

An Event-Driven Method to Simulate Filippov Systems with Accurate Computing of Sliding Motions

PETRI T. PIIROINEN

National University of Ireland, Galway
and

YURI A. KUZNETSOV

Utrecht University and IMPB RAS, Pushchino

This article describes how to use smooth solvers for simulation of a class of piecewise smooth systems of ordinary differential equations, called Filippov systems, with discontinuous vector fields. In these systems constrained motion along a discontinuity surface (so-called sliding) is possible and requires special treatment numerically. The introduced algorithms are based on an extension to Filippov's method to stabilise the sliding flow together with accurate detection of the entrance and exit of sliding regions. The methods are implemented in a general way in MATLAB and sufficient details are given to enable users to modify the code to run on arbitrary examples. Here, the method is used to compute the dynamics of three example systems, a dry-friction oscillator, a relay feedback system and a model of an oil well drill-string.

Categories and Subject Descriptors: G.1.7 [Numerical Analysis]: Ordinary Differential Equations—*Initial value problems*; G.1.0 [Numerical Analysis]: General—*Numerical algorithms*; J.2 [Computer Applications]: Physical Sciences and Engineering

General Terms: Algorithms

Additional Key Words and Phrases: Numerical simulation, piecewise-smooth differential equations, Filippov systems, sliding solutions

ACM Reference Format:

Piironen, P. T. and Kuznetsov, Y. A. 2008. An event-driven method to simulate Filippov systems with accurate computing of sliding motions. *ACM Trans. Math. Softw.*, 34, 3, Article 13 (May 2008), 24 pages. DOI = 10.1145/1356052.1356054 <http://doi.acm.org/10.1145/1356052.1356054>

This work was supported by the EU EP5 Project Siconos (Grant no. IST-2001-37172).

Authors' addresses: P. T. Piironen, Department of Mathematical Physics, National University of Ireland, Galway, Ireland; email: petri.piironen@nuigalway.ie; Y. A. Kuznetsov, Department of Mathematics, Utrecht University, The Netherlands; and Institute of Mathematical Problems of Biology, Russian Academy of Sciences, Pushchino, Moscow Region, 142290 Russia; email: yu.kuznetsov@math.uu.nl.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org. © 2008 ACM 0098-3500/2008/05-ART13 \$5.00 DOI 10.1145/1356052.1356054 <http://doi.acm.org/10.1145/1356052.1356054>

1. INTRODUCTION

Reliable numerical tools for simulation of mechanical (e.g., gears and brakes) and electrical (e.g., relay systems and DC-DC converters) systems play an important part in the analysis and development of such systems. What characterises these systems is that they are often modelled by sets of ordinary differential equations (ODEs) of varied complexity. There is a wide variety of numerical methods for solving ODEs and many of them are routinely used in established software, for example, MATLAB [Shampine and Reichelt 1997; Ashino et al. 2000]. However, most of these algorithms require that the ODEs are sufficiently smooth, while more realistic models such as those mentioned above typically include some kind of discontinuities. Systems with discontinuities are often referred to as *piecewise smooth* (PWS) systems, where the discontinuities could be either in the states or in the right hand sides (the vector field or its derivatives) of the ODEs (see e.g., Filippov [1988]). These kinds of systems require special numerical treatment during simulation which will be apparent below.

There are obvious differences in the treatment of systems with state jumps (e.g., due to impacts in mechanical systems) and discontinuous vector fields (e.g., due to switches in electronics). In the present paper we will focus on the latter type, usually referred to as *Filippov systems* (see Filippov [1988], Leine and Nijmeijer [2004]). The most important feature of Filippov systems is the possibility of motions constrained to some subset of the state space. Such constrained motion is often referred to as *sliding* [di Bernardo et al. 1999, 2002] (or *sticking* in the context of friction systems [Galvanetto and Bishop 2000; Galvanetto 2001]).

A first step towards understanding dynamics of piecewise smooth systems is often to perform direct numerical simulation (DNS), where it is of great importance that the time and location of any nonsmooth events are resolved as accurately as possible, for example, in a nonsmooth system solver [Gear and Østerby 1984]. This idea can be compared with another idea for simulating nonsmooth systems, which is to recast the nonsmoothness in terms of a complementarity system formulation [Brogliato 2003]. Then one can use time stepping methods accompanied with linear complementarity problem (LCP) solvers to simulate the systems without the need for accurate event detection. That is, the solver can only note that one or more events have occurred during a time step without finding the actual event time and location (see further Moreau [1999], Jean [1999]). Such methods have proven to be effective in simulating mechanical systems with a large number of constraints. However, they suffer from the disadvantage that they are typically only low-order algorithms and nonsmooth events can be lost. The focus of this work is to accurately detect the interaction of smooth and nonsmooth dynamics. Therefore we require high order algorithms to solve the smooth vector fields, a way of stabilising the sliding flow, and an accurate event detection algorithm. Such a simulator can then be used to compute Poincaré maps and to continue limit-cycles and their sliding bifurcations under parameter changes in general Filippov systems, as fixed points of these maps. Up to now, most analysis of Filippov systems has been

(semi-)analytic, and usually limited to small systems [di Bernardo et al. 2002]. However, a numerical tool that has successfully been implemented to analyse Filippov systems is SLIDECONT [Dercole and Kuznetsov 2005]. It is based on the widely used numerical package AUTO [Doedel et al. 1997] that can continue solutions to nonlinear boundary-value problems via orthogonal collocation. To some extent, SLIDECONT has the ability to continue equilibria, limit cycles, and their sliding bifurcations, but to date it still lacks the capability to perform DNS of Filippov systems and automatically switch between sliding and nonsliding motions. In this paper an algorithm for simulation of Filippov systems will be presented that precisely fills this gap; that is, it solves for the sliding flow directly and automatically switches between free and constrained motion. Since only smooth systems are to be solved numerically, well known methods together with appropriate error estimates can be applied between the switches.

The rest of the article is organized as follows. In Section 2 we introduce and define Filippov systems and explain what characterises them. A description of the numerical algorithm for simulating these Filippov systems is presented in Section 3. Three examples, a dry-friction oscillator, a relay feedback system and a drill-string model, are presented in Section 4, along with instructions for users who want to use the downloadable programs. The actual MATLAB files used for simulation of Filippov systems are presented and explained in Appendix A, including instructions for potential users. Finally, in Section 5 we summarise this paper with some concluding remarks regarding the proposed algorithm, for example, we mention situations where the algorithm might possibly face some problems. We also discuss future work on methods to locate and continue limit cycles and sliding bifurcations of Filippov systems.

2. FILIPPOV SYSTEMS

As mentioned in Section 1 we will consider dynamical systems defined by discontinuous vector fields, so called Filippov systems. What characterises such a system is the division of the state space into disjoint subregions, such that in each such region the defining vector field is smooth. The boundaries between the different regions will be referred to as *discontinuity surfaces*. In this section only a brief introduction to Filippov systems will be given, and for a more thorough exposition of this topic see Filippov [1988], di Bernardo et al. [2002], Kuznetsov et al. [2003], Leine and Nijmeijer [2004].

A general dynamical system can be written as

$$\dot{x} = f(x), \quad x \in \mathbf{R}^n, \quad (1)$$

where the vector field $f(x)$ can be either smooth or piecewise smooth. Let us first assume that the state space consists of only two regions, S_i and S_j , separated by a discontinuity surface Σ_{ij} , which is defined by a smooth scalar function $h_{ij}(x)$ such that

$$\Sigma_{ij} = \{x \in \mathbf{R}^n \mid h_{ij}(x) = 0\}, \quad (2)$$

and where

$$S_i = \{x \in \mathbf{R}^n \mid h_{ij}(x) > 0\} \quad \text{and} \quad S_j = \{x \in \mathbf{R}^n \mid h_{ij}(x) < 0\}. \quad (3)$$

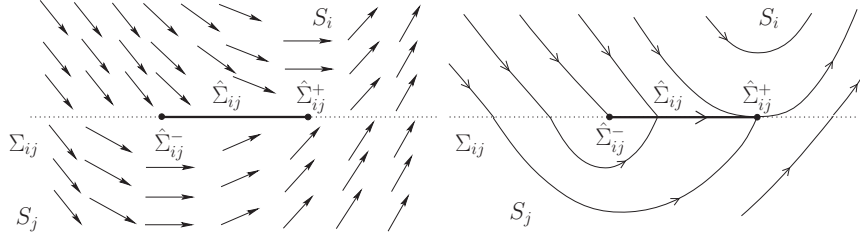


Fig. 1. The vector fields F_i and F_j near a stable sliding region $\hat{\Sigma}_{ij}$ (left) and some corresponding trajectories (right).

Hence, (1) can be rewritten as

$$\dot{x} = \begin{cases} F_i(x), & x \in S_i, \\ F_j(x), & x \in S_j, \end{cases} \quad (4)$$

where F_i and F_j are sufficiently smooth. In the rest of this article it is also assumed that F_i and F_j are defined in the whole state space, even if they are only used in their respective regions.

If the vector fields F_i and F_j are locally both pointing away from or both pointing towards the discontinuity surface Σ_{ij} the dynamics is assumed to be locally constrained to the surface, as depicted in the left part of Figure 1, and the motion is said to be sliding. The open subset $\hat{\Sigma}_{ij}$ of the surface Σ_{ij} where the vector fields are both pointing towards or away from Σ_{ij} is often referred to as the *sliding surface*. In Figure 1 the sliding surface $\hat{\Sigma}_{ij} \subset \Sigma_{ij}$ is a line segment between two points, $\hat{\Sigma}_{ij}^-$ and $\hat{\Sigma}_{ij}^+$.

For notational reasons, it is useful here to define the *Lie derivative* $\mathcal{L}_F(h)(x)$ of a smooth function $h(x)$ along a smooth vector field $F(x)$ as

$$\mathcal{L}_F(h)(x) := \left. \frac{d}{dt} h(\xi(t)) \right|_{t=0} = \left. \frac{dh(x)}{dx} \frac{d\xi(t)}{dt} \right|_{t=0} = \left\langle \frac{dh(x)}{dx}, F(x) \right\rangle, \quad (5)$$

where $\xi(t)$ a solution to $\dot{\xi} = F(\xi)$ that satisfies $\xi(0) = x$ (see, e.g., Arnol'd [1973]). Thus, if it holds that

$$\mathcal{L}_{F_i - F_j}(h_{ij})(x) < 0, \quad x \in \hat{\Sigma}_{ij}, \quad (6)$$

then $\hat{\Sigma}_{ij}$ is stable, while if

$$\mathcal{L}_{F_i - F_j}(h_{ij})(x) > 0, \quad x \in \hat{\Sigma}_{ij}, \quad (7)$$

the sliding surface is unstable.

Using *Utkin's equivalent control* [Utkin 1992] the dynamical system (4) can be extended to include the vector field on the sliding surface such that

$$\dot{x} = F_{ij}(x), \quad x \in \hat{\Sigma}_{ij}, \quad (8)$$

where

$$F_{ij}(x) = \frac{F_i(x) + F_j(x)}{2} + \frac{F_j(x) - F_i(x)}{2} \mu_{ij}(x), \quad (9)$$

and $-1 \leq \mu_{ij}(x) \leq 1$. Since the motion is constrained to $\hat{\Sigma}_{ij}$, F_{ij} must be tangent to $\hat{\Sigma}_{ij}$, i.e., $\mathcal{L}_{F_{ij}}(h_{ij})(x) = 0$, which yields

$$\mu_{ij}(x) = -\frac{\mathcal{L}_{F_i+F_j}(h_{ij})(x)}{\mathcal{L}_{F_j-F_i}(h_{ij})(x)}. \quad (10)$$

It is notable that $F_{ij} = F_i$ and $F_{ij} = F_j$ when $\mu_{ij}(x) = -1$ and $\mu_{ij}(x) = 1$, respectively, and also $\mathcal{L}_{F_j-F_i}(h_{ij})(x) \neq 0$ for $x \in \hat{\Sigma}_{ij}$ since the vector fields always point toward or away from the discontinuity surface. The surfaces defining the borders of the sliding surface are thus given by

$$\hat{\Sigma}_{ij}^- = \{x \in \hat{\Sigma}_{ij} \mid \mu_{ij}(x) = -1\} \text{ and } \hat{\Sigma}_{ij}^+ = \{x \in \hat{\Sigma}_{ij} \mid \mu_{ij}(x) = 1\}. \quad (11)$$

These borders are composed of so called *tangent points* [Kuznetsov et al. 2003] and will be referred to as *tangent surfaces* (cf. the points $\hat{\Sigma}_{ij}^-$ and $\hat{\Sigma}_{ij}^+$ in Figure 1).

We next define solutions of the Filippov system (4) by concatenating standard solutions in S_i and S_j and (maximal) sliding solutions in $\hat{\Sigma}_{ij}$ (see the right part of Figure 1). To assure the uniqueness of such a solution, it is sufficient to assume that it does not visit points of Σ_{ij} , where *both* vectors F_i and F_j are tangent to Σ_{ij} . In what follows, we call such solutions and their corresponding orbits *generic*.

Finally, let us discuss briefly a relationship between the above described construction and the commonly used *Filippov's convex method* [Filippov 1988] (see further Section 3.2). The original Filippov approach consists of replacing (4) by the differential inclusion

$$\dot{x} \in \begin{cases} \{F_i(x)\}, & x \in S_i, \\ \overline{\text{co}}(F_i(x), F_j(x)), & x \in \Sigma_{ij}, \\ \{F_j(x)\}, & x \in S_j, \end{cases} \quad (12)$$

where $\overline{\text{co}}(F_i, F_j)$ is the minimal closed convex set containing F_i and F_j ; that is,

$$\overline{\text{co}}(F_i, F_j) = \{f \in \mathbf{R}^n : f = F_i + \lambda(F_j - F_i), \lambda \in [0, 1]\}.$$

A solution to this differential inclusion is an absolutely continuous function $x(t)$ that satisfies (12) for almost all t from its domain of definition. Under assumed smoothness of F_{ij} and h_{ij} , Theorem 2 by Filippov [1988] (Chapter 2, pp. 110–111) implies forward uniqueness of those solutions to (12), which do not visit unstable sliding surfaces, and those points of Σ_{ij} where both vectors F_i and F_j are tangent to Σ_{ij} . Moreover, this unique solution coincides with the one constructed above. It should be noted that in Filippov's approach solutions starting at unstable sliding surfaces are not unique, while they generically have this property in our formulation, where they are constrained to $\hat{\Sigma}_{ij}$. This difference is not important in most applications, since solutions starting away from the unstable sliding surface can never reach it.

3. THE EVENT-DRIVEN SIMULATION METHOD

Under some circumstances (e.g., if small linear systems are considered) it is certainly possible to find explicit expressions for the solutions of the ODE that

describes sliding if the vector fields in the non-sliding regions are given. However, the idea here is to present a numerical algorithm where the user only provides the different vector fields and information about the discontinuity surfaces and then the vector fields for the sliding regions are automatically computed by a routine which uses Equations (2), (8)–(10). The method that has been chosen here to simulate Filippov systems is similar to the hybrid system approach, where integrations of smooth ODEs are mixed with discrete maps and vector field switches. In practice, it means that an initial value problem is solved for one of the possible smooth dynamical systems given in Equation (1) until the orbit reaches one of the predefined surfaces. At such a point the vector field is possibly switched, depending on the state at that time instance (see further Section 3.1.1).

Because of our choice of strategy to integrate Filippov systems, it is very important to have a reliable ODE solver that is accompanied by an accurate routine to locate discontinuity and tangent surface crossings (see Section 2). In what follows a surface crossing will be called an *event* and a scalar function defining an *event surface* is referred to as the *event function*. The existence of event detection routines will be here assumed. For instance in MATLAB event detection routines are built-in and can easily be used together with the likewise built-in ODE solvers to integrate orbits and to locate events along them as precisely as the accuracy of MATLAB allows (for more details of the MATLAB ODE routines, see Champine and Reichelt [1997], Ashino et al. [2000]). However, standard methods, for example, secant type methods, can easily be implemented and have proven to be fast and reliable. The type of events to be detected also play an important role in how to numerically deal with them and how sensitive the event detection needs to be.

In Section 3.1 we will focus on the basic ideas behind the simulation algorithm for one discontinuity surface and also give a schematic description on how the algorithm works. However, in Section 3.2 we give a short explanation on how to extend this method with further surfaces, and in Section 4.3 a system with two discontinuity surfaces will be presented and numerically examined.

3.1 Algorithm for a Single Discontinuity Surface

We can write the full dynamical system with one discontinuity surface Σ_{ij} as

$$\dot{x} = \begin{cases} F_i(x), & x \in S_i, \\ \hat{F}_{ij}(x), & x \in \hat{\Sigma}_{ij}, \\ F_j(x), & x \in S_j, \end{cases} \quad (13)$$

where the sliding vector field $F_{ij}(x)$ is given by (9). In addition, since the dynamics is constrained to $\hat{\Sigma}_{ij}$ during sliding, we substitute $F_{ij}(x)$ by its *regularization* $\hat{F}_{ij}(x)$ in order to avoid drift away from $\hat{\Sigma}_{ij}$. How this is done is discussed and explained further in appendix B.

As mentioned in Section 2 the state space for a Filippov system with one discontinuity surface is locally divided into three disjoint regions, namely, S_i , S_j , and Σ_{ij} . Further, as seen in Figure 2, the state space can also be divided into two regions, \hat{M}_{ij} and M_{ij} , by the two *extended tangent surfaces* Σ_{ij}^- and Σ_{ij}^+ ,

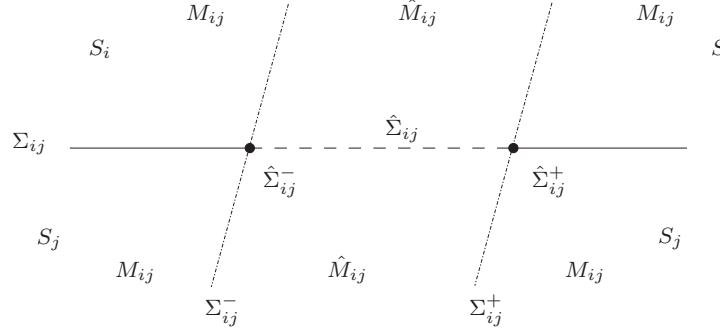


Fig. 2. The different regions that the state space is divided into in a neighbourhood of the sliding surface $\hat{\Sigma}_{ij}$.

which are defined by

$$\Sigma_{ij}^- = \{x \in \mathbf{R}^n \mid \mu_{ij}(x) = -1\} = \{x \in \mathbf{R}^n \mid \mathcal{L}_{F_i}(h_{ij}) = 0\}, \quad (14)$$

$$\Sigma_{ij}^+ = \{x \in \mathbf{R}^n \mid \mu_{ij}(x) = 1\} = \{x \in \mathbf{R}^n \mid \mathcal{L}_{F_j}(h_{ij}) = 0\}, \quad (15)$$

where μ_{ij} is defined as in Equation (10). For later reference, by using (14) and (15) the two regions \hat{M}_{ij} and M_{ij} are defined as

$$\hat{M}_{ij} = \{x \in \mathbf{R}^n \mid |\mu_{ij}(x)| < 1\} \text{ and } M_{ij} = \{x \in \mathbf{R}^n \mid |\mu_{ij}(x)| \geq 1\}. \quad (16)$$

Notice the similarity with the tangent surfaces defined in Equation (11). Also, since it is assumed that F_i and F_j are defined in the whole state space, μ_{ij} is defined everywhere except for points where both F_i and F_j are tangent to the surface $h_{ij}(x) = \text{const}$. Recall that a subset of such points have been excluded from generic orbits in Section 2.

This division into disjoint subregions makes it relatively straightforward to implement in a numerical algorithm and reduces the number of checks that have to be made every time the discontinuity surface Σ_{ij} is crossed. Also, since we are always looking for the surfaces given by $\mathcal{L}_{F_i}(h_{ij})(x) = 0$ and $\mathcal{L}_{F_j}(h_{ij})(x) = 0$ it makes the algorithm more robust for the location of events for orbits that hit the discontinuity surface almost tangentially.

3.1.1 Event Location, Event Functions, and Event Variables. When simulating Filippov systems using an event-driven scheme it is important to locate events, for example, discontinuity surface or a tangent surface crossing, as accurately as possible (within a given tolerance). Therefore, to make an automatic algorithm robust, specific events need to be predefined as region-dependent. If it is assumed that a system has a total of m possible events in each region then an *event list* $e(x, t)$ can be defined as

$$e(x, t) = (e_1(x, t), \dots, e_m(x, t)), \quad (17)$$

where each element $e_k(x, t) \in \mathbf{R}$ is an event function that defines an event surface that can be reached by the state vector or the time.

Here we define the event functions as

$$e_1(x, t) := h_{ij}(x), \quad e_2(x, t) := \mathcal{L}_{F_i}(h_{ij})(x), \quad e_3(x, t) := \mathcal{L}_{F_j}(h_{ij})(x),$$

(cf. Equations (2), (14), and (15)).

In order for the solver to know what vector field to use and what events to look for, a number of *event variables* will be introduced. Thus, to keep track of which of the regions S_i , S_j , and $\hat{\Sigma}_{ij}$ the state variable $x(t)$ is in we introduce the event variables s_1 , s_2 , and s_3 , respectively. Further, to keep track of if the state variable is in M_{ij} or \hat{M}_{ij} we introduce the event variables s_4 and s_5 , respectively. Letting $s_{ij} = (s_1, s_2, s_3, s_4, s_5)^T$ and using the same state space division as in Figure 2 we can give the event variables the following values

$$s_1(x) = \begin{cases} 1, & x \in S_i, \\ -1, & x \notin S_i, \end{cases} \quad s_2(x) = \begin{cases} 1, & x \in S_j, \\ -1, & x \notin S_j, \end{cases} \quad (18)$$

$$s_3(x) = \begin{cases} 1, & x \in \hat{\Sigma}_{ij}, \\ -1, & x \notin \hat{\Sigma}_{ij}, \end{cases} \quad (19)$$

$$s_4(x) = \begin{cases} 1, & x \in M_{ij}, \\ -1, & x \notin M_{ij}, \end{cases} \quad s_5(x) = \begin{cases} 1, & x \in \hat{M}_{ij}, \\ -1, & x \notin \hat{M}_{ij}. \end{cases} \quad (20)$$

Depending on in what region $x(t)$ is in and to what region it continues the event parameters are changed at that event accordingly. For instance, assume $x \in S_i \cup \hat{M}_{ij}$ before it crosses the discontinuity surface Σ_{ij} then we will have

$$s_{ij} = (1, -1, -1, -1, 1)$$

before the crossing and

$$s_{ij} = (-1, -1, 1, -1, 1)$$

after. This means that the orbit is sliding along $\hat{\Sigma}_{ij}$ after the surface crossing and this information is passed to the solver so that the correct vector field is used in the solving process.

Following this discussion, each of our original event functions e_k can be seen as one of two different kinds, namely $e_k^+(x, t)$ and $e_k^-(x, t)$, where the former corresponds to one in which the surface crossings are only detected when e_k is increasing and the latter when e_k is decreasing.

In Table I we have listed what surface to look for in each region, the list of event variables in each region, and also from which direction a zero crossing of an event function is to be looked for. A dash in the table means that there is no need to look for this zero crossing since this event cannot happen unless another event has occurred first.

3.1.2 User-Defined Event Functions. Often dynamical systems are transformed from nonautonomous to autonomous form by extending the state variables by adding the (scaled) time as a new variable. Typically, if a system is periodic, for instance due to a periodic forcing, with period T it might be of interest to locate the end of the period, where time could for instance be reset to zero, see, for example, Section 4.1. Also, a specific time is often used as a

Table I. Event Variables and Functions

$x \in$	$s_{ij} =$	e_1	e_2	e_3
$S_i \cup M_{ij}$	(1, -1, -1, 1, -1)	$e_1^- = 0$	$e_2^+ = 0$	$e_3^+ = 0$
$S_i \cup \hat{M}_{ij}$	(1, -1, -1, -1, 1)	$e_1^- = 0$	$e_2^\pm = 0$	$e_3^\mp = 0$
$S_j \cup M_{ij}$	(-1, 1, -1, 1, -1)	$e_1^+ = 0$	$e_2^\pm = 0$	$e_3^\pm = 0$
$S_j \cup \hat{M}_{ij}$	(-1, 1, -1, -1, 1)	$e_1^+ = 0$	$e_2^\pm = 0$	$e_3^\mp = 0$
$\hat{\Sigma}_{ij}$	(-1, -1, 1, -1, 1)	—	$e_2^\pm = 0$	$e_3^\mp = 0$

Poincaré section when analysing recurrent dynamics. For instance, if we assume x_k is the state variable of the (scaled) time and t_P is the time that is of interest, we can then introduce a surface Λ_P defined by the function $h_P(x) = 0$, where $h_P(x) = x_k - t_P$. Then it is only necessary to look for Λ_P for increasing values of $h_P(x)$. Notice that it is possible to introduce any other event function of the user's preference, for example to define other Poincaré surfaces.

3.1.3 The Algorithm. In the flow chart in Figure 3 we present an overview of the algorithm for simulating Filippov systems with one discontinuity surface. A description of what the program does in the different boxes is explained in Algorithm 1. Notice that this overview assumes that an event location routine is built in together with the ODE solver.

Algorithm I. A description of the numbered boxes in the flow chart in Figure 3 is given here.

1. Initialize the program with solver type, solver properties, names of user-created files, parameter values, initial conditions and simulation time.
 2. Find the initial event variables s_i for $i = 1, \dots, 5$.
 3. Solve the current ODE until an event occurs or if the final simulation time has been reached. Use vector field F_i if $s_1 = 1$, F_j if $s_2 = 1$ and \hat{F}_{ij} if $s_3 = 1$.
 4. Check if the current time is equal to the final time.
 5. Check in which region the state is in by using the event variables s_i .
 6. —8. Check which event that occurred.
 9. Set $s_1 = -s_1$ and $s_2 = -s_2$ since we are moving from region $S_i \cup M_{ij}$ to $S_j \cup M_{ij}$ (or $S_j \cup M_{ij}$ to $S_i \cup M_{ij}$), and then continue to A. (See Figure 2.)
 10. Set $s_4 = -s_4$ and $s_5 = -s_5$ since we are moving from region $S_i \cup M_{ij}$ to $S_i \cup \hat{M}_{ij}$ (or $S_j \cup M_{ij}$ to $S_j \cup \hat{M}_{ij}$), and continue to A. (See Figure 2.)
 11. Set $s_1 = -1$, $s_2 = -1$ and $s_3 = -s_3$ since we are moving from region $S_i \cup \hat{M}_{ij}$ to $\hat{\Sigma}_{ij}$ (or $S_j \cup \hat{M}_{ij}$ to $\hat{\Sigma}_{ij}$), and continue to A. (See Figure 2.)
 12. Set $s_4 = -s_4$ and $s_5 = -s_5$ since we are moving from region $S_i \cup \hat{M}_{ij}$ to $S_i \cup M_{ij}$ (or $S_j \cup \hat{M}_{ij}$ to $S_j \cup M_{ij}$), and continue to A. (See Figure 2.)
 13. Set $s_1 = -s_1$, $s_3 = -s_3$, $s_4 = -s_4$ and $s_5 = -s_5$ since we are moving from region $\hat{\Sigma}_{ij}$ to $S_i \cup M_{ij}$, and continue to A. (See Figure 2.)
 14. Set $s_2 = -s_2$, $s_3 = -s_3$, $s_4 = -s_4$ and $s_5 = -s_5$ since we are moving from region $\hat{\Sigma}_{ij}$ to $S_j \cup M_{ij}$, and continue to A. (See Figure 2.)
-

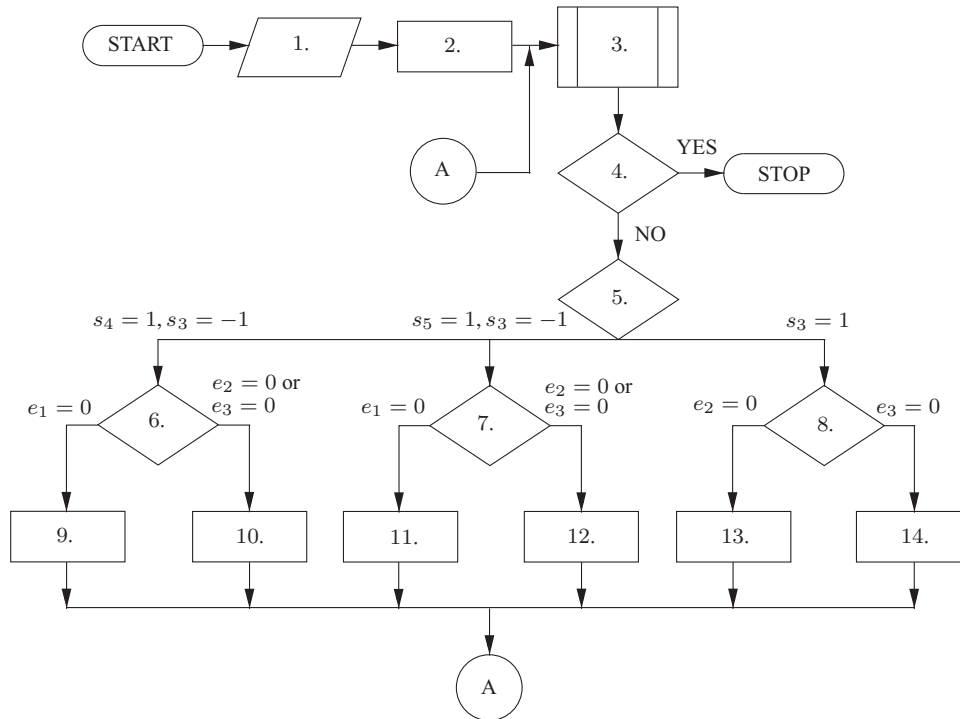


Fig. 3. A schematic flow chart of the algorithm for simulation of Filippov systems with one discontinuity surface. A description of the numbered boxes is given in Algorithm 1.

3.2 A Number of Discontinuity Surfaces

In the previous discussion we have considered only one discontinuity surface. However, it is straightforward to extend the methods to an arbitrary number of surfaces. The only thing one has to do is to ensure uniqueness of orbits constrained to the discontinuity surfaces. This is done by only considering a special class of Filippov systems where the vector fields in the different regions of state space are linearly dependent, as we shall see. To simulate orbits along the various discontinuity surfaces we will here use Filippov's convex method instead of Utkin's equivalence method (as was the case in Section 2) to show how that approach can be implemented but also since it is slightly more straightforward. Notice also that the notation in this section therefore differs somewhat from the one-surface case.

Assume we have a general dynamical system (1) and M discontinuity surfaces Σ_i defined by M functions $h_i(x) = 0$. These surfaces divide the state space into a number of disjoint sets, where the vector fields are different. As mentioned above we have specific rules on what the vector fields can be to guarantee uniqueness of orbits constrained to the discontinuity surfaces. The idea we propose here is to introduce a base vector field that is valid when $h_i(x) > 0$ for all i and for each surface crossing we make an addition Δ_i to the base vector field that is used after the surface crossing. Using this idea and Filippov's

convex method, the vector field for the whole domain, including the sliding regions, can be written as

$$F(x) = F_{\text{orig}}(x) + \sum_{i=1}^M \Delta_i(x) \mu_i(x), \quad (21)$$

where

$$\mu_i(x) = \begin{cases} 1, & h_i(x) < 0, \\ [0, 1] & h_i(x) = 0, \\ 0, & h_i(x) > 0. \end{cases} \quad (22)$$

For each surface we have thus defined a function $\mu_i(x)$, in a similar way to the single surface case, that determines the active vector field (cf. Equations (9) and (12)).

In the same way as described earlier in Section 3.1.1 the program has to keep track of whether surfaces have been crossed or not, so that the correct values of the μ_i s are used, and thus the correct vector field is integrated. The only difference from the one-surface case is that each discontinuity surface requires its own set of event variables and event functions.

Since sliding can occur along more than one discontinuity surface simultaneously, the values of the corresponding $\mu_i \in [0, 1]$ have to be determined. This can be done by using the same approach as in the one-surface case. To give an example, and without lack of generality, assume that the orbit will slide along the surfaces Σ_i and Σ_j . Then we know that $h_i(x) = 0$ and $h_j(x) = 0$, respectively, and we want $\mathcal{L}_F(h_i)(x) = 0$ and $\mathcal{L}_F(h_j)(x) = 0$ to hold. By using these conditions, eqs. (21) and (22) we get that

$$\begin{pmatrix} \mu_i(x) \\ \mu_j(x) \end{pmatrix} = - \begin{pmatrix} \mathcal{L}_{\Delta_i} h_i(x) & \mathcal{L}_{\Delta_j} h_i(x) \\ \mathcal{L}_{\Delta_i} h_j(x) & \mathcal{L}_{\Delta_j} h_j(x) \end{pmatrix}^{-1} \begin{pmatrix} \mathcal{L}_F(h_i)(x) \\ \mathcal{L}_F(h_j)(x) \end{pmatrix}, \quad (23)$$

where

$$\tilde{F}(x) = F_{\text{orig}}(x) + \sum_{k=1, k \neq i, j}^M \Delta_k(x) \mu_k(x). \quad (24)$$

Furthermore, a similar regularization of the sliding vector field as discussed in Appendix B (see especially Equation (50)) is possible by introducing M constants C_i , where each such constant has a positive value if the systems is sliding along the i th surface and is zero otherwise.

The main disadvantage with this approach is that the number of surfaces and event locations grow quickly with the number of discontinuity surfaces which naturally increases the simulation time. However, this approach for two surfaces has been implemented in the drill-string example in Section 4.3.

4. EXAMPLES

In this section we will present some results obtained by using the implemented programs described in this paper, and presented in Appendix A, to simulate three different systems. These systems have been chosen to highlight the variety of Filippov systems and different dynamical behaviour that the simulation

tool can handle. The first system is a nonlinear dry-friction oscillator with one discontinuity surface that has a rich dynamics and characteristic *stick-slip* motion. The second system is a linear relay feedback system with one discontinuity surface that exhibits chaotic behaviour. The third example is a nonlinear drill-string system with two discontinuity surfaces. For these examples we will give all necessary information so that the MATLAB simulation routines in Appendix A can be used. In all examples the ODE-solver ode45 (4th-order Runge-Kutta) has been used with MATLAB's built-in event detection routines. Also, all files used in these examples can be downloaded from <http://www.maths-physics.nuigalway.ie/users/Piironen/Filippov/>.

4.1 A Dry-Friction Oscillator

Stick-slip motion is a well-known, but not fully understood, behaviour in many mechanical systems with friction. The most simple examples showing this kind of behaviour are dry-friction oscillators. Therefore they have drawn a lot of attention and been widely studied. Here we will focus on an undamped dry-friction oscillator with one degree of freedom given by

$$\ddot{y} + y = \sin(\omega t) - F \operatorname{sgn}(\dot{y}), \quad (25)$$

where ω is the forcing frequency and F is the magnitude of the Coloumb friction force. This system has been described by Feigin [1994] and extensively analysed by Kowalczyk [2003] and Kowalczyk and Piironen [2006].

To be able to use the proposed strategy for integrating (25) we will consider the equivalent autonomous first order system

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F \operatorname{sgn}(x_2) \\ \omega \end{pmatrix}, \quad (26)$$

where $x = (x_1, x_2, x_3)^T = (y, \dot{y}, \omega t \bmod 2\pi)^T$. This implies that the discontinuity surface Σ_{12} is defined as

$$\Sigma_{12} = \{x \in \mathbf{R}^3 \mid H_{12}(x) = 0\},$$

which divides the state space into two disjoint regions

$$S_1 = \{x \in \mathbf{R}^3 \mid H_{12}(x) > 0\}, \quad S_2 = \{x \in \mathbf{R}^3 \mid H_{12}(x) < 0\},$$

where

$$H_{12}(x) = x_2.$$

Now it is easy to rewrite (26) as

$$\dot{x} = \begin{cases} F_1(x), & x \in S_1 \\ F_2(x), & x \in S_2, \end{cases} \quad (27)$$

where

$$F_1 = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F \\ \omega \end{pmatrix}, \quad F_2 = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) + F \\ \omega \end{pmatrix}.$$

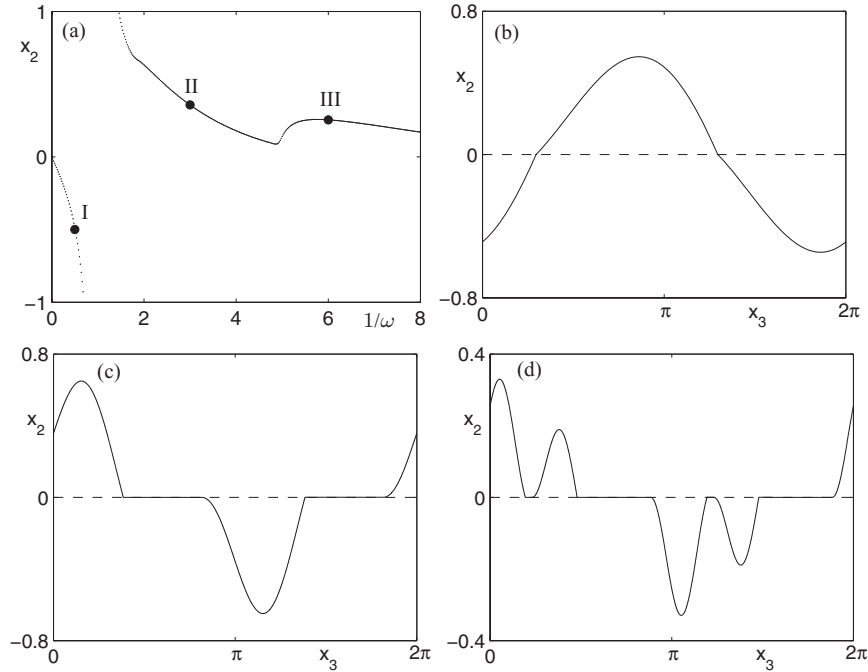


Fig. 4. Bifurcation diagram and time histories for the dry-friction oscillator. (a) A brute-force bifurcation diagram for recurrent motions, where x_2 is plotted against $1/\omega$ at the Poincaré section $x_3 = 2\pi$. The parameter values at the indicated points I, II, and III, are $(F = 0.4, \omega = 0.5)$, $(F = 0.4, \omega = 3)$ and $(F = 0.4, \omega = 6)$, respectively. The panels (b), (c) and (d) show the limit cycles corresponding respectively to the points I, II, and III.

The first approach to analyse the dynamics of a system of this kind is often to make a parameter sweep and create a brute-force bifurcation diagram, which could reveal stable attractors for the given parameter ranges. In Figure 4(a) a brute-force bifurcation diagram of the system (27) is depicted, where the parameter F was kept fixed at 0.4 and the frequency ω varied. For each value of the frequency the system was integrated for 500 forcing periods (corresponding to an integration time $500 \times 2\pi/\omega$); for the final 100 periods the variable x_2 was recorded and plotted every time the orbit reached the Poincaré section defined by the function

$$h_P(x) = x_3 - 2\pi.$$

It is clear from Figure 4(a) that we have a large effect at $\omega = 1$ due to resonance, as expected. Further we see that there are drastic changes in the bifurcation diagram at $1/\omega \approx 1.8$ and $1/\omega \approx 4.8$. To understand how the dynamics differ from one parameter range to the next the limit cycles corresponding to the frequencies at the Roman numerals I, II, and III in Figure 4(a) were located and are shown in Figure 4(b), (c), and (d), respectively. In Figure 4(b), where $1/\omega = 0.5$, we see a limit cycle without any sliding segments, but as we increase $1/\omega$ to 3 (point II in Figure 4(a)) we clearly see in Figure 4(c) that the limit cycle has two segments of sliding motion. Finally, as we increase $1/\omega$ further to 6 (point III

in Figure 4(a)) the limit cycle includes an even greater number of sliding segments, as seen in Figure 4(d). These results indicate that at least two nonsmooth transitions (or sliding bifurcations [Kuznetsov et al. 2003], [di Bernardo et al. 2002]) have occurred along the parameter sweep. The proposed simulation algorithm can now be extended to accurately detect where these transitions occur. In fact, in Kowalczyk and Piironen [2006] the transition points of this particular system are accurately detected and the sliding bifurcations are continued under parameter variations, and a two-parameter bifurcation diagram (in F and ω) is presented to show how the sliding bifurcations organise the global dynamics.

This particular example shows that the proposed algorithm can be used not only for direct numerical simulations of systems with sliding motion but also as a building block for continuation algorithms that follow branches of sliding bifurcations. The results have helped us to understand what happens to the dynamics in mechanical systems with sliding segments (also referred to as *stick-slip* motion).

4.2 A Relay Feedback System

Relay feedback is one of the most commonly used control techniques in practical applications, such as temperature control and mechanical and electromechanical systems. A single-input single-output relay feedback system can be written as

$$\dot{x} = Ax + Bu, \quad (28)$$

$$y = Cx, \quad (29)$$

$$u = -\text{sgn}(y), \quad (30)$$

or

$$\dot{x} = \begin{cases} Ax - B, & Cx > 0 \\ Ax + B, & Cx < 0 \end{cases} \quad (31)$$

where $x \in \mathbf{R}^n$ is the state vector, and $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{n \times 1}$, $C \in \mathbf{R}^{1 \times n}$ are constant matrices. Here we will take a closer look at a particular system analysed by di Bernardo et al. [2001], where

$$A = \begin{pmatrix} -(2\zeta\omega + 1) & 1 & 0 \\ -(2\zeta\omega + \omega^2) & 0 & 1 \\ -\omega^2 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ -2\sigma \\ 1 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad (32)$$

and where $x = (x_1, x_2, x_3)^T$ is the state vector and $h \in \mathbf{R}$ is the control parameter. From this we can also conclude that the discontinuity surface Σ_{12} is defined by

$$\Sigma_{12} = \{x \in \mathbf{R}^3 \mid H_{12}(x) = 0\}, \quad (33)$$

and the two disjoint regions are thus given by

$$S_1 = \{x \in \mathbf{R}^3 \mid H_{12}(x) > 0\} \quad \text{and} \quad S_2 = \{x \in \mathbf{R}^3 \mid H_{12}(x) < 0\}, \quad (34)$$

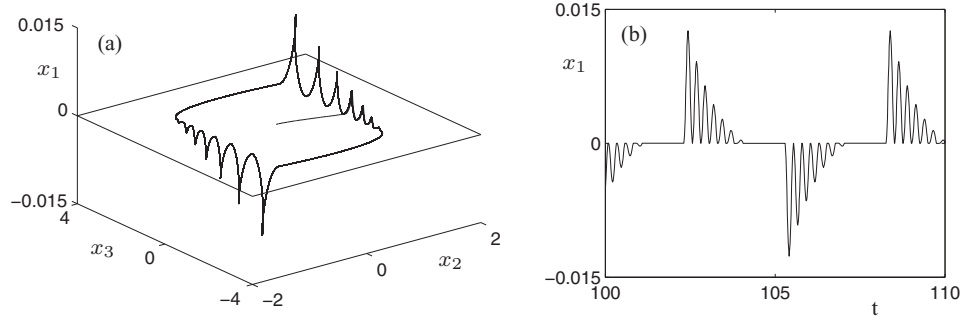


Fig. 5. State-space and time-history diagrams for the relay feedback system. (a) A state space diagram and (b) a time-history diagram of the state variable x_1 showing the eventual stable periodic motion, for $\zeta = 0.05$ and $\omega = 25$.

where

$$H_{12}(x) = x_1. \quad (35)$$

To be able to use our method we write the dynamical system (31) as

$$\dot{x} = \begin{cases} F_1(x), & x \in S_1, \\ F_2(x), & x \in S_2, \end{cases} \quad (36)$$

where the two vector fields are

$$F_1(x) = \begin{pmatrix} -(2\zeta\omega + 1)x_1 + x_2 - 1 \\ -(2\zeta\omega + \omega^2)x_1 + x_3 + 2 \\ -\omega^2x_1 + x_2 - 1 \end{pmatrix}, \quad F_2(x) = \begin{pmatrix} -(2\zeta\omega + 1)x_1 + x_2 + 1 \\ -(2\zeta\omega + \omega^2)x_1 + x_3 - 2 \\ -\omega^2x_1 + x_2 + 1 \end{pmatrix}. \quad (37)$$

In Figure 5(a) we see the dynamics, and eventually the limit cycle, for a particular parameter combination ($\zeta = 0.05$, $\omega = 25$) showing both unconstrained and sliding motions. The initial condition is on the sliding surface, as seen in Figure 5. This does not represent a problem for the simulator. Note also the large number of events that take place per period including 18 separate departures from the sliding surface.

A brute force bifurcation diagram, where $\zeta = -0.07$ was held fixed and σ varied, is shown in Figure 6(a). There we see a period-adding sequence where high periodic motion or chaos is observed between periodic windows, and where each period include many intervals of sliding motion. Similar period-adding behaviour can also be seen in impacting systems and piecewise linear maps. In Figure 6(b) chaotic motion is depicted for $\zeta = -0.07$ and $\sigma = 10$ to highlight the great complexity of the motion.

Here we have shown that the simulation method can be used to analyse systems with a high number of sliding segments per period by calculating bifurcation diagrams. Together with a continuation code, as discussed in Section 4.1, a bifurcation diagram constitutes a powerful tool to analyse linear Filippov system that arise in many electronic systems.

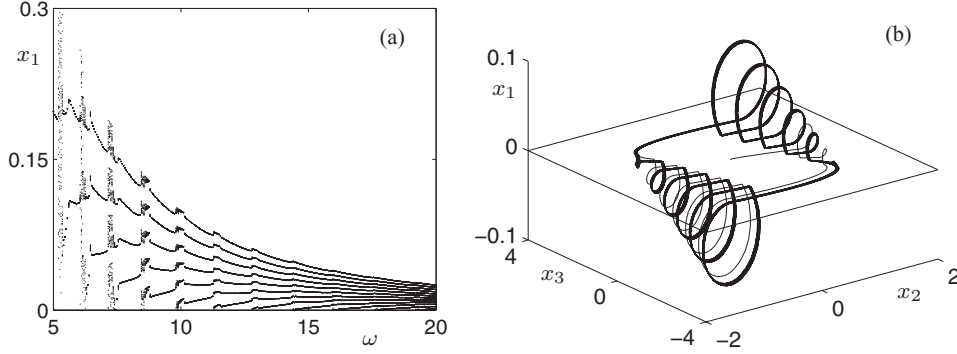


Fig. 6. State-space and time-history diagrams for the relay feedback system. (a) A bifurcation diagram showing x_1 versus ω for $\zeta = -0.07$, where the Poincaré surface is given by $\mathcal{L}_{F_1}(H)(x) = 0$. (b) A state space diagram for $\zeta = -0.07$, $\omega = 10$, where the initial condition is taken to be on the sliding surface.

4.3 A Drill-String System

A simple model of the motion of a drill-string consists of two discs separated by a string with a brake connected to the lower disc. The upper disc represents a rotary table to which a drive motor is connected via a gear box and the string. The lower disc represents the drill-string with the bottom-hole-assembly at the drill-rig and the additional brake implements the friction forces between the drill bit and the bore hole. Such a drill-string system is described by Mihajlovic et al. [2006] and modeled by

$$J_u \ddot{\theta}_u + k_\theta (\theta_u - \theta_l) + T_{fu}(\theta_u) = k_m u, \quad (38)$$

$$J_l \ddot{\theta}_l - k_\theta (\theta_u - \theta_l) + T_{fl}(\theta_l) = 0, \quad (39)$$

where θ_u (θ_l), J_u (J_l) and T_{fu} (T_{fl}) are respectively the angle, the moment of inertia and friction torque of the upper (lower) disc. Furthermore, k_θ is the torsional stiffness of the drill string, u is the input voltage and k_m is a motor constant. Making the assumption used by Mihajlovic et al. [2006] on the friction laws at the upper and lower discs, we have

$$T_{fu}(\dot{\theta}_u) = \begin{cases} T_{fu}^+ = T_{sup} + b_{up}\dot{\theta}_u, & \dot{\theta}_u > 0 \\ T_{fu}^- = -T_{sun} + b_{un}\dot{\theta}_u, & \dot{\theta}_u < 0 \end{cases} \quad (40)$$

and

$$T_{fl}(\dot{\theta}_l) = \begin{cases} T_{fl}^+ = T_{sl} + T_1 \left(1 + \frac{2}{1 + e^{\beta_1 \dot{\theta}_l}}\right) + T_2 \left(1 + \frac{2}{1 + e^{\beta_2 \dot{\theta}_l}}\right), & \dot{\theta}_l > 0, \\ T_{fl}^- = -T_{sl} - T_1 \left(1 + \frac{2}{1 + e^{-\beta_1 \dot{\theta}_l}}\right) - T_2 \left(1 + \frac{2}{1 + e^{-\beta_2 \dot{\theta}_l}}\right), & \dot{\theta}_l < 0, \end{cases} \quad (41)$$

where b_{up} , b_{un} , T_{sup} , T_{sun} , T_{sl} , T_1 , T_2 , β_1 and β_2 are all constants. By letting

$$x = (x_1, x_2, x_3)^T = (\theta_u - \theta_l, \dot{\theta}_u, \dot{\theta}_l)^T$$

the drill-string system ((38)–(39)) can be written as

$$\dot{x} = \begin{pmatrix} x_2 - x_3 \\ \frac{k_m}{J_u}u - \frac{k_\theta}{J_u}x_1 - \frac{1}{J_u}T_{fu}(x_2) \\ k_\theta x_1 - \frac{1}{J_l}T_{fl}(x_3) \end{pmatrix}. \quad (42)$$

From Equations (40) and (41), we can determine that there are two discontinuity surfaces Σ_1 and Σ_2 defined by

$$\Sigma_1 = \{x \in \mathbf{R}^3 \mid H_1(x) = 0\}, \quad \Sigma_2 = \{x \in \mathbf{R}^3 \mid H_2(x) = 0\}, \quad (43)$$

where

$$H_1(x) = x_2 \quad \text{and} \quad H_2(x) = x_3. \quad (44)$$

Notice that we are using the methodology and notation introduced in Section 3.2. Now it is straightforward to write (42) in our preferred format:

$$\dot{x} = F_1(x) + \Delta_1(x)\mu_1(x) + \Delta_2(x)\mu_2(x), \quad (45)$$

where

$$F_1(x) = \begin{pmatrix} x_1 - x_2 \\ \frac{k_m}{J_u}u - \frac{k_\theta}{J_u}x_1 - T_{fu}^+(x_2) \\ \frac{k_\theta}{J_l}x_1 - T_{fl}^+(x_3) \end{pmatrix}, \quad (46)$$

$$\Delta_1(x) = \begin{pmatrix} 0 \\ T_{fu}^+(x_2) - T_{fu}^-(x_2) \\ 0 \end{pmatrix} \quad \text{and} \quad \Delta_2(x) = \begin{pmatrix} 0 \\ 0 \\ T_{fl}^+(x_3) - T_{fl}^-(x_3) \end{pmatrix}. \quad (47)$$

Here it is important to point out that the four possible vector fields are linearly dependent, so that we have a uniquely defined sliding vector field in the intersection between discontinuity surfaces. Since this system has two discontinuity surfaces, due to the Coloumb friction models at the upper and lower discs, it is possible for the discontinuity surfaces to cross, which practically means that the upper and lower disc can get stuck at the same time. However, for the example examined here limit cycles with sliding along one surface (Σ_2) only has been found. In Figure 7 we show four limit cycles I, II, III, and IV corresponding to output voltages $u = 1$, $u = 3$, $u = 3$ and $u = 5$, respectively. The limit cycle I does not have any sliding segments but as the voltage is increased to 2 the limit cycle has undergone a grazing-sliding bifurcation and both the periodic solutions II and III clearly have a sliding segment. As the voltage is increased further the limit cycle changes its character via a *switching-sliding* bifurcation. As seen in Figure 7(b) the limit cycle has an extra discontinuity surface crossing, but the number of sliding segments remains the same.

This is a good example where numerical experiments show how dynamics change under parameter variations, thus giving an impulse for further analytical and numerical investigation.

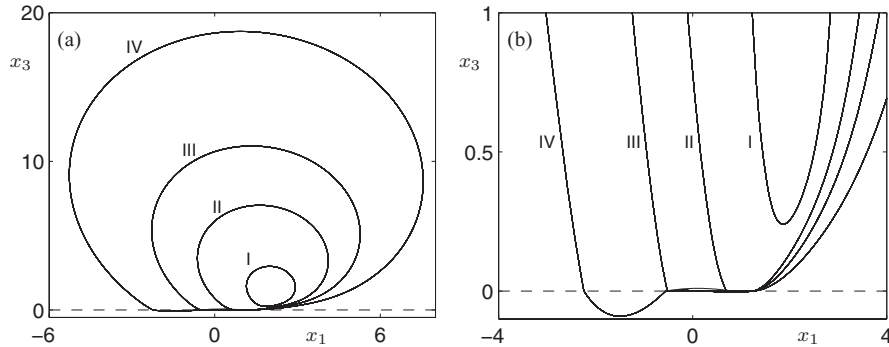


Fig. 7. State-space diagrams for the drill-string system. (a) A projection of the state diagram showing x_3 against x_1 for four limit cycles 'I', 'II', 'III' and 'IV' corresponding to $u = 1$, $u = 2$, $u = 3$ and $u = 5$, respectively. (b) A close up of panel (a) to show the transformation from the non-sliding limit cycle to sliding limit cycles via two sliding bifurcations. The rest of the system parameters are $k_m = 3.5693$, $k_\theta = 0.078$, $J_u = 0.4765$, $J_l = 0.0326$, $T_{sup} = 0.3216$, $T_{sun} = 0.3026$, $b_{up} = b_{un} = 1.9667$, $T_{sl} = 0.0940$, $T_1 = 0.0826$, $T_2 = -0.2910$, $\beta_1 = 6.3598$ and $\beta_2 = 0.0786$

5. CONCLUSIONS AND OUTLOOK

The introduced methods give a simple way to automatically simulate generic orbits of Filippov systems using a hybrid system approach, and where the user only has to introduce information about the vector fields and discontinuity surfaces. Although many research groups have developed simulation environments for specific nonsmooth problems, the authors are unaware of a general Filippov system solver, such as the one proposed here. Therefore we hope that this software, and the ideas upon which it is built, will be used by both applied mathematicians and engineers for such simulations, and hopefully make the whole community think of these problems in a more general setting.

The obvious strength of the proposed simulation code is that it is relatively simple to use, as long as the user has some knowledge of MATLAB. One problem with the hybrid system approach though is that the combinatoric complexity of the code grows quickly as the number of discontinuity surfaces is increased. Also, we choose MATLAB since it is relatively simple to test ideas, implement code and use the final product. The downside is of course that MATLAB is much slower than an equivalent program written in, for instance, C, C++ or FORTRAN. So if the code is to be used for larger-scale problems, for example, with many discontinuity surfaces, such an implementation is preferable. Also, for even more efficient calculations it would be useful to have a script that initially generates a code for the user-specific problem, in terms of the state space dimension and the number of discontinuity surfaces.

As with any hybrid method a potentially dangerous phenomenon is *chatter* or *Zeno phenomenon* [Budd and Dux 1994], that is, the appearance of infinitely many switches in a finite time interval. The way to deal with this is to include an additional local map to step over the tail of the chatter sequence. Another possible problem can occur for a point on the discontinuity surface, where two vector fields are tangent to the discontinuity surface and the sliding vector field is not defined, see (10). In generic planar Filippov systems without parameters, these

singular sliding points do not appear. However, they appear in generic one-parameter families of planar Filippov systems as collisions of tangent points (see Kuznetsov et al. [2003]), as well as in generic n -dimensional parameter-independent Filippov systems with $n \geq 3$ and their families as intersections of the tangent surfaces (see, e.g., Filippov [1988](Chapter 5)). Therefore, it is also necessary to describe the behaviour of the proposed method near such singularities. In planar Filippov systems, singular sliding points nonisolated in the sliding surface are not harmful, since the sliding vector field can be extended by continuity to these points (if we neglect infinitely wdegenerate cases, see [Kuznetsov et al. 2003]). Note that isolated singular sliding points are usually considered as equilibria. Numerical experiments show that our method, indeed, steps over non-isolated singular points correctly. In multidimensional Filippov systems, the situation is more involved and requires additional theoretical analysis. However, preliminary numerical experiments demonstrate robust behaviour of the code also in these cases.

As mentioned earlier, the algorithm can also be used as a building block for a continuation algorithm that follow both periodic orbits in one parameter and codimension-one bifurcations in two parameters, using Poincaré maps (i.e., shooting). Such an implementation will be discussed in future works [Kowalczyk and Piiroinen 2006]. We also hope to develop an interactive general simulation and continuation environment that supports both discontinuous vector fields and state jumps, so that the user specifies a surface as continuous, Filippov, or impacting.

APPENDIXES

A. HOW TO USE THE CODE

The method described in the present paper in the case of a single discontinuity surface has been implemented in MATLAB. Here we will give some details on how to use the programs. Templates for the files can be found in Piiroinen [2007]. There one can also find the files used for the examples in Section 4, including the drill-string example with two discontinuity surfaces.

The program consists of three files,

```
run_oscillator.m
filippov.m
vectorfields.m
```

and possibly the optional files

```
pfunction.m
jacobians.m
```

The file `run_oscillator.m` is the main program in which integration time, parameters, initial conditions, names of the vector field, Jacobian matrix, and Poincaré function files, the name of the ODE solver and its properties (e.g., error tolerances) are introduced by the user. Notice that as with any Matlab simulation one should use the solver that fits the problem best, for example, `ode15s` for stiff problems, and the error tolerances should also be changed depending on the

amount of accuracy that is expected. This `run_oscillator.m` file also calls the file `filippov.m`, which takes care of the event handling and also calls the MATLAB built-in ODE solver and event detector. In the vector field file `vectorfields.m`, which is used by `filippov.m`, the user adds the different vector fields, the function defining the discontinuity surface, the normal to the discontinuity surface, and possibly the function defining a Poincaré surface. In `pfunction.m` the user defines the action to be taken at the Poincaré surface. Finally, to make location of near-grazing trajectories within the sliding surface more robust the Jacobian matrices corresponding to the different vector fields can be added in the file `jacobians.m`. In many cases this is not necessary, but if one expects to have adding-sliding bifurcations this is a good idea.

It has to be noted that no particular effort has been made to optimize the code at this stage. The main reason is to make it as transparent as possible so that users can make suitable changes themselves.

Let us now show explicitly what to write in the different files specified by the user. The system we will use in this exposition is the same dry-friction oscillator as described in Section 4.1, where a more thorough introduction to this system is presented. Let the dynamical system be given by

$$\dot{x} = \begin{cases} F_1(x), & x_2 > 0, \\ F_2(x), & x_2 < 0, \end{cases} \quad (48)$$

where $x = (x_1, x_2, x_3)^T$,

$$F_1(x) = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F \\ \omega \end{pmatrix}, \quad F_2(x) = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) + F \\ \omega \end{pmatrix}, \quad (49)$$

and F and ω are constants. The user needs to adjust the different files as follows.

run_oscillator.m

The following has to be given by the user:

ODE-solver

`solver = 'ode45'`; (The choice of MATLAB ODE solver.)

MATLAB's ODE-solver properties

`opts = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.1);`

Name of the vector field file

`vfields = 'vectorfields';`

Name of the Jacobians file

`vfields = 'jacobians';`

Name of the Poincaré function file

`pfunction = 'pfunction';`

Filippov parameter (See appendix B)

`C = 1;`

Parameter list

`F = 0.4; omega = 1/3; params = [F,omega];`

Initial conditions

```
y0 = [1,2,0];
```

Integration time

```
T = 2*pi/omega*10; tspan = [0,T]; (10 periods of the periodic forcing.)
```

The output that will be given by `filippov.m` is a list of the time t , and the corresponding values of the states y , the times t_e at the events, the state vector y_e at the events, an index list i_e of the events, and the event variables se at the events (cf. the event handling in `MATLAB`).

`vectorfields.m`

In this file the user has to specify the following:

Parameters

```
F = params(1); omega = params(2);
```

The two vector fields

```
F1 = [y(2); -y(1)+sin(y(3))-F; omega]; ( $H > 0$ )
```

```
F2 = [y(2); -y(1)+sin(y(3))+F; omega]; ( $H < 0$ )
```

The function defining the discontinuity surface

```
H = y(2);
```

The gradient of H

```
dH = [0,1,0];
```

The Poincaré surface

```
h = y(3)-2*pi; (The system is  $2\pi$ -periodic in  $x_3$ .)
```

Location direction for the Poincaré section

```
hdir = 1;
```

`jacobians.m`

In this file the user has to specify the following:

Parameters

```
(No parameters needed)
```

The two Jacobians

```
J1 = [0,1,0; -1,0,cos(y(3)); 0,0,0]; ( $H > 0$ )
```

```
J2 = J1; ( $H < 0$ )
```

The second derivative of H

```
d2H = zeros(3,3);
```

`pfunction.m`

In this file the user has to specify the following:

Parameters

```
(No parameters needed)
```

Action at Poincaré section

```
y1 = [y(1), y(2), 0]; (Reset of the scaled time.)
```

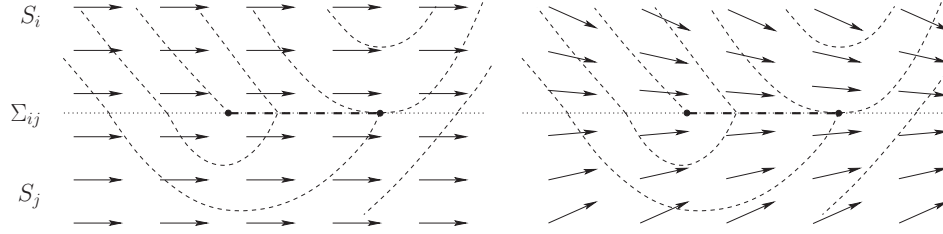


Fig. 8. The sliding vector fields F_{ij} (left) and \hat{F}_{ij} (right) in a neighbourhood of the discontinuity surface Σ_{ij} .

B. REGULARIZING THE SLIDING VECTOR FIELD

Since a numerically constructed sliding solution might not follow the discontinuity surface exactly, we have to consider the vector field F_{ij} not only for $x \in \hat{\Sigma}_{ij}$ but in a neighbourhood of $\hat{\Sigma}_{ij}$. Under the above introduced genericity conditions, Equations (9) and (10) can be used to define the vector field F_{ij} in such a neighbourhood, if we formally allow $|\mu_{ij}(x)|$ to be greater than 1. Note that orbits of the extended vector field F_{ij} pass through the boundary of the sliding surface $\hat{\Sigma}_{ij}$, i.e., the tangent surfaces $\hat{\Sigma}_{ij}^{\pm}$. This property, that is illustrated in Figure 8 (left), is essential for the event detection and location described in the next section. However, this extended vector field has a family of *invariant surfaces* $h_{ij} = \text{const}$, one of which is Σ_{ij} given by $h_{ij} = 0$. Therefore, for a numerical solution of the sliding equation, there is the (unwanted) possibility of drifting away from the discontinuity surface, to one of the other invariant surfaces. This occurs due to accumulation of numerical errors combined with the neutral stability of the surface $\hat{\Sigma}_{ij}$ for $\dot{x} = F_{ij}(x)$ (see the left panel of Figure 8). One way to avoid such a drift is to make the sliding surface $\hat{\Sigma}_{ij}$ attracting as long as the motion governed by the sliding vector field (9) is constrained to it. This can be done by introducing a new sliding vector field \hat{F}_{ij} by adding a *small* term to the original vector field F_{ij} that makes the sliding surface *locally attractive* (see the right panel of Figure 8). For example, this vector field can be defined as

$$\hat{F}_{ij}(x) = F_{ij}(x) - Ch_{ij}(x) \left(\frac{dh_{ij}}{dx}(x) \right)^T, \quad (50)$$

where C is a positive constant. It is clear that $\hat{\Sigma}_{ij}$ is locally attracting since the new term is orthogonal to $\hat{\Sigma}_{ij}$ and points towards the surface. Furthermore, for $x \in \hat{\Sigma}_{ij}$, we have

$$Ch_{ij}(x) \left(\frac{dh_{ij}}{dx}(x) \right)^T = 0,$$

which is exactly what we want since the extra term does not interfere with the sliding vector field as long as the solution stays on Σ_{ij} . It is always possible to choose $C > 0$ such that the motion in the normal direction will be faster than sliding along the discontinuity surface. However, one should avoid choosing C too big because in this case the ODE system $\dot{x} = \hat{F}_{ij}(x)$ becomes stiff. Similar techniques to make constraint surfaces attractive are used in numerical

integration of differential-algebraic equations (DAEs) (see, e.g., [Ascher and Petzold 1998]).

When $|\mu_{ij}(x)| > 1$ for $x \in \Sigma_{ij}$, then the motion will not be locally constrained to Σ_{ij} , and we note that $F_{ij} = \hat{F}_{ij} = F_i$ if $\mu_{ij} = -1$ and $F_{ij} = \hat{F}_{ij} = F_j$ if $\mu_{ij} = 1$. Therefore, to compute a solution passing through a point where $|\mu_{ij}(x)| > 1$, we must switch to F_i if $\mu_{ij}(x) < -1$ and F_j if $\mu_{ij}(x) > 1$.

ACKNOWLEDGMENTS

The authors would like to thank Prof Alan Champneys and Dr Arne Nordmark for valuable comments, and Dr Mihajlovic for drawing our attention to the drill-string model.

REFERENCES

- ARNOLD, V. I. 1973. *Ordinary Differential Equations*. MIT Press, Cambridge, MA.
- ASCHER, U. M. AND PETZOLD, L. R. 1998. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, PA.
- ASHINO, R., NAGASE, M., AND VAILLANCOURT, R. 2000. Behind and beyond the MATLAB ODE suite. *Comput. Math. Appl.* 40, 491–512.
- BROGLIATO, B. 2003. Some perspectives on the analysis and control of complementarity systems. *IEEE Trans. Auto. Control* 48, 918–935.
- BUDD, C. J. AND DUX, F. 1994. Chattering and related behaviour in impact oscillators. *Phil. Trans. Roy. Soc. London A* 347, 365–389.
- DERCOLE, F. AND KUZNETSOV, YU. A. 2005. SlideCont: An Auto97 driver for sliding bifurcation analysis. *ACM Trans. Math. Softw.* 31, 95–119.
- DI BERNARDO, M., JOHANSSON, K. H., AND VASCA, F. 1999. Sliding orbits and their bifurcations in relay feedback systems. In *Proceedings of the 38th IEEE Conference on Decision and Control*. Phoenix, AZ.
- DI BERNARDO, M., JOHANSSON, K. H., AND VASCA, F. 2001. Self-oscillations and sliding in relay feedback systems: Symmetry and bifurcations. *Int. J. Bifurcat. Chaos* 11, 4, 1121–1140.
- DI BERNARDO, M., KOWALCZYK, P., AND NORDMARK, A. 2002. Bifurcations of dynamical systems with sliding: derivation of normal-form mappings. *Physica D* 170, 175–205.
- DOEDEL, E. J., CHAMPNEYS, A. R., FAIRGRIEVE, T. F., KUZNETSOV, YU. A., SANDSTEDTE, B., AND WANG, X. 1997. *Auto97: Continuation and bifurcation software for ordinary differential equations (with HomCont)*. Computer Science, Concordia University, Montreal, Canada, <ftp.cs.concordia.ca/doedel/doc/auto>.
- FEIGIN, M. I. 1994. *Forced Oscillations in Systems With Discontinuous Nonlinearities*. Nauka, Moscow USSR. (In Russian).
- FILIPPOV, A. F. 1988. *Differential Equations with Discontinuous Righthand Sides*. Kluwer Academic Publishers.
- GALVANETTO, U. 2001. Some discontinuous bifurcations in a two block stick-slip system. *J. Sound Vibration* 284, 4, 653–669.
- GALVANETTO, U. AND BISHOP, S. R. 2000. Dynamics of a simple damped oscillator undergoing stick-slip vibrations. *Meccanica* 34, 337–347.
- GEAR, C. W. AND ØSTERBY, O. 1984. Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Softw.* 10, 23–44.
- JEAN, M. 1999. The non-smooth contact dynamics method. *Comput. Meth. Appl. Mechan. Engin.* 177, 3-4, 235–257.
- KOWALCZYK, P. 2003. Analytical and numerical investigations of sliding bifurcations in n-dimensional piecewise-smooth systems. Ph.D. thesis, Department of Engineering Mathematics, University of Bristol, Bristol, UK.
- KOWALCZYK, P. AND PIROINEN, P. T. 2006. Numerical and analytical investigations of sliding bifurcations in a dry-friction oscillator. *Physica D*, to appear.

- KUZNETSOV, YU. A., RINALDI, S., AND GRAGNANI, A. 2003. One-parameter bifurcations in planar Filippov systems. *Int. J. Bifur. Chaos Appl. Sci. Engin.* 13, 2157–2188.
- LEINE, R. I. AND NIJMEIJER, H. 2004. Dynamics and bifurcations of non-smooth mechanical systems. Lecture Notes in Applied and Computational Mechanics, vol. 18. Springer-Verlag, Berlin, Germany.
- MIHAJLOVIC, N., VAN DE WOUW, N., HENDRIKS, M., AND NIJMEIJER, H. 2006. Friction-induced limit cycling in flexible rotor systems: An experimental drill-string set-up. *Nonlinear Dynamics* 46, 237–291.
- MOREAU, J. J. 1999. Numerical aspects of the sweeping process. *Comput. Meth. Appl. Mechan. Engin.* 177, 3-4, 329–349.
- PIIROINEN, P. 2007. Filippov system solver for Matlab. Department of Mathematical Physics, National University of Ireland, Galway, Galway, Ireland, www.maths-physics.nuigalway.ie/users/Piiroinen/Filippov/.
- SHAMPINE, L. F. AND REICHEL, M. W. 1997. The MATLAB ODE suite. *SIAM J. Sci. Comput.* 18, 1–22.
- UTKIN, V. I. 1992. *Sliding Modes in Control Optimization*. Springer-Verlag, Berlin, Germany.

Received June 2005; revised July 2006; accepted February 2007