t_0, t_1) \\ W_k e^{-\alpha(t-t_k)}, & t \in [t_k, t_{k+1}),\ \forall k > 0, \end{cases} \tag{5.10}$$

with $W_0 \geq V(\xi_0)$, $W_k = V(x_k(t))$, and $0 < \alpha < |\lambda_{\max}|$, where $\lambda_{\max}$ is the solution of the maximum generalized eigenvalue defined in Chapter 1.

5.3 Self-Triggered Algorithm

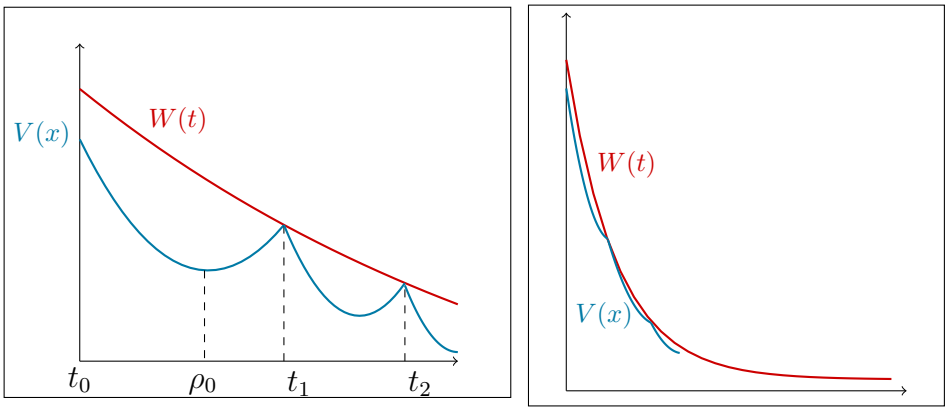
The goal of a self-triggered algorithm is to predict the time t_{k+1}, knowing the time t_k, the state vector $x(t_k)$, and the system model. Let $Z(t) = W(t) - V(\xi(t))$. The time t_{k+1} is the time at which the following equation is verified

$$Z(t) = 0. \tag{5.11}$$

Equation (5.11) depends on the time and implicitly on the state $\xi(t)$ which depends on the time through a transition matrix as seen from Equation (5.5). This configuration renders Equation (5.11) extremely difficult, if not impossible, to solve analytically. For this reason, we propose a numerical solution to Equation (5.11), where the instant t_{k+1} is computed through a root-finding algorithm.

A numerical scheme needs an initial value, and our first guess would be to initialize the root-finding algorithm at instant t_k in order to predict the instant t_{k+1}. However, the instants t_k and t_{k+1} can be relatively far apart, and as a result, the algorithm may fail to converge. Then, we have to initialize our algorithm at a later time instant. Let ρ_k denote the first time instant at which the PLF reaches a local minimum after the time t_k. The instant ρ_k is a good candidate for an initial value, and in what follows, we evaluate its use in the root-finding algorithm.

Depending on the dynamics of the system and the choice of the decay rate α, the evolution of the PLF after an update of the control can be classified into two categories. Either the PLF reaches its minimum value within the interval $[t_k, t_{k+1})$, or the PLF intersects the threshold before it has time to reach a minimum value. The two cases can be further described as follows:



(a) Case $\rho_k \leq t_{k+1}$. (b) Case $\rho_k > t_{k+1}$.

Figure 5.1: Shape of the PLF for different choices of α.

1. Case $\rho_k \leq t_{k+1}$:
   From experimental observations, this case is the most frequent of the two. It occurs when α is far enough from $|\lambda_{\max}|$, and the threshold

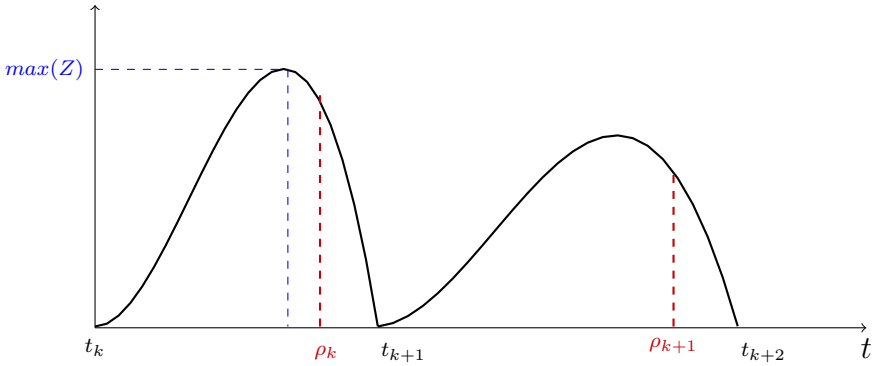


Figure 5.2: $Z(t)$, the difference between $W(t)$ and $V(x(t))$ in two sampling intervals where $\rho_k \leq t_{k+1}$.

decreases slowly compared to the PLF. After an update of the control, the latter has enough time to decrease, reach a minimum value at time ρ_k, and increase again before intersecting with the threshold at time t_{k+1}. This case is depicted in Figure 5.1a.

At time ρ_k, we know that $t_{k+1} > \rho_k$ if $Z(\rho_k) > 0$, indicating that the threshold is still above the PLF at ρ_k. In this scenario, ρ_k is an ideal choice for initializing the root-finding algorithm. First, because ρ_k is closer to t_{k+1} than t_k. Secondly, if we examine the difference $W(t) - V(\xi(t))$ shown on Figure 5.2 for two successive intervals $[t_k, t_{k+1})$ and $[t_{k+1}, t_{k+2})$, we see that in each of these two intervals, the difference goes through a stationary point. If we initialize the root-finding algorithm with t_k, the presence of the stationary point could throw the algorithm off track. However, ρ_k is always located after the stationary point, as seen on Figure 5.2, and as confirmed by the following proof. At the stationary point, $dZ(t)/dt = 0$, i.e.

$$\frac{dV(\xi(t))}{dt} = -\alpha W_k e^{-\alpha(t-t_k)},$$

showing that the derivative of the PLF is negative at the stationary point, meaning that the PLF is still in the decreasing stage. Thus the instant ρ_k, at which the derivative of the PLF vanishes, must occur after this stationary point.

2. Case $\rho_k > t_{k+1}$:
This case is depicted in Figure 5.1b, and is the rarest of the two cases, according to experimental observations. It generally results from a bad

choice of closed-loop eigenvalues and the decay rate α. If α is chosen close enough to $|\lambda_{\max}|$, and if, after an update of the control law, the PLF and the threshold decrease with approximately similar speeds, the two functions might intersect before the PLF reaches its minimum value. Then the time instant t_{k+1} precedes the instant ρ_k. Similarly to the previous case, we know that we are in this case if at time ρ_k, $Z(t_k) < 0$.

Even though this case is possible, it can be avoided by choosing α further from $\lambda_{\max}$, by placing the poles far enough from the imaginary axis, and making sure that their damping ratio is acceptable. If this case happens, we still can use ρ_k as an upper bound for t_{k+1}, and locate the root in the interval $[t_k, \rho_k]$.

The time ρ_k is not known in advance and needs to be computed. For one-dimensional systems, where vector x, and matrices A and B are scalars, the instants ρ_k can be computed analytically. For higher-dimensional systems, ρ_k can only be determined numerically. The numerical solution is found by a minimization algorithm which searches for the local minimum of $V(\xi(t))$ that occurs immediately after t_k.

5.3.1 Minimization Stage

5.3.1.1 One-Dimensional Systems

This case is simpler and helps us illustrate our approach. So, we consider the first order LTI system described as

$$\begin{aligned} \dot{x}(t) &= ax(t) + bu(t), \\ y(t) &= cx(t), \end{aligned} \tag{5.12}$$

where $x(t), u(t) \in \mathbb{R}$, and $a, b, c \in \mathbb{R}^*$, $\forall t > 0$.

Let x_k denote $x(t_k)$. The event-triggered control law is given by $u(t) = -Kx_k$ and System (5.12) in its closed-loop form is given by

$$\dot{x}(t) = ax(t) - bKx_k, \quad \forall t \in [t_k, t_{k+1}), \tag{5.13}$$

Since we assumed that $a \neq 0$, the augmented system described by Equation (5.4) is not needed for the scalar case. The differential equation (5.13) admits a unique solution for $t > t_k$, given by

$$x(t) = \left(\frac{bK}{a} + (1 - \frac{bK}{a}) e^{a(t-t_k)} \right) x_k. \tag{5.14}$$

To System (5.12), we associate a Lyapunov-like function of the form

$$V(x(t)) = px(t)^2, \tag{5.15}$$

where $p > 0$ is a solution to the Lyapunov inequality

$$2p(a - bK) \leq -q, \tag{5.16}$$

where $q > 0$ is a user-defined design parameter.

The minimum of $V(x(t))$ corresponds to

$$0 = \frac{dV(x(t))}{dt} = 2p(ax(t) - bKx_k)x(t). \tag{5.17}$$

Equation (5.17) admits two solutions, $x(t) = 0$, and $x(t) = bKx_k/a$. However, the solution $x(t) = bKx_k/a$ is impossible as it is equivalent to

$$\left(\frac{bK}{a} + (1 - \frac{bK}{a})e^{a(t-t_k)}\right) x_k = \frac{bKx_k}{a},$$
$$e^{a(t-t_k)}\left(1 - \frac{bK}{a}\right) = 0.$$

We know that $e^{a(t-t_k)} \neq 0$ and we cannot choose K such that $bK/a = 1$ or else we would destabilize the system. Therefore, in the scalar case, $dV/dt = 0$, if and only if $x(t) = 0$.

Consequently, the local minima of $V(x(t))$ occur only when $x(t) = 0$ and ρ_k can be directly computed from Equation (5.14)

$$\left((1 - \frac{bK}{a})e^{a(\rho_k - t_k)} + \frac{bK}{a}\right) x_k = 0.$$

We know that $x_k \neq 0$, because at $t = t_k$, $V(x_k) = px_k^2 = W(t_k) \neq 0$, hence $x_k \neq 0$. Therefore, the times ρ_k are given by the expression

$$\rho_k = \frac{1}{a} \log\left(\frac{bK}{bK - a}\right) + t_k. \tag{5.18}$$

We can always take the logarithm of $bK/(bK - a)$ because this is always a positive quantity, as can be seen from the following proof.

- Case $a > 0$:
  The feedback gain is chosen such that $a - bK < 0$. Then, $bK - a > 0$, and $bK > a > 0$. Since the numerator and denominator are both positive, then $bK/(bK - a) > 0$. Moreover, $bK/(bK - a) > 1$, proving that the ρ_k computed by Equation (5.18) occurs indeed after t_k.

- Case $a < 0$:
  If the open-loop system is already stable, the objective of the control is certainly to place the pole further to the left. Then, the feedback gain is chosen such that $a - bK < a < 0$. Then, we must have $bK > 0$ and $bK - a > 0$. Consequently, as in the previous case, $bK/(bK - a) > 0$. Even if in this case $bK/(bK - a) < 1$, the ρ_k given by Equation (5.18) still occurs after t_k.

Equation (5.18) is independent of x_k, indicating that the interval $[t_k, \rho_k]$ has the same length for all k.

5.3.1.2 Higher-Dimensional Case

As no analytical solution is available, we introduce a minimization algorithm for determining the first instant ρ_k that minimizes $V(\xi(t))$ for $t > t_k$. The minimization algorithm is a modified Newton algorithm. In this algorithm, a Newton step is first computed in the direction of descent of V, then scaled through a line search to ensure that $V(\xi(t))$ decreases enough at each iteration.

To compute the Newton step, we need to calculate the first and second time derivatives of $V(\xi(t))$. For simplicity, in what follows, we refer to the first and second derivatives as $\nabla_{\mathrm{t}} V$ and $\nabla^2_{\mathrm{t}} V$.

$$\nabla_{\mathrm{t}} V = \xi(t)^T \begin{bmatrix} M & L \\ L^T & 0_{n\times n} \end{bmatrix} \xi(t), \tag{5.19}$$

where $M = (A - BK)^T P + P(A - BK)$ and $L = PBK$, and

$$\nabla^2_{\mathrm{t}} V = \xi(t)^T \begin{bmatrix} \Lambda & \Gamma \\ \Gamma^T & \gamma \end{bmatrix} \xi(t), \tag{5.20}$$

where

$$\begin{aligned} \Lambda &= (A - BK)^T M + M(A - BK) + (A - BK)^T L^T + L(A - BK), \\ \Gamma &= (A - BK)^T L + MBK + LBK, \\ \gamma &= L^T BK + K^T B^T L. \end{aligned}$$

The minimization algorithm is detailed in Algorithm 1. The algorithm starts by an initial guess $\rho^{(0)} = t_k$. We then set the maximum number of iterations $MaxIter$ to a randomly large number. Its only purpose is to avoid an infinite loop in case the algorithm does not converge. The variable *iter*

monitors the number of iterations, and is incremented at each iteration until $MaxIter$ is reached or a solution is found.

The core of the algorithm works as follows. A Newton step is computed by first computing $\xi(\rho)$, $\nabla_t V$ and $\nabla_t^2 V$, using Equations (5.5), (5.19) and (5.20), respectively. If $\nabla_t^2 V$ is too small, below a tolerance tol_1, we set $\nabla_t^2 V = tol_1$ with lines 5 and 6, as suggested in [59], to avoid dividing by a small value. The Newton step is computed as $-\nabla_t V / |\nabla_t^2 V|$. The absolute value of the second derivative is a modification of the original Newton algorithm and is added to ensure that the Newton direction is a descent direction of the function to be minimized, in this case V. The original Newton search direction $-\nabla_t V / \nabla_t^2 V$ is a direction of descent of V if (see [60])

$$-\nabla_t V \frac{\nabla_t V}{\nabla_t^2 V} < 0.$$

Clearly, if $\nabla_t^2 V < 0$, the search direction is not a direction of descent of V, and we need to search in the opposite direction. Luckily, since our problem is one-dimensional ($V(\xi(t))$ is a scalar function of t), it suffices to reverse the sign of the second derivative when it is negative, or simply to take $|\nabla_t^2 V|$.

The Newton step is then scaled through a backtracking line search (lines 9 through 10) [61] to ensure that $V(\xi(t))$ decreases enough after each iteration. The scaling parameter s is first taken as 1. We require a decrease of the function by a percentage κ_1, usually taken between 0.01 and 0.3 (1 and 30%). As long as $V(\xi(t))$ has not decreased enough, s is decreased by a factor β. The parameter β is usually taken between 0.1 for a crude search, and 0.8 for a more accurate search. Figure 5.3 illustrates this procedure. It shows that as long as $V(\xi(\rho + s\Delta\rho))$ is above the line $V(\xi(\rho)) + \kappa_1 \nabla_t V s\Delta\rho$, the parameter s is decreased. The lower blue dashed line represents the case where $\kappa_1 = 1$, and the upper red dashed line shows a smaller decrease $\kappa_1 < 1$.

Once a suitable step has been found, a new iterate of ρ is computed. The algorithm terminates when the change in the value of ρ from an iteration to another becomes too insignificant, or below a value tol_2.

Using Algorithm 1, we can obtain a fast convergence. First, because many time consuming computations can be carried out offline. This is the case for matrices M, L, Γ, Λ and γ. Even the introduction of a backtracking line search, which is generally time consuming, does not slow the algorithm considerably. From our experimental trials, we observed that the line search

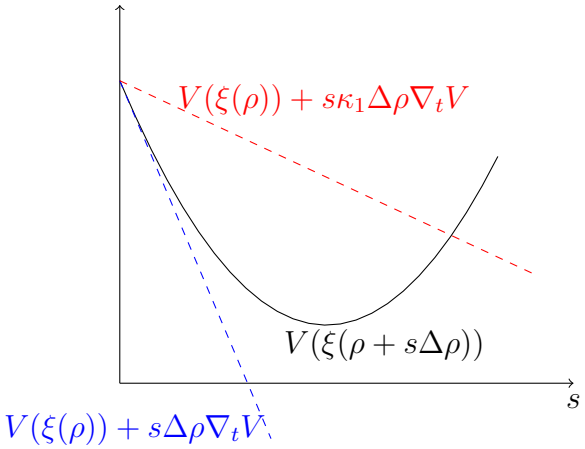


Figure 5.3: Backtracking Line search [61]

is needed at most once in every interval, and becomes unnecessary as we approach the minimizer. Therefore, we noticed through our experiments that the algorithm's execution time is negligible compared to the length of the interval $t_{k+1} - t_k$.

Finally, since the success of a minimization algorithm depends on the convexity of the function to be minimized, we should say a few words about the convexity of the function $V(\xi(t))$. Using the second derivative, we should be able to determine the convexity of $V(\xi(t))$. However, due to the presence of terms such as $L(A-BK)$ and MBK, no conclusion can be made as to the sign of the second derivative. From our observations, it can take positive or negative values on the interval $[t_k, t_{k+1})$. What we do know is that $V(\xi(t))$ goes necessarily through a local minimum before increasing again, and near the minimum $\nabla_t^2 V$ is positive. Problems of convexity arise around t_k, when we are relatively far from ρ_k, and where $\nabla_t^2 V < 0$. As explained earlier, the search direction is a direction of ascent in this case. However, this is not a problem, as we introduced the modification $|\nabla_t^2 V|$ [59].

5.3.2 Root-Finding Stage

The root-finding algorithm that we use is a hybrid between Newton's method and the bisection method. Newton's method has a quadratic convergence rate near the root and would speed up the algorithm. However, we do not know the behavior of the difference $Z(t)$, so to prevent failures, we safeguard the algorithm with bisection, which is a globally convergent method.

Algorithm 1 Minimization Algorithm

procedure MINIMIZATION
  $\rho \leftarrow t_k$
  while $iter \leq MaxIter$ **do**
    compute $\xi(\rho)$, $\nabla_t V(\xi(\rho))$, $\nabla_t^2 V(\xi(\rho))$
    if $\nabla_t^2 V < tol_1$ **then**
      $\nabla_t^2 V \leftarrow tol_1$
    $\Delta\rho \leftarrow -\nabla_t V / |\nabla_t^2 V|$
    $s \leftarrow 1$
    while $V(\xi(\rho + s\Delta\rho)) \geq V(\xi(\rho)) + \kappa_1 \nabla_t V \; s \; \Delta\rho$ **do**
      $s \leftarrow \beta s, \quad \beta \in (0,1), \kappa_1 \in (0, 0.5)$
    $tmp \leftarrow \rho$
    $\rho \leftarrow \rho + s\Delta\rho$
    if $|tmp - \rho| < tol_2$ **then**
      return ρ
    $iter++$

The algorithm works as follows. A pre-processing stage identifies the interval, denoted by $[t_{\min}, t_{\max}]$, in which the root is located. Then, starting from the middle of this interval, a new iterate is computed with Newton's method. If the new iterate is located within the previously identified interval, it is accepted. Otherwise, the Newton iterate is rejected and instead a bisection iterate is computed. The bisection iterate is the mid-point of the search interval.

To apply Newton's algorithm for finding the roots of function Z, we need both the function $Z(t)$ and its first derivative. The expression of $Z(t)$ is

$$Z(t) = W_k e^{-\alpha(t-t_k)} - \xi(t)^T \; \mathcal{P} \; \xi(t), \tag{5.21}$$

where $\xi(t)$ is given by equation (5.5).
The first derivative with respect to time, along the trajectories of $\xi(t)$ is

$$\frac{dZ(t)}{dt} = -W_k \alpha e^{-\alpha(t-t_k)} - \nabla_t V. \tag{5.22}$$

Algorithm 2 describes the procedure to delimit the interval $[t_{\min}, t_{\max}]$ in which the root t_{k+1} is located. We first define two time instants t_1 and t_2 to recursively estimate the search interval. As explained earlier, we are faced with two situations; $Z(\rho_k) < 0$, in which we know that $t_{k+1} < \rho_k$, or $Z(\rho_k) > 0$ and in that case $t_{k+1} \geq \rho_k$.

Algorithm 2 Interval Finding

```
1: procedure PRE-PROCESSING
2:     t_1 ← ρ_k
3:     if Z(t_1) < 0 then
4:         θ ← −κ_2(ρ_k − t_k),  0 < κ_2 ≤ 0.5
5:     else
6:         θ ← κ_2(ρ_k − t_k)
7:     t_2 ← ρ_k + θ
8:     while Z(t_1)Z(t_2) ≥ 0 do
9:         t_2 ← t_2 + θ
10:        if t_2 ≤ t_k then
11:            t_2 ← t_2 − θ, θ ← θ/2
12:            t_2 ← t_2 + θ
13:    t_max ← max(t_1, t_2)
14:    t_min ← t_max − |θ|
```

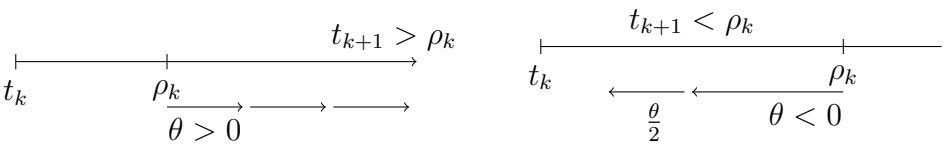
In both cases, we initialize t_1 with the value ρ_k. The procedure starts by picking a parameter θ, that we scale on the time lapse $\rho_k - t_k$. The scaling factor κ_2 is chosen between 0 and 0.5, depending on how crude we want the search to be. The sign of θ depends on the sign of $Z(\rho_k)$. If $Z(\rho_k) < 0$, $\theta < 0$, as in line 4 of Algorithm 2. Otherwise, $\theta > 0$, as in line 6.

Afterward, starting from $t_2 = t_1 + \theta$, t_2 is either increased or decreased until $Z(t_1)$ and $Z(t_2)$ have different signs, indicating that the root has been crossed. Lines 10 and 11 of Algorithm 2 affect only the case where $t_{k+1} < \rho_k$ and where decreasing t_2 can lead to $t_2 \leq t_k$, if the search is too crude. If this happens, we just restore t_2 to its previous value, before it crossed t_k, we decrease θ, and we resume the search.

When the search is over, we set $t_{\max} = \max(t_1, t_2)$, and $t_{\min}$ is simply $t_{\max} - |\theta|$. These operations are performed in lines 13 and beyond.

This process is illustrated in Figure 5.4. Figure 5.4a represents the case $\rho_k < t_{k+1}$, and illustrates how θ is added to ρ_k until we find $t_{\max}$. Figure 5.4b represents the case $\rho_k > t_{k+1}$. It shows how θ is subtracted from ρ_k to locate $t_{\max}$. If by adding θ we get a value less than t_k, we add $\theta/2$ instead, and so on.

Once we locate the root inside an interval, we can start the root-finding process. Algorithm 3 describes the root-finding procedure. It is a slightly



(a) Case $\rho_k < t_{k+1}$ (b) Case $\rho_k > t_{k+1}$

Figure 5.4: Locating the root inside an interval.

modified version of the hybrid Newton-bisection algorithm found in [62]. To make the notations shorter, from now on we refer to $dZ(t)/dt$ as $\nabla_t Z(t)$.

Algorithm 3 Root-Finding Algorithm

procedure NEWTON-BISECTION
  if $Z(t_{\min}) == 0$ **then**
    return $t_{\min}$
  if $Z(t_{\max}) == 0$ **then**
    return $t_{\max}$
  $t \leftarrow (t_{\min} + t_{\max})/2$
  $\Delta t \leftarrow t_{\max} - t_{\min}, \quad \Delta t_{\text{old}} \leftarrow \Delta t$
  compute $Z(t)$, $\nabla_t Z(t)$
  $step \leftarrow \frac{Z(t)}{\nabla_t Z(t)}$
  while $iter \leq MaxIter$ **do**
    if $t_{\min} \geq t - step$ *or* $t_{\max} \leq t - step$ *or* $\frac{|\Delta t_{\text{old}}|}{2} < |step|$ **then**
      $\Delta t_{\text{old}} \leftarrow \Delta t$
      $\Delta t \leftarrow (t_{\max} - t_{\min})/2$
      $t \leftarrow t_{\min} + \Delta t$
    else
      $\Delta t_{\text{old}} \leftarrow \Delta t$
      $\Delta t \leftarrow step$
      $t \leftarrow t - \Delta t$
    if $|\Delta t| < tol_3$ **then** return t
    if $Z(t) > 0$ **then** $t_{\min} \leftarrow t$
    else $t_{\max} \leftarrow t$

The algorithm starts by making sure that neither $t_{\min}$ nor $t_{\max}$ are the root, the procedure is exited if it is the case. Checking whether $t_{\min}$ is a root or not should be performed before the pre-processing, but for the sake of separation, we include it in the root-finding algorithm at this stage. A

maximum number of iterations is the same $MaxIter$ as for the minimization procedure. The current iterate t is initialized as the midpoint of the interval $[t_{\min}, t_{\max}]$.

The variables Δt and Δt_{old} store the current and the former step lengths, respectively. We compute $Z(t)$ and $\nabla_t Z(t)$ in order to compute the Newton step. The condition on line 10 of Algorithm 3 decides whether a Newton step is taken or rejected. If by taking the Newton step we exceed $t_{\max}$ or regress below $t_{\min}$ or if Newton's algorithm is too slow, the Newton step is rejected, and a bisection step is taken instead. Lines 11 to 13 represent a bisection step, whereas lines 15 to 17 represent the case where the Newton step is taken.

After the new iterate is computed, we evaluate $Z(t)$ at that point. If $Z(t)$ is positive, the new iterate is located before the root, and it becomes $t_{\min}$. Otherwise, the current iterate become $t_{\max}$. The algorithm terminates when the change in t between two consecutive iterates is too small, i.e. when the step length becomes smaller than a tolerance tol_3.

5.3.3 Summary of the Self-Triggered Algorithm

The three steps of the self-triggered algorithm, described separately so far, are grouped in the order in which they are called, in Algorithm 4.

Algorithm 4 Self-Triggered Algorithm

```
1: procedure SELF-TRIGGERED
2:     MINIMIZATION(t_k)
3:     return ρ_k
4:     if Z(ρ_k) == 0 then
5:         return ρ_k
6:     PRE-PROCESSING(ρ_k)
7:     return t_min, t_max
8:     NEWTON-BISECTION(t_min,t_max)
9:     return t_{k+1}
```

5.4 Numerical Simulation

5.4.1 A First Example

Consider the following third order LTI system [25],

$$\dot{x}(t) = \begin{bmatrix} 1 & 1 & 0 \\ -2 & 0 & 4 \\ 5 & 4 & -7 \end{bmatrix} x(t) + \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} u(t),$$

with initial state $x_0 = [-2\ 3\ 5]^T$.
The system is unstable with poles at -8.58, 0.58, 2.00. We stabilize the system with a state-feedback control law with feedback gain

$$K = \begin{bmatrix} 8.38 & 26.36 & 10.38 \end{bmatrix},$$

that places the poles at $-1.14 \pm 1.35i$, -5.71. Solving the generalized eigenvalue problem gives $\lambda_{\max} = 2.28$ and

$$P = \begin{bmatrix} 275.7 & 1025.5 & 577.9 \\ 1025.5 & 3840.1 & 2173.5 \\ 577.9 & 2173.5 & 1234.1 \end{bmatrix}. \tag{5.23}$$

We select $\alpha = 2.18\ \mathrm{s}^{-1}$ and $W_0 = 1.3V(x_0)$. We simulate the system's operation for 7 s, with a sampling period $T_s = 10^{-3}$.

For the minimization and root-finding algorithms, we select the maximum number of iterations $MaxIter = 50$. We initialize the minimization algorithm at $t_k + \delta$ with $\delta = T_s = 10^{-3}$. The second derivative $\nabla_{\mathrm{t}}^2 V$ has to be above the tolerance $tol_1 = 10^{-20}$. Algorithm 1 terminates when the change in ρ does not exceed $tol_2 = 10^{-5}$. For the line search, we require a function decrease of 1% at each iteration, or $\kappa_1 = 0.01$. To achieve this decrease, we choose $\beta = 0.35$. For the pre-processing stage, we select $\kappa_2 = 0.25$.

The tolerance tol_3, at which the root-finding algorithm terminates, is set dynamically. Such a choice is motivated by the exponential decrease of $W(t)$, which tends to zero as time tends to infinity. If tol_3 is constant, at some point, $W(t)$ can decrease below this tolerance, and so does $V(\xi(t))$, leading to a small $Z(t)$ that could be mistaken for the root, when there is actually no intersection. Therefore, we index tol_3 on W_k. As long as $W_k > 1$, $tol_3 = 10^{-5}$. When $W_k < 1$, it can be written in the form $W_k = \omega.10^{-\phi}$, where $0 < \omega < 10$ and $\phi \in \mathbb{N}$, and tol_3 is modified to $tol_3 = 10^{-5}.10^{-\phi}$.

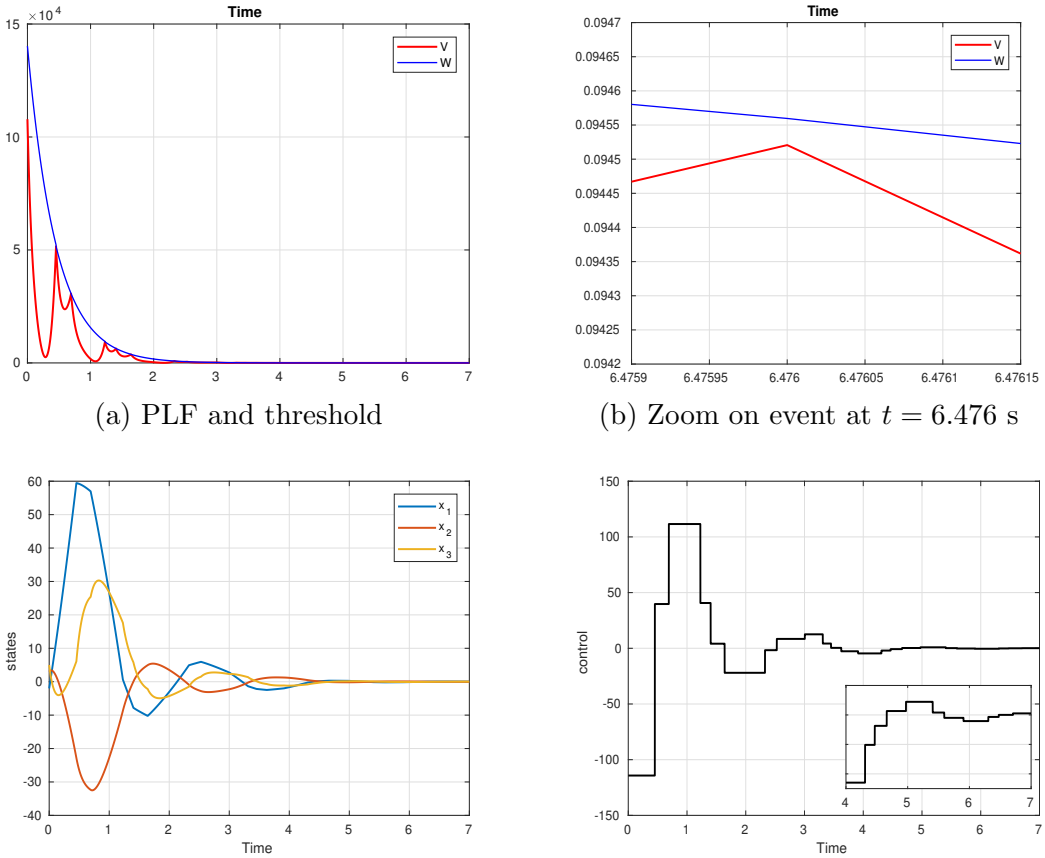
At $t = 0$, we apply the control law $u(t_0) = -Kx_0$ and we compute the instant t_1 using the self-triggered algorithm. The system is then at rest, only maintaining a control value of $u(t_0)$, until the clock signal displays the time t_1. At this point, the operation is repeated.

Figure 5.5a shows the time evolution of the functions $V(\xi(t))$ and $W(t)$. It shows that $V(\xi(t))$ decreases at every intersection with $W(t)$, which proves that the algorithm manages to identify correctly the times at which these events occur, inducing an update of the control law. Even when the two functions approach zero, the intersections are still detected as shown on Figure 5.5b, which singles out an event at $t = 6.476$ s and $W(t) = 0.0948$.

The zoom on the event at $t = 6.476$ s shows that the update of the control law is carried out one time step before the intersection occurs. This is due to the fact that the control can only be updated at multiples of the simulation sampling period T_s. For this reason, when an intersection is predicted somewhere between sampling instants $t = 6.476$ s and $t = 6.477$ s, we update the control law at the earlier instant, $t = 6.476$ s, to prevent the PLF from crossing the threshold. In the event-triggered version, events were detected after they occurred, that is why at time t_k, we used to have $V(x(t_k)) \geq W(t_k)$. The self-triggered implementation can then prevent these digressions.

The three state variables, shown on Figure 5.5c, tend to equilibrium and the system stabilizes around $\pm 5\%$ of its equilibrium value within 6.94 s. The stabilizing control law is shown on Figure 5.5d. This figure shows the uneven distribution of updates in time. Figure 5.5d also includes a zoom on the control in the time interval $[4\text{ s}, 7\text{ s}]$, which emphasizes the asynchronous and scattered nature of the updates, and which is not visible on the larger figure. In 7 s of simulation time, stabilizing the system required 23 updates of the control law.

Table 5.1 lists the first six event times with the corresponding inter-event times $(t_k - t_{k+1})$ and running times of the self-triggered control algorithm. We notice that for our experimental conditions, the algorithm's running time is much smaller than the corresponding inter-event time, allowing the online use of the algorithm. Moreover, the running time decreases as we go further in time, the highest running time being the first call of the algorithm, but this call can be made offline. Eventually, the running time settles around 0.002 s. Additionally, matrices M, L, Λ, Γ and γ are computed offline, and thus do not affect running time.



(a) PLF and threshold

(b) Zoom on event at $t = 6.476$ s

(c) States

(d) Self-triggered control

Figure 5.5: Simulation results of self-triggered control.

From here, we can see the advantages of the self-triggered algorithm, as the algorithm requires only 2 ms, and for the rest of the time until the next update instant, the CPU is either at rest, or free to run other tasks. Conversely, the event-triggered control algorithm has to check the event-triggering conditions at all instants, never really freeing the CPU.

5.4.2 Example of the Case $\rho_k > t_{k+1}$

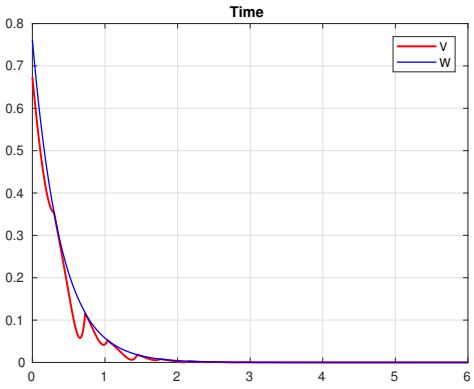
The case where an intersection between $V(\xi(t))$ and $W(t)$ occurs before the PLF reaches its minimum at time ρ_k is rare. From our experiments, we noticed that this case happens when the poles are too close to the imaginary axis, or when their damping is insufficient. It also happens when α is too close to $|\lambda_{\max}|$. We managed to catch an instance of this case when simulating the SISO system in Chapter 1 with $\alpha = 2.566\ \mathrm{s}^{-1}$.

Table 5.1: The first 6 events

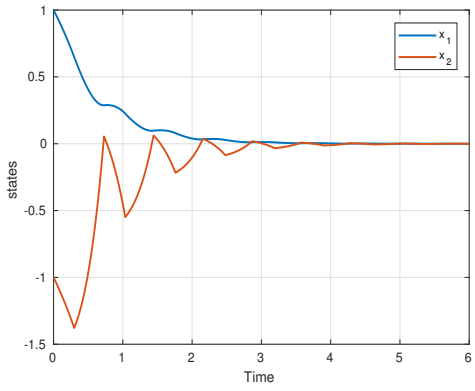
Update time	Inter-event time	Running time
0.453	0.453	0.0481
0.691	0.238	0.0081
1.228	0.537	0.0043
1.403	0.175	0.0029
1.641	0.238	0.0089
2.328	0.687	0.0030

Apart from the value of α, the rest of the experimental conditions mentioned in Chapter 1 are kept the same. The resulting graphs of the PLF and the threshold function are represented in Figure 5.6a. The first event occurs at $t_1 = 0.298$ s, whereas the minimum of $V(\xi(t))$ on the interval $[t_0, t_1)$ occurs at $\rho_0 = 0.3$ s. We can see that the algorithm manages to find the right update time in this case as well. The state variables are shown on Figure 5.6b.

Even though we are dealing with the first update instant, which can be usually computed offline, it is still interesting to compare the algorithm's execution time to the length of the interval $[t_0, t_1)$, especially since in this case, the inter-event time is shorter. The entire self-triggered control algorithm (minimization stage and root-finding stage) converges to the right answer in 0.048 s, which is much less than the length of the interval $t_1 - t_0 = 0.298$ s.



(a) PLF and threshold

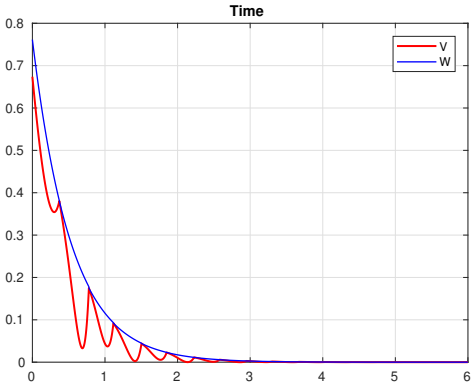


(b) States

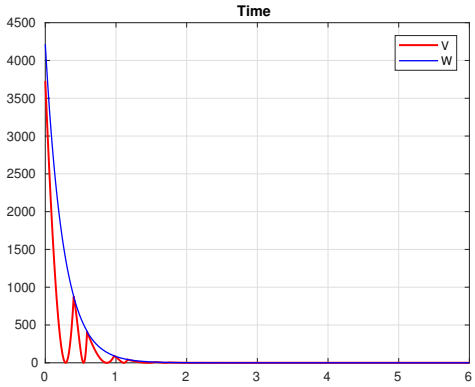
Figure 5.6: Simulation of the SISO system where $t_1 < \rho_0$.

As stated earlier, the case $\rho_k > t_{k+1}$ can be avoided by taking a smaller α, or by moving the eigenvalues further to the left. Figure 5.7a represents

the PLF and the threshold for a smaller α, $\alpha = 1.866\ \text{s}^{-1}$. Figure 5.7b represents the PLF and the threshold when moving the closed-loop poles further to the left to -2 and -3, when they used to be at $-1.33 \pm 1.4i$, while keeping $\alpha = 2.566\ \text{s}^{-1}$. We can see that in both cases $\rho_0 < t_1$.



(a) PLF and threshold for a smaller value of α



(b) PLF and threshold for different closed-loop poles

Figure 5.7: Simulation of the SISO system where $\rho_0 < t_1$.

5.4.3 One-Dimensional Case

We consider the following scalar system

$$\dot{x}(t) = 0.2x(t) + u(t), \tag{5.24}$$

with initial state $x_0 = 1$.

We pick the stabilizing feedback gain $K = 1.5$. We find a Lyapunov-like function with $p = 1.5$, and a threshold with $\alpha = 2.6\ \text{s}^{-1}$, and $W_0 = 2.5$.

Figure 5.8 shows the time evolution of the PLF and the threshold functions, where the updates were carried out using the self-triggered algorithm. The points marked by stars represent the local minima of the PLF predicted by Equation (5.18),

$$\rho_0 = 0.625\ \text{s},$$
$$\rho_1 = 1.479\ \text{s},$$
$$\rho_2 = 2.310\ \text{s}.$$

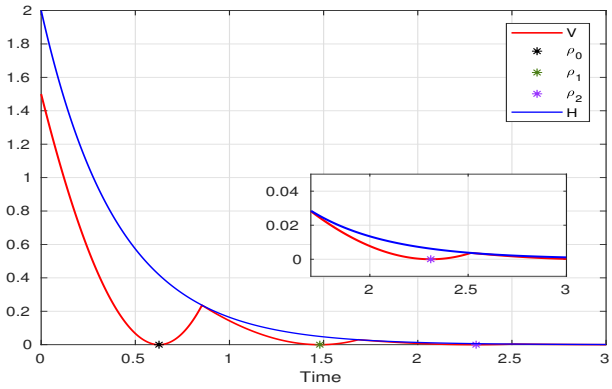


Figure 5.8: The PLF and threshold with the predicted values of ρ_0 and ρ_1.

The predicted values correspond to the registered minima of the PLF. Besides, the plot also confirms the fact that the minima always correspond to $V(x(t)) = 0$ and thus $x(t) = 0$. This also tells us that in the scalar case, we can use the analytical expressions and the minimization algorithm interchangeably, as they both lead to the same results. We have tested both approaches on this system, and both gave the exact same results.

5.5 Conclusion

In this chapter, we have introduced a self-triggered control algorithm that computes the sampling instants of the control law via numerical methods. In the control algorithm, the events correspond to the intersections between the pseudo-Lyapunov function of the system and an upper threshold function. We have shown that an exact solution was very hard, if not impossible, to find, but a numerical solution can be computed through a root-finding algorithm. Root-finding algorithms are divided into global methods and local methods. Global methods require to locate the next execution instant within an interval, which can be an arduous task in our case, as we only know the previous sampling instant. Local methods relax this requirement, but involve the risk of divergence or convergence to the wrong solution, as two consecutive events can be relatively far apart in time.

To counter these problems, we precede the root-finding algorithm by a minimization algorithm that locates the local minimizer of the PLF. This minimizer then serves as a starting point for the root-finding algorithm. The root-finding algorithm is a hybrid between a local and a global algorithm, the global algorithm ensures convergence to the right solution, whereas the

local algorithm ensures speed.

We have tested the self-triggered control algorithm on multiple numerical examples and proved its efficiency. We observed that the algorithm converges to the right solution even if the next sampling takes place before the minimum is reached. However, we maintain that this case is undesirable and should be avoided by a more careful pole placement and choice of rate of decay. The algorithm's running time is very small compared to a single sampling interval, and becomes negligible as we approach the equilibrium point. We have also derived the exact formulas to compute the minimizer in the case of first order systems, where the state is a scalar.

We have also noticed during the experimentation of the algorithm that both the minimizing and root-finding algorithms are very sensitive to the chosen tolerances. We have shown how to set some these tolerances using the properties of the system. The others depend on the LTI system under study, and more importantly on the overall simulation time. As the simulation time gets longer, the PLF and the threshold reach smaller values, and therefore, smaller tolerances are needed.

General Conclusion

In this thesis, we developed event-triggered control techniques for several classes of systems. These classes involve LTI systems, switched linear systems and nonlinear systems. We also introduced methods that successfully address both the problems of stabilizing an unstable system and driving a system to track a reference trajectory. We evaluated each of the event-triggered control methods based on their ability to achieve either stability, reference tracking, or both, as well as their ability to guarantee a minimum inter-event time and to reduce the number of updates of the control law. The latest of these methods, the event-triggered control method for nonlinear systems, has not yet been proven to satisfy the latter requirement, and the proof is left for a future work.

All the event-triggered control methods developed for linear time invariant systems, whether they are destined for the stabilization of an unstable system or for the tracking of a reference trajectory, involve a comparison between a Lyapunov-like function and a threshold. The threshold can be decreasing in time or constant. This approach is particularly suited for this class of systems, as a stabilized linear system readily admits a quadratic Lyapunov function. The suitable threshold function for the PLF can then be easily found. In the case of an exponentially decreasing threshold, the two functions are designed simultaneously by solving a maximum generalized eigenvalue problem. In the case of a constant threshold, we have established the relationship between the maximum desired tracking error and the upper bound on the PLF. The threshold can be computed via this relationship.

Even though the exponentially decreasing threshold was used for stabilization and the constant threshold for tracking, the decreasing threshold can also be used to solve a tracking problem. However, the constant threshold is more suited for tracking problems. This is due, at first, to the fact that the constant threshold ensures only a practical stability, while a stabilization control is most often designed to achieve a stronger form of stability. Secondly,

because the event-triggered control algorithm with a constant threshold δ requires a starting value $V(x_0)$ inside the region enclosed between $V = 0$ and $V = \delta$. While this requirement can be met in the tracking algorithm by setting the initial state of the reference system such that $V(x_0 - x_{r_0}) < \delta$, x_0 being the initial state of the controlled system, we generally have no control on the initial state of a system to be stabilized.

The methods that we present have the advantage of being easy to use and involving very few parameters unlike most available event-triggered control methods. The latter methods often require the selection and fine-tuning of several parameters, and this operation is vital to the successful application of these methods. By contrast, the methods that we propose contain very few parameters most of which are found through an optimization procedure. The only parameters that the user has to select are the closed-loop eigenvalues of the system, and the maximum allowable tracking error for tracking problems.

Because of this, one might argue that the absence of tuning parameters leaves the user with very few degrees of freedom. After all, even the choice of the closed-loop eigenvalues can be constrained by actuator saturation. However, the software solutions for solving the maximum generalized eigenvalue problem offer some tuning parameters. For example, using MATLAB's *'gevp'* function, the user can control the feasibility radius, which is a bound on the magnitude of the solution. Furthermore, we developed these algorithms to increase the inter-event times, but if the user wants shorter intervals, a maximum inter-event time can be imposed. As a result, some of the intervals that are deemed too long are shortened, producing a smoother response.

In the linear case, we gave rigorous proofs of the existence of a minimum inter-event time. We also provided a rough lower bound on the time lapse between two events. This estimation represents a worst case scenario, while we have observed that in reality, much larger intervals separate the events. In most cases, we observed scattered, unevenly separated events. But, we also observed some phenomena that we could not entirely explain. For example, while simulating the SISO system in Chapter 1, we noticed that for this system, the events were almost equidistant. Possible explanations include the linearity of the system which makes for a predictable behavior. Or, it could be that the state aligns itself on one of the eigenvalues, producing a repeated behavior. For the MIMO system, we observed that events occur in clusters, separated by long periods of time, a phenomenon that we could not explain either.

The discrete-time implementation of the event-triggered control algorithms added an extra challenge. While developing these methods, we assumed that the event-triggering conditions were monitored continuously, while in reality they are monitored periodically. For this reason, the intersection between the PLF and threshold cannot be caught by the event generator, which only detects the event after it happened. To best imitate the workings of the continuous-time algorithm, an update of the threshold function was necessary. In the case of a decaying threshold, the value of the threshold was updated to the value of the PLF at the instant at which the event is detected. Since we ensured a fast time decay of the threshold, this operation does not destabilize the system. In the case of a constant threshold, the threshold is updated at the event instant, and then decreased again once the PLF falls back below the original value of the threshold. In this case, the system can be destabilized if other events happen before the threshold resumes its original value. While solving a tracking problem, we suggested a way to update the reference state instead, but this was a costly operation and its use should be weighed against the cost of having no update at all. Updating neither the threshold nor the reference state results in a few additional updates of the control law.

In the discrete-time implementation of the self-triggered control algorithm we have the advantage of knowing the event instant before it happens. Therefore, we have the choice of carrying out the update one sampling instant ahead of the event or one instant after. Even though we chose to update the control law one instant before the event, an update of the threshold was necessary in order to keep the algorithms consistent with the actual implementation and not run the risk of divergence.

For nonlinear systems, we developed an event-triggered control strategy through contraction analysis. This approach has the advantage of allowing us to provide a detailed procedure for building the event-triggering conditions for a large class of nonlinear systems, but is especially easy to use for nonlinear systems affine in the control. Conversely, most of the methods found in the literature rely on a generic Lyapunov function that the user has to search for. When using this method on an example, we noticed a considerable reduction of the number of updates, compared to a periodic implementation, while managing to stabilize the originally unstable system.

However, the development of the event-triggered control method using contraction analysis is still at an early stage and we still need to clarify some of its aspects. For instance, the conditions for the existence of the trans-

formation matrix $\Theta(x)$ have not yet been well established. In addition, the method that we use to find $\Theta(x)$ obliges us to use a specific form of the control law, whereas if we could find another way to determine Θ independently of the control, we could design more general, and better suited control laws. Furthermore, we still need to prove the existence of an inter-event time.

In short, we can conclude that the event-triggered control strategies introduced in this work manage to achieve the goals demanded from a control algorithm, stability and reference tracking, with minimum control effort. These strategies manage to reduce the communications between the CPU and the plant considerably, leading to a more reasonable use of the communication resources. On the other hand, these methods can deteriorate the quality of the response. For this reason, we need to tune the parameters so as to find the right balance between quality of the response and number of updates. We also need to identify the right applications that readily allow for these types of methods, the ones that are not too sensitive to a decrease in the quality of the response.

Future Work

In addition to points that still need clarification and that we mentioned in the conclusion, this work paves the way for several future research options. We list some of these perspectives below:

- **Disturbance rejection**
  In the systems that we have dealt with so far, no disturbance was acting on any part of the plant. However, in reality, control designers are often confronted to systems with external disturbances, measurement noise or model uncertainty. Therefore, we want to find out how the event-triggered control algorithm would perform under the influence of some form of disturbance, and what would be the modifications to bring to the algorithm to ensure disturbance rejection. One possibility could be to add an observer that would measure the effect of the disturbance and integrate this measure into the event-triggered control algorithm. This approach is especially suited for the reference tracking algorithm, as the measure of the disturbance could easily be included into the model of reference system.

- **Improvements on the nonlinear control method**
  The event-triggered nonlinear control algorithm that we introduced offers promising results, both in terms of the reduction of the number of

updates and the ease with which it can be used. However, we still need to write a rigorous proof of the existence of a minimum inter-event time. This task is rendered difficult by the lack of well established results in contraction analysis and by the points that still need clarification. We mentioned earlier the problem of determining the existence of a transformation matrix Θ for each system. The second point involves the possibility of using the control law that we want. In the procedure that we currently use, Θ and a control law have to be computed simultaneously, imposing on us a certain form of the control law. However, we notice that Θ is the solution of the following partial differential equation in space

$$\frac{\partial \Theta(x)}{\partial x} f(x) + \Theta(x) \frac{\partial f(x)}{\partial x} = F\Theta(x),$$

where $\dot{x} = f(x)$ is the closed-loop system dynamics, and F is a user-defined negative definite matrix. This partial differential equation cannot be currently solved as we ignore its boundary conditions. But if we can determine these conditions, we will be able to design our own stabilizing control law, then use this equation to compute $\Theta(x)$ numerically.

- **Robust and nonlinear self-triggered control algorithm**
  The self-triggered control algorithm relies entirely on the system model, and therefore, can be sensitive to model uncertainties and disturbances. Thus, one other perspective of research is to make this algorithm robust to uncertainties. Additionally, the self-triggered control algorithm can be extended to nonlinear systems, either by using Lyapunov methods as in the linear case, or by using contraction analysis.

Bibliography

[1] P. Ellis, "Extention of phase plane analysis to quantized systems," *IRE Transactions on Automatic Control*, vol. 4, pp. 43 – 59, 1959.

[2] C. Draper, W. Wrigley, and J. Hovorka, *Inertial Guidance*. Pergamon Press, Oxford, 1960.

[3] K. J. Åström and B. Bernhardsson, "Comparison of periodic and event based sampling for first-order stochastic systems," in *14th Triennial IFAC World Congress, Beijing, China*, 1999.

[4] W. Heemels, R. Gorter, A. van Zijl, P. van den Bosch, S. Weiland, W. Hendrix, and M. Vonder, "Asynchronous measurement and control: a case study on motor synchronization," *Control Engineering Practice*, vol. 7, no. 12, pp. 1467 – 1482, 1999.

[5] K.-E. Årzén, "A simple event-based PID controller," in *14th Triennial IFAC World Congress, Beijing, China*, 1999.

[6] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[7] D. Lehmann and J. Lunze, "Event-based control: A state-feedback approach," in *2009 European Control Conference*, pp. 1716–1721, IEEE, 2009.

[8] W. M. Heemels, M. T. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 847–861, 2013.

[9] X.-M. Zhang and Q.-L. Han, "Event-triggered $\mathcal{H}_\infty$ control for a class of nonlinear networked control systems using novel integral inequalities," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 4, pp. 679 – 700, 2016.

[10] C. Peng and T. C. Yang, "Event-triggered communication and $\mathcal{H}_\infty$ control co-design for networked control systems," *Automatica*, vol. 49, no. 5, pp. 1326 – 1332, 2013.

[11] A. Ferrara, A. N. Oleari, S. Sacone, and S. Siri, "An event-triggered Model Predictive Control scheme for freeway systems," *51st IEEE Conference on Decision and Control. Maui, Hawaii, USA*, 2012.

[12] A. Selivanov and E. Fridman, "Distributed event-triggered control of diffusion semilinear PDEs," *Automatica*, vol. 68, pp. 344 – 351, 2016.

[13] N. Espitia, A. Girard, N. Marchand, and C. Prieur, "Event-based control of linear hyperbolic systems of conservation laws," *Automatica*, vol. 70, pp. 275 – 287, 2016.

[14] S. Li and B. Xu, "Co-design of event generator and controller for event-triggered control system," in *30th Chinese Control Conference*, 2011.

[15] M. Abdelrahim, R. Postoyan, J. Daafouz, and D. Nesić, "Co-design of output feedback laws and event-triggering conditions for lienar systems," in *53rd IEEE Conference on Decision and Control, Los Angeles, California, USA*, 2014.

[16] S. Tabouriech, A. Seuret, J. M. G. da Silva, and D. Sbarbaro, "Observer-based event-triggered control co-design for linear systems," *IET Control Theory and Applications*, vol. 10, no. 18, pp. 2466 – 2473, 2016.

[17] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," vol. 55, pp. 2030–2042, Sept. 2010.

[18] M. Mazo, A. Anta, and P. Tabuada, "On self-triggered control for linear systems: Guarantees and complexity," in *Proceedings of the European Control Conference*, (Budapest, Hungary), August 2009.

[19] N. Meslem and C. Prieur, "Event-based controller synthesis by bounding methods," *European Journal of Control*, vol. 26, pp. 12–21, 2015.

[20] X. Wang and M. Lemmon, "On event design in event-triggered feedback systems," *Automatica*, vol. 47, pp. 2319–2322, 2011.

[21] S. Boyd and L. E. Ghaoui, "Method of centers for minimizing generalized eigenvalues," *Linear Algebra and its Applications*, vol. 188 - 189, pp. 63 – 111, 1993.

[22] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in convex programming*. SIAM, 1994.

[23] A. Seuret, C. Prieur, and N. Marchand, "Stability of nonlinear systems by means of event-triggered sampling algorithms," *IMA Journal of Mathematical Control and Information*, vol. 31, no. 3, pp. 415–433, 2014.

[24] M. Heymann, F. Lin, G. Meyer, and S. Resmerita, "Analysis of zeno behaviors in hybrid systems.," in *41st IEEE Conference on Decision and Control, Las Vegas, Nevada USA*, 2002.

[25] R. C. Dorf and R. Bishop, *Modern control systems*. Pearson Education, Inc., 2014.

[26] MathWorks, "Yaw damper design for a 747® jet aircraft." http://fr.mathworks.com/help/control/examples/yaw-damper-design-for-a-747-jet-aircraft.html.

[27] N. Marchand, S. Durand, and J. F. G. Castellanos, "A general formula for event-based stabilization of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 5, pp. 1332–1337, 2013.

[28] D. Liberzon, *Switching in systems and control*. Birkhäuser, 2003.

[29] R. C. Wong, H. A. Owen, and T. G. Wilson, "A fast algorithm for the time-domain simulation of switched-mode piecewise-linear systems," in *IEEE Power Electronics Specialists Conference, Gaithersburg, MD, USA*, 1984.

[30] C. M. Wolf and M. W. Degner, "Computationally efficient event-based simulation of switched power systems and AC machinery," in *IEEE Conference on Energy Conversion Congress and Exposition (ECCE)*, 2016.

[31] Z. Sun and S. S. Ge, *Switched linear systems control and design*. Springer, 2005.

[32] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems*, vol. 19, no. 5, pp. 59–70, 1999.

[33] W. P. Dayawansa and C. F. Martin, "A converse Lyapunov theorem for a class of dynamical systems which undergo switching," *IEEE Transactions on Automatic Control*, vol. 44, no. 4, pp. 751–760, 1999.

[34] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308 – 322, 2009.

[35] P. Li, Y. Kang, Y.-B. Zhao, and J. Zhou, "Dynamic event-triggered control for networked switched linear systems," in *36th Chinese Control Conference, Dalian, China*, 2017.

[36] A. Girard, "Dynamic triggering mechanisms for event-triggered control," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1992–1997, 2015.

[37] Y. Qi, P. Zeng, and W. Bao, "Event-triggered and self-triggered H_∞ control of uncertain switched linear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems.*, vol. To appear, 2018.

[38] T.-F. Li and J. Fu, "Observer-based dynamic output event-triggered control of switched systems," in *36th Chinese Control Conference, Dalian, China*, 2017.

[39] L. I. Allerhand and U. Shaked, "Robust state-dependent switching of linear systems with dwell time," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 994–1001, 2013.

[40] V. Montagner, V. Leite, R. Oliveira, and P. Peres, "State feedback control of switched linear systems: an LMI approach," *Journal of Computational and Applied Mathematics*, vol. 194, pp. 192–206, 2006.

[41] N. Otsuka and S. Sekiguchi, "Quadratic stabilization and transit problems for switched linear systems via state feedback," in *Third International Conference on Mathematics and Computers in Sciences and in Industry, Chania, Greece*, August 2016.

[42] G. Wang, Q. Zhang, C. Yang, and C. Sue, "Stabilization of continuous-time randomly switched systems via the LMI approach," *Applied Mathematics and Computation*, vol. 266, pp. 527 – 538, 2015.

[43] S. Durand, B. Boisseau, J.-J. Martinez-Molina, N. Marchand, and T. Raharijaona, "Event-based LQR with integral action," in *2014 IEEE Emerging Technology and Factory Automation (ETFA)*, (Barcelona, Spain), pp. 1–7, IEEE, Sept. 2014.

[44] P. Tallapragada and N. Chopra, "On event triggered trajectory tracking for control affine nonlinear systems," in *50th IEEE Conference on*

Decision and Control and European Control Conference (CDC-ECC), (Orlando, Florida, USA), pp. 5377–5382, IEEE, Dec. 2011.

[45] P. Tallapragada and N. Chopra, "On event triggered tracking for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 58, pp. 2343–2348, Sept. 2013.

[46] M. Shao, F. Hao, and X. Chen, "Event-triggered and self-triggered asymptotically tracking control of linear systems," in *32nd Chinese Control Conference (CCC)*, (Xi'an, China), pp. 6650–6655, IEEE, July 2013.

[47] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, issue 6, pp. 683 – 696, June 1998.

[48] W. Lohmiller and J.-J. E. Slotine, "Contraction analysis: A practical approach to nonlinear control applications," *Proceeding of the 1998 IEEE international conference on control applications, Trieste, Italy*, 1998.

[49] W. Lohmiller, *Contraction analysis of nonlinear systems*. PhD thesis, Massachusetts Institute of Technology, 1999.

[50] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 3^{rd} ed., 2000.

[51] S. Durand, J. F. Guerrero-Castellanos, and R. Lozano-Leal, "Self-triggered control for the stabilization of linear systems," in *9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2012.

[52] M. Velasco, P. Martí, and E. Bini, "Optimal-sampling-inspired self-triggered control," in *International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, 2015.

[53] M. Kishida, "Event-triggered control with self-triggered sampling for discrete-time uncertain systems," *IEEE Transactions on Automatic Control*, 2018.

[54] K. Kobayashi and K. Hiraishi, "Self-triggered optimal control of linear systems using convex quadratic programming," in *AMC2014-Yokohama*, (Yokohama, Japan), March 14 - 16 2014.

[55] X. Wang and M. D. Lemmon, "Self-triggered feedback control systems with finite-gain $\mathcal{L}_2$ stability," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 452–467, 2009.

[56] X. Wang and M. D. Lemmon, "Self-triggered feedback systems with state-independent disturbances," in *2009 American Control Conference*, (St. Louis, MO, USA), June 2009.

[57] K. Kobayashi and K. Hiraishi, "Self-triggered model predictive control with delay compensation for networked control systems," in *38th Annual Conference on IEEE Industrial Electronics Society*, 2012.

[58] E. Henriksson, D. E. Quevedo, E. G. W. Peters, H. Sandberg, and K. H. Johansson, "Multiple-loop self-triggered model predictive control for network scheduling and control," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2167–2181, 2015.

[59] W. Murray, "Newton-type methods," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[60] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*. Elsevier Academic Press, 1986.

[61] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, New York, 2014.

[62] W. H. Press, S. Teukolsky, B. P. Flannery, and W. T. Vetterling, *Numerical recipes : the art of scientific computing*. Cambridge University Press, third edition ed., 2007.