Discretized Bounded Sliding Mode Control for Simulation of Differential-Algebraic Systems

Farshad Rum

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

October 2004

Your file Votre référence
ISBN: 0-494-04427-6
Our file Notre référence
ISBN: 0-494-04427-6

ABSTRACT

Discretized Bounded Sliding Mode Control for Simulation of

Differential-Algebraic Systems

Farshad Rum

Sliding mode control has recently proved to be a highly effective method for state space modeling of differential-algebraic equation systems (DAEs). Sliding control realizations have great potential for simulation since they allow more computationally efficient robust modeling approximations to be constructed for DAE systems. However, efficient discretization of such methods poses a significant problem due to the well known chattering phenomena that often occurs due to limited computational bandwidth. While some errors are inevitable due to limited bandwidth, the chattering phenomenon can be reduced by minimizing the frequency at which the system crosses the sliding surface. In this work, we find relations between controller parameters, error bounds, and the crossing frequency. They are then used to synthesize efficient discretized sliding mode realizations that optimize crossing frequency and the associated controller sampling period. Together, these results form an efficient discretized bounded sliding mode control approach for simulation of differential-algebraic systems.

Table of contents

List of figures

1 Introduction

1.1 Motivation and literature review

Differential-algebraic equation (DAE) systems are described by a set of differential equations and algebraic constraints. They are also known as descriptor systems [1]. Examples of such problems range from dynamical systems derived from the method of Lagrange multipliers [2], multibody systems [3], object oriented simulation [4], and process control [5]. One of the main difficulties for control and simulation of DAE systems is that they are not expressed in an explicit state space form.

Solution of DAEs has gained considerable attention from a numerical perspective. Backward differentiation formulas [6], BDF, have proven to work effectively, mainly because of their stability properties even when time steps are relatively large. However this time step, which is imposed by the fastest evolving part in the entire system, is used to solve a set of nonlinear equations as big as the size of the slow and fast dynamics combined. Additionally since these methods are implicit we have to iteratively solve this large set of nonlinear equations to advance one step in time. Since the number of iterations required is unknown the method is not deterministic and not well suited for real time simulation.

An important property of BDF methods is their ability to stably solve stiff sets of equations even when slow and fast motions are so tightly intermingled that it is hard (if not impossible) to make them decoupled. Nevertheless the direct drawback is that they

are not well suited to multi rate solution and therefore the large set of equations mentioned above has to be solved simultaneously. In the case of DAEs if differential equations are not stiff and the only fast dynamics is due to algebraic constraints there is no benefit obtained from the above mentioned property.

The shortcomings of BDF methods previously noted make them undesirable for real time simulations. An alternative and effective approach for real time DAE simulation based on sliding control [7] has been developed that allows the systematic construction of explicit state space approximations. The state space approximations can be simulated explicitly without iteration and therefore can be used in real time simulations. The approximations, referred to as realizations, can be developed for a large class of nonlinear DAEs. This approach based on boundary layer sliding control is known as the singularly perturbed sliding manifold approach (SPSM) to DAE realization [8]. Chapter 2 gives a short review of this method followed by a detailed example application [9] in Appendix A.

The main drawback of the SPSM method is that it normally results in stiff singularly perturbed dynamics that are difficult to simulate. An alternative approach is to use sliding control without boundary layer approximation. In this case discretization of such methods poses a significant problem due to the well known chattering phenomena that often occurs due to limited sampling rate. It limits us from exactly following the sliding manifold. This problem is due to the underlying assumption in continuous sliding control theory that there is infinite sampling rate available.

An alternative approach based on discrete time sliding mode (DTSM) control theory [10] has been designed to eliminate chattering and restrict the motion to the sliding

surface at fixed sample times. Although this method works well for discrete time systems it faces difficulties when applied to continuous time systems. Most of these issues are related to the fact that the discrete time model is only an approximation to the actual continuous system. In between sampling intervals control activity may cause significant changes in the continuous system, which may not be represented by the discrete time approximation.

Zhao and Utkin [11] have developed a well suited method for discretization of sliding control that contains the errors in between sample times (see figure 1.1 (a)). It provides a discrete controller directly applied to the continuous time system that ensures motion on the sliding manifold at sample times and also guarantees the motion is kept within specified bounds between control updates. This method exploits the allowable amount of error by continuously monitoring the sliding variable deviations. As a result of monitoring the motion, sampling periods are no longer constant. While this method is highly effective for control applications, the implementation proceeds by iteration. In order to advance one sampling period it solves a set of nonlinear equations and monitors the motion during every iteration to check for error bound violations until none occur. This makes it difficult to apply the approach for simulation of DAE realizations since iterative methods cannot normally guarantee deterministic execution times required by real time simulations.

1.2 Thesis contributions and outline

Discretization methods that use a pre-specified sampling rate normally result in varying error bounds, whereas methods that guarantee error bounds lead to a variable sampling rate. In the first case a study on error bounds becomes relevant, while in the second case we need to study the controller sampling period required to achieve error bounds within the available sampling bandwidth. The new approach presented in this thesis for discretized sliding control can be implemented in either manner depending on whether monitoring the sliding variables between sample times is desirable or not. The control is updated when any of the monitored sliding variables hits the boundary layer surface (figure 1.1(b)). This proposed method is non iterative which makes it particularly well suited for simulation of sliding control DAE realizations compared to the previous iterative approaches.

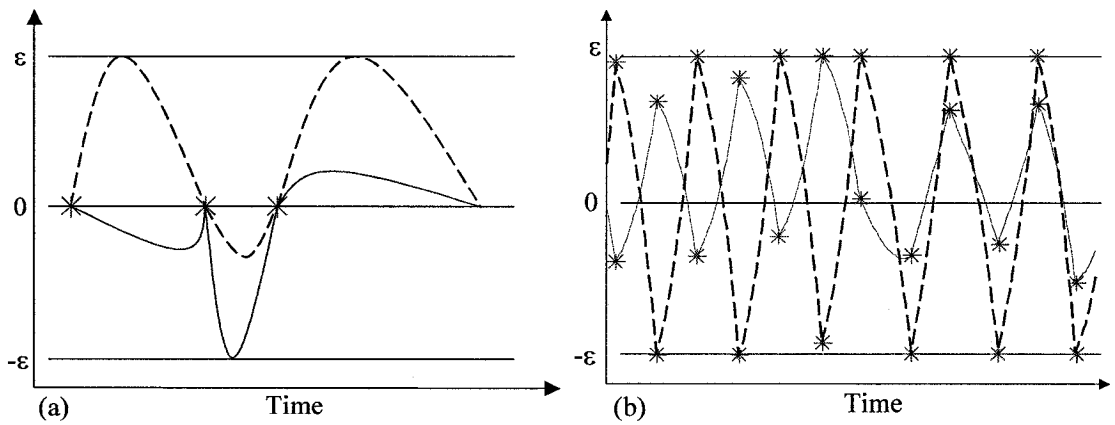


Figure 1.1. Typical s–trajectories for (a) Zhao and Utkin (1996) and (b) our approach.

For our approach we find the minimum controller sampling period necessary to guarantee a desired error bound. In order to synthesize a controller with an optimal sampling period we establish bounds on the gains necessary to ensure a crossing condition. The results are then combined to establish a guaranteed minimum bound on the sampling period in terms of the controller parameters. Optimization of this worst case bound yields the maximum permissible controller sampling period. During simulation of sliding realizations, computational effort is almost directly proportional to sampling rate. Therefore, computational effort will be minimized by the proposed method. It will also help reduce chattering and excitation of unmodeled dynamics for hardware in the loop simulations, since the crossing frequency will also be minimized.

Another major issue is robustness. This property is desirable even in simulation problems where modelling approximations correspond to uncertainty. In addition to protecting against round off errors, robustness can help systematic reduction of computations involved in updating control inputs. Robustness implies that the realization will converge even when some computational terms are intentionally neglected. Gordon [7] has previously developed robustness conditions to address this problem for continuous time sliding control. In this thesis, the analogous robustness conditions are developed for discretized sliding control realizations. It is shown that key robustness properties are preserved by our approach with some additional requirements.

The discretized sliding control approach for simulation of DAEs is presented in chapter 3. Design and Optimization issues are investigated in chapter 4, followed by investigation of the reaching phase dynamics. In chapter 5 the new approach is applied to simulation of a double pendulum DAE system. Together, these results form an efficient

discretized bounded sliding mode control (DBSMC) approach for simulation of differential-algebraic systems.

2 Background on Realization of DAE systems using sliding mode control

An overview of sliding mode control based DAE realization is presented in this chapter. Consider the following semi-explicit nonlinear DAE system

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{z}) \tag{2.1}$$

$$\mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{z}) \tag{2.2}$$

where $\mathbf{x} \in \Re^n$, $\mathbf{z} \in \Re^m$, $\mathbf{f} : \Re \times \Re^n \times \Re^m \to \Re^n$, and $\mathbf{g} : \Re \times \Re^n \times \Re^m \to \Re^m$.

One of the main difficulties with control and simulation of DAE systems is that they are not expressed in an explicit state space form required by most control and simulation methods. In previous investigations it has been found that sliding mode control can be effectively used to develop explicit state space approximations (realizations) of DAE systems [8]. This approach based on singularly perturbed sliding manifolds is referred to as SPSM hereafter. Assuming that equations (2.1-2.2) are sufficiently differentiable and possess a solution for **x** and **z** with consistent initial conditions, an important property of a DAE system known as the index can be defined [6].

Definition 2.1 The minimum number of times that all or part of the constraint equations (2.2) have to be differentiated with respect to time in order to solve for $\dot{\mathbf{z}}$ as a continuous function of t, **x**, and **z** is the *index* of the DAE (2.1,2.2).

The above definition from the field of numerical analysis happens to be closely related to the notion of output feedback linearization [12] in the area of systems and control. If we consider $\dot{\mathbf{z}}$ as the input to the system represented by equation (2.1) and take the constraints (2.2) as outputs of the system, the solution of a DAE becomes a nonlinear control problem (figure 2.1). This opens the door to a rich variety of nonlinear control schemes that can be used to address the associated DAE realization problem given by the following equation

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(t, \mathbf{x}, \mathbf{z}) \\ \dot{\mathbf{z}} &= \mathbf{v} \\ \mathbf{w} &= \mathbf{g}(t, \mathbf{x}, \mathbf{z}) \end{aligned} \tag{2.3}$$

It can be shown that a sliding controller applied to (2.3) which forces the outputs to small bounded errors will yield a close approximation to the DAE system (2.1-2.2) [8]. This will result in an explicit state space realization that can be used with a large class of simulation and control methods. Furthermore, it can be shown that if the DAE system is locally exponentially stable then the internal dynamics of the sliding control realization will also be stable [7]. For the methods proposed in this thesis it will be assumed that this condition is satisfied so that the realization is stable.

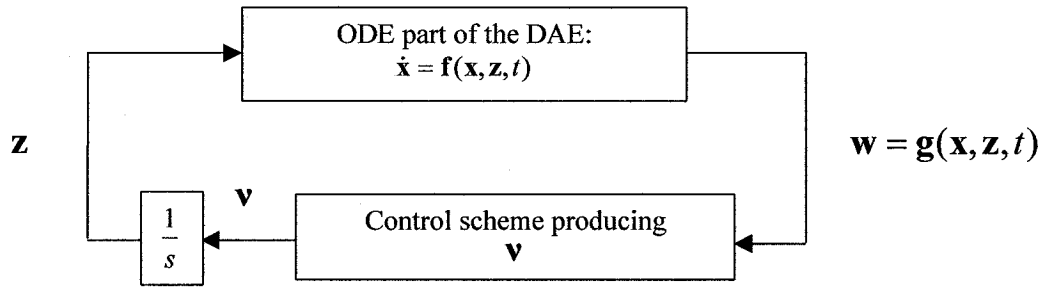


Figure 2.1. Block diagram of the control approach to DAE realization.

The SPSM method consists of a sliding controller that forces the system (2.3) to sliding surfaces of the form

$$s_i = \left(\mu_i \frac{d}{dt} + 1\right)^{r_i - 1} w_i \quad , \quad \mu_i > 0 \qquad \text{for } 1 \le i \le m \tag{2.4}$$

The equation $\dot{\mathbf{z}} = \mathbf{v}$ is then used to calculate $\mathbf{z}$.

The sliding surfaces given by equation (2.4) are characterized by a positive constant μ_i and the corresponding index, r_i for each constraint. It is assumed that the DAE has a vector index [7], $\mathbf{r} = [r_1, ..., r_m]^T$, which is analogous to the vector relative degree [12] of (2.3). The index of a constraint is defined as the minimum number of times it has to be differentiated with respect to time for all or some parts of $\dot{\mathbf{z}}$ to appear. Consequently elements of the $\mathbf{z}$ vector have to appear in the $r_i - 1^{th}$ time derivative so that $w_i^{(r_i-1)} = w_i^{(r_i-1)}(t, \mathbf{x}, \mathbf{z})$. Defining this time derivative as $\Omega_i = w_i^{(r_i-1)}$ for each constraint, one additional derivative of $\boldsymbol{\Omega}$ yields

$$\dot{\boldsymbol{\Omega}} = \frac{\partial \boldsymbol{\Omega}}{\partial \mathbf{z}} \dot{\mathbf{z}} + \frac{\partial \boldsymbol{\Omega}}{\partial t} + \frac{\partial \boldsymbol{\Omega}}{\partial \mathbf{x}} \mathbf{f} = \mathbf{J}_{\Omega} \mathbf{v} + \boldsymbol{\beta}(t, \mathbf{x}, \mathbf{z}) \tag{2.5}$$

with $\mathbf{J}_{\Omega} = \frac{\partial \boldsymbol{\Omega}}{\partial \mathbf{z}}$, $\boldsymbol{\beta}(t, \mathbf{x}, \mathbf{z}) = \frac{\partial \boldsymbol{\Omega}}{\partial t} + \frac{\partial \boldsymbol{\Omega}}{\partial \mathbf{x}} \mathbf{f}$ and $\mathbf{v} = \dot{\mathbf{z}}$.

Note that the vector index is defined to exist if the Jacobian matrix $\mathbf{J}_{\Omega}$ is nonsingular in a domain around the DAE solution.

Recalling that $\mathbf{v}(t)$ is our control input vector, nonsingularity of the Jacobian matrix allows us to force the motion in a desired direction since

$$\dot{\mathbf{s}}(t) = \mathbf{J}_S \mathbf{v} + \boldsymbol{\alpha}(t, \mathbf{x}, \mathbf{z}) \tag{2.6}$$

where

$$\begin{aligned} \mathbf{J}_S &= \operatorname{diag}\left(\mu_i^{r_i - 1}\right) \mathbf{J}_{\Omega} \\ \alpha_i &= \mu^{r_i - 1} \beta_i(t, \mathbf{x}, \mathbf{z}) + \sum_{j=1}^{r_i - 1} \frac{(r_i - 1)! \mu_i^{r_i - j - 1}}{(r_i - j - 1)! \, j!} w_i^{(j)} \quad i = 1, \ldots, m \end{aligned} \tag{2.7}$$

Selecting $\mathbf{v} = \mathbf{J}_S^{-1}\left(\dot{\mathbf{s}}_d(t) - \boldsymbol{\alpha}(t, \mathbf{x}, \mathbf{z})\right)$ yields any desired motion, $\dot{\mathbf{s}}_d$, with respect to sliding surfaces and enables us to make them attractive. The proposed sliding control based realization has the form

$$
\begin{aligned}
&\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x},\mathbf{z}) \\
&\dot{\mathbf{z}} = \mathbf{v} \\
&\mathbf{w} = \mathbf{g}(t,\mathbf{x},\mathbf{z}) \\
&\hat{\mathbf{J}}_s \mathbf{v} = -\hat{\boldsymbol{\alpha}} - \mathbf{K}\,\mathrm{diag}(\mathrm{sign}(s_i)) \\
&s_i = \left(\mu_i \frac{d}{dt} + 1\right)^{r_i - 1} w_i \quad , \quad \mu_i > 0 \qquad \text{for } 1 \le i \le m
\end{aligned}
\tag{2.8}
$$

In many applications numerical approximations and parametric uncertainties lead to imperfect approximations of $\boldsymbol{\alpha}$ and $\mathbf{J}_\Omega$ that we denote by $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{J}}_\Omega$. In simulation problems these terms can be computed exactly, however, the computations are often costly and time consuming. As a result it is often desirable to incorporate approximations $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{J}}_\Omega$ in the DAE realization (2.8). In section 3.4 we present a result (theorem 3.3) that shows how we can systematically stabilize the system in the presence of these approximations. There is an additional advantage associated with approximating the Jacobian matrix $\mathbf{J}_\Omega$ for large-scale systems. It will be illustrated in section 3.4 that it is possible to avoid inverting the original $\mathbf{J}_\Omega$, which is potentially very large and sparse, by using an approximate inverse that is more efficient to compute denoted by $\hat{\mathbf{J}}_\Omega^{-1}$.

3 Discretized bounded sliding mode control (DBSMC) for realization of DAE systems

After a sliding control based realization has been designed, efficient discretization for simulation purposes remains a significant issue due to the potential problem of chattering and the associated simulation errors. To address this problem a new approach for efficient realization of DAE systems based on discretized bounded sliding mode control (DBSMC) is developed in this chapter. The proposed method is a sample and hold discretization of the continuous time sliding control that guarantees the distance of motion from the sliding surfaces to be within pre specified bounds. Control updates are evaluated at certain instances to force the motion back to desired sliding layers and are held fixed until the next sample time.

Experience has shown that most of the computational effort in simulating the DAE realization (2.8) is used in computing the input $\mathbf{v}$. Therefore, the proposed approach will focus on maximizing the controller sampling period (or minimizing the frequency). We carry out all analysis in continuous time and make no assumptions on how the system dynamics associated with $\mathbf{x}$ and $\mathbf{z}$ are discretized. The user has the flexibility to discretize the dynamics in an appropriate manner (using Euler integration or some other method). In practice, the simulation time step for the fast dynamics of $\mathbf{z}$ can be the same as the controller sampling period, and the time step for the slow dynamics of $\mathbf{x}$ can be significantly larger.

In the proposed scheme we will evidently obtain the largest sampling period if we postpone controller updates until a sliding variable error grows out of its bounds, which

can be determined by monitoring the sliding surface variables in between sample times (see figure 1 (b)). This results in DBSMC with monitoring which yields a variable period discretization approach. Alternatively, we can make updates at a sufficiently small constant sampling period to avoid the monitoring cost. In a best case scenario, the sampling period of this approach might be close to the average sampling period of DBSMC with monitoring. In some cases it might have less computational cost in situations when monitoring is computationally costly. It will be shown that both approaches are analogous to minimizing the frequency at which the system crosses the sliding surface, which will also act to reduce the effects of chattering. The two monitoring approaches given in sections 4.1 and 3.2 are first presented as general sliding control discretization methods. They are then specialized to discretization of sliding control realizations in section 3.3, resulting in a new approach for discretized DAE realization. The robustness conditions for the new method are then determined in section 3.4.

3.1 Discretized bounded sliding mode control with monitoring

Consider a previously designed sliding controller as follows

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t, \mathbf{u})$$
$$u_i(\mathbf{x}, t) = \begin{cases} u_i^+(\mathbf{x}, t) \text{ when } s_i(\mathbf{x}) > 0 \\ u_i^-(\mathbf{x}, t) \text{ when } s_i(\mathbf{x}) < 0 \end{cases} \quad \text{for } i = 1, \ldots, m. \tag{3.1}$$

where $\mathbf{x} \in \Re^n$ is the state vector and $\mathbf{u} = [u_1,...,u_m]^T \in \Re^m$ is the vector of control inputs. Each $s_i(\mathbf{x})$ is a sliding surface and all control functions $u_i^+(\mathbf{x},t)$ and $u_i^-(\mathbf{x},t)$ are continuous. Assuming that a certain amount of deviation is allowed around the sliding surfaces

$$|s_i(\mathbf{x})| \le \varepsilon_i, \text{ for } i = 1,...,m. \tag{3.2}$$

the discretized bounded sliding mode control with monitoring is then given by

$$\begin{aligned} &\dot{\mathbf{x}} = f(\mathbf{x},t,\mathbf{u}) \\ &u_i(\mathbf{x},t) = \text{ZOH}\left(\begin{cases} u_i^+(\mathbf{x},t) + \rho_i^+(\mathbf{x},t) \text{ when } s_i(\mathbf{x}) > 0 \\ u_i^-(\mathbf{x},t) + \rho_i^-(\mathbf{x},t) \text{ when } s_i(\mathbf{x}) < 0 \end{cases} \text{for } i = 1,...,m \text{ if } \exists i : |s_i(\mathbf{x})| \ge \varepsilon_i\right) \end{aligned} \tag{3.3}$$

where ZOH(.) is the familiar zero order hold operator and $\rho_i^+(\mathbf{x},t)$ and $\rho_i^-(\mathbf{x},t)$ are continuous functions necessary to meet an extra crossing condition detailed in the following paragraphs.

In classical sliding control the main objective is forcing the motion onto sliding manifolds and thereby reducing the dynamics to a simple linear system. Conditions of the form $s(t)\dot{s}(t) \le 0$ must be ensured for all the sliding variables to ensure that the motion is locally attractive to the desired surfaces. In this thesis we additionally consider a new issue referred to as *discretization disturbance*. It is a measure of the rate of change in the system under control from the time the last control signal was computed.

Consequently, system deviations become more prominent as one tries to enlarge the sampling period. Figure 3.1 shows a typical scenario where the motion has been initially directed towards the sliding surface and later deviated by an opposing discretization disturbance.

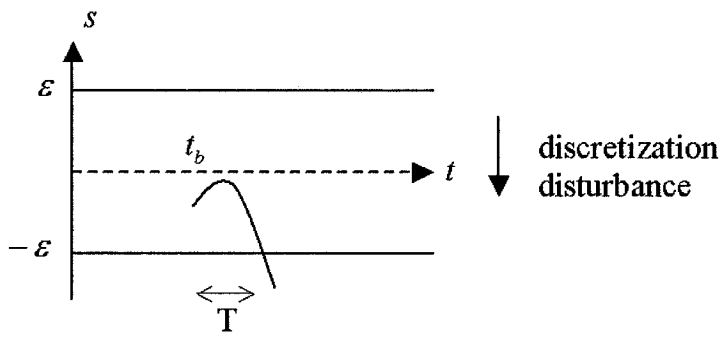


Figure 3.1. S–trajectory initially moving towards the sliding surface with a large opposing discretization disturbance.

This kind of situation can especially become a problem when the system is close to a boundary layer (see figure 3.2) and a very small sampling period is imposed.

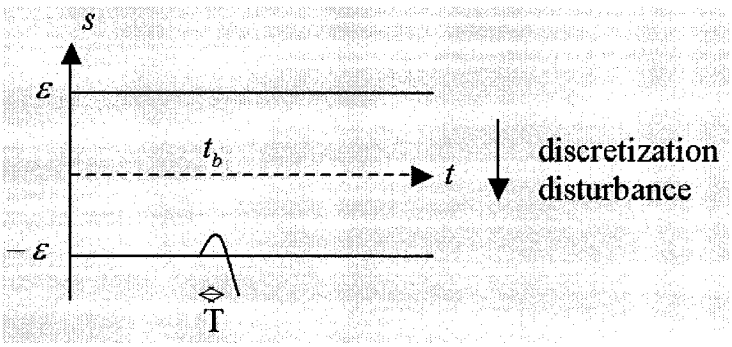


Figure 3.2. An initially attractive sliding surface may lead to very small time steps.

In order to avoid such scenarios we need to enforce an extra condition on the motion in each interval. The motion is not only initially forced to move towards the sliding surface (classical $u_i^+(\mathbf{x},t)$ and $u_i^-(\mathbf{x},t)$ terms), but it is also provided enough input so that it can exit from the other side of its boundary layer (figure 3.3). We call this extra requirement the *crossing condition* and satisfy it through supplementary $\rho_i^+(\mathbf{x},t)$ and $\rho_i^-(\mathbf{x},t)$ control terms (see theorem 4.1 for a sufficient amount of control inputs). This additional criterion ensures a minimum amount of displacement in motion, namely ε_i, before another update becomes necessary. As a result the crossing condition allows us to exploit the maximum allowable amount of deviations to produce larger sampling periods.

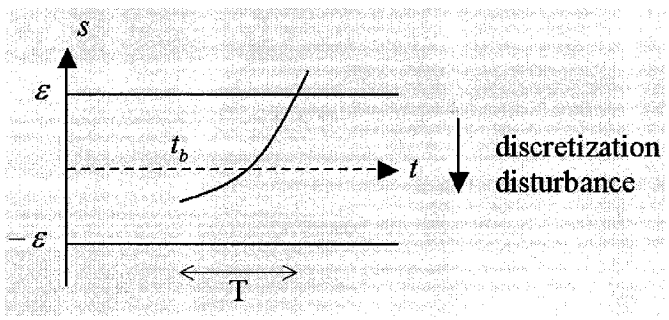


Figure 3.3. Motion supplied with enough input to exit from the other side of the boundary layer.

Given the finite rate of motion with respect to sliding surfaces and the minimum amount of displacement, ε_i required by the crossing condition there is a corresponding minimum amount of time required for each interval. We refer to this lower bound imposed by each sliding surface as T_i^{min}. It is evident that the lower bound on the sampling period is the smallest of these values, $T^{min} = \min(T_i^{min})$ for $i = 1,...,m$. If this sampling period is within the available control bandwidth we can ensure that once the motion enters a boundary

layer it will remain inside thereafter. Finally, one should note that no monitoring of motion is necessary for each sliding surface variable before its corresponding T_i^{min}.

3.2 Discretized bounded sliding mode control without monitoring

By guaranteeing the crossing condition in a DBSMC design we know that starting anywhere inside boundary layers we are still within bounds for $T \le \min(T_i^{min})$ $i = 1,...,m$. Therefore, the following constant rate control scheme can guarantee bounds for all motions.

$$\dot{\mathbf{x}} = f(\mathbf{x},t,\mathbf{u})$$
$$u_i(\mathbf{x},t) = \mathrm{ZOH}\left(\begin{cases} u_i^+(\mathbf{x},t) + \rho_i^+(\mathbf{x},t) \text{ when } s_i(\mathbf{x}) > 0 \\ u_i^-(\mathbf{x},t) + \rho_i^-(\mathbf{x},t) \text{ when } s_i(\mathbf{x}) < 0 \end{cases} \text{ for } i = 1,...,m \text{ if } t \ge t^{last} + T^{min} \right) \quad (3.4)$$

Here t^{last} is the time of the last control update and $T^{min} = \min(T_i^{min})$ for $i = 1,...,m$. Note that this corresponds to a simple sample and hold discretization of continuous sliding control with extra gains to ensure that the sampling rate is small enough to keep us inside boundary layers.

3.3 DBSMC approach for realization of DAE systems

In this section we apply the approaches of sections 3.1 and 3.2 to realization of DAEs resulting in the DBSMC approach to DAE realization. Following the procedure outlined in those sections we obtain the block diagram shown in figure 3.4 for the DBSMC control configuration.

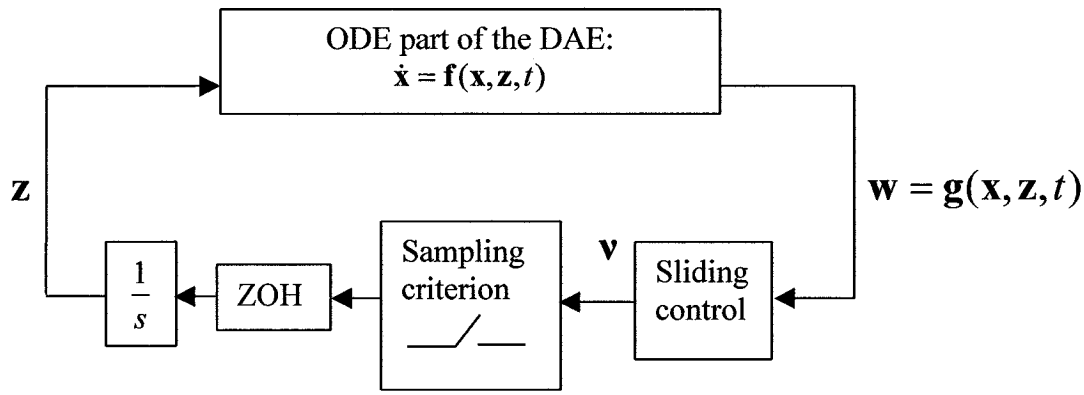


Figure 3.4. Block diagram of DAE realization using DBSMC.

As mentioned in the previous sections, once the input signal $\mathbf{v}$ is updated it is held constant until the next sample time. If $\mathbf{J}_\Omega$ and $\boldsymbol{\beta}$ from equation (2.5) remain constant in each interval, then $\mathbf{w}^{(r)}$ would stay fixed and the s-trajectory would be a polynomial in time. However, changes in the system under control can be large and the resulting discretization disturbance does not allow us to assume $\mathbf{w}^{(r)}$ is constant in each

interval. It is assumed that during each interval the rate of change of this variable is bounded from above by M_i for each sliding motion.

$$\frac{\left|w_i^{(r)} - w_{b,i}^{(r)}\right|}{t - t_b} \le M_i \quad i = 1,...,m \tag{3.5}$$

where $w_{b,i}^{(r)}$ represents $w_i^{(r)}$ evaluated at the beginning of the time interval t_b.

Remark 3.1 Eligibility to assume (3.5) comes from the fact that both $\mathbf{J}_\Omega$ and $\boldsymbol{\beta}$ in normal form (2.5) are smooth vector fields and the plant is bounded (outputs are bounded by definition and the internal dynamics is assumed to be BIBO stable). Recalling that $\mathbf{v}$ is constant in each interval and taking one more time derivative of (2.5) gives

$$\mathbf{w}^{(r+1)} = \dot{\mathbf{J}}_\Omega(t,\mathbf{x},\mathbf{z})\mathbf{v} + \dot{\boldsymbol{\beta}}(t,\mathbf{x},\mathbf{z}). \tag{3.6}$$

Now expanding the time derivative of $\boldsymbol{\beta}$

$$\dot{\boldsymbol{\beta}}(t,\mathbf{x},\mathbf{z}) = \frac{\partial \boldsymbol{\beta}}{\partial t} + \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{x}}\mathbf{f}(t,\mathbf{x},\mathbf{z}) + \frac{\partial \boldsymbol{\beta}}{\partial \mathbf{z}}\mathbf{v} \tag{3.7}$$

it is easy to see that $\dot{\boldsymbol{\beta}}$ is bounded due to smoothness of $\boldsymbol{\beta}$ and plant's stability. Similar arguments are also applicable to $\mathbf{J}_\Omega$; therefore, $\mathbf{w}^{(r+1)}$ is bounded in each interval.

Before proceeding to establish a guaranteed performance for the DBSMC scheme we need the following lemma.

Lemma 3.1 Consider a constraint with index *r* and define

$$x(t) = \frac{d(s(t) - (\mu \frac{d}{dt})^{r-1} w(t))}{dt} \tag{3.8}$$

if $|s(t)| \leq \varepsilon$ for $t \geq 0$ then

$$|x(t)| \leq \frac{2(2^{r-1} - 1)\varepsilon}{\mu} \text{ for } t \geq 0 \tag{3.9}$$

Proof.

From the block diagram in figure 3.5:

$$X(p) = \frac{p\left((\mu p + 1)^{r-1} - (\mu p)^{r-1}\right)}{(\mu p + 1)^{r-1}} S(p) = \frac{1}{\mu} \sum_{j=1}^{r-1} \left(\frac{\mu p}{\mu p + 1} \right)^{j} S(p)$$

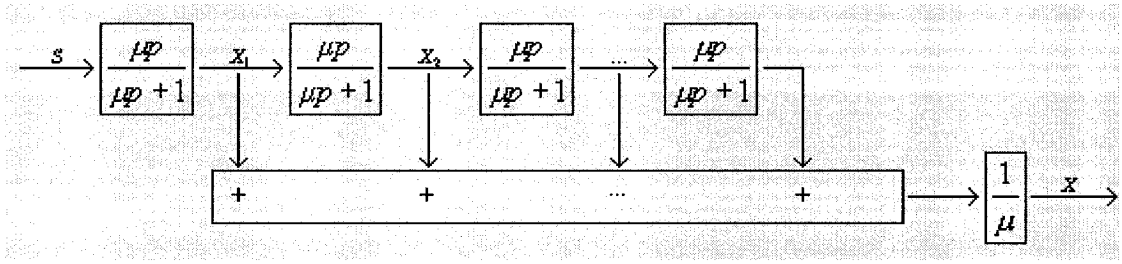


Figure 3.5. Block diagram used to calculate bounds on $x(t)$.

The bounds on the output of each block is twice as much as the bounds on its input; for example for the first block

$$\frac{\mu p}{\mu p+1} = 1 - \frac{1/\mu}{p+1/\mu}$$

$$|x_1(t)| = \left| \int_0^t s(\tau)(\delta(t-\tau) - \frac{e^{-\frac{t-\tau}{\mu}}}{\mu})d\tau \right| \le \varepsilon \int_0^t \left| \delta(t-\tau) - \frac{e^{-\frac{t-\tau}{\mu}}}{\mu} \right| d\tau \le 2\varepsilon$$

Similarly

$$x_2(t) \le 2^2\varepsilon$$
$$\vdots$$
$$x_{r-1}(t) \le 2^{r-1}\varepsilon$$

Therefore

$$|x(t)| \le \sum_{j=1}^{r-1} 2^j \frac{\varepsilon}{\mu} = \frac{2(2^{r-1}-1)\varepsilon}{\mu}. \ \square$$

Remark 3.2 There is another way to obtain the bounds on the above function. According to [13] the error, w, and its derivatives up to order r-1 are bounded by

$$\left|\frac{d^j w_i}{dt^j}\right| \le \frac{2^j \varepsilon}{\mu^j} \tag{3.10}$$

if $|s| \le \varepsilon$. By expanding the expression for $x(t)$ and using these bounds we could obtain the bounds on $x(t)$; however, results will be rather conservative. For example for $r = 3$ we would obtain

$$|x(t)| = |2\mu\ddot{w}(t) + \dot{w}(t)| \le 2\mu|\ddot{w}(t)| + |\dot{w}(t)| \le 10\frac{\varepsilon}{\mu} \tag{3.11}$$

compared to $|x(t)| \le 6\frac{\varepsilon}{\mu}$ obtained by lemma 3.1. Similarly for r=4, $|x(t)| \le 38\frac{\varepsilon}{\mu}$ by this method and $|x(t)| \le 14\frac{\varepsilon}{\mu}$ is given by lemma 3.1.

We can now establish a guaranteed lower bound on the sampling period for each individual constraint.

Theorem 3.1 If the crossing condition is satisfied, the time interval necessary for a constraint with index r to cross the sliding surface and exit from the other side of its boundary layer is greater than

$$T^{min} = \frac{-\left(|\dot{s}_b| + \frac{4(2^{r-1}-1)\varepsilon}{\mu}\right) + \sqrt{\left(|\dot{s}_b| + \frac{4(2^{r-1}-1)\varepsilon}{\mu}\right)^2 + 4(|s_b| + \varepsilon)\mu^{r-1}M}}{2\mu^{r-1}M} \qquad (3.12)$$

where $\frac{|w^{(r)} - w_b^{(r)}|}{t - t_b} \le M$ during the interval of crossing. The variables $w_b^{(r)}$, s_b and $\dot{s}_b$ represent $w^{(r)}$, s and $\dot{s}$ respectively evaluated at the beginning of the interval t_b.

Proof. Using (2.4) for initial time t_b and ending time t_e of the interval we have

$$s_b = \left(\mu\frac{d}{dt} + 1\right)^{r-1} w(t_b) \text{ and } s_e = \left(\mu\frac{d}{dt} + 1\right)^{r-1} w(t_e).$$

Now define $f(t) = s(t) - (\mu\frac{d}{dt})^{r-1} w(t)$. According to the mean value theorem we can find a point in the interval such that $\dot{f}(t_m) = \frac{f(t_e) - f(t_b)}{T}$ where $|\dot{f}(t_m)| \le \frac{(2^r - 2)\varepsilon}{\mu}$ due to lemma 3.1.

Therefore,

$$\left|\frac{f(t_e) - f(t_b)}{T}\right| \le \frac{(2^r - 2)\varepsilon}{\mu}$$

$$|s_e - s_b| = |s_b| + \varepsilon = \left|(\mu\frac{d}{dt})^{r-1}(w_e - w_b) + f(t_e) - f(t_b)\right|$$

$$\le \mu^{r-1}\left|w_e^{(r-1)} - w_b^{(r-1)}\right| + \frac{(2^r - 2)\varepsilon}{\mu}T \Rightarrow \left|w_e^{(r-1)} - w_b^{(r-1)}\right| \ge \frac{|s_b| + \varepsilon - \frac{(2^r - 2)\varepsilon}{\mu}T}{\mu^{r-1}}$$

We specify the desired $\dot{s}$ at the beginning of the interval.

$$\dot{s}_b = \dot{f}(t_b) + \mu^{r-1} w_b^{(r)} \Rightarrow w_b^{(r)} = \frac{\dot{s}_b - \dot{f}(t_b)}{\mu^{r-1}}.$$

Using lemma 3.1 for the beginning point yields the following upper bound on $\left|w_b^{(r)}\right|$

$$\left|w_b^{(r)}\right| \le \frac{|\dot{s}_b| + |\dot{f}(t_b)|}{\mu^{r-1}} \le \frac{|\dot{s}_b| + \dfrac{(2^r - 2)\varepsilon_i}{\mu}}{\mu^{r-1}}$$

Now using the mean value theorem there is a point, *m1*, in the interval for which we can write

$$\left|w_m^{(r)}\right| = \frac{\left|w_e^{(r-1)} - w_b^{(r-1)}\right|}{T}$$

$$\frac{\left|w_m^{(r)} - w_b^{(r)}\right|}{t_m - t_b} \le M \Rightarrow \left|\left|w_m^{(r)}\right| - \left|w_b^{(r)}\right|\right| \le \left|w_m^{(r)} - w_b^{(r)}\right| \le M(t_m - t_b) \le MT \Rightarrow$$

$$\left|w_m^{(r)}\right| \le \left|w_b^{(r)}\right| + MT \le \frac{|\dot{s}_b| + \dfrac{(2^r - 2)\varepsilon_i}{\mu}}{\mu^{r-1}} + MT$$

$$T = \frac{\left|w_e^{(r-1)} - w_b^{(r-1)}\right|}{\left|w_m^{(r)}\right|} \ge \frac{\dfrac{|s_b| + \varepsilon - \dfrac{(2^r - 2)\varepsilon}{\mu} T}{\mu^{r-1}}}{\dfrac{|\dot{s}_b| + \dfrac{(2^r - 2)\varepsilon}{\mu}}{\mu^{r-1}} + MT} \Rightarrow$$

$$T \geq \frac{-\left(|\dot{s}_b| + \frac{4(2^{r-1}-1)\varepsilon}{\mu}\right) + \sqrt{\left(|\dot{s}_b| + \frac{4(2^{r-1}-1)\varepsilon}{\mu}\right)^2 + 4(|s_b| + \varepsilon)\mu^{r-1}M}}{2\mu^{r-1}M}. \quad \square$$

Remark 3.3 Note that the above result can alternatively provide us with maximum system deviations for each sliding surface variable when only a certain amount of sampling rate is available. Observing that $|\dot{s}_b| = k$ we can write

$$|s(t)| \leq \frac{\mu^{r-1}MT + k}{\frac{1}{T} - \frac{4(2^{r-1}-1)}{\mu}} \tag{3.13}$$

Equation (3.10) will then yield the resulting bounds on error, $w(t)$, and its derivatives up to $w^{(r-1)}(t)$. This information is applicable to any discretized implementation of SMC.

Denoting by k_i^c the minimum amount of gain that ensures the crossing condition we have the following bound for DBSMC realization.

Theorem 3.2 Consider the DBSMC scheme applied to DAE realization, equation (2.8), with $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}$, $\hat{\mathbf{J}}_s = \mathbf{J}_s$. If for all constrains the gain k_i is enough to ensure the crossing condition, $k_i \geq k_i^c$, and there exists M_i such that during the interval $\frac{\left|w_i^{(r)}(t) - w_{b,i}^{(r)}\right|}{t - t_b} \leq M_i$ then the sampling periods imposed by this constraint after it enters its boundary layer are bounded from below by

$$T_i^{min} = \frac{-\left(k_i + \frac{4(2^{r_i-1}-1)\varepsilon_i}{\mu_i}\right) + \sqrt{\left(k_i + \frac{4(2^{r_i-1}-1)\varepsilon_i}{\mu_i}\right)^2 + 4\varepsilon_i\mu_i^{r-1}M_i}}{2\mu_i^{r-1}M_i} \qquad (3.14)$$

and the sampling period that ensure all motions remain in their boundary layers is the minimum of these values,

$$T^{min} = \min(T_i^{min}) \; i = 1,...,m. \qquad (3.15)$$

Proof. If we assume that the gains are sufficiently large to guarantee the crossing condition we can use the result of theorem 3.1. On the other hand $|\dot{s}_{b,i}| = k_i$ at the beginning of each interval and $|s_b| \geq 0$. Substituting this result into equation (3.12) we obtain

$$T_i^{min} = \frac{-\left(k_i + \frac{4(2^{r_i-1}-1)\varepsilon_i}{\mu_i}\right) + \sqrt{\left(k_i + \frac{4(2^{r_i-1}-1)\varepsilon_i}{\mu_i}\right)^2 + 4\varepsilon_i\mu_i^{r-1}M_i}}{2\mu_i^{r-1}M_i}.$$

Finally, $T^{min} = \min(T_i^{min}) \; i = 1,...,m$ is the period imposed by all motions collectively. □

3.4 Robustness conditions

The robustness property of sliding controllers represents one of its main advantages over other control methods. Fortunately the proposed method has similar

robustness properties to the previously proposed sliding control realization methods. Recalling the following result for SPSM realization [7]

Theorem 3.3 Consider the following DAE realization (2.8) where $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{J}}_s$ are perturbed versions of $\boldsymbol{\alpha}$ and $\mathbf{J}_s$ respectively. If the above realization has a vector index and

$$\mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K} - \boldsymbol{\Gamma} \text{ is positive diagonally dominant} \tag{3.16}$$

the above control renders the sliding surfaces attractive. The expression $\boldsymbol{\Gamma} = \left| \text{diag}[\boldsymbol{\alpha} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\hat{\boldsymbol{\alpha}}] \right|$ is a combined uncertainty due to perturbations and parametric uncertainties.

Perturbations in simulation problems exist in a similar manner to those in physical control systems. Sometimes they are unintentional such as round off errors. However, there is also another type of perturbation that is intentionally introduced as a result of model simplifications made to reduce computational costs. In either case we represent the value of $\boldsymbol{\alpha}$ that is practically used to update equations by $\hat{\boldsymbol{\alpha}}$. Following the classical sliding control approach we assume that although real values of $\boldsymbol{\alpha}$ are unknown (or expensive to find) we have an available upper bound on the difference $\left|\alpha_i - \hat{\alpha}_i\right|$ for each sliding surface. The following theorem reveals that this uncertainty can be handled through an increase in gains, similar to continuous time sliding control.

Theorem 3.4 Consider the discretization scheme applied to a DAE realization (2.8) with $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{J}}_s = \mathbf{J}_s$. If for all constraints

$$k_i \geq k_i^c + |\alpha_i - \hat{\alpha}_i| \tag{3.17}$$

then the crossing condition is robustly satisfied and sampling periods imposed by this constraint are bounded from below by the value given in equation (3.15).

Proof. The main difference here is that the actual, $\dot{\mathbf{s}}^a$, are not necessarily equal to the desired values, $\dot{\mathbf{s}}^d$, at the beginning of intervals; however, since $\hat{\mathbf{J}}_S = \mathbf{J}_S$ we can write

$$\begin{cases} \dot{\mathbf{s}}^a = \mathbf{J}_S \mathbf{v} + \boldsymbol{\alpha} \\ \mathbf{v} = \hat{\mathbf{J}}_S^{-1}(\dot{\mathbf{s}}^d - \hat{\boldsymbol{\alpha}}) \end{cases} \Rightarrow \dot{\mathbf{s}}^a = \dot{\mathbf{s}}^d + (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}).$$

Now given $k_i \geq k_i^c + |\alpha_i - \hat{\alpha}_i|$ for each constraint we can write

$$-\text{sign}(s_i^b)\dot{s}_i^a = -\text{sign}(s_i^b)(k_i + (\alpha_i - \hat{\alpha}_i)) \geq k_i^c + |\alpha_i - \hat{\alpha}_i| - \text{sign}(s_i^b)(\alpha_i - \hat{\alpha}_i) \geq k_i^c$$

Meaning firstly that $\text{sign}(s_i^b)\dot{s}_i^a < 0$ and the motion sets off in the right direction (heading to other side of the layer) and secondly $|\dot{s}_i^a| \geq k_i^c$ so it can cross the boundary layer. Therefore the crossing condition is satisfied and the bound given by equation (3.15) holds for this case too. □

The above result is comparable to the continuous time criterion [14]

$$k_i \geq |\alpha_i - \hat{\alpha}_i| + \eta_i \quad , \quad \eta_i > 0 \qquad \text{for } 1 \leq i \leq m. \tag{3.18}$$

Although a small positive η_i ensures stability in discrete control of continuous systems, we need to satisfy a crossing condition to avoid situations such as figure 3.2 so that a certain amount of performance can be guaranteed.

Remark 3.4 The general expression for α_i is given by

$$\alpha_i = \mu^{r_i - 1}\beta_i(t, \mathbf{x}, \mathbf{z}) + \sum_{j=1}^{r_i - 1} \frac{(r_i - 1)!\mu_i^{r_i - j - 1}}{(r_i - j - 1)!\, j!} w_i^{(j)} \tag{3.19}$$

The terms in the second expression have already been evaluated in the process of calculating s_i so a choice of

$$\hat{\alpha}_i = \sum_{j=1}^{r_i - 1} \frac{(r_i - 1)!\mu_i^{r_i - j - 1}}{(r_i - j - 1)!\, j!} w_i^{(j)} \tag{3.20}$$

does not involve much computational overhead. Theorem 3.4 then yields the following sufficient condition

$$k_i \geq k_i^c + \mu^{(r_i - 1)}|\beta_i|. \tag{3.21}$$

Similar to the previous case it can be shown that our discretized control has similar robustness properties in the presence of both disturbances and parametric uncertainties.

Theorem 3.5 Consider the discretization scheme applied to a DAE realization (2.8) with $\hat{\boldsymbol{\alpha}}$, $\hat{\mathbf{J}}_s$. The crossing condition is satisfied and the sampling periods imposed by constraints are bounded from below by the value given in equation (3.15) if the following condition is satisfied

$$\mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K} - \boldsymbol{\Gamma} \text{ is positive diagonally dominant} \tag{3.22}$$

where $\boldsymbol{\Gamma} = \text{diag}\left[\left|\boldsymbol{\alpha} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\hat{\boldsymbol{\alpha}}\right| + \mathbf{K}^c\right]$.

Proof. Our proof follows similar SMC robustness theorems [14], [7] with additional k_i^c term added to meet the crossing condition. At the beginning of each interval all inputs are updated according to

$$\mathbf{v} = \hat{\mathbf{J}}_s^{-1}\left(-\hat{\boldsymbol{\alpha}} - \mathbf{K}\,\text{diag}[\text{sign}(s_i)]\right)$$

At this time the derivative of sliding motions is given by

$$\dot{\mathbf{s}} = \boldsymbol{\alpha} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\hat{\boldsymbol{\alpha}} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K}\,\text{sign}(\mathbf{s})$$

Expanding for the i^{th} component yields

$$\dot{s}_i = \gamma_i - D_{i1}\,\text{sign}(s_1) - \ldots - D_{ii}\,\text{sign}(s_i) - \ldots - D_{im}\,\text{sign}(s_m)$$

where

$$\boldsymbol{\gamma} = \boldsymbol{\alpha} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\hat{\boldsymbol{\alpha}}, \quad \mathbf{D} = \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K}$$

Attractivity of this sliding surface and the crossing condition can then be enforced if

$$D_{ii} > |\gamma_i| + |D_{i1}| + \ldots + |D_{im}| + k_i^c$$

since it implies both $\text{sign}(s_i^b)\dot{s}_i^a < 0$ and $|\dot{s}_i^a| \geq k_i^c$.

This condition is equivalent to the matrix $\mathbf{D} - \text{diag}(|\boldsymbol{\gamma}| + \mathbf{K}^c)$ being positive diagonally dominant; hence, the condition (3.22).

□

4 Design and Optimization of DBSMC

This chapter addresses key design and optimization issues associated with DBSMC. Included are conditions for sufficient gain selection to ensure the crossing condition, optimization of the controller sampling period and an overview of the DBSMC design procedure. The performance of system is considered in terms of our lower bound estimate of the sampling period (3.15) and formulated in terms of the controller parameters. Theorem 4.1 provides us with a sufficient gain that ensures the crossing condition and allows us to reduce the parameter dependency of relation (3.14) to a relation involving only μ.

4.1 Gain selection for crossing condition

Lemma 4.1 Consider a constraint with index r and recall $x(t)$ from equation (3.8). The absolute value of function $y(t)$ given by

$$y(t) = x(t) - \frac{\mu}{r-1}\frac{d^2\left(s(t) - \left(\left(\mu\frac{d}{dt}\right)^{r-1} + (r-1)\left(\mu\frac{d}{dt}\right)^{r-2}\right)w(t)\right)}{dt^2} \tag{4.1}$$

is bounded from above by

$$|y(t)| \le \frac{2\left(2^r - r - 1\right)}{r-1}\frac{\varepsilon}{\mu} \text{ for } t \ge t_r \tag{4.2}$$

after $s(t)$ reaches inside the boundary layer, $|s(t)| \le \varepsilon$, at t_r.

Proof. Analyzing the expression in the frequency domain gives

$$X(p) - Y(p) = \frac{\mu}{r-1}p^2\frac{\left((\mu p+1)^{r-1} - (\mu p)^{r-1} - (r-1)(\mu p)^{r-2}\right)}{(\mu p+1)^{r-1}}S(p) =$$

$$\frac{1}{\mu(r-1)}\sum_{k=1}^{r-2}k\left(\frac{\mu p}{\mu p+1}\right)^{k+1}S(p)$$

Combining with the Laplace transform of $x(t)$ from lemma 3.1 we obtain

$$Y(p) = \frac{1}{\mu}\sum_{j=1}^{r-1}\left(\frac{\mu p}{\mu p+1}\right)^j S(p) - \frac{1}{\mu(r-1)}\sum_{j=1}^{r-1}(j-1)\left(\frac{\mu p}{\mu p+1}\right)^j S(p) =$$

$$\frac{1}{\mu}\sum_{j=1}^{r-1}\left(1-\frac{j-1}{r-1}\right)\left(\frac{\mu p}{\mu p+1}\right)^j S(p)$$

The block diagram of this transfer function is similar to the one in lemma 3.1 except that this time, each bock is multiplied by a gain before summation. In this case we obtain

$$|y(t)| \le \sum_{j=1}^{r-1}(1-\frac{j-1}{r-1})2^j\frac{\varepsilon}{\mu} = \frac{2\left(2^r - r - 1\right)}{r-1}\frac{\varepsilon}{\mu}.$$

□

Replacing (3.5) with the stronger assumption

$$w_i^{(r+1)}(t) \le M_i, \quad i = 1,...,m \tag{4.3}$$

on all intervals and using the result in lemma 4.1 one can write the following theorem.

Theorem 4.1 A sufficient gain that guarantees the crossing condition for any state inside the boundary layer is given by

$$k^* = \frac{\mu^r M}{r-1} + \frac{2(2^r - r - 1)}{r-1}\frac{\varepsilon}{\mu}. \tag{4.4}$$

Proof. The case when initially $s_b < 0$ will be investigated here. The other case can be proved similarly. First we show how the motion can be written in terms of the above gain.

$$\ddot{s}(t) = \frac{d^2\left(\left(\mu\frac{d}{dt}\right)^{r-1}\right)}{dt^2}w(t) + (r-1)\frac{d^2\left(\left(\mu\frac{d}{dt}\right)^{r-2}\right)}{dt^2}w(t) + \frac{d^2\left(s(t) - \left(\left(\mu\frac{d}{dt}\right)^{r-1} + (r-1)\left(\mu\frac{d}{dt}\right)^{r-2}\right)w(t)\right)}{dt^2} =$$

$$\frac{d^2\left(\left(\mu\frac{d}{dt}\right)^{r-1}\right)}{dt^2}w(t)+\frac{r-1}{\mu}\left(\dot{s}(t)-y(t)\right)\geq$$

$$-\mu^{(r-1)}M+\frac{r-1}{\mu}\left(\dot{s}(t)-y(t)\right)\geq-\mu^{r-1}M+\frac{r-1}{\mu}\left(\dot{s}(t)-\frac{2\left(2^r-r-1\right)}{r-1}\frac{\varepsilon}{\mu}\right)$$

$$=\frac{r-1}{\mu}\left(\dot{s}(t)-k^*\right)$$

Therefore there exist positive variable $\eta(t)>0$ such that $\ddot{s}(t)=\eta(t)\left(\dot{s}(t)-k^*\right)$. This LPV system then yields $\dot{s}(t)\geq k^*$ thereafter if initially $\dot{s}(t_b)\geq k^*$. Choosing k^* at the beginning of each interval satisfies the initial condition requirement and completes the proof. □

4.2 Optimization of controller sampling period

The following corollary is useful in Optimization of the controller sampling period.

Corollary 4.1 The controller gain k^c is by definition the minimum amount of gain that ensures the crossing condition. According to theorem 4.1 we can guarantee this condition by selecting k^*. Combining this gain with the results of theorems 3.2 or 3.5 yields sufficient robust gains. Substituting this gain into equation (3.14) we can derive the worst case guaranteed lower bound for sampling periods exclusively in terms of μ_i. This equation can then be maximized to find optimal values of μ_i which will result in an optimal lower bound and controller sampling period.

As mentioned in theorem 4.1, values of gains given by (4.4) are only sufficient and once proper values for μ_i are obtained we can improve the performance by experimentally using smaller gains (figure 4.1) that still satisfy the crossing condition.

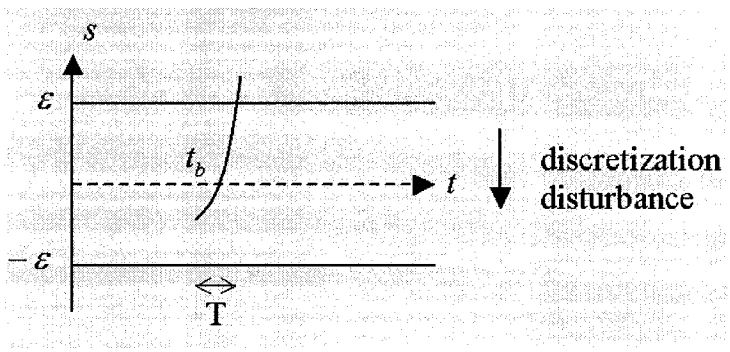


Figure 4.1. An excessive amount of input produces a short crossing time.

These smaller gains as well as less conservative error bounds lead to more realistic cost functions that can improve the optimality of the controller sampling period.

In order to improve optimality further we make an attempt to develop less conservative cost functions. The following lemma gives a less conservative result for the case of index three DAEs which can be generalized in a similar manner to higher index DAEs.

Lemma 4.2 In the period of time that a motion is inside the boundary layer the absolute value of the function $x(t) = 2\mu\ddot{w}(t) + \dot{w}(t)$ is bounded from above by

$$|x(t)| \le \frac{4+2e^{-3}}{\mu}\varepsilon . \tag{4.5}$$

Proof. For notational simplicity define $\lambda = 1/\mu$. To express the evolution of $x(t)$ in terms of the sliding motion we express their relation in the frequency domain

$$X(p) = \lambda\left[\frac{p}{p+\lambda} + \left(\frac{p}{p+\lambda}\right)^2\right]S(p)$$

And find the bounds in time domain by inverting the Laplace transform

$$\frac{|x(t)|}{\varepsilon} = \int_0^t \left|2\lambda\delta(\tau) - 3\lambda^2 e^{-\lambda\tau} + \lambda^3\tau e^{-\lambda\tau}\right| d\tau \le$$

$$2\lambda + \int_0^{3/\lambda}\left(3\lambda^2 e^{-\lambda\tau} - \lambda^3\tau e^{-\lambda\tau}\right)d\tau + \int_{3/\lambda}^{T}\left(-3\lambda^2 e^{-\lambda\tau} + \lambda^3\tau e^{-\lambda\tau}\right)d\tau$$

$$= \begin{cases} 4\lambda + \lambda^2 t e^{-\lambda t} - 2\lambda e^{-\lambda t} & t \le 3/\lambda \\ 4\lambda + 2\lambda e^{-3} + 2\lambda e^{-\lambda t} - \lambda^2 t e^{-\lambda t} & t > 3/\lambda \end{cases}$$

As shown in figure 4.2 the above upper bound, denoted by f(t) is bounded from above by $\left(4+2e^{-3}\right)\lambda$.

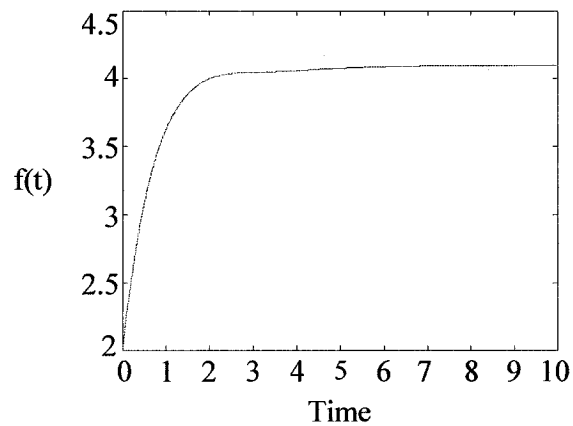


Figure 4.2. A less conservative bound on x(t) for index three DAEs.

□

Therefore a less conservative lower bound for index three DAEs can be written as

$$T_i^{min} = \frac{-\left(k_i + \frac{4(2+e^{-3})\varepsilon_i}{\mu_i}\right) + \sqrt{\left(k_i + \frac{4(2+e^{-3})\varepsilon_i}{\mu_i}\right)^2 + 4\mu_i^2\varepsilon_i M_i}}{2\mu_i^2 M_i}. \tag{4.6}$$

In finding the bounds in (3.14) and (4.6) we are assuming the sliding motion to have an unlimited frequency range. However, in most applications the frequency content of the sliding motion can be practically bounded. The following result illustrates how this fact might affect cost function bounds in practice.

Lemma 4.3 If a constraint error, $w(t)$, has its frequency spectrum bounded by pulsation ω_m, then for the period of time that its corresponding motion is inside its boundary layer the absolute value of function $x(t) = 2\mu\ddot{w}(t) + \dot{w}(t)$ is bounded from above by

$$|x(t)| \le \left(2\mu\omega_m^2 + \omega_m\right)\varepsilon. \tag{4.7}$$

Proof. Using (3.10) we have $w(t) \le \varepsilon$ that gives us $\left|\frac{d^n}{dt^n}w(t)\right| \le (\omega_m)^n \varepsilon$ for all derivatives due to Bernstein lemma [15]. The result then follows by substituting this last inequality into the expression for $x(t)$. □

Improvements in the Optimization problem become particularly practical in on line parameter tuning schemes where we have access to more realistic bounds on errors, their derivatives, and disturbances.

4.3 Controller design

The results presented in this thesis can be combined into a design procedure for an optimal DBSMC controller as illustrated in figure 4.3.

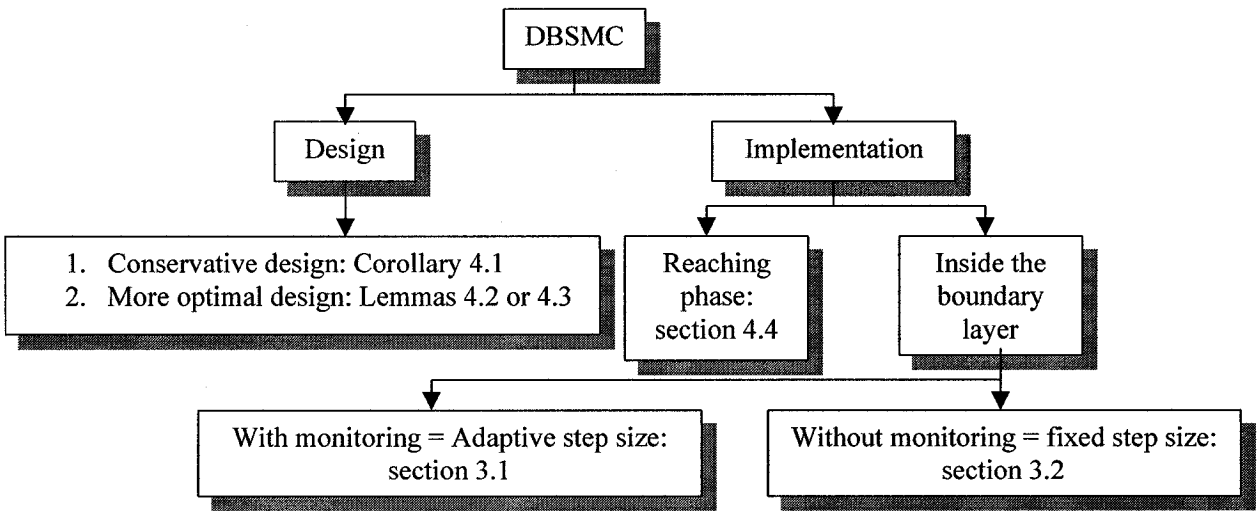


Figure 4.3. Design and Implementation of DBSMC.

For optimal gains and controller sampling period we can use Corollary 3.1. Less conservative values can be obtained for the special cases of index 3 DAEs and outputs with bounded frequency spectrum using Lemmas 4.2 and 4.3 respectively. The DBSMC method can be applied in the reaching phase using the methods outlined in chapter 4.

Inside the boundary layer we can use a varying sampling period described in section 3.1 if we monitor the sliding surface variables and update the controller when the errors reach their specified values. If monitoring is computationally costly or varying sampling period is not desired we can use DBSMC without monitoring (section 3.2) with a constant controller sampling period. Together these results provide a systematic approach for designing efficient discretized sliding control realizations.

4.4 Reaching phase dynamics

The focus of this thesis is on the sliding stage of sliding mode control. For inconsistent initial conditions, however, the motion might begin outside of the boundary layer. This initial stage of motion known as the reaching phase is known to take a finite time in continuous sliding control [14]. The following two algorithms are proposed for the reaching phase of the DBSMC approach before the motion is contained in the desired boundary layers

1. Contracting boundary layers

Given any initial conditions we can always set the initial width of the boundary layers large enough to contain them. Thus, $\varepsilon_k^i = \max(|s_k^i|, \varepsilon^i)$ $i = 1,...,m$ where k is the step number. It is then possible to apply the previous theory, which guarantees (due to the crossing condition) that all motions will converge towards their sliding manifolds. We can then divide the motions into an "inside" region for $\varepsilon_n^i = \varepsilon^i$ and an "outside" region

for $\varepsilon_n^i = |s_n^i|$. A possible way to ensure contraction of outside boundary layers is to update inputs either when the first outside trajectory hits its sliding surface or an inside trajectory reaches the other side of its boundary layer (see figure 4.4). The above procedure can be repeated until all outside trajectories move inside. Since the contraction rate of outside trajectories does not approach zero (owing to the crossing condition) and each interval takes a finite time (theorem 3.2), this procedure has a finite reaching time similar to continuous sliding control. This algorithm also guarantees that once a trajectory has reached its boundary layer it remains inside thereafter. Note that the sudden changes in boundary width are within the scope of the theory presented in this thesis since all of the results were obtained by separately studying the motion in finite time intervals, which may have their own boundary widths.

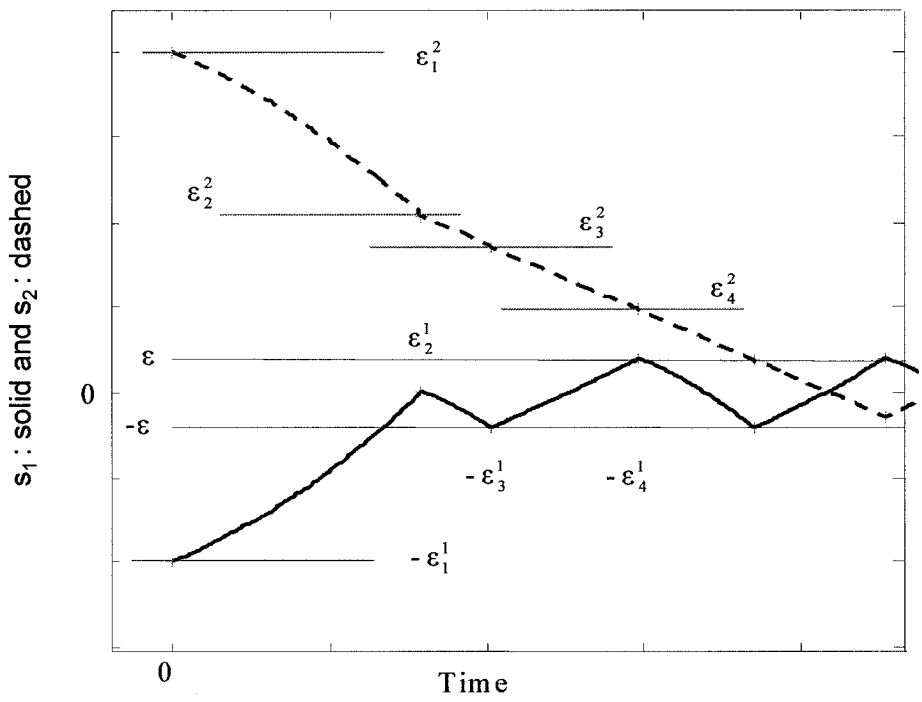


Figure 4.4. Contracting boundary layers during the reaching phase.

2. Attracting surfaces

This algorithm involves the adjusting of gains each time we change the width of the boundary layers to satisfy the crossing condition in the new interval. We can alternatively use the original gains that were obtained for the desired boundary layer. This may not be enough to guarantee that trajectories will reach their sliding surfaces when they are far away. However, even with insufficient gains the motion is directed towards the sliding surface (local attractivity) for a finite amount of time before it is diverted by a possibly large discretization disturbance. If the inputs are updated each time an "outside" trajectory is starting to move away or when an "inside" trajectory reaches the other side of its boundary layer (see figure 4.5) we can guarantee that after a finite time all trajectories will be contained in the desired boundary layers. Therefore, this algorithm provides both finite reaching time and locking properties.

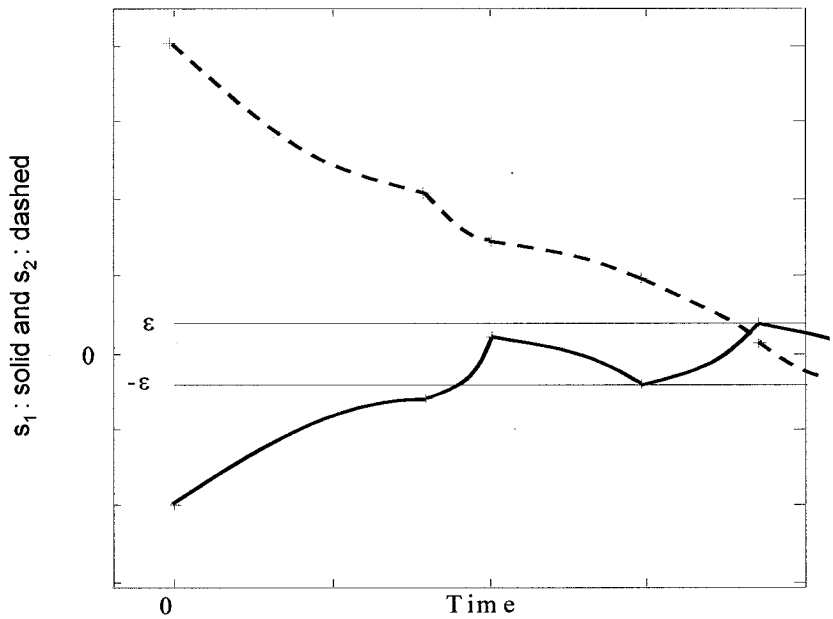


Figure 4.5. Attracting surfaces during the reaching phase.

5 Application of DBSMC

In this chapter the DBSMC approach is applied for simulation of a double pendulum DAE system (figure 5.1).

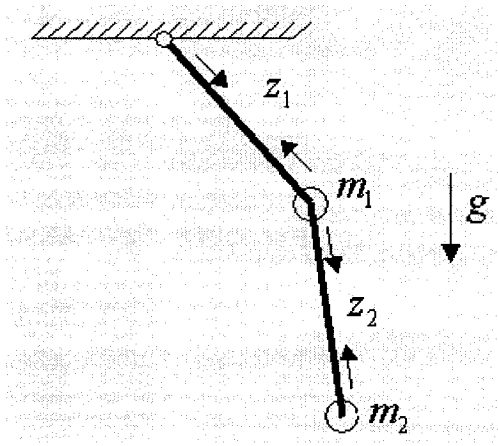


Figure 5.1. Double pendulum DAE system

5.1 Mechanical model and sliding control formulation

Writing the equations of motion in Cartesian coordinates yields the following DAE system

$$\begin{cases} \dot{\mathbf{R}}_1(t) = \mathbf{V}_1(t) \\ m_1\dot{\mathbf{V}}_1(t) = -z_1\hat{\mathbf{r}}_{01}(t) + z_2\hat{\mathbf{r}}_{12}(t) - m_1 g\hat{k} \\ \dot{\mathbf{R}}_2(t) = \mathbf{V}_2(t) \\ m_2\dot{\mathbf{V}}_2(t) = -z_2\hat{\mathbf{r}}_{12}(t) - m_2 g\hat{k} \end{cases} \qquad (5.1)$$

$$\begin{cases} 0 = L_1(t) - L_1^0 \\ 0 = L_2(t) - L_2^0 \end{cases}$$

where $\vec{\mathbf{R}}(t), \vec{\mathbf{V}}(t), m,$ and $L(t)$ represent position, velocity, mass, and length of each link respectively. Instantaneous lengths are denoted by $L(t)$ and each L^0 is the desired length of each pendulum link. The unit direction vectors are defined by

$$\hat{\mathbf{r}}_{ij} = \frac{\mathbf{R}_j - \mathbf{R}_i}{|\mathbf{R}_j - \mathbf{R}_i|}. \qquad (5.2)$$

Denoting the error in constraints by w_1 and w_2 and differentiating with respect to time we can write for link 1

$$\begin{aligned} w_1(t) &= L_1(t) - L_1^0 \\ \dot{w}_1(t) &= \hat{\mathbf{r}}_{01}(t).\mathbf{V}_1(t) \\ \ddot{w}_1(t) &= \frac{|\mathbf{V}_1(t)|^2 - \dot{w}_1(t)^2}{L_1(t)} - g\hat{\mathbf{r}}_{01}(t).\hat{k} + \hat{\mathbf{r}}_{01}(t).\left(\frac{-z_1\hat{\mathbf{r}}_{01}(t) + z_2\hat{\mathbf{r}}_{12}(t)}{m_1}\right) \end{aligned} \qquad (5.3)$$

$$\begin{aligned} \beta_1(t) &= \frac{d}{dt}\left(\frac{|\mathbf{V}_1(t)|^2 - \dot{w}_1(t)^2}{L_1(t)} - g\hat{\mathbf{r}}_{01}(t).\hat{k}\right) + \frac{1}{m_1}\frac{d}{dt}\left(\hat{\mathbf{r}}_{01}(t).\hat{\mathbf{r}}_{12}(t)\right)z_2 \\ \alpha_1(t) &= \mu_1^2\beta_1(t) + 2\mu_1\ddot{w}_1(t) + \dot{w}_1(t) \end{aligned} \qquad (5.4)$$

Similar expressions can be derived for the other link. It can be shown that the DAE index is three for both constraints. Therefore, equation (2.4) yields

$$s_i(t) = \mu_i^2 \ddot{w}_i(t) + 2\mu_i \dot{w}_i(t) + w_i(t) \quad , \quad i = 1, 2 \tag{5.5}$$

and the associated Jacobian matrix is given by

$$\mathbf{J}_s = \begin{bmatrix} -\dfrac{\mu_1^2}{m_1} & \dfrac{\mu_1^2}{m_1}\hat{\mathbf{r}}_{01}.\hat{\mathbf{r}}_{12} \\ \dfrac{\mu_2^2}{m_2}\hat{\mathbf{r}}_{01}.\hat{\mathbf{r}}_{12} & -2\dfrac{\mu_2^2}{m_2} \end{bmatrix}. \tag{5.6}$$

5.2 Simulation parameters

In this simulation we select $m_1 = m_2 = 0.1\,(\text{kg})$, $L_1^0 = L_2^0 = 2.0\,(\text{m})$ and the gravitational constant to be $g = 9.8\,(\text{m/s}^2)$. The simulation initial conditions are given zero initial velocities, but in order to test the reaching phase performance of the algorithm some initial errors in constraints are introduced using the initial positions

$$\mathbf{R}_1(0) = [-1, 0, 0]^T, \mathbf{R}_2(0) = [-5, 0, 0]^T.$$

Substituting the initial conditions above and setting $z_1(0) = z_2(0) = 0$ in (5.5) we have: $s_1(0) = w_1(0) = -1$ and $s_2(0) = w_2(0) = 2$. The specified error bounds are

$\varepsilon_1 = \varepsilon_2 = 0.2$. Therefore both constraints are initially outside the desired boundary layers. In this example the *attracting surfaces* reaching phase approach proposed in chapter 4 was employed (see figure 5.4 (a)).

5.3 Controller design and simulation results

To avoid the cost of computing exact α an approximation $\hat{\alpha}$ was used. Using remark 3.4 we select $\hat{\alpha}_i = 2\mu_i \ddot{w}_i + \dot{w}_i$, $i = 1,2$. This approximation reduced $|\alpha - \hat{\alpha}|_{max}$ to nearly half compared to selecting $\hat{\alpha}_i = 0$. According to theorems 3.2 and 3.5 this helps us obtain larger sampling periods. From remark 3.4 our choice of $\hat{\alpha}$ yields: $|\alpha_i - \hat{\alpha}_i| = \mu_i^2 |\beta_i|$. Since β depends only on the original system dynamics it does not depend on the sliding manifold parameters μ_i selected.

Initial simulations provided the following bounds: $|\beta_1| \le 1.3 \times 10^3$, $|\beta_2| \le 2.3 \times 10^3$, $M_1 = 9 \times 10^4$, $M_2 = 11 \times 10^4$. Choosing the gains according to (3.21) and (4.4) to ensure the crossing condition yields

$$k_i = \frac{\mu_i^3 M_i}{2} + 4\frac{\varepsilon_i}{\mu_i} + \mu_i^2 |\beta_i|, \; i = 1,2 \tag{5.7}$$

Optimizing the corresponding cost functions (controller sampling period) yields μ_1=0.065 and μ_2=0.060 (see figure 5.2 for the cost function of the second link).

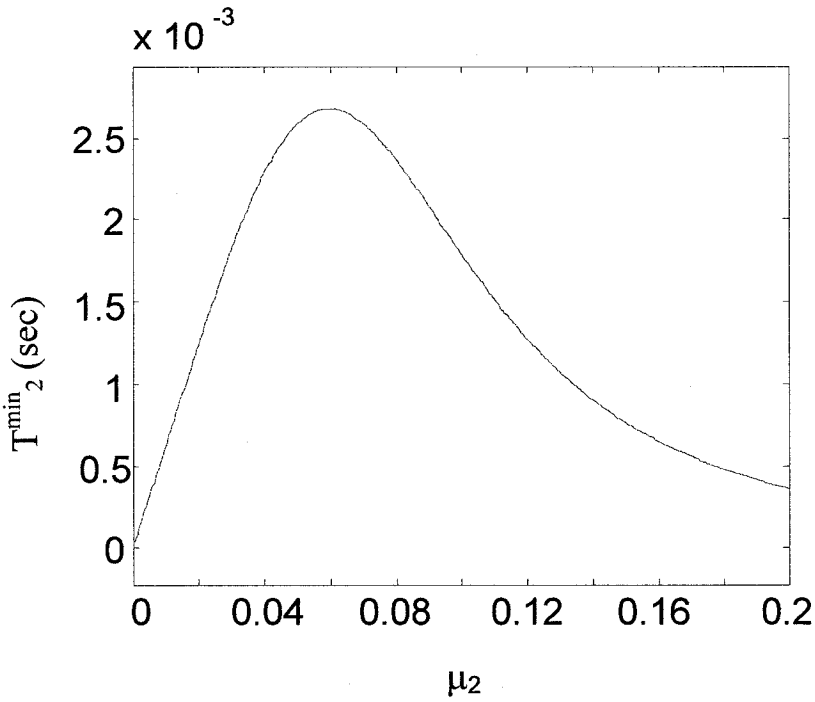


Figure 5.2. Cost function for the second link with sufficiently large input

The controller gains can now be lowered to provide a more optimal solution. It was found that choosing the following gains

$$k_i = \mu_i^2 |\beta_i|, i = 1,2 \tag{5.8}$$

which correspond to the gains that satisfy the robustness condition, was also enough in practice to satisfy the crossing condition. In order to demonstrate the improved sampling period Optimization the lowered gains were substituted into equation (4.6). The new optimal designs then yield μ_1=0.076 and μ_2=0.066 (see figure 5.3 for the cost function of the second link).

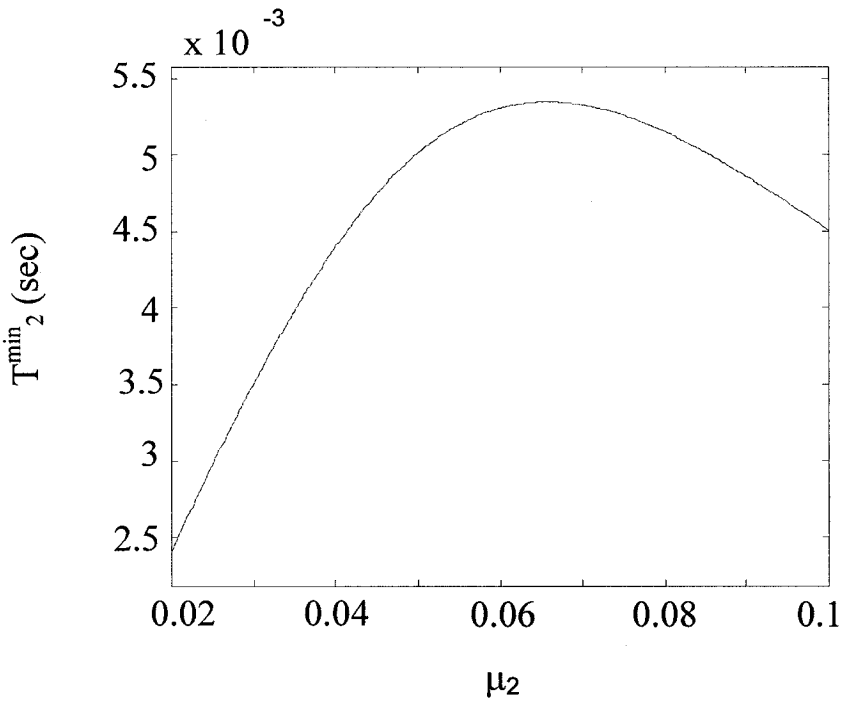


Figure 5.3. Cost function for the second link with relaxed gains.

The optimal parameter values (see figure 5.3) guarantee a minimum sampling period $T_{min} = 0.005$ (s). The DBSMC approach with monitoring (see figure 5.4 (b)) was applied to the system. The minimum and average values of sampling periods obtained during simulations are $T_{min} = 0.016$ (s) and $T_{av} = 0.025$ (s). Therefore, the minimum measured value is above the predicted bound $T_{min} = 0.005$ as expected. Figures 5.4a and 5.5 depict the evolution of s-trajectories and errors with time.

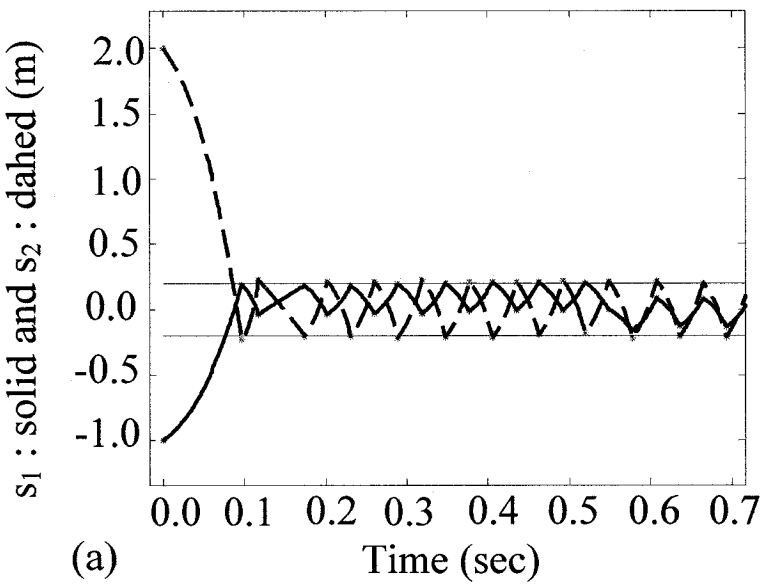
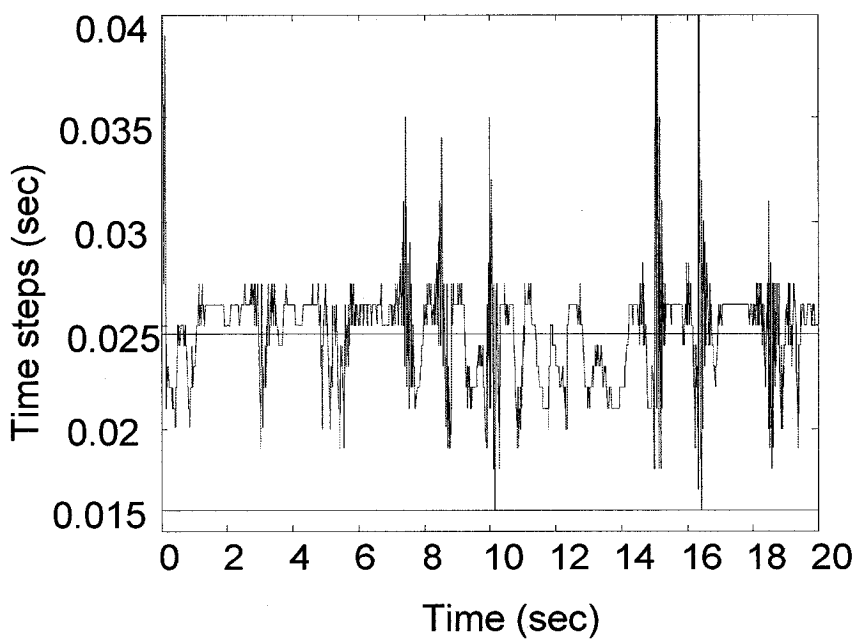


Figure 5.4. (a) Evolution of s–trajectories for each link,



(b) Time steps used to ensure the desired bounds.

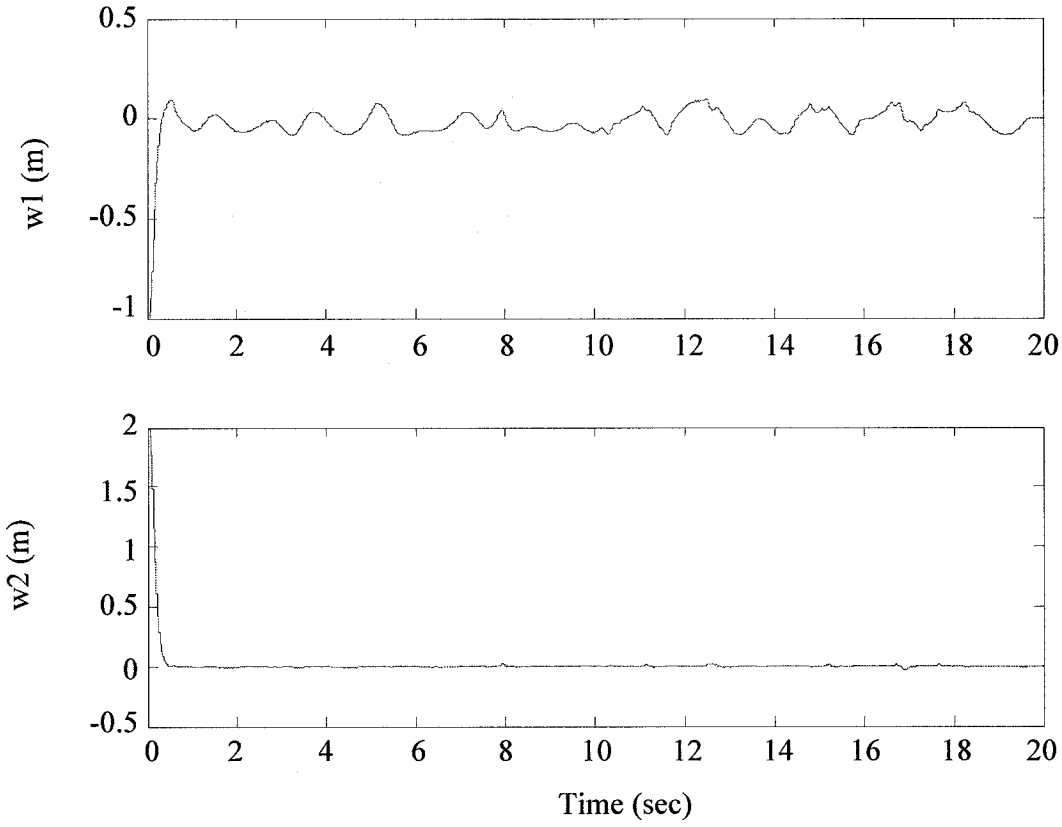


Figure 5.5. After a short reaching time the desired bounds on the length of each link are satisfied.

The implementation of DBSMC without monitoring can be applied by using an sampling period of T_{min} = 0.005 (s) for the entire simulation (see figure 5.6). This results in five times higher sampling frequency and computational load compared to the mean value T_{av}=0.025 (s) for DBSMC with monitoring. It is also apparent that DBSMC without monitoring restricts the sliding motion to ε_1=0.06 and ε_2=0.08 which is smaller than the specified error bounds of ε=0.2. Thus, DBSMC with monitoring appears to be the preferable method of implementation for this type of DAE system since the computational cost of monitoring is small for this problem.

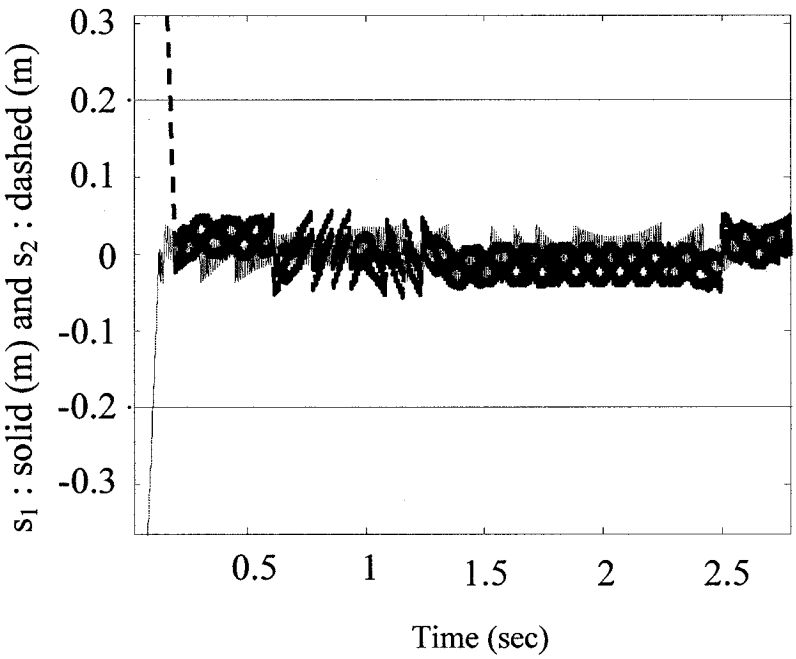


Figure 5.6. Sliding motion with constant time-step.

6 Conclusions and future work

In this thesis, a new approach for efficient realization of DAE systems based on discretized bounded sliding mode control is presented. The main results are summarized as follows:

- Key relations are developed between controller parameters, error bounds, and the controller sampling period in theorem 3.2.
- Necessary conditions are derived that allow making systematic approximations and cutting on the computational overhead in theorem 3.5.
- Sufficient inputs are derived that ensure the crossing condition in theorem 4.1.
- They are then used to synthesize efficient discretized sliding mode realizations that optimize the controller sampling period and reduce the crossing frequency in corollary 3.1.

As a future work, application of DBSMC to large scale problems such as deformable structures [9] is recommended. Moreover in this thesis we mainly focused on keeping the DAE constraints small; in other terms, looking at (2.8) we solved an output regulation problem while a DAE also includes internal states and dynamics, which are not controllable through inputs. This issue is particularly important when we want to investigate how close our DAE realization is to the original DAE. Another important question is if our DAE realization will affect important qualities of the DAE such as controllability/observability. Gordon [7] has studied these problems for the case of

systems with continuous control inputs. We recommend investigation of these issues for discrete inputs like the ones suggested in chapter 3 as a future work.

References

[1] Lunberger, D. G., 1977, Dynamic Equations in Descriptor Form. *IEEE Transactions on Automatic Control*, vol. 22, no. 3, 312-321.

[2] Hubert Hahn : Rigid Body Dynamics of Mechanisms: Applications. Springer-Verlag October 2003.

[3] Campbell, S. L., 1995, High-index differential algebraic equations. *Mech. Struct. and Mach.*, 23(2), 199-222.

[4] Gu, B., Asada, H.H., 2001, Co-Simulation of Algebraically Coupled Dynamic Subsystems. *American Control Conference,* 2273-2278.

[5] Kumar, A., and Daoutidis, P.: Control of Nonlinear Differential-Algebraic-Equation Systems with Applications to Chemical Processes. Chapman & Hall/CRC, Research Notes in Mathematics Series 1999.

[6] Brenan, K., Campbell, S., and Petzold, L.: Numerical Solution of Initial Value Problems in Differential-Algebraic Equations. Amsterdam: North Holland 1989.

[7] Gordon, B.W.: State Space Modelling of Differential-Algebraic Systems using Singularly Perturbed Sliding Manifolds. Ph.D Thesis, MIT, Mechanical Engineering Dept., August (1999)

[8] Gordon, B. W., and Asada, H., 2000, Modeling, realization, and simulation of thermo-fluid systems using singularly perturbed sliding manifolds. *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 122, no. 4, 699-707.

[9] Rum, F. and Gordon, B. W., 2004, Simulation of Deformable Objects Using Sliding Mode Control with Application to Cloth Animation. *International Conference on Computational Science,* 2004, 292-299.

[10] Drakunov, S. V. and Utkin, V. I., 1990, Sliding mode in dynamic systems. *Int. J. Contr.*, vol. 55, pp. 1029–1037.

[11] Zhao, F. and Utkin, V. I., 1996, Adaptive simulation and control of variable-structure control systems in sliding regimes. *Automatica, Volume 32, Issue 7, Pages 1037-1042*

[12] Isidori, A. : Nonlinear control systems : an introduction . Berlin; New York : Springer-Verlag, 1989.

[13] Slotine, J.-J.E., 1984, Sliding Controller Design for Nonlinear Systems. *Int. J.*

Control. Vol. 40, page 2.

[14] Utkin, V.I.: Sliding Modes in Optimization and Control. Springer-Verlag 1993.

[15] Papoulis, A. : Signal Analysis. McGraw-Hill 1977.

[16] Reeves, W. T., 1983, Particle Sytems: A technique for modelling a class of fuzzy objects. *Computer Graphics Proceedings SIGGRAPH,* 17, 359-376.

[17] Desbrun, M., Meyer, M., Barr, A.H., 2000, Interactive Animation of Cloth-Like Objects for Virtual Reality. *Journal of Vizualisation and Computer Animation.*

Appendix A: Simulation of Deformable Objects using Sliding Mode Control

A.1 Problem formulation

We model the flexible object as a collection of distributed masses connected to each other via rigid/flexible connections, which is also referred to as a particle system [16]. These models have the ability to capture complex dynamical behaviors and are well suited to animation needs [17]. All the forces either internal or external simply depend on location and velocity of particles therefore in order to simulate such systems we only need to compute forces on each particle and two simple integrations will yield positions and velocities.

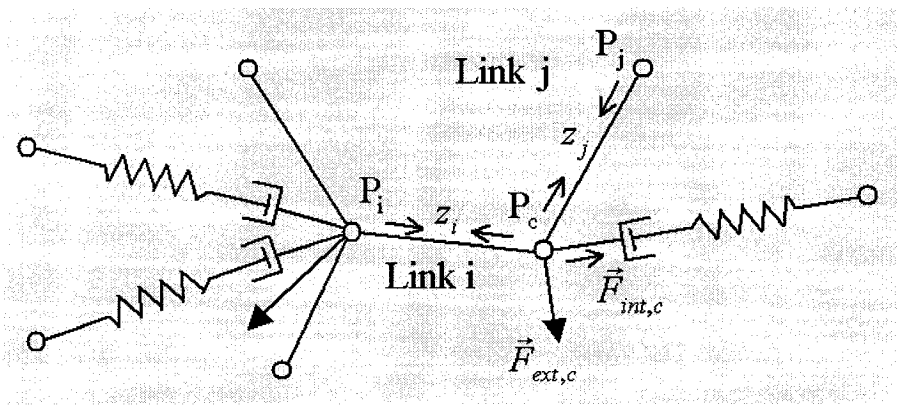


Figure A.1. A generic 3D section of a flexible object modelled as a particle system

As shown in Fig. A.1 we can generally categorize the forces on particles as internal forces due to flexible connections, $\vec{\mathbf{F}}_{int}$, that are responsible for shear and bending behavior of the object, internal forces due to rigid connections, z, and finally the external forces, $\vec{\mathbf{F}}_{ext}$, that represent interaction forces between the object and the environment such as collisions, contacts, gravity, wind, *etc.* Take note that we are not modeling the rigid links as springs but rather keep their forces as unknowns for the controller to determine. As a result we have the following ODE:

$$\begin{cases} \dot{\vec{\mathbf{R}}}_k = \vec{\mathbf{V}}_k \\ \dot{\vec{\mathbf{V}}}_k = \dfrac{1}{m_k}\left(\sum_j z_j \hat{\mathbf{r}}_{kj} + \vec{\mathbf{F}}_{int,k} + \vec{\mathbf{F}}_{ext,k}\right), \quad k = 1,\ldots,n_p \end{cases} \tag{A.1}$$

with these constraints

$$0 = L_i - L_i^0, \quad i = 1,\ldots,n_l \ . \tag{A.2}$$

Here $\vec{\mathbf{R}}_k$ and $\vec{\mathbf{V}}_k$ represent the position and velocity of each particle, n_p is the number of particles, $\hat{\mathbf{r}}_{kj} = \dfrac{\vec{\mathbf{R}}_j - \vec{\mathbf{R}}_k}{\left\|\vec{\mathbf{R}}_j - \vec{\mathbf{R}}_k\right\|}$ is the unit vector from particle k to j that is at the other end of the rigid link connecting them. For constraints, L_i is the instantaneous length of the i^{th} link, L_i^0 is its desired length and n_l represents the total number of links.

By permitting the length of links to change as much as ε_i the constraints will be the following inequalities:

$$\left|L_i - L_i^0\right| \le \varepsilon_i, \quad i = 1,...,n_l \ . \tag{A.3}$$

A.2 Designing the SPSM controller

We start application of the SPSM method by introducing the following error variable:

$$w_i = L_i - L_i^0 \tag{A.4}$$

Differentiating w.r.t. time (see Fig. A.1) one obtains:

$$\dot{w}_i = \hat{\mathbf{r}}_{ci}.\left(\vec{\mathbf{V}}_i - \vec{\mathbf{V}}_c\right) \tag{A.5}$$

$$\ddot{w}_i = \frac{\left\|\vec{\mathbf{V}}_i - \vec{\mathbf{V}}_c\right\|^2 - \dot{w}_i^{\,2}}{L_i} + \hat{\mathbf{r}}_{ci}.\left(\vec{\dot{\mathbf{V}}}_i - \vec{\dot{\mathbf{V}}}_c\right). \tag{A.6}$$

Since

$$\vec{\dot{\mathbf{V}}}_c = \frac{1}{m_c}\left(\sum_j z_j \hat{\mathbf{r}}_{cj} + \vec{\mathbf{F}}_{int,c} + \vec{\mathbf{F}}_{ext,c}\right) \tag{A.7}$$

We can see that z terms appear in $\ddot{w}$. Therefore, according to definition of index of a DAE [6] our problem is of index three. The sliding surface designed by the SPSM method will then be:

$$s_i = \mu^2 \ddot{w}_i + 2\mu \dot{w}_i + w_i \tag{A.8}$$

where μ is a positive parameter that determines the dynamics of the fast motion. The SPSM method then designs a controller that forces the motion to the above desired dynamics. In order to see the effect of μ on error we recall the following result from [13]:

Differentiating $\ddot{w}_i$ w.r.t time and packing the $\dddot{w}$ vector we can write:

$$\dddot{\mathbf{w}} = \mathbf{J}_\Omega \mathbf{v} + \boldsymbol{\beta} \tag{A.9}$$

where

$$\mathbf{J}_\Omega = \frac{\partial \dddot{\mathbf{w}}}{\partial \mathbf{z}} \tag{A.10}$$

Substituting in eq. (A.8) we obtain:

$$\dot{\mathbf{s}} = \mu^2 \ddot{\mathbf{w}} + 2\mu\dot{\mathbf{w}} + \mathbf{w} = \mu^2 \mathbf{J}_\Omega \mathbf{v} + \mu^2 \boldsymbol{\beta} + 2\mu\dot{\mathbf{w}} + \mathbf{w} \tag{A.11}$$

Now defining

$$\mathbf{J}_s = \mu^2 \mathbf{J}_\Omega \tag{A.12}$$

$$\boldsymbol{\alpha} = \mu^2 \boldsymbol{\beta} + 2\mu\dot{\mathbf{w}} + \mathbf{w} \tag{A.13}$$

we can write (A.11) as:

$$\dot{\mathbf{s}} = \mathbf{J}_s \mathbf{v} + \boldsymbol{\alpha} \tag{A.14}$$

Using the results in chapter two we compute $\mathbf{v}$ by the following controller:

$$\mathbf{v} = -\hat{\mathbf{J}}_s^{-1}\left(\hat{\boldsymbol{\alpha}} + \mathbf{K}\,\mathrm{diag}\left[\mathrm{sat}\left(\frac{s_i}{\varepsilon_i}\right)\right]\right) \tag{A.15}$$

the motion will converge to its desired error bound after a short reaching phase and will stay there thereafter, if the following conditions are satisfied:

1- $\mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K} - \left|\,\mathrm{diag}[\boldsymbol{\alpha} - \mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\hat{\boldsymbol{\alpha}}]\,\right|$ has to be positive diagonally dominant.

2- $\mathbf{J}_s\hat{\mathbf{J}}_s^{-1}\mathbf{K}$ must be uniformly positive definite. (A.16)

The sat(.) function used in eq. (A.15) is indeed the linear saturation function shown in Fig. A.2 and is given by:

$$\text{sat(u)} = \begin{cases} \text{u} & |\text{u}| < 1 \\ \text{sign(u)} & |\text{u}| \geq 1 \end{cases} \tag{A.17}$$

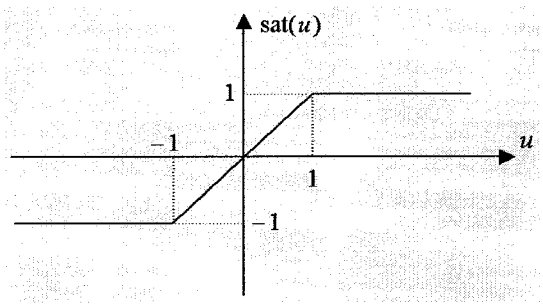


Figure A.2. The linear saturation function

It is used to smooth the control and help us avoid the chattering phenomenon [13] common to sliding mode control methods.

In this appendix we simply invert the real jacobian matrix, $\hat{\mathbf{J}}_s^{-1} = \mathbf{J}_s^{-1}$, thus reducing (A.16) to:

1- $\mathbf{K}$ - $\left|\, \text{diag}[\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}] \,\right|$ has to be positive diagonally dominant.

2- $\mathbf{K}$ must be uniformly positive definite. (A.18)

The above criteria can then be easily satisfied by a large enough $\mathbf{K}$. All we are left with is choosing $\hat{\boldsymbol{\alpha}}$ and computing the $\mathbf{J}_s$ matrix.

A.2.1 Choice of $\hat{\boldsymbol{\alpha}}$

The exact expression for $\boldsymbol{\alpha}$ is given by eq. (A.13):

$$\boldsymbol{\alpha} = \mu^2\boldsymbol{\beta} + 2\mu\ddot{\mathbf{w}} + \dot{\mathbf{w}}$$

One can notice that $\ddot{\mathbf{w}}$ and $\mathbf{w}$ have already been evaluated in the process of calculating $\mathbf{s}$ and choice of

$$\hat{\boldsymbol{\alpha}} = 2\mu\ddot{\mathbf{w}} + \dot{\mathbf{w}} \qquad \text{(A.19)}$$

does not involve much computational overhead. If we further take our gain matrix to be diagonal, $\mathbf{K} = \text{diag}(k_i)$, the only sufficient condition we need to meet becomes:

$$k_i \geq \mu^2|\beta_i| \qquad \text{(A.20)}$$

Given the fact that β_i mainly depends on the ODE (A.9) that is under control we do not have to retune the gains each time we try a new value for μ.

A.2.2 Computing the $\mathbf{J}_s$ matrix

Consider the generic link i in Fig. A.1 and the particle, P_c connecting it to link j. Using definition (A.10) with equations (A.6) and (A.7) yields:

$$[\mathbf{J}_\Omega]_{i,j} = \begin{cases} -\dfrac{\hat{\mathbf{r}}_{ci} \cdot \hat{\mathbf{r}}_{cj}}{m_c} & \text{if } i \neq j \\ -\dfrac{1}{m_{i1}} - \dfrac{1}{m_{i2}} & \text{if } i = j \text{ and the none of the two end particles of link i has a acceleration constraint} \\ -\dfrac{1}{m_{i,free}} & \text{if } i = j \text{ and only one end particle of link i does not have an acceleration constraint} \\ 0 & \text{if links i and j have no nodes in common} \end{cases} \tag{A.21}$$

In the above equation masses of the two end particles of link i are denoted by m_{i1} and m_{i2} and mass of the particle in link i that does not have an acceleration constraint is represented by $m_{i,free}$. Examples of an acceleration constraint could include when the particle is fixed at its place or when it is attached to another object, which is considerably more massive compared to the cloth. In the latter situation the acceleration of the attaching particle is mainly governed and constrained by the corresponding point in that object, for example consider the attachment points in animation of parachute or sail for cloth, or the connecting point of hair to an object.

Finally take note that the size of the jacobian matrix that has to be inverted is equal to the number of rigid links, n_l.

Remark A.1. In simulations that we performed, the algorithm proved to be robust against programming errors that yielded a slightly wrong $\mathbf{J}_s$. Aside from the fact that an amount of error in permitted by eq. (A.16), if the user makes a mistake in recognizing if a particle's acceleration is or is not constrained, the constraint could be considered as a neglected external force on that particle. This simply induces an error in $\hat{\boldsymbol{\alpha}}$ and as demonstrated by (A.18) can be robustly cancelled by choosing a big $\mathbf{K}$, which apparently does not involve any additional overhead. This fact can be especially handy when our flexible object dynamically changes its connections with other objects, *e.g.* when the sails are torn and taken away by a strong wind! A more common case happens in interactive animations where some points of the flexible object are dynamically chosen and moved by the user.

Let us now illustrate the method by applying it to a simple case. It can be considered as a model for hair or chain simulation.

A.2.3 Example

Consider the simple particle system shown in Fig. A.3 consisting of two rigid links. ODE (A.1) in this case is:

$$\begin{cases} \dot{\vec{\mathbf{R}}}_1(t) = \vec{\mathbf{V}}_1(t) \\ m_1 \dot{\vec{\mathbf{V}}}_1(t) = -z_1 \hat{\mathbf{r}}_{01}(t) + z_2 \hat{\mathbf{r}}_{12}(t) - m_1 g \hat{k} \\ \dot{\vec{\mathbf{R}}}_2(t) = \vec{\mathbf{V}}_2(t) \\ m_2 \dot{\vec{\mathbf{V}}}_2(t) = -z_2 \hat{\mathbf{r}}_{12}(t) - m_2 g \hat{k} \end{cases} \tag{A.22}$$

with inequalities:

$$\begin{cases} \left| L_1(t) - L_1^0 \right| \le \varepsilon_1 \\ \left| L_2(t) - L_2^0 \right| \le \varepsilon_2 \end{cases} \tag{A.23}$$

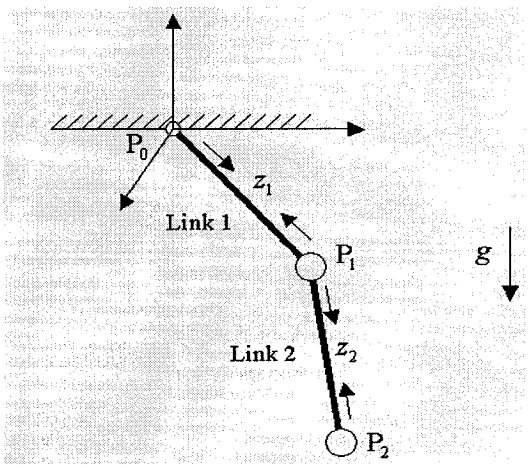


Figure A.3. A simple particle system for hair

Particle 0 of link 1 is fixed and has its acceleration constrained to zero and its other end that is particle 1 is not constrained; therefore, $\left[\mathbf{J}_\Omega\right]_{11} = -\frac{1}{m_1}$. Link 2 does not have any

constrained particles at its ends and therefore $[\mathbf{J}_\Omega]_{22} = -\frac{1}{m_1} - \frac{1}{m_2}$. Particle 1 is shared by the two links and so the jacobian matrix in this case becomes:

$$\mathbf{J}_s = \mu^2 \begin{bmatrix} -\frac{1}{m_1} & -\frac{\hat{\mathbf{r}}_{10}.\hat{\mathbf{r}}_{12}}{m_1} \\ -\frac{\hat{\mathbf{r}}_{10}.\hat{\mathbf{r}}_{12}}{m_1} & -\frac{1}{m_1} - \frac{1}{m_2} \end{bmatrix}. \qquad \text{(A.24)}$$